ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.04

СОГЛАШЕНИЯ «POSIX» ПО ОБРАБОТКЕ ОПЦИЙ И АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ

О.В. СУХОРУКОВ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматривается стандарт «POSIX» касательно соглашений по обработке аргументов командной строки, которые можно рассматривать как совокупность договоренностей между разработчиком операционной системы и прикладным программистом. Представлены основные определения соглашений, приводятся примеры использования, комментарии автора.

Работа с командной строкой – это тот базис, который предоставляет гибкость и мощь UNIX/Linux систем, а соблюдение соглашений «POSIX» (Portable Operating System Interface) касательно опций и аргументов командной строки облегчает и стандартизирует процесс взаимодействия с командной оболочкой – поведение незнакомой программы на конкретную команду становится ожидаемым.

Компьютерная программа – это последовательность инструкций, написанных на одном или более языке программирования и предназначенных для исполнения вычислительной машиной [1]. Программа может получать данные извне. На начальном этапе исполнения основным способом получения данных может служить обработка аргументов командной строки.

В программах, созданных на линейке языков «C» таким способом, является чтение аргументов функции «main()» [2, с. 58], которая служит в качестве начальной точки исполнения программы [3]. В этом случае данные называют «аргументами программы», которые не стоит отождествлять с «аргументами командной строки». Это разные понятия. Дело в том, что программа получает свои аргументы от родительского процесса, в качестве которого не всегда выступает командная оболочка. Таким процессом, к примеру, может оказаться браузер, в котором в настоящее время никакой «командной строки» нет [2, с. 58].

В момент запуска программы функции «main()» передаются два аргумента. В первом, обычно называемом «argc» (сокращение от *argument count*), стоит количество аргументов, заданных в командной строке. Второй параметр, называемый «argv» (от *argument vector*), является указателем на массив символьных строк, содержащих сами аргументы. Для работы с этими строками обычно используются указатели нескольких уровней. По общепринятому соглашению «argv[0]» – это имя вызываемой программы, так что значение «argc» никогда не бывает меньше 1 [4, с. 150].

Аргументом командной строки может служить любой набор символов. Между собой аргументы обычно разделяются пробелами либо символами табуляции, за исключением случаев, когда разделитель экранирован кавычками [3]. Несмотря на то, что организация и дальнейшее использование аргументов строго не регламентируются и могут быть ограничены лишь фантазией программиста, в «UNIX/Linux» среде существует общепринятая договоренность с набором рекомендаций по их структуре и применению.

Такой набор рекомендаций описывается в так называемом стандарте «POSIX» (*Portable Operating System Interface*), который можно рассматривать как совокупность договоренностей между разработчиком операционной системы и прикладным программистом. «Договоренность» означает, прежде всего, одинаковость интерпретации (семантики) слов и выражений [5]. Помимо обработки командной строки стандарт «POSIX» описывает множество базовых, системных сервисов, необходимых для функционирования прикладных программ, доступ к которым предоставляется посредством интерфейса, специфицированного для языка «C», командного языка и общеупотребительных служебных программ [6; 7].

«POSIX» не ограничен рамками «UNIX/Linux» среды. Существуют операционные системы (ОС) «независимого происхождения» (например, системы реального времени), предоставляющие необходимые сервисы и тем самым поддерживающие выполнение «POSIX-совместимых» приложений. Можно утверждать, что следование стандарту «POSIX» облегчает перенос приложений практически на любую операционную платформу. При этом дополнительные усилия по повышению мобильности, прилагаемые на этапе разработки, безусловно, окупаются [6].

В первую очередь, стандарт «POSIX» разделяет аргументы командной строки на «опции» и «операнды» (независимые аргументы).

«Опции» являются специальными управляющими аргументами командной строки, которые призваны изменять поведение программы или предоставлять программе дополнительную информацию. По старому соглашению, опции начинаются с дефиса (знак минус) и состоят из единственной буквы [8, с. 38]. Это

так называемые короткие опции. В новом соглашении также существует поддержка длинных опций, начинающихся с двух, подряд стоящих дефисов (знаков минусов). Длинная опция может состоять из любого количества символов, исключающих символ разделитель [7].

«Операндом» в этом случае считаются все остальные аргументы командной строки, не являющиеся «опциями». Имя вызываемой программы «argv[0]» функции «main()» также является «операндом».

Основные перефразированные соглашения стандарта «POSIX» по опциям и аргументам (операндам) [7] представлены в таблицах 1 и 2.

Таблица 1 – соглашения «POSIX» по коротким опциям и аргументам

№	POSIX	Пример	Комментарий
1	Имя опции должно быть простым буквенно-цифровым символом. Опции с множеством цифр не допускаются	-a -b -c -1 -2 -3	Опция «-W» зарезервирована для производителей, реализующих утилиты «POSIX»
2	Все опции должны начинаться с символа «-»		Символ «-» отличает опцию от аргумента. (Последовательность из двух символов «--» отличает длинную опцию от короткой)
3	Для опций, не требующих аргументов (операндов), должно быть возможно объединение нескольких опций после единственного символа «-»	foo -a -b -c foo -abc	Оба примера должны интерпретироваться одинаково
4	Если опция требует аргумент, он должен быть отделен от опции пробелом	fgrep -f patfile	На практике это соглашение применяется редко – опция и ее аргумент (операнд) обычно могут находиться в одной строке: «fgrep -fpatfile» (такое поведение реализуется функциями «getopt()» и «getopt_long()»)
5	Аргументы опций не должны быть необязательными	fgrep -f <patfile>	Это означает, что если в документации программы указано, что опции требуется аргумент, этот аргумент должен присутствовать всегда, иначе программа потерпит неудачу. Однако, как и с предыдущим пунктом, это одно из тех соглашений, которое не всегда поддерживается. К примеру, «GNU getopt()» предусматривает необязательные аргументы опций, поскольку считает, что они могут быть полезны
6	Если опция принимает аргумент, который может иметь несколько значений, программа должна получать этот аргумент в виде одной строки со значениями, разделенными запятыми или иным установленным разработчиком разделителем	prog -u pasha,masha,dasha prog -u «pasha masha dasha»	В первом примере разделение осуществляется запятыми, во втором – пробелами, аргументы опции в этом случае экранируются кавычками.
7	Опции должны находиться в командной строке первыми, перед аргументами (операндами)	prog -abc arg1 arg2	Версии «getopt() Unix» придерживаются этого соглашения, версии «GNU getopt()» по умолчанию этого не делают, однако это можно настроить
8	Порядок, в котором приведены опции, не должен играть роли. Однако, для взаимно исключающих опций, когда одна опция перекрывает установки другой, последняя считается главной	prog -abc prog -bca prog -cab	Все примеры должны интерпретироваться одинаково
9	Порядок, в котором приведены аргументы, имеет для программы значение	prog arg1 arg2 arg3	Каждая программа должна четко документировать последовательность расположения используемых аргументов
10	Программы, читающие или записывающие именованные файлы, должны трактовать единственный аргумент «-» как означающий стандартный ввод или стандартный вывод, в зависимости от того, что подходит программе		Если операнд задает читаемый или записываемый файл, то знак минус на его месте используется только для обозначения стандартного ввода (или стандартного вывода, если из контекста ясно, что специфицируется выходной файл)

Следует отметить, что даже многие стандартные программы не следуют всем указанным соглашениям. Главной причиной является историческая совместимость, многие такие программы предшествовали систематизации этих соглашений [8, с. 40].

«Длинные опции» – это естественное развитие стандарта «POSIX» в ответ на тенденции развития информационных технологий. Компьютеры стали быстрее, память увеличилась в геометрической прогрессии, возможности программ стали шире, а алгоритмы сложнее [9]. Отпала необходимость в экономии места, появились технологии интеллектуального дополнения ввода, при которых нивелировалась разница в скорости ввода между короткими опциями и их длинными соотношениями. Длинные опции легче запомнить, и они предоставляют возможность стандартизировать их применение среди всех утилит GNU. Хорошим примером будет являться опция «--help», которая обычно не изменяет своего значения среди утилит GNU, чего нельзя сказать по поводу ее сокращенной версии «-h».

Поскольку длинные опции начинаются с «--», совместное применение их с короткими опциями не конфликтует с предшествующими соглашениями «POSIX» [7]. Длинные опции GNU имеют свои собственные соглашения, реализованные в функции «getopt_long()». Основные перефразированные соглашения стандарта «POSIX» по длинным опциям представлены в таблице 2.

Таблица 2 – соглашения «POSIX» по длинным опциям

№	POSIX	Пример	Комментарий
1	Каждая коротка опция (один символ) должна иметь также свой вариант в виде длинной опции	-h --help	Оба примера должны интерпретироваться одинаково
2	Дополнительные специфические для GNU опции не нуждаются в соответствующей короткой опции	--trulala	В документации к стандарту «POSIX» в разделе 4.9 прилагается таблица стандартизированных длинных опций по утилитам GNU [7]. Опция, не входящая в эту таблицу, считается специфической для GNU (см. пример) и не нуждается в соответствующей короткой опции, однако тем же стандартом «POSIX» рекомендуется ее создать
3	Длинную опцию можно сократить до кратчайшей строки, которая остается уникальной	--verbose (--verbo) --verbatim (--verba)	В примере приложение регламентирует две опции «--verbose» и «--verbatim». Стандартом «POSIX» допускается при использовании опций сокращать их имена до минимального значения, про которых программа сможет их идентифицировать. Минимальные допустимые значения в этом случае будут «--verbo» и «--verba» соответственно
4	Аргументы опций могут быть необязательными		Для таких опций считается, что аргумент присутствует, если он находится в одной строке с опцией. Однако в существующей реализации «getopt_ long()» это работает лишь для коротких опций. Например, если «-x» такая опция и дана строка «foo -xYANKEES -y», аргументом «-x» является «YANKEES». То для «foo -x -y», у опции «-x» нет аргументов [8, с. 40]
5	Программы могут разрешить длинным опциям начинаться с одного символа «-»		Такое поведение типично для многих программ X Window [8, с. 40]. По мнению автора, такое допущение в соглашении идет в разрез с соглашениями «POSIX» по коротким опциям и является грубейшей ошибкой
6	Специальный элемент командной строки «--», который не является ни опцией, ни операндом, обозначает конец опций. Все последующие слова трактуются как операнды, даже если они начинаются со знака минус	prog -- -a -b --opt	В текущем примере «-a», «-b», «--opt» интерпретируются не как опции, а не как аргументы

Проанализировав соглашения в таблицах 1, 2, можно подытожить сказанное выше следующим выводом: опции согласно стандарту «POSIX» – это унифицированный интерфейс для передачи данных в программу через аргументы [2, с. 59].

Механизм опций стандарта «POSIX» позволяет разделить все аргументы программы на следующие категории:

1 Опции – различают:

- короткие (односимвольные);
- длинные (многосимвольные).

2 Зависимые аргументы (аргументы опций) – это аргументы, наличие которых у опции предполагается. Например, имя файла, которое указывается вслед за опцией «-o» компилятора «gcc», является зависимым аргументом. Опции, не требующие наличия зависимых аргументов, называют флагами. Как правило, одна опция может принимать только один такой аргумент.

3 Свободные аргументы – это аргументы, которые не связаны напрямую с опциями. Например, имя каталога, передаваемое программе «rmdir», является свободным аргументом.

Процедура обработки командной строки считается тривиальной задачей. Массив «argv» содержит аргументы командной строки, «argc» указывает, сколько их было передано, а стало быть для работы с командной строкой программам вполне достаточно параметров функции «main()». Но только до тех пор, пока программист не решит следовать соглашениям «POSIX».

Анализ таблиц 1, 2 раскрыл огромное количество нюансов, которое необходимо реализовать, это и объединение опций, и разделение опций на длинные и короткие, контроль обязательных и необязательных аргументов и т. п.

Еще примерно в 1980-х годах группа поддержки «UNIX» для «System III» в «AT&T» заметила, что каждая программа «UNIX» использовала для разбора аргументов свои собственные методики. Чтобы облегчить работу программистов в упрощении написания кода, придерживающегося стандартных соглашений, была разработана функция «getopt()». Функция GNU «getopt_long()» предоставляет совместимую с «getopt()» версию, а также упрощает разбор длинных опций в описанной ранее форме [8, с. 43]. Однако эти функции до сих пор являются актуальными и используются практически без изменений. Первоначально функции были разработаны для «C», для использования в других языках, таких как «java», «Perl», «Python», «PHP» и др., существуют соответствующие оболочки, так называемые фреймворки над функциями, позволяющие задействовать их в своем коде [10].

Подавляющее большинство ПО в UNIX/Linux среде используют для своей работы аргументы командной строки. Даже программы с графическим интерфейсом, которым, казалось бы, и вовсе не нужна консоль, интерпретируют десятки входных параметров. Это может быть задание начальных настроек, включение отладочного режима или просто вывод версии программы без ее запуска [10]. И это не только дань традиции, такой подход позволяет обмениваться выходными данными [11], использовать функционал других программ, работать с программой через «голую» консоль, не имея графического окружения, объединять, дополнять, и даже изменять функционал программ, не изменяя их кода [12]. То есть работа с командной строкой это и есть тот базис, который предоставляет гибкость и мощь UNIX/Linux систем, а соблюдение соглашений «POSIX» касательно опций и аргументов командной строки, облегчает и стандартизирует процесс взаимодействия с командной оболочкой – поведение незнакомой программы на конкретную команду становится ожидаемым.

ЛИТЕРАТУРА

1. Интернет, компьютеры, софт и прочий Hi-Tech [Электронный ресурс] // Что такое компьютерная программа: сайт. – Электрон. текстов. дан. – [Б. м.], 2010. – Режим доступа: http://xbb.uz/soft/Chto_takoe_kompjuternaja_ programma. – Загл. с экрана.
2. Иванов, Н.Н. Программирование в Linux [Текст] : самоучитель / Н.Н. Иванов. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2012. – 400 с.
3. MSDN [Электронный ресурс] // Функция main и выполнение программ: сайт. – Электрон. текстов. дан. – [Б. м.], 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/3ze4ytsc.aspx. – Загл. с экрана.
4. Керниган, Б. Язык программирования Си [Текст] : учеб. метод. пособие / Б. Керниган, Д. Ритчи; пер. с англ. – 3-е изд., испр. – СПб. : Невский Диалект, 2001. – 352 с.
5. Романюк, С. Сюрпризы POSIX : сайт / С. Романюк // Журнал. Открытые Системы [Электронный ресурс]. – Электрон. текстов. дан. – [Б. м.], 1999. – Режим доступа: http://citforum.ru/operating_systems/articles/posix.shtml. – Загл. с экрана.
6. ИНТУИТ. Национальный открытый университет [Электронны ресурс] // В. Галатенко, Программирование в стандарте POSIX : сайт. – Электрон. текстов. дан. – [Б. м.], 2015. – Режим доступа: http://www.intuit.ru/studies/courses/47/47/lecture/1397. – Загл. с экрана.
7. GNU Operating System [Электронный ресурс] // GNU Coding Standards : сайт. – Электрон. текстов. дан. – [Б. м.], 2015. – Режим доступа: http://www.gnu.org/prep/standards. – Загл. с экрана.
8. Роббинс А. Linux: программирование в примерах [Текст] : учеб. метод. пособие / А. Роббинс; пер. с англ. – М. : КУДИЦ-ОБРАЗ, 2005. – 656 с.
9. Белорусов, А. Перспективы развития мирового рынка высоких технологий [Текст] / А. Белорусов, В. Вовченко // Белорусский журнал международного права и международных отношений. – 2002. – № 2. – С. 80–85.
10. Журнал LinuxFormat [Электронный ресурс] // Правильные аргументы : сайт. – Электрон. текстов. дан. – [Б. м.], 2015. – Режим доступа: http://wiki.linuxformat.ru/wiki/LXF112:Getopt. – Загл. с экрана.
11. Heap AltLinux [Электронный ресурс] // Ввод, вывод и конвейер: сайт. – Электрон. текстов. дан. – [Б. м.], 2008. – Режим доступа: http://heap.altlinux.org/modules/linux_pipeline/index.html. – Загл. с экрана.
12. OpenNET [Электронный ресурс] // Advanced Bash-Scripting Guide – Электрон. текстов. дан. – [Б. м.], 2016. – Режим доступа: http://www.opennet.ru/docs/RUS/bash_scripting_guide/. – Загл. с экрана.

УДК 004.04

НОВАЯ ПАРАДИГМА ОБРАБОТКИ АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ

О.В. СУХОРУКОВ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Описываются существующие методы обработки аргументов командной строки. Приводятся как их сильные, так и слабые стороны. Анализируя методы, выдвигаются два основных критерия, формирующих цель авторской разработки. Первый критерий – универсальность метода. Второй – создание максимального комфорта, нативно понятного интерфейса, быстрого старта при первом использовании не требующего глубокого изучения документации.

Универсальность авторского метода заключается в том, что за обработку опций отвечает отдельное приложение, принимающее в виде конфигурационного файла или строки структуру используемых опций, а также командную строку, требующую разбора. А после ее разбора и обработки выдает по требованию всю необходимую информацию.

Используя для декларации опций в конфигурационном файле сценарный язык программирования «JavaScript», автор добивается не только нативно понятного синтаксиса, но и тем самым увеличивает его гибкость и эффективность.

Технологии программирования не стоят на месте, эволюционируют сама парадигма программирования, определяющая не только совокупность идей и понятий, а также сам стиль написания компьютерных программ [1]. Выходя на более высокий уровень абстракций, код, по нашему мнению, становится более интуитивно понятным, читабельным. Развитие вычислительной техники все больше разворачивает стиль написания программ от кода, «как это будет удобно компилятору, процессору», до кода, «как это будет удобно человеку». Если рассматривать разработку библиотек, применяемых при создании программного обеспечения, во-первых, разработка ведется уже не только с учетом максимальной скорости работы ее методов, но и, во-вторых, с учетом создания максимального комфорта, быстрого старта при их использовании, нередко даже в ущерб первому.

Функции «getopt()» и «getopt_long()», являющиеся на сегодняшний день де-факто стандартным инструментом обработки аргументов командной строки в UNIX/Linux среде, были разработаны еще в 1980-х годах [2, с. 43]. Изначально функции создавались для языка «C», однако практически для любого языка на сегодняшний день существуют специальные библиотеки, реализующие их функционал или работающие на их базе. К примеру, «Argp», созданный для линейки языков «C» [3], «getopts» – для «bash» [4], «optparse» – для «Python» [5] и пр.

Функции «getopt()», «getopt_long()», а также их производные выглядят на сегодняшний день, на наш взгляд, крайне архаично. Для столь тривиальной, далеко не основной, но крайне важной задачи, как обработка аргументов командной строки [6], потребуется потратить, возможно, не один человеко-час, чтобы разобраться, как правильно ими пользоваться. При этом, по личному опыту, хорошим примером будет заполнение структур «Argp» [3], далеко не факт, что, единожды разобравшись, как это делается, вам не придется при следующем использовании разбираться снова.

Описанная проблема формирует очевидную цель – при разработке нового метода обработки аргументов командной строки приоритетной задачей является создание максимального комфорта, нативно понятного интерфейса, быстрого старта, при первом использовании не требующего глубокого изучения документации.

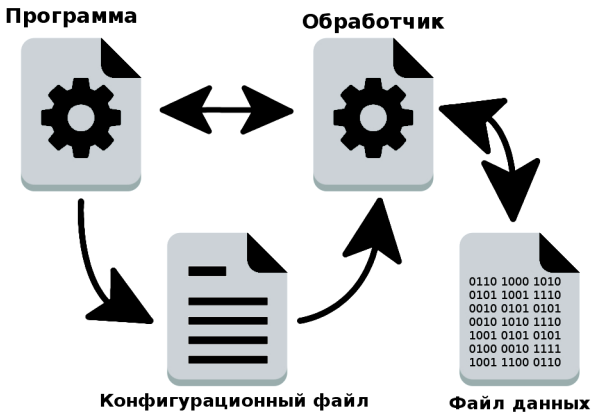
Не только нам интерфейс описанных методов казался излишне запутанным, устаревшим или имеющим недостаточный функционал. По этому поводу были написаны многочисленные статьи, не раз принимались и до сих пор принимаются в ряде случаев успешные попытки создать более простой интерфейс, при этом расширяющий функционал существующих методов. Хорошим примером будет служить библиотека «shflags», которая использует все ту же функцию «getopt()», но помимо парсинга опций умеет также и проверять их значения и даже самостоятельно обзывает переменные для опций согласно их длинному имени [7]. Подобные (удачные) аналоги стандартных функций можно отыскать практически в любом языке, к примеру, более продвинутая библиотека «argparse» для «Python» [8], заменяющая собой уже ставшую в ряде случаев стандартной, библиотеку «optparse» [5].

За кажущимся выходом из положения кроется не очевидная на первый взгляд проблема. Функции «getopt()» и «getopt_long()» были универсальными, так как были единственными, а значит не приходилось вникать в нюансы и отличия многочисленных методов. Сейчас же получается, что в каждом языке программирования, тем или иным способом устраняя недостатки первоначальных функций, имеется своя

реализация обработки аргументов командной строки, со своей библиотекой и набором методов, которая с каждой последующей версией все больше удаляется от исходных функций 80-х годов.

Автору, позиционирующему себя как системный программист-администратор, часто приходится писать небольшие вспомогательные программы, при этом быстро переключаясь с одного языка на другой. Это могут быть как всевозможные скрипты, написанные на одном из языков сценария, так и программы, написанные с использованием линейки языков «C». Обработка командной строки с учетом всех возможных соглашений «POSIX» (POSIX – Portable Operating System Interface) не может являться тривиальной задачей [6], что, в свою очередь, влияет на сложность применения всех известных автору методов обработки. Зачастую, по опыту автора, код обработки командной строки может оказаться длиннее и даже сложнее, чем код основной программы, к примеру, короткого скрипта. При этом постоянные переключения между методами обработки опций, разных языков программирования лишь усугубляют положение, дополнительно отнимая время и усилия.

Поэтому необходимо разработать, не только простой, но еще и обязательно универсальный метод обработки аргументов командной строки, не требующий переключения между языками. Это один из простых способов создания универсального метода, если за обработку опций будет отвечать отдельное приложение (рисунок). В этом случае «программа» передает «обработчику» в виде конфигурационного файла или строки структуру используемых опций, а также командную строку, требующую обработки. «Обработчик» разбирает командную строку согласно переданной структуре опций, сохраняет полученные данные во временный файл и завершает свою работу. По мере необходимости «программа» посылает запросы «обработчику», «обработчик» сверяется с файлом и отправляет «программе» соответствующий ответ.



Общая схема обработки

Схема с обработчиком в виде отдельной программы – цена, которую необходимо платить за универсальность. Такой способ нельзя назвать быстрым, однако обработка аргументов командной строки – это процедура, которая обычно и не требуется, – быстрых вычислений. Количество запросов можно сократить до минимума, вплоть до получения всех «формализованных» данных в ответ на передачу командной строки. Если скорость обмена информацией между «программой» и «обработчиком» важна, тогда по примеру баз данных вместо разовых вызовов достаточно открыть сессию, при которой «обработчик» будет завершать работу не после отработки очередного запроса, а после истечения определенного интервала времени. Если во время ожидания поступает новый запрос – интервал ожидания продлевается. При такой схеме работы «обработчику» не требуется при каждом запросе считывать или сбрасывать данные во временный файл, он сделает это только после истечения времени ожидания. Файл данных удаляется «обработчиком», по запросу, в конце жизненного цикла «программы».

Если за универсальность метода отвечает схема с обработчиком в виде отдельной программы, то за комфорт и нативно понятный интерфейс будет отвечать структура конфигурационного файла. На роль конфигурационного файла можно выбрать любой формат – от «YAML» [9], «XML» [10], «JSON» [11] до собственных разработок, однако, по нашему мнению, лучшим решением для формирования конфигурационного файла будет использование языка «JavaScript» [12].

Формат конфигурационного файла «YAML», к примеру, хотя и можно назвать «дружественным», однако нельзя назвать стандартным. Разделение синтаксических элементов производится посредством последовательности знаков тире завершающих специальным символом [9]. Это может оказаться непривычным для людей, не знакомых с форматом, а значит, может принести с собой дополнительные неудобства в работе.

Формат «XML» имеет избыточную разметку, за которой в итоге теряются основные данные [10].

«JSON», в отличие от «XML», имеет минималистскую разметку, не бросающуюся в глаза [11], а его расширенный формат – «JSON5» – поддерживает комментарии, имеет более упрощенный формат, допуская наличие мелких ошибок (к примеру, допускается наличие запятой после последнего элемента в объекте или списке) и пр. [13].

«JSON» основан на «JavaScript» [11], и его расширенный формат, в принципе, уже идеально подходит для использования в схеме «программа – обработчик». Однако используя вместо формата «JSON5», язык программирования, на котором он был основан, – «JavaScript» [12], мы не ограничиваем себя форматом, а получаем весь доступный функционал языка программирования, что открывает новые широкие возможности.

Использование сценарных языков программирования в конфигурационных файлах – метод не новый, но превосходно себя зарекомендовавший. Хорошим примером будет являться конфигурационный файл домена «XEN», являющийся, по сути, скриптом на языке «Python». А значит, в конфигурационном файле могут применяться любые конструкции этого языка, делая его гибким и эффективным. Например, можно объединить конфигурационные файлы нескольких машин в один, и для запуска любой из них использовать этот файл, с параметром, указывающим, домен какой машины должен стартовать и т.д. [14]. При стандартном использовании, в частности инициализации переменных, языки программирования по способу применения не отличаются от способа, применяемого в простых конфигурационных файлах, к примеру, в том же файле «INI», распространенном в «MS Windows» [15], а значит, не усложняют код.

Используя «JavaScript», «обработчик» в этом случае будет выступать в роли интерпретатора, производя построчный анализ, обработку и выполнение программы, описанную в конфигурационном файле. Так как код интерпретатора «JavaScript» в виде «обработчика» реализуется самостоятельно и полностью доступен, значит доступна и возможность дополнения, изменения среды, относящейся к языку программирования «JavaScript», согласно которому будет производиться построчный анализ и обработка выполнения программы. Согласно стандарту «POSIX» аргументы командной строки делятся на опции и аргументы (операнды). Аргументы, в свою очередь, делятся на свободные и зависимые (аргументы опций) [6]. Для реализации деления по типу и дальнейшего описания аргументов командной строки введем в среду языка программирования «JavaScript» два дополнительных конструктора для объекта опции и аргумента с именами «Option» и «Argument» соответственно. Конструкторы объектов будут являться частью языка и доступны для использования в конфигурационном файле наравне с другими конструкциями «JavaScript».

Конструктор объекта «Option» имеет свойство «keys», представляющее собой массив, в котором будут храниться строковые идентификаторы коротких и длинных имен, означающих эту опцию. К примеру, программой декларируется стандартная опция вывода справки, длинное имя которой согласно соглашениям «POSIX» будет «--help». Тогда для инициализации этой опции в конфигурационном файле (скрипте) необходимо ввести следующую команду:

```
var opt = new Option();
opt.keys = ["help"];
```

В первой строке производится декларация указателя «opt1» на объект и вызов конструктора. Во второй строке – инициализация массива. Конструктор объекта «Option» принимает элементы массива «keys» в виде аргументов, поэтому предыдущий код можно переписать в одну строку:

```
var opt = new Option("help");
```

Соглашением «POSIX» не дано определение короткого имени опции для вызова справки [16], однако часто в качестве такого имени используется символ «?» или «h». В этом случае команда декларации может иметь следующий вид:

```
var opt = new Option("help", "?", "h");
```

Пусть имя программы будет «program», тогда следующие команды при ее вызове будут интерпретироваться одинаково и после обработки вызывать справку:

```
program --help
program -?
program -h
```

Конструктор объекта «Argument» из основных свойств имеет имя объекта «objectName», и флаг «norequest» при значении «true» – указывающий, что аргумент является не обязательным. Имя объекта – это имя, которое будет выводиться в справке. Конструктор объекта принимает значения этих свойств в виде аргументов. Зависимость аргумента от опции определяется, тем остоятельством, является ли указатель на объект «Argument» свойством опции или нет.

К примеру, декларация независимой, обязательной опции «ARG» имеет следующий вид:

```
var arg = new Argument("ARG", false);
```

Декларация этого же аргумента, но в качестве зависимого, от вышеописанной опции, аргумента «opt», имеет следующий вид:

```
opt.arg = new Argument("ARG", false);
```

Это лишь, неполный пример синтаксиса, дополнительно введенных структур «JavaScript», реализующихся в «обработчике», так как формат данной статьи не позволяет раскрыть его полностью, однако должен хорошо демонстрировать принцип использования. Для декларации использующихся опций и аргументов достаточно создать соответствующие объекты и передать их декларацию в виде конфигурационного файла вместе с командной строкой «обработчику», который выполнит разбор командной строки и выдаст по требованию всю необходимую информацию. Распространенный и очень гибкий синтаксис «JavaScript» делает декларацию опций нативно понятной, а схема «программа – обработчик» делает метод универсальным.

ЛИТЕРАТУРА

1. Большая Энциклопедия Нефти Газа [Электронный ресурс] // Развитие – программирование : сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://www.ngpedia.ru/id360477p1.html. – Загл. с экрана.
2. Роббинс, А. Linux: программирование в примерах : пер. с англ. [Текст]: учеб.-метод. пособие / А. Роббинс. – М. : КУДИЦ-ОБРАЗ, 2005. – 656 с.
3. GNU Operating System [Электронный ресурс] // Parsing Program Options with Argp : сайт. – Электрон. текстов. дан. – [Б. м.], 2015. – Режим доступа: http://www.gnu.org/software/libc/ manual/html_node/Argp. html# Argp. – Загл. с экрана.
4. CIT Forum [Электронный ресурс] // getopts – разбор опций команды: сайт. – Электрон. текстов. дан. – [Б. м.], 2016. – Режим доступа: http://citforum.ru/operating_systems/manpages/GETOPTS.1.shtml. – Загл. с экрана.
5. Python [Электронный ресурс] // optparse – Parser for command line options: сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: https://docs.python.org/dev/library/optparse.html. – Загл. с экрана.
6. Сухоруков – соглашения «Posix» по обработке опций и аргументов командной строки.
7. Хабрахабр [Электронный ресурс] // Администрирование → bash скрипт с поддержкой длинных (gnu-style) опций : сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: https://habrahabr.ru/post/133860. – Загл. с экрана.
8. Python [Электронный ресурс] // Argparse Tutorial : сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: https://docs.python.org/3/howto/argparse.html. – Загл. с экрана.
9. The Offical YAML Web Site [Электронный ресурс] // YAML 1.2 : сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://yaml.org/. – Загл. с экрана.
10. XML.com [Электронный ресурс] // XML : сайт. – Электрон. дан. – [Б. м.], 2014. – Режим доступа: http://www.xml.com/. – Загл. с экрана.
11. JSON [Электронный ресурс] // Introducing JSON : сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://www.json.org/. – Загл. с экрана.
12. JavaScript [Электронный ресурс] // JavaScript: сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://www.javascript.com/. – Загл. с экрана.
13. JSON5 [Электронный ресурс] // JSON5: сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://www.json5.org/. – Загл. с экрана.
14. XGU [Электронный ресурс] // Конфигурационный файл XEN : сайт. – Электрон. дан. – [Б.м.], 2009. – Режим доступа: http://xgu.ru/wiki/Конфигурационный_файл_Xen. – Загл. с экрана.
15. Non GNU [Электронный ресурс] // INI formats: сайт. – Электрон. дан. – [Б. м.], 2016. – Режим доступа: http://www.nongnu.org/chmspec/latest/INI.html. – Загл. с экрана.
16. GNU Operating System [Электронный ресурс] // GNU Coding Standards: сайт. – Электрон. текстов. дан. – [Б. м.], 2015. – Режим доступа: http://www.gnu.org/prep/standards. – Загл. с экрана.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ ГРАФИЧЕСКОГО ВЕБ-ИНТЕРФЕЙСА НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ «ВЕБ-ПРИЛОЖЕНИЕ ПО НЕКОММЕРЧЕСКИМ ПЕРЕВОЗКАМ ПАССАЖИРОВ»

Н.А. БОБКОВ
(Представлено: Д.Ф. ПАСТУХОВ)

Представлен анализ технологий, используемых для реализации интерфейсов веб-приложений. Приводится пример реализации веб-интерфейса одной из страниц веб-приложения. Показаны подходы к проектированию графического веб-интерфейса.

С развитием Интернет-технологий стало появляться все больше веб-приложений, тематика которых совершенно различна.

Веб-приложение – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется преимущественно на сервере, обмен информацией происходит по сети.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент – сервер». Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него [1].

Разработка веб-приложений – это мощный инструмент, который позволяет реализовывать бизнес-идеи и создавать успешные программные продукты. Именно веб-приложения помогают автоматизировать процесс работы с крупными потоками информации (клиентская база, каталог товаров, филиальная сеть, документы и т.д.).

Front-end и back-end – термины в программной инженерии, которые различают согласно принципу разделения ответственности между внешним представлением и внутренней реализацией соответственно. Front-end – это абстракция, которая предоставляет пользовательский интерфейс. Например, в проектировании программного обеспечения Model-View-Controller архитектура обеспечивает front-end и back-end между базой данных, компонентами обработки данных и пользователями [1].

Разработка веб-приложений по системе front – end и back – end подразумевает иерархическое разделение процесса создания ресурса на две части, на разработку пользовательского интерфейса (фронтэнда) и его программно-административной части (бэкэнда).

В настоящее время веб-приложения уже сопоставимы по своим возможностям с классическими приложениями. Но при этом могут быть доступны в любом месте и в любое время на компьютере, планшете или мобильном устройстве и зачастую имеют меньшую совокупную стоимости владения. Эти особенности делают веб-технологии очень привлекательными для решения широкого спектра задач.

Средства решения задачи. Рассмотрим основные средства реализации фронтэнда веб-приложения по некоммерческим перевозкам пассажиров.

Front – end разработка – это работа по созданию публичной части приложения, с которой непосредственно контактирует пользователь, и функционал, который обычно обыгрывается на клиентской стороне (в браузере). Составляющая часть приложения Front – end отвечает за вывод определенной информации пользователю и по факту совершения им каких-либо действий в веб-приложении, интерпретацию ее в вид, понятный программам, относящимся к бэкэнду.

В работе был использован Bootstrap – CSS/HTML фреймворк для создания веб-приложений, который содержит ряд преимуществ, благодаря которым он считается самым популярным среди себе подобных. Другими словами, это набор инструментов для вёрстки.

Рассмотрим основные преимущества Bootstrap:

- скорость работы – благодаря множеству готовых элементов верстка с бутстрапом занимает значительно меньше времени;
- масштабируемость – добавление новых элементов не нарушает общую структуру;
- легкая настраиваемость – редактирование стилей производится путем создания новых css-правил, которые исполняются вместо стандартных;
- большое количество шаблонов;
- огромное сообщество разработчиков;
- широкая сфера применения – Bootstrap используется в создании тем для практически любой CMS (OpenCart, Prestashop, Magento, Joomla, Bitrix, WordPress и любые другие), в том числе для одностраничных приложений [3]. Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

Основные инструменты Bootstrap:

- сетки – заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140 px относится к классу .span2 (.col-md-2 в третьей версии фреймворка), который можно использовать в CSS описании документа;
- шаблоны – фиксированный или резиновый шаблон документа;
- типографика – описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т.д.;
- медиа – представляет некоторое управление изображениями и видео;
- таблицы – средства оформления таблиц, вплоть до добавления функциональности сортировки;
- формы – классы для оформления форм и некоторых событий происходящих с ними;
- навигация – классы оформления для табов, вкладок, страничности, меню и тулбара;
- алерты – оформление диалоговых окон, подсказок и всплывающих окон [1].

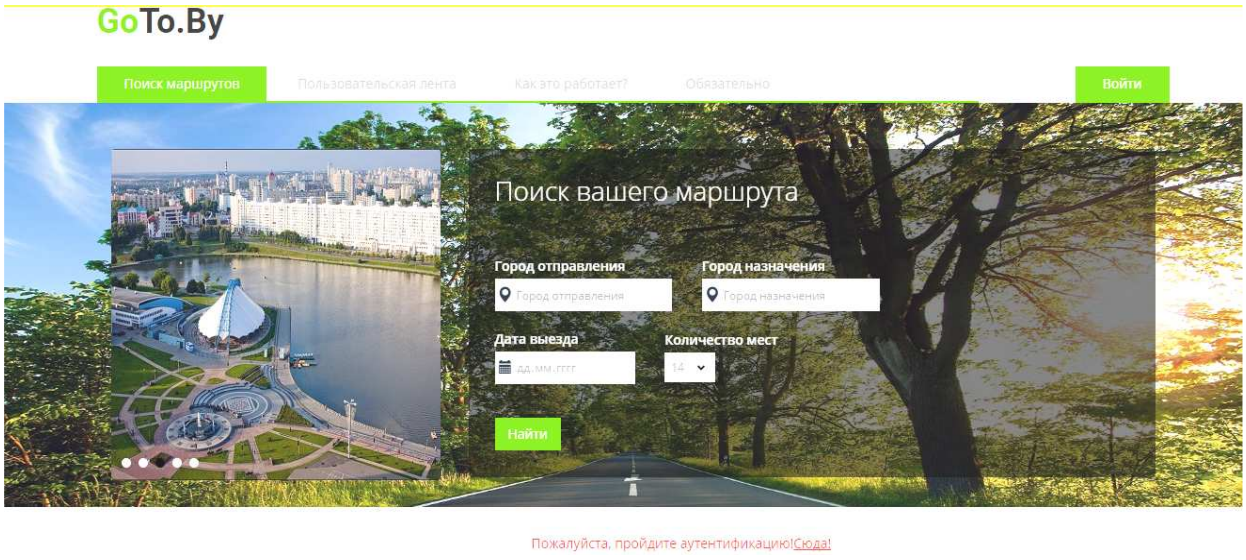
Также при реализации веб-приложения использовалась JSP (Java Server Pages) – технология, позволяющая создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML или XML, и JSP-элементы, которые конструируют динамическое содержимое. Кроме этого могут использоваться библиотеки JSP-тегов, а также EL (Expression Language) для внедрения Java-кода в статичное содержимое JSP-страниц [1].

JSP является платформонезависимой, переносимой и легко расширяемой технологией для разработки веб-приложений. Основная идея JSP очень проста – сама страница представляет из себя шаблон с уже заготовленными HTML-тэгами, между которыми надо вставить нужные данные.

JSP страницы имеют расширение .jsp и размещаются там же, где и обычные web-страницы. Структура таких страниц может состоять из пяти конструкций: HTML, комментарии, скриптовые элементы, директивы и действия. JSP-страница при компиляции преобразуется в сервлет со статическим содержимым, которое направляется в поток вывода, связанный с методом service. Поэтому при первом запросе этот процесс может вызвать небольшую задержку. Скриптовые элементы позволяют указать код на языке Java, который впоследствии станет частью конечного сервлета, директивы дают возможность управлять всей структурой сервлета, а действия служат для задания существующих используемых компонентов, а также для контроля над поведением движка JSP. Для упрощения работы со скриптами имеются заранее определенные переменные, такие как request, response, pageContext, session, out, application, config, page, exception [1; 2].

Код JSP-страницы транслируется в Java-код сервлета с помощью компилятора JSP-страниц Jasper, а затем компилируется в байт-код виртуальной машины java (JVM). Контейнеры сервлетов, способные исполнять JSP-страницы, написаны на языке Java. JSP-страницы загружаются на сервере и управляются из структуры специального Java server packet, который называется Java EE Web Application. Обычно страницы упакованы в файловые архивы .war и .ear.

Проектирование интерфейса. В главном окне приложения можно осуществить поиск маршрутов без просмотра контактной информации водителей, просмотреть последние двадцать фотографий, загруженные пользователями, а также изучить принципы работы системы (рисунок).



Главное окно приложения

При входе на сайт приложения для каждого пользователя создается уникальная сессия, которая сохраняется на всем протяжении нахождения и пользования данным ресурсом. До того момента, пока пользователь не авторизовался, ему будет присвоена анонимная роль, в соответствии с которой он может просматривать только закладки, расположенные на главной странице. После авторизации пользователю добавятся те или иные полномочия, в зависимости от его роли в системе.

При выборе пункта меню «Пользовательская лента» перед пользователем появляется окно, которое содержит последние двадцать добавленных фотографий с описанием местоположения съемки.

На вкладке «Как это работает» пользователь может узнать о работе системы в целом. Данная вкладка содержит следующие пункты:

- для водителей – описание действий водителя, для комфортной поездки;
- для попутчиков – описание действий попутчиков, для удобства при переезде.

Вкладка «Обязательно» несет информационную задачу. Её цель – донести до пользователей, что будет, если нарушить политику приложения.

При выборе пункта меню «Войти» перед пользователем появляется окно, которое можно использовать как при авторизации в веб-приложении, так и для регистрации нового аккаунта. При нажатии на надпись «Забыли пароль» приложение переведет ваш запрос на страничку, содержащую всего одно текстовое поле для ввода адреса электронной почты, к которой привязан пользовательский аккаунт.

Если у пользователя не существует аккаунта в системе, он должен будет нажать на надпись: «Зарегистрируйтесь». В появившемся окне регистрации, в левой части экрана, расположены две кнопки для быстрой регистрации через социальные сети «Вконтакте» и «Facebook». В правой части экрана расположена стандартная регистрация в приложении. При регистрации проверяется оригинальность псевдонима, используемого на сайте при последующей авторизации, и адрес электронной почты.

По окончании любой регистрации перед пользователем появится окно, информирующее его об успешном создании аккаунта в приложении.

После успешной регистрации или авторизации в приложении пользователь перенаправляется в личный кабинет для редактирования информации о себе. Здесь в левой части экрана расположена фотография пользователя. При использовании быстрой регистрации фотография заимствуется из социальной сети, а при использовании стандартной регистрации веб-приложение устанавливает фотографию по умолчанию. Пользователь может изменить ее в любое удобное время. В правой части экрана располагается блок личной информации. При первом входе необходимо заполнить адрес электронной почты и контактный номер телефона для получения доступа к поиску маршрутов. Также при использовании быстрой регистрации рекомендуется изменить пароль, так как система устанавливает пароль по умолчанию «goto.by».

Пользователь может отметить свои предпочтения в поездке. При поиске маршрута по предпочтениям они будут учитываться.

В заключение отметим, что в настоящее время веб-технологии быстро развиваются, они дают разработчикам все больше возможностей, а их производительность растет. В итоге уменьшаются преимущества нативных приложений, а разработка все больше уходит в веб-среду. Веб-приложения прочно вошли в нашу жизнь, и многими из них мы пользуемся, даже не задумываясь, например поиском Google или почтовым сервисом Gmail.

При разработке веб-интерфейсов необходимо брать во внимание не только хороший дизайн, который помогает сконцентрироваться на задачах пользователя, но и учитывать то, что любой интерфейс должен быть интуитивно понятным для пользователя – он должен моментально понимать, какой следующий шаг он может совершить. Веб-интерфейс должен обладать такими качествами, как кроссплатформенность, адаптивность для различных размеров экранов, динамичность и корректная работа в различных браузерах.

Очевидно, что плюсов от использования веб-приложений гораздо больше, чем минусов, а использование различных фреймворков и технологий позволяет значительно сократить время реализации веб-приложений без потери их качества.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia. org/wiki/. – Дата доступа: 23.09.2016.
2. Metanit – сайт о программировании [Электронный ресурс]. – Режим доступа: http://metanit.com. – Дата доступа: 23.09.2016.
3. Hinex – компания, специализирующаяся в веб-разработке и Интернет-рекламе [Электронный ресурс]. – Режим доступа: http://hinex.ru/. – Дата доступа: 23.09.2016.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ АРХИТЕКТУРЫ СЕРВЕРНОЙ ЧАСТИ НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ «ВЕБ-ПРИЛОЖЕНИЕ ПО НЕКОММЕРЧЕСКИМ ПЕРЕВОЗКАМ ПАССАЖИРОВ»

Н.А. БОБКОВ
(Представлено: Д.Ф. ПАСТУХОВ)

Анализируются технологии, используемые при реализации серверной части веб-приложений. Рассмотрен фрейморк Spring FrameWork. Показаны подходы к проектированию архитектуры серверной части на примере веб-приложения «веб-приложение по некоммерческим перевозкам пассажиров».

Стремительное развитие Интернет-технологий послужило толчком к появлению веб-приложений. Разработка веб-приложений – это не просто создание сайтов, это системы электронных платежей и интернет-банкинг, корпоративные порталы, включающие в себя документооборот, почту, календарь и множество других функций.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент – сервер». Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет ее клиенту по сети с использованием протокола HTTP.

Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере. Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль [1].

Серверная часть веб-приложения – это программа или скрипт на сервере, обрабатывающая запросы пользователя (точнее, запросы браузера). База данных – программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент.

Разработка веб-приложений по системе front – end и back – end подразумевает иерархическое разделение процесса создания ресурса на две части: на разработку пользовательского интерфейса (фронтэнда) и его программно-административной части (бэкэнда).

Средства решения задачи

Рассмотрим основные средства реализации бэкэнда веб-приложения по некоммерческим перевозкам пассажиров.

Бэкэнд development – это процесс программирования веб-приложения и наполнения его функционалом. Создание ядра приложения, разработка платформы, наполнение его основным функционалом и создание административной зоны – это и есть бэкэнд разработка.

Бэкэнд производит обработку пользовательской информации, полученной из front-офиса, и возвращает front – end результат в понятной ему форме. Бэкэнд-программирование – это веб программирование, целью которого является реализация серверной стороны сайта, интеграция базы данных и связь ее с пользовательской (front-end) стороной. Разработка бэкэнда сайта также включает настройку и установку на сервер необходимого программного обеспечения. Проще говоря, фронтэнд передает информацию и команды от пользователя в бэкэнд, а тот, в свою очередь, производит их обработку. Или если уж совсем просто, то front – end создается для посетителя сайта, а back – end для его администратора [3].

В работе были использованы сервлеты. Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос – ответ. Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP – специфичные сервлет классы.

Жизненный цикл сервлета состоит из следующих шагов:

1. В случае отсутствия сервлета в контейнере:

- класс сервлета загружается контейнером;
- контейнер создает экземпляр класса сервлета;
- контейнер вызывает метод init(). Этот метод инициализирует сервлет и вызывается, в первую очередь, до того, как сервлет сможет обслуживать запросы. За весь жизненный цикл метод init() вызывается только один раз.

2. Обслуживание клиентского запроса. Каждый запрос обрабатывается в своем отдельном потоке. Контейнер вызывает метод service() для каждого запроса. Этот метод определяет тип пришедшего запроса и распределяет его в соответствующий этому типу метод для обработки запроса. Разработчик сервлета

должен предоставить реализацию для этих методов. Если поступил запрос, метод для которого не реализован, вызывается метод родительского класса и обычно завершается возвращением ошибки инициатору запроса.

3. В случае если контейнеру необходимо удалить сервлет, он вызывает метод destroy(), который снимает сервлет из эксплуатации. Подобно методу init(), этот метод тоже вызывается единожды за весь цикл сервлета [1].

Также использовался контейнер сервлетов Apache Tomcat (в старых версиях – Catalina). Apache Tomcat (в старых версиях – Catalina) – контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces(JSF). Написан на языке Java.

Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования. Он используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish [2].

Особо важной использованной технологией был фреймворк Spring, свободно распространяемый фреймворк, созданный Родом Джонсоном. Главная цель направлена на упрощение разработки приложений на языке Java. В своем устремлении на сложности, связанные с разработкой на языке Java, фреймворк Spring использует четыре ключевые стратегии:

- легковесность и ненасильственность благодаря применению простых Java-объектов (POJO);
- слабое связывание посредством внедрения зависимостей и ориентированности на интерфейсы;
- декларативное программирование через аспекты и общепринятые соглашения;
- уменьшение объема типового кода через аспекты и шаблоны.

Spring Framework имеет довольно широкую функциональность и активно используется при разработке сложных бизнес-приложений. Spring Framework может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании:

- inversion of Control контейнер: конфигурирование компонент приложений и управление жизненным циклом Java объектов;
- фреймворк аспектно-ориентированного программирования: работает с функциональностью, которая не может быть реализована возможностями объектно-ориентированного программирования на Java без каких-либо потерь;
- фреймворк доступа к данным: работает с системами управления реляционными базами данных на Java платформе;
- некоторые другие фреймворки, рассмотрение которых отложим до лучших времен.

С целью помочь в решении всех проблем при создании приложения был разработан веб-фреймворк, входящий в состав Spring. Опираясь на шаблон модель – представление – контроллер (Model – View – Controller, MVC), фреймворк Spring MVC помогает строить веб-приложения, столь гибкие и слабо связанные, как сам фреймворк Spring Framework.

Каждый раз, когда пользователь щелкает на ссылке или отправляет форму в веб-браузере, запрос отправляется на работу. Наш запрос работает курьером, перенося информацию из одного места в другое. С момента, когда он покинет браузер, и до момента, когда вернется ответ, запрос сделает несколько остановок, каждый раз сбрасывая часть информации и подбирая что-то взамен [1; 2].

Первой остановкой на пути запроса является DispatcherServlet. Как и большинство веб-фреймворков на языке Java, фреймворк Spring MVC пропускает все входящие запросы через единственный сервлет входного контроллера. Входной контроллер (front controller) является типичным шаблоном проектирования веб-приложений, где единственный сервлет берет на себя ответственность за передачу всех запросов остальным компонентам приложения, выполняющим фактическую их обработку. В Spring MVC входным контроллером является DispatcherServlet.

Задача контроллера Dispatcher Servlet состоит в том, чтобы передать запрос контроллеру Spring MVC. Контроллер – это компонент Spring, обрабатывающий запрос. Но приложение может иметь несколько контроллеров, и входному контроллеру Dispatcher Servlet требуется помощь, чтобы определить, какому контроллеру передать запрос. Поэтому контроллер Dispatcher Servlet консультируется с одним или несколькими механизмами отображения и выясняет, где будет следующая остановка запроса. При принятии решения механизм отображения, в первую очередь, руководствуется адресом URL в запросе.

Как только будет выбран соответствующий контроллер, Dispatcher Servlet отправляет запрос в путь к выбранному контроллеру. Достигнув контроллера, запрос отдаст часть своего груза (информацию, отправленную пользователем) и терпеливо будет ждать, пока контроллер обработает эту информацию. (На самом деле хорошо спроектированный контроллер сам почти не занимается обработкой информации, вместо этого он делегирует ответственность за обработку одному или нескольким служебным объектам.)

В результате работы контроллера часто появляется некоторая информация, которая должна быть передана назад пользователю и отображена в браузере. Эта информация называется моделью. Но отправки обратно необработанной информации недостаточно, перед отправкой ее следует представить в удобном для пользователя формате, обычно в HTML. Для этого информация должна быть передана в одно из представлений, которыми обычно являются JSP.

Последнее, что должен сделать контроллер, – упаковать вместе модель и имя представления для отображения результатов в браузере. Затем он отсылает запрос вместе с моделью и именем представления обратно входному контроллеру Dispatcher Servlet [1].

Методы решения задач. В соответствии с выполняемыми функциями, веб-приложение можно разбить на несколько модулей.

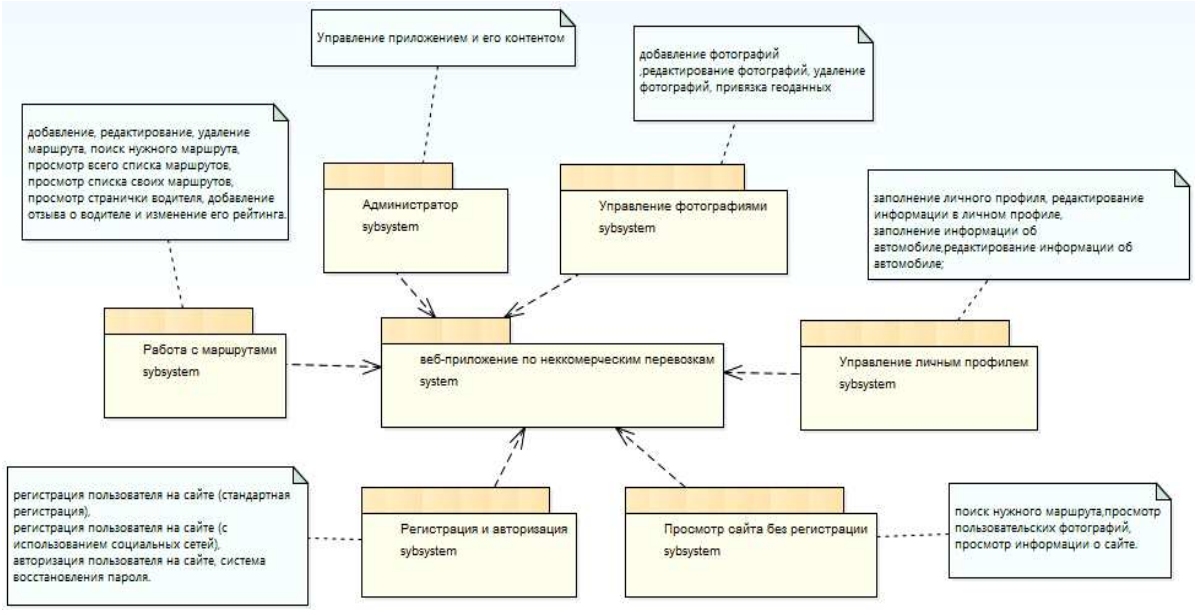
1. Модуль просмотра сайта без регистрации:
   - поиск нужного маршрута;
   - просмотр пользовательских фотографий;
   - просмотр информации о сайте.
2. Модуль регистрации и авторизации:
   - регистрация пользователя на сайте (стандартная регистрация);
   - регистрация пользователя на сайте (с использованием социальных сетей);
   - авторизация пользователя на сайте;
   - система восстановления пароля.
3. Модуль работы с личными данными:
   - заполнение личного профиля;
   - редактирование информации в личном профиле;
   - заполнение информации об автомобиле;
   - редактирование информации об автомобиле;
   - просмотр списка отзывов.
4. Модуль работы с маршрутами:
   - добавление маршрута;
   - редактирование маршрута;
   - удаление маршрута;
   - поиск нужного маршрута;
   - просмотр всего списка маршрутов;
   - просмотр списка своих маршрутов;
   - просмотр странички водителя;
   - добавление отзыва о водителе и изменение его рейтинга.
5. Модуль работы с фотографиями:
   - добавление и загрузка фотографии;
   - редактирование фотографии;
   - удаление фотографии;
   - прикрепление геоданных к фотографии;
   - просмотр всего списка фотографий;
   - просмотр выбранной фотографии.
6. Модуль администратора:
   - все вышеперечисленные функции;
   - удаление отзывов;
   - редактирование отзывов;
   - удаление пользовательских профилей;

Функциональная структура системы представлена на рисунке.

Веб-приложение содержит следующие классы:

- Admin Controller содержит методы, необходимые для работы администратора в приложении;
- Login Controller содержит методы, необходимые для авторизации в приложении;
- Rersonal Controller содержит методы, необходимые для реализации функциональности в личном кабинете пользователя;
- Photo Controller содержит методы, для загрузки, редактирования и удаления фотографий;
- Recalls Controller содержит методы, необходимые для функционирования системы отзывов;
- Registration Controller содержит методы, необходимые для регистрации в приложении;
- Routes Controller содержит методы, необходимые для регистрации в приложении;
- Vk Service Impl содержит методы, необходимые для регистрации в приложении через социальную сеть «Вконтакте»;

- Fb Service Impl содержит методы, необходимые для регистрации в приложении через социальную сеть «Facebook»;
- Fb Service содержит методы и поля, необходимые для получения данных из социальной сети «Facebook»;
- Vk Service содержит методы и поля, необходимые для получения данных из социальной сети «Вконтакте»;
- Fotos Service Impl содержит методы, необходимые для корректного отображения фотографий в приложении;
- Recalls Service Impl содержит методы, необходимые для успешного функционирования системы отзывов и системы рейтинга водителей;
- User Service Impl содержит методы, необходимые для реализации функциональности, отвечающей за деятельность пользователя в приложении;
- Routes Service Impl содержит методы, необходимые для реализации функциональности, отвечающей за корректное отображение информации о маршрутах;
- Valid Email – класс, проверяющий корректность электронной почты пользователя;
- Sec Security Config – класс, содержащий информацию о кодировании пользовательских паролей.



Функциональная структура системы

Заключение. Проектирование и разработка программного средства должны осуществляться с учетом современных технологий, поставленных требований и правил к веб-приложениям. Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создается один раз для произвольно выбранной платформы и на ней разворачивается.

При разработке серверной части веб-приложения необходимо выбирать такие средства и методы, которые позволят в полной мере реализовать необходимый функционал. Использование сервлетов и контейнеров для их хранения, различных фреймворков, таких как Spring и не только, а также разработка веб-приложения по шаблону MVC значительно ускоряет и упрощает процесс разработки серверной части веб-приложения без потери функциональности.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/. – Дата доступа: 23.09.2016.
2. Metanit – сайт о программировании [Электронный ресурс]. – Режим доступа: http://metanit.com. – Дата доступа: 23.09.2016.
3. Hinex – компания, специализирующаяся в веб-разработке и Интернет-рекламе [Электронный ресурс]. – Режим доступа: http://hinex.ru/. – Дата доступа: 23.09.2016.

УДК 371.261; 004.418

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ГЕНЕРАЦИИ ДОКУМЕНТАЦИИ ОБ УСПЕВАЕМОСТИ ШКОЛЬНИКОВ В ТЕКСТОВОМ И ТАБЛИЧНОМ ФОРМАТАХ

А.В. ВОЙТЕХОВИЧ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматривается принцип работы программного обеспечения для генерации и экспорта в MS Word и MS Excel отчетов работников школы по успеваемости школьников. Программный продукт позволяет значительно сократить количество времени и трудозатрат для оформления типовой документации, в которой часто дублируются входные данные.

Существует немалое количество программных продуктов мониторинга успеваемости учащихся для учреждений образования различного уровня, но большинство из них рассчитано на классы/группы большой численности. Такие программы чаще всего требуют профессионального подхода к администрированию. В маленьких (сельских) школах такие специалисты встречаются крайне редко, а количество отчетов различного вида за каждую четверть не намного меньше, чем у больших школ. Дело в том, что отчетность зависит от количества классов, а не от количества учеников в них (от 1 до 10 в маленьких школах и от 20 до 30 – в больших) [1].

Для сокращения трудозатрат по работе с журналами и оформлению итоговой отчетности за текущую четверть (или учебный год) разработана многопользовательская система «Мониторинг успеваемости школьников», систематизирующая и значительно упрощающая данный процесс.

Программный продукт состоит из двух частей:

- серверная: база данных;
- клиентская: настольное оконное приложение.

Взаимодействие с системой мониторинга происходит через клиентское приложение, в котором реализован простой и удобный интерфейс пользователя. Для начала работы необходимо пройти авторизацию, пользователь указывает свои учетные данные (логин и пароль).

После прохождения аутентификации пользователь получает доступ к системе, а на экране отображается главное окно приложения, представленное на рисунке 1.

Рисунок 1. – Скриншот главного окна приложения

В раздел работы с данными может зайти только администрация школы – директор и завучи. При выборе пункта работы с данными на экран будет выведено окно, изображенное на рисунке 2.

Для работы с данными необходимо перейти на нужную вкладку приложения, затем либо нажать на кнопку добавления записи, либо выбрать запись из списка и нажать или на кнопку удаления, или на кнопку редактирования записей. Далее сформируется окно с пустыми ячейками, либо с данными выбранной записи. После обработки данных необходимо нажать на кнопку «Сохранить» (или «Удалить» при удалении записи) и на экран выведется информационное сообщение о том, что операция прошла успешно, текущее окно закроется и снова отобразится окно работы с данными.

Для ввода оценок учитель должен перейти в пункт меню «Ввод оценок» и выбрать, какие именно оценки будут введены. Стоит заметить, что оценки за экзамены могут ввести лишь представители администрации, а оценки за поведение – только классные руководители.

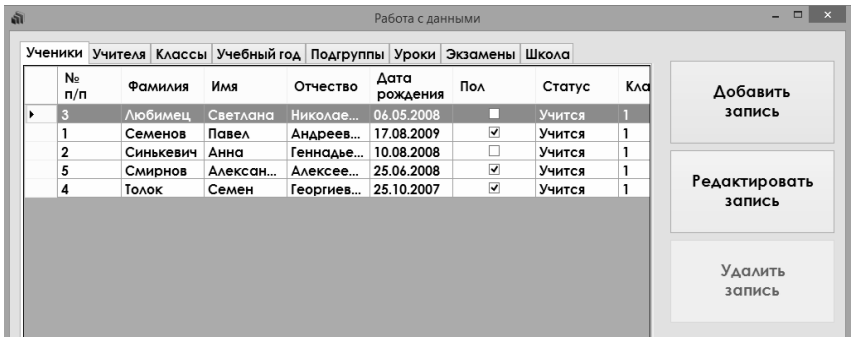


Рисунок 2. – Скриншот окна работы с данными

При выборе пункта меню «Оценки по предметам» появится окно выставления оценок. В нем необходимо выбрать учебный год, класс или подгруппу и предмет, по которому будут выставляться оценки. Затем нужно нажать на кнопку «Выставить оценки» и выбрать ученика. Для сохранения выставленных оценок нужно нажать на кнопку «Сохранить», а для возвращения значений по умолчанию – кнопку «Отмена».

После сохранения данных на экран выводится информационное сообщение. Для выхода из режима выставления оценок по данному предмету нужно нажать кнопку «Закончить выставление оценок» и можно будет выбрать другой урок, либо кнопку «Назад», после чего данное окно закроется, а на экране вновь появится окно меню выставления оценок. Окно ввода оценок по предметам представлено на рисунке 3.

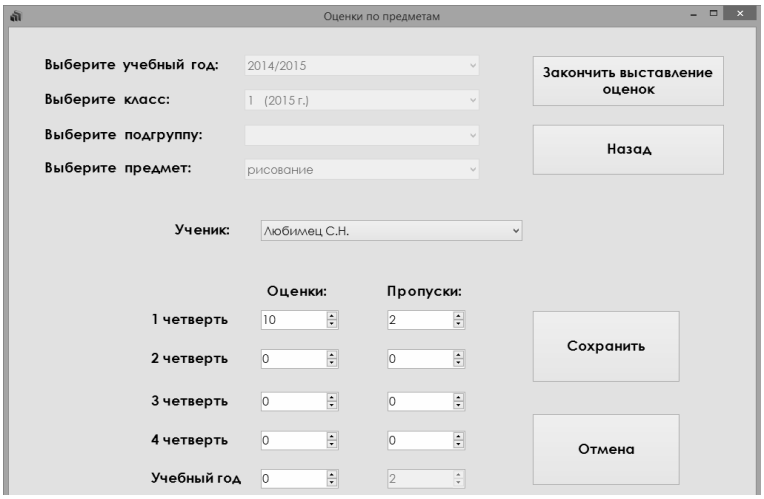


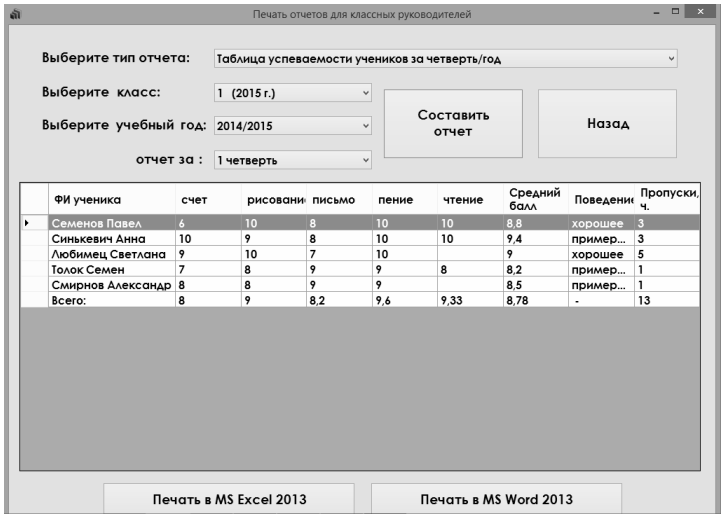
Рисунок 3. – Скриншот окна ввода оценок по предметам

При выборе пункта меню «Оценки за поведение» появится соответствующее окно. В нем необходимо выбрать учебный год, класс и ученика. За поведение можно выставить только одну из следующих оценок: плохое, неудовлетворительное, удовлетворительное, хорошее, примерное.

Выставление оценок за экзамены происходит аналогичным образом, только выбираются учебный год, экзамен и ученик, затем выставляется оценка и нажимается кнопка «Сохранить».

Для генерации и печати итоговой документации необходимо в главном меню выбрать пункт «Печать отчетов». Отчеты разделены на три категории: отчеты учителей; отчеты классных руководителей; отчеты администрации. Это позволяет, во-первых, ограничить доступ неуполномоченных учителей к информации, доступ к которой им необязателен и нежелателен, во-вторых, систематизировать работу с документацией для классных руководителей и администрации, так как им необходимо обрабатывать гораздо больше разного типа документов.

После перехода в нужный раздел необходимо выбрать тип отчета, указать все необходимые данные и нажать кнопку «Составить отчет». Сгенерированный отчет представлен на рисунке 4.

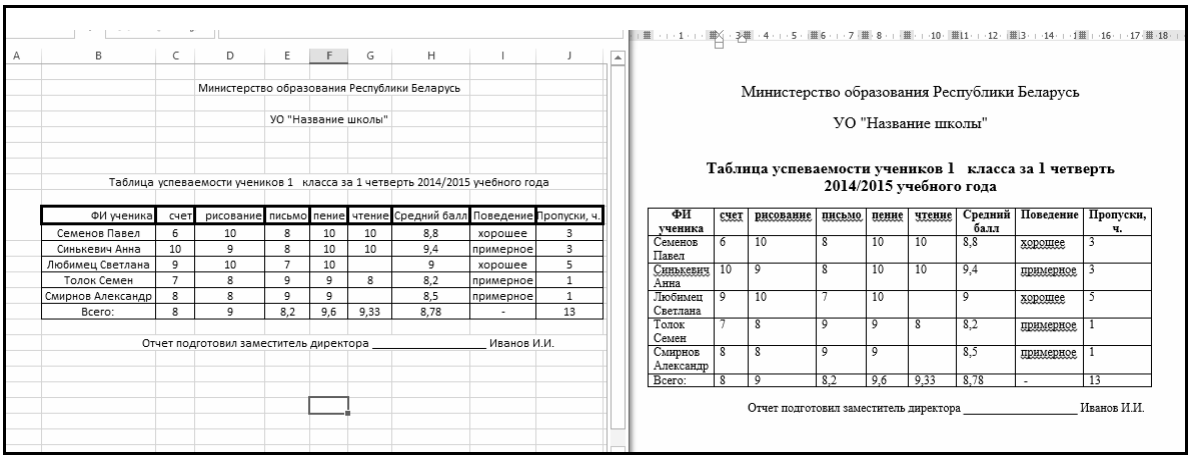


ФИ ученика	счет	рисовани	письмо	пение	чтение	Средний балл	Поведени	Пропуски, ч.
Семенов Павел	6	10	8	10	10	8,8	хорошее	3
Синькевич Анна	10	9	8	10	10	9,4	пример...	3
Любимец Светлана	9	10	7	10		9	хорошее	5
Толок Семен	7	8	9	9	8	8,2	пример...	1
Смирнов Александр	8	8	9	9		8,5	пример...	1
Всего:	8	9	8,2	9,6	9,33	8,78	-	13

Рисунок 4. – Скриншот окна генерации отчетов классных руководителей

После этого будут доступны кнопки импорта данных из сгенерированной таблицы в MS Excel и MS Word. После нажатия на эти кнопки происходит выгрузка данных в документ соответствующего типа [2]. К этим данным добавляются заглавия и реквизиты составителя.

Пример выгрузки данных в MS Excel 2013 и MS Word 2013 представлены на рисунках 4 и 5.



Министерство образования Республики Беларусь

УО "Название школы"

Таблица успеваемости учеников 1 класса за 1 четверть 2014/2015 учебного года

ФИ ученика	счет	рисование	письмо	пение	чтение	Средний балл	Поведение	Пропуски, ч.
Семенов Павел	6	10	8	10	10	8,8	хорошее	3
Синькевич Анна	10	9	8	10	10	9,4	примерное	3
Любимец Светлана	9	10	7	10		9	хорошее	5
Толок Семен	7	8	9	9	8	8,2	примерное	1
Смирнов Александр	8	8	9	9		8,5	примерное	1
Всего:	8	9	8,2	9,6	9,33	8,78	-	13

Отчет подготовил заместитель директора _____ Иванов И.И.

Министерство образования Республики Беларусь

УО "Название школы"

Таблица успеваемости учеников 1 класса за 1 четверть 2014/2015 учебного года

ФИ ученика	счет	рисование	письмо	пение	чтение	Средний балл	Поведение	Пропуски, ч.
Семенов Павел	6	10	8	10	10	8,8	хорошее	3
Синькевич Анна	10	9	8	10	10	9,4	примерное	3
Любимец Светлана	9	10	7	10		9	хорошее	5
Толок Семен	7	8	9	9	8	8,2	примерное	1
Смирнов Александр	8	8	9	9		8,5	примерное	1
Всего:	8	9	8,2	9,6	9,33	8,78	-	13

Отчет подготовил заместитель директора _____ Иванов И.И.

Рисунок 5. – Скриншот результатов выгрузки отчетов в MS Excel и MS Word

Для оценки эффективности использования данного программного продукта был произведен его тестовый запуск в сельской средней школе, в каждом классе которой учится от 1 до 8 человек. Первый этап работы с системой – заполнение базы данных учеников, учителей и расписания – оказался довольно трудоемким и продолжительным, но этот недостаток невелик, так как база будет актуальна в течение всего учебного года, а в августе будет необходимо только добавить новых учеников, удалить выпускников и внести новое расписание. Второй этап (внесение в систему оценок) – оказался гораздо проще и быстрее, чем предполагалось, несмотря на непривычный для учителей формат работы с оценками. Скоростью и качеством генерации отчетов за 4-ю четверть и весь 2015/2016 учебный год все пользователи остались довольны.

В итоге приложение значительно ускорило работу с документами для учителей и администрации за счет исключения дублирования действий по вводу данных в систему, автоматизации расчетов и возможности генерации документов в соответствии со стандартами Министерства образования.

ЛИТЕРАТУРА

1. Справки внутришкольного контроля за успеваемостью учащихся [Электронный ресурс]. – Режим доступа: http://www.vshk.by/uspev.html. – Дата доступа: 08.02.2016.
2. Работа с серверами автоматизации Word и Excel в Visual Studio .Net [Электронный ресурс]. – Режим доступа: http://wladm.narod.ru/C_Sharp/componentbegin.html. – Дата доступа: 10.04.2016.

УДК 004.624, 004.912

ИМПОРТ И ЭКСПОРТ ДАННЫХ ИЗ ПРИЛОЖЕНИЙ, НАПИСАННЫХ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ C#, В ДОКУМЕНТЫ MS WORD И MS EXCEL

А.В. ВОЙТЕХОВИЧ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Исследуется вопрос оптимизации ручного труда при работе с данными, хранящимися в электронном виде. Также автор рассматривает методы решения этой проблемы применительно к языку программирования C# и документам MS Word и MS Excel.

Сегодня большая часть документации, которая передается между людьми, отделами и учреждениями, хранится в электронном формате. Это позволяет сократить расходы на печать документов, легко вносить в них поправки, создавать новые документы на основе шаблонных и собирать информацию из многих источников в единый ресурс. При этом не всегда всю информацию удобно хранить только в текстовом или только в табличном формате. При работе с документами периодически возникают проблемы дублирования данных, различных значений для одного и того же понятия в разных копиях документов (например, указаны разные даты для одного события или различная стоимость в двух накладных по одному товару), проблемы некорректного отображения данных и тому подобные. К тому же при работе с числовыми данными в текстовых документах часто приходится производить вычисления вручную, а при обнаружении ошибки повторять весь процесс заново.

Для решения проблем обработки большого объема данных разработано бессчетное множество программ, которые обеспечивают ввод, хранение, обработку и вывод данных, связанных с указанной предметной областью. Все они работают с прикрепленными к ним базами данных, которые обеспечивают корректное хранение данных без излишнего их дублирования. Такие системы можно разделить на два типа: справочные системы; автоматизированные информационные системы [1].

Справочные системы используются для получения справочной информации. Администратор или ответственное лицо вносят данные в систему вручную, а пользователи с помощью модуля поиска находят и просматривают нужную им в данный момент информации. Примером таких систем являются справочные терминалы на железнодорожных и автобусных вокзалах, электронные каталоги, справочные терминалы в книжных магазинах.

Автоматизированные информационные системы, в отличие от справочных, могут не только обрабатывать и хранить информацию, но и получать в качестве входных данных документы, оформленные в установленном формате, а также генерировать отчеты, выгружать их в документы различных форматов и печатать. Некоторые информационные системы кроме этих функций могут выполнять и иные, более специфические для своих предметных областей, другие же поддерживают только часть из вышеперечисленных функций.

Для того чтобы обеспечить передачу данных между автоматизированными информационными системами и внешними документами, необходимо изучить принципы импорта и экспорта данных в различных языках программирования. В случае импорта или экспорта в файлы формата txt, xml и тому подобных достаточно функций чтения и записи в файл, открытый в текстовом режиме. Разница состоит только в представлении данных – в текстовом файле данные будут структурированы с помощью пробелов, символов переноса строки или специальных, заранее условленных, символов и их последовательностей, а в xml-подобных файлах – с помощью тегов.

В случае работы с doc- и xls-файлами дела обстоят хуже, так как эти файлы помимо полезных данных хранят большой объем информации о форматировании текста и разметке документов. Для того чтобы стороннее приложение получило доступ к данным в таком формате, лучше всего использовать специализированные библиотеки. Библиотеки для работы с файлами MS Word и MS Excel для разных языков программирования работают по единому принципу. Файл открывается в фоновом режиме для чтения или записи соответствующим приложением (Word или Excel) и передача данных производится не напрямую между документом и пользовательским приложением, а по цепочке: пользовательское приложение -> функции из специализированной библиотеки -> приложение для работы с документами данного типа -> документ и обратно.

Рассмотрим функции специализированных библиотек импорта и экспорта данных для приложений, написанных на языке программирования C# Microsoft.Office.Interop.Excel и Microsoft.Office.Interop.Word. Стоит отметить, что эти библиотеки не входят в стандартный пакет библиотек и их необходимо скачать и подключить к проекту вручную. При этом нужно учесть некоторые их особенности. В связи с тем, что сейчас активно используются минимум три версии пакета Microsoft Office (2007, 2010, 2013), а в некоторых учреждениях до сих пор используется пакет 2003 года, нужно скачать библиотеку, предназначенную

для конкретной версии пакета. Это связано с тем, что в новых версиях постоянно добавлялись новые макросы, элементы разметки документа, а также постепенно менялся формат представления данных. Тем не менее часть функций осталась неизменной. На основе этого была организована обратная совместимость документов старых версий с новыми пакетами. Таким образом, библиотека, предназначенная для пакета 2010, будет корректно работать на компьютере, на котором установлен MS Office 2010, а если будет установлен пакет более ранней версий, она может некорректно отображать часть информации и выбрасывать ошибки в случае вызова функций, отсутствующих в приложении. Если же на компьютере будет установлен MS Office 2013, то приложение не сможет связаться с документом, выбрасывает ошибку и завершает свою работу.

Работа с текстовыми и табличными файлами проводится аналогично, поэтому рассмотрим импорт и экспорт данных на примере работы с документом MS Excel [2].

После скачивания библиотеки ее необходимо переместить в папку ресурсов и подключить к проекту. Программист получает возможность использовать все функции из подключенных библиотек, но их вызов сопровождается префиксом Microsoft.Office.Interop.Excel или Microsoft.Office.Interop.Word. Для того чтобы укоротить запись функций и получить возможность подключить другую версию специализированных библиотек, не меняя исходного кода всего приложения, перед описанием главного класса приложения можно прописать глобальный алиас:

using Exc = Microsoft.Office.Interop.Excel.

Алиас – это специфический синоним названию библиотеки, прописанный программистом вручную, который используется на равных правах со стандартным названием библиотеки. После добавления вышеописанного алиаса функции работы с документами MS Excel можно вызывать с коротким префиксом – Exc.

Приложение MS Excel, с которым работает пользовательское приложение, работает на основе древовидной структуры, корень которой – объект Application. Этот объект содержит одну или более книг, ссылки на которые хранятся в свойстве Workbooks. Каждая книга является объектом Workbook, которая, в свою очередь, содержит одну или более страниц, ссылки на которые хранятся в свойстве Worksheets, и может содержать диаграммы – свойство Charts. Страницы содержат объекты ячейки или группы ячеек, ссылки на которые доступны через объект Range. Еще ниже в иерархии строки и столбцы. Объект Chart содержит ссылки на серии линий, легенды и тому подобные элементы.

В первую очередь, при работе с документами необходимо создать глобальный объект класса Application, затем программист получает возможность открыть его в режиме чтения или записи либо создать новый документ.

Ниже представлен фрагмент программы, которая открывает существующий файл, объединяет ячкейки A1–E1, добавляет в них надпись «Тестовый запуск прошел успешно», сохраняет изменения и закрывает файл. Каждая строка исходного кода пронумерована для удобства объяснения действий, производимых в той или иной строке.

```
1:  Excel.Application excel = new Excel.Application();
2:  excel.Visible = false;
3:
4:  Excel.Workbook fileExcel;
5:  fileExcel = excel.Workbooks.Open("C:/file1.xlsx");
6:
7:  Excel.Sheets excelsheets =  excelapp.Worksheets;
8:  Excel.Worksheet excelworksheet = (Excel.Worksheet)excelsheets.get_Item(1);
9:
10: Excel.Range excelcells=excelworksheet.get_Range("A1",Type.Missing);
11: excelcells=excelcells.get_Offset(0,4);
12: excelcells.Value2 = “Тестовый запуск прошел успешно”;
13:
14: fileExcel.Save();
15: fileExcel.Close();
16: excel.Quit();
```

В первой строке инициализируется глобальный объект, через который программа получит доступ к документу Excel и одновременно запускается приложение MS Excel в фоновом режиме. Во второй строке мы указываем, что фоновое окно, в котором будет открыт файл, не будет отображаться на экране. Если вместо «false» указать «true», то окно будет выведено на экран без связанного с ним документа. В строках 4–5 происходит создание и инициализация переменной, хоторая хранит ссылку на рабочую книгу. Функция excel.Workbooks.Open открывает документ, полный путь к которому указан в круглых скобках. Если файл поврежден, отсутствует или имеет неверное расширение, то приложение выдаст ошибку и автоматически закроется. Такие ошибки необходимо отслеживать, но в рамках простейшего примера в этом необходимости нет.

В переменной excelsheets (строка 7) будут храниться ссылки на все листы, существующие в ранее открытой книге, а в переменную excelworksheet в строке 8 заносится ссылка на первый лист книги с помощью функции get_Item(). Если в качестве параметра указать не единицу, а другое число, будет открыта страница с указанным порядковым номером. При этом стоит предусмотреть проверку на корректность указанного числа, так как оно не может быть нулевым, отрицательным или большим, чем количество листов в открытой книге.

Функция get_Range() возвращает ссылку на ячейку, указанную в первом параметре (строка 10), а функция get_Offset() позволяет выделить диапазон ячеек и объединить их. Диапазон указывается с помощью двух чисел: первое показывает, сдвигается ли граница диапазона вниз относительно нижнего края ячейки и насколько, а второе – сдвигается ли граница диапазона вправо относительно правого края ячейки и насколько. В случае указания отрицательных чисел сдвиг происходит в обратную сторону. В строке 11 представленного исходного кода в переменную excelcells вместо ссылки на ячейку A1 заносится ссылка на новообразованный диапазон, который состоит из объединенных ячеек A1, B1, C1, D1 и E1. Значение ячейки устанавливается с помощью свойства Value2, как показано в строке 12. Далее остается только сохранить внесенные изменения, закрыть документ и разорвать связь с приложением MS Excel с помощью функции Quit(). Данный процесс описан в строках 14–16 исходного кода. Считывание данных происходит аналогичным образом. Результат выполнения данной программы представлен на рисунке 1.

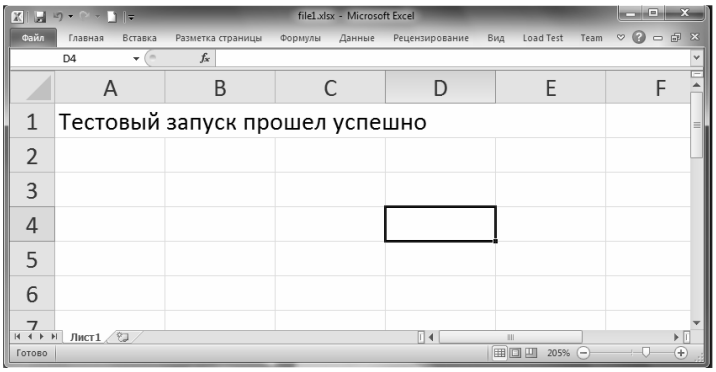


Рисунок 1. – Результат работы программы

При работе с текстовыми документами происходит тот же процесс. Разница состоит лишь в иерархии объектов, вложенных в корневой объект Applications. Эта иерархия представлена на рисунке 2.

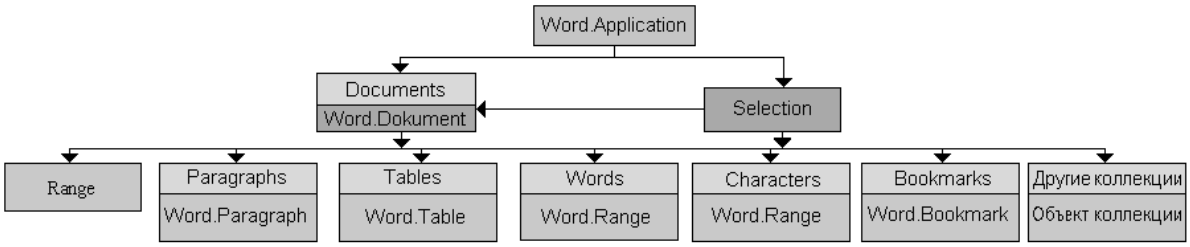


Рисунок 2. – Иерархия основных объектов Word, необходимых при разработке приложения

Таким образом, можно сказать, что изучение импорта и экспорта данных из приложения, написанного на языке программирования C#, в текстовые и табличные документы сводится к изучению иерархии объектов в корневом элементе Applications для Word и Excel, а также изучению основных принципов и функций работы с ними. Несмотря на то, что при первом рассмотрении тема кажется слишком обширной и трудной, после изучения базового материала и нескольких попыток применить полученные знания на практике выясняется, что работа с внешними документами сводится к механическому переписыванию шаблонных наборов функций, а самая большая проблема при работе с ними – это опасность запутаться в очередности и позиции выводимых и считываемых данных.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия: Информационная система [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Информационная_система. – Дата доступа: 10.04.2016.
2. Работа с серверами автоматизации Word и Excel в Visual Studio .Net [Электронный ресурс]. – Режим доступа: http://wladm.narod.ru/C_Sharp/componentbegin.html. – Дата оступа: 10.06.2016.

УДК 004.42+371.263

АДАПТИВНОЕ ТЕСТИРОВАНИЕ И АЛГОРИТМЫ ЕГО ПРОВЕДЕНИЯ АНАЛИЗ АКТУАЛЬНОСТИ ЗАДАЧИ РАЗРАБОТКИ СИСТЕМЫ АДАПТИВНОГО ТЕСТИРОВАНИЯ

С.С. ВОЙТЕХОВИЧ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматриваются адаптивное тестирование и алгоритмы его проведения. Показана актуальность задачи разработки системы адаптивного тестирования. Дано определение понятия адаптивного тестирования. Описаны некоторые алгоритмы его проведения, рассмотрены достоинства и недостатки каждого из них.

Адаптивное тестирование – разновидность тестирования, при котором порядок предъявления заданий (или трудность заданий) зависит от ответов испытуемого на предыдущие задания [1]. Говоря более простым языком, адаптивное тестирование – это процесс проверки знаний, который пытается эмитировать работу реального человека, проводящего устный опрос испытуемого. При устной проверке знаний реального испытуемого преподаватель чаще всего старается оценивать уровень знаний испытуемого и не задавать вопросов, на которые испытуемый заведомо не сможет ответить. Подобный подход зачастую является самым быстрым и точным при проверке знаний, но имеет один большой недостаток – невозможность оценки знаний сразу у большого количества испытуемых. Данный недостаток должны были решить разные виды письменных заданий. Но недостатком подобных вариантов стало время, необходимое на проведение контроля и проверку ответов учащихся. Эту проблему по большей части решают разного рода тестовые задания, которые, как правило, требуют небольшого времени на ответ. Но и данное решение не лишено недостатков. Во-первых, скорость написание теста значительно уменьшается при большом количестве заданий. Это также способствует рассеиванию внимания тестируемого, что может привести к случайным ошибкам. Во-вторых, при большом количестве заданий есть вероятность случайного получения положительной оценки посредством угадывания правильного ответа. В-третьих, все испытуемые имеют доступ ко всем заданиям, что позволяет в процессе обсуждения заданий определить правильные ответы перебором. Данная проблема приводит к быстрой потере актуальности тестов при их многократном использовании.

По сути, адаптивное тестирование решает многие из представленных проблем. Адаптивные алгоритмы тестирования позволяют снизить количество показываемых испытуемому заданий, что ускоряет процесс тестирования и способствует более длительной актуальности набора тестовых заданий. Все еще остается проблема с возможным угадыванием ответов, но за счет того, что испытуемому будет предоставлено меньше заданий, вероятность угадывания уменьшается. Более того, грамотная установка критериев анализа уровня сложности заданий с большой вероятностью приведет завершение теста путем угадывания к неудовлетворительному результату.

У адаптивного тестирования также есть недостатки. Из-за сложности алгоритмов применение адаптивных тестов практически невозможно без использования компьютера. Из этого следует еще один недостаток, который свойственен любым компьютерным системам тестирования – сложность анализа вольного ответа испытуемого. То есть при составлении теста необходимо учесть возможные варианты ответа без возможности запросить у испытуемого развернутый ответ. Но если имеется достаточное количество компьютеров для выполнения оценки знаний, и необходима проверка большого количества испытуемых, данные недостатки не являются критическими.

Самым большим недостатком адаптивного тестирования является сложность создания базы тестовых заданий. Сложность заключается в том, что, во-первых, база тестовых заданий должна быть достаточно большой, так как при малом количестве тестовых заданий оценка знаний становится неэффективной, во-вторых, каждому вопросу должно быть присвоено значение сложности. Таким образом, использование алгоритмов адаптивного тестирования не окупит затрат времени на его подготовку, если нужен одноразовый тест.

Первый недостаток можно исключить в процессе разработки системы для проведения адаптивного тестирования, если дать возможность многоразового использования каждого вопроса независимо от теста, в рамках которого он был создан. Такой подход позволит создавать общую базу заданий и использовать каждый из них по необходимости. Таким образом, даже создание одноразового теста становится более простой задачей.

Для обхода второго недостатка можно использовать методы машинного обучения. То есть если позволить нескольким испытуемым, уровень подготовленности которых заведомо известен, пройти все задания тестовой базы, то на основании их результатов автоматически можно оценить каждое из заданий. Далее, в ходе использования базы тестовых заданий можно продолжать анализировать ответ по каждому из них, на основании чего корректировать их сложность.

Как видно, большинство недостатков адаптивных алгоритмов тестирования можно обойти, но разработка подобной системы является достаточно сложной задачей.

Существует множество алгоритмов проведения адаптивного тестирования. При этом нет четко выраженных стандартов в решении данного вопроса. В результате изучения данной темы были выделены следующие адаптивные алгоритмы:

- *алгоритм процентного соотношения*. Данный алгоритм применим в том случае, когда в один тест собраны вопросы из разных тем и нет возможности их сгруппировать по темам, или же в том случае, когда все вопросы относятся к одной теме. В рамках данного алгоритма составитель вопросов должен разбить вопросы по уровням сложности и настроить параметры тестирования. Параметры тестирования определяют условия перехода тестируемого на следующие уровни сложности. Например, при существовании 5-ти уровней сложности и начале теста с 3-го уровня, составитель тестов определяет, по скольким последним вопросам будет проходить решение о переходе на другой уровень и при каком проценте правильных или не правильных ответов на вопросы будет происходить изменения уровня сложности. Допустим, что при оценке четырех последних выполненных заданий и параметрах 25% нижний предела и 75% верхний предел будет определено, сколько процентов заданий было выполнено верно, и в зависимости от значения решается, на какой уровень переходит тестируемый, т.е. если процент меньше нижнего предела, то происходит понижение уровня, процент между значениями нижнего и верхнего предела – уровень не изменяется, больше верхнего предела – происходит повышение уровня. Признаками завершения теста могут быть лимит времени, длительное пребывание тестируемого на одном уровне сложности или исчерпание количества вопросов текущего уровня;

- *алгоритм «мягкого» тестирования*. Применим, если вопросы теста можно точно разбить на темы. Алгоритм заключается в следующем: все темы собираются в общую группу и из каждой темы выбирается по вопросу некоторого уровня сложности. Если тестируемый ошибается при ответе на вопрос некоторой темы, данная тема исключается из общей группы и из нее вопрос на следующем цикле не отбирается. При исключении всех тем из группы, в группу снова вносятся все темы и так до выполнения условия завершения тестирования. Условиями завершения тестирования могут быть лимит времени, критерий количества вопросов или критерий количества циклов выборки вопросов. В рамках тем уровень сложности вопросов выбирается по алгоритму процентного соотношения. У данного алгоритма есть недостатки, в частности тестируемый с низким уровнем знаний может, хорошо зная одну тему, закончить тест с хорошим результатом;

- *алгоритм «жесткого» тестирования*. Идентичен алгоритму «мягкого» тестирования, за исключением того, что из группы тем исключаются те вопросы, на который тестируемый дает правильный ответ. Основной недостаток данного способа очевиден – тестируемый с хорошими знаниями может окончить тест с неудовлетворительным результатом, если плохо будет знать хотя бы одну из тем;

- *комбинированный алгоритм*. Объединяет в себе алгоритмы «мягкого» и «жесткого» тестирования. Один из алгоритмов выбирается в процессе тестирования, т.е. если тестируемый показывает хорошие знания, то выбирается алгоритм «жесткого» тестирования, иначе – алгоритм «мягкого» тестирования. Следовательно, в зависимости от выбранного в процессе тестирования алгоритма, тестируемые оцениваются по-разному. Данный алгоритм решает проблемы двух предыдущих, но требует дополнительных данных об уровне знаний испытуемого, достаточных для принятия решения о выборе алгоритма.

Как видно из приведенных выше алгоритмов, для использования каждого из них необходимо оперировать большим количеством переменных для каждой сессии тестирования. Именно этот факт делает использование алгоритмов адаптивного тестирования практически невозможным без использования компьютера. Более того, этот же фактор делает системы адаптивного тестирования сложными для реализации. Это привело к малому распространению подобных систем. Из систем, которые предоставляют возможность использования адаптивных алгоритмов, можно выделить систему управления курсами «Moodle» [2] и система АСТ-ТЕСТ [3]. При этом только система АСТ-ТЕСТ направлена на проведение именно адаптивного тестирования. Система «Moodle» дает возможность самых минимальных настроек, которые сводятся к открытию следующего теста при успешном прохождении предыдущего, либо открытию еще одного теста в ином случае.

Таким образом, разработка систем адаптивного тестирования на сегодняшний день является актуальной задачей. Этому способствует малое распространение подобных систем и повсеместное использование компьютерной техники для улучшения учебного процесса.

ЛИТЕРАТУРА

1. Словарь терминов [Электронный ресурс]. – Режим доступа: http://www.profcareer.ru/lib/dicty.php. – Дата доступа: 25.09.2016.
2. Moodle [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Moodle. – Дата доступа: 25.09.2016.
3. Сайт независимого центра тестирования качества обучения [Электронный ресурс]. – Режим доступа http://www.ast-centre.ru. – Дата доступа: 26.09.2016.

УДК 004.42

РЕАЛИЗАЦИЯ ВЕБ-СИСТЕМЫ ДЛЯ ПРОВЕДЕНИЯ АДАПТИВНОГО ТЕСТИРОВАНИЯ

С.С. ВОЙТЕХОВИЧ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Рассматривается один из вариантов реализации системы адаптивного тестирования в качестве веб-ресурса. Анализируютя основные проблемы, возникающие при реализации подобной системы как веб-ресурса. Как простейший вариант реализации системы адаптивного тестирования разработано приложение, которое выполняет все поставленные цели.

Адаптивное тестирование – разновидность тестирования, при которой порядок предъявления заданий (или трудность заданий) зависит от ответов испытуемого на предыдущие задания [1]. По сути, адаптивное тестирование является развитием идей простого тестирования и исправляет некоторые ее недостатки, такие как продолжительность тестирования при большом количестве заданий и быстрая потеря актуальности тестовых заданий при многократном использовании. Алгоритмы адаптивного тестирования, при правильной реализации могут сократить количество предоставляемых заданий из тестовой базы без потери точности оценивания. Платой за подобную эффективность стала сложность алгоритмов адаптивного тестирования и невозможность их использования без применения компьютера. Более того, усложняется процесс построения базы тестовых заданий, так как необходимо вводить классификацию сложности для заданий. Также адаптивное тестирование становится неэффективным при малом количестве заданий. Таким образом, создание адаптивного теста для одноразовой проверки знаний является не окупаемой по времени задачей.

Из сказанного выше следует, что система адаптивного тестирования должна обеспечивать многоразовое использование заданий и тестов, а таже задавать критерии сложности для заданий. Более того, эффективность системы можно увеличить, если дать возможность одновременного проведения тестирования группой испытуемых. Следовательно, реализация системы адаптивного тестирования как настольного приложения без клиент-серверной архитектуры является задачей, не имеющей смысла, учитывая уровень развития современных сетевых технологий.

Следовательно, при выборе способа реализации серверного приложения встает выбор – реализовывать настольное клиент-серверное приложение, либо реализовать веб-приложение. Каждый из подходов имеет свои достоинства и недостатки. В частности, настольное клиент-серверное приложение позволяет более жестко контролировать условия проведения тестирования, но требует установку приложений системы как на серверную, так и клиентскую часть. Это требует дополнительных усилий от конечного пользователя и усложняет портирование приложения на разные платформы. Касательно веб-приложения, можно наблюдать противоположную ситуацию – клиентский интерфейс приложения можно запустить на любом устройстве, которое имеет веб-браузер, а установка требуется только на серверной части системы. При этом сложно контролировать условие проведения теста посредством веб-интерфейса. Более того, в веб-интерфейсе сложнее реализовать некоторые интерактивные элементы, которые в интерфейсе настольного приложения реализуются достаточно легко. Но отсутствие необходимости полного портирования приложения на другие платформы (в том числе и мобильные) перекрывает многие недостатки веб-архитектуры. Данный факт стал решающим при выборе веб-архитектуры для разрабатываемой системы.

Для разработки системы было решено использовать язык программирования Java в связке с фреймворком Spring. Данная связка повсеместно используется для построения корпоративных веб-систем и хорошо себя зарекомендовала. В качестве системы управления базой данных (СУБД) была выбрана MySQL. Это бесплатная СУБД с достаточным функционалом для решения поставленной задачи. Более того, она достаточно проста в управлении, что облегчает процесс установки серверной части приложения. В качестве сервера веб-приложений была использована Apache Tomcat как наиболее популярный вариант. Более того, Apache Tomcat имеет версии практически для всех современных операционных систем.

Разрабатываемую систему было решено разделить на две части: модуль организации тестов и модуль тестирования. В первую часть включены все средства взаимодействия составителя тестов и испытуемых, в том числе модули регистрации составителей тестов и испытуемых, модуль организации курсов для группирования тестов и модуль доступа к тестам. Вторая часть системы включает модуль создание тестовых заданий и модуль их отображения. Разработанный модуль создания тестовых заданий поддерживает создание следующих типов заданий:

- одиночный ответ;
- множественный ответ;
- точный ответ.

Малое количество типов тестовых заданий компенсируется расширяемой архитектурой модуля создания заданий. Текст вопросов было решено хранить в HTML формате, что позволило менять форматирование текста вопроса. Для поддержки подобной возможности была использована библиотека nicEdit, написанная на JavaScript. Данная библиотека позволила добавить в веб-интерфейс полнофункциональный текстовый редактор (рис. 1).

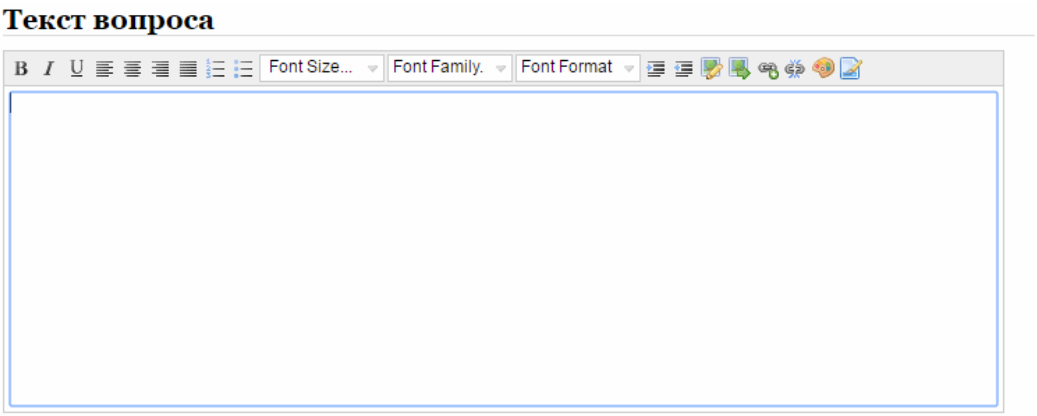


Рисунок 1. – Текстовый редактор из библиотеки nicEdit

При создании теста есть возможность задать перечень настроек, характеризующих работу адаптивного алгоритма. В разработанной системе реализуется алгоритм процентного соотношения, в рамках которого анализируется некоторое количество последних ответов испытуемого. В зависимости от процента правильных ответов принимается решение: повышать сложность заданий, понижать или оставлять неизменной. Значения, задающие количество анализируемых ответов, граничные значения для изменения уровня, а также ограничение времени на тест, задается на форме создания теста (рис. 2).

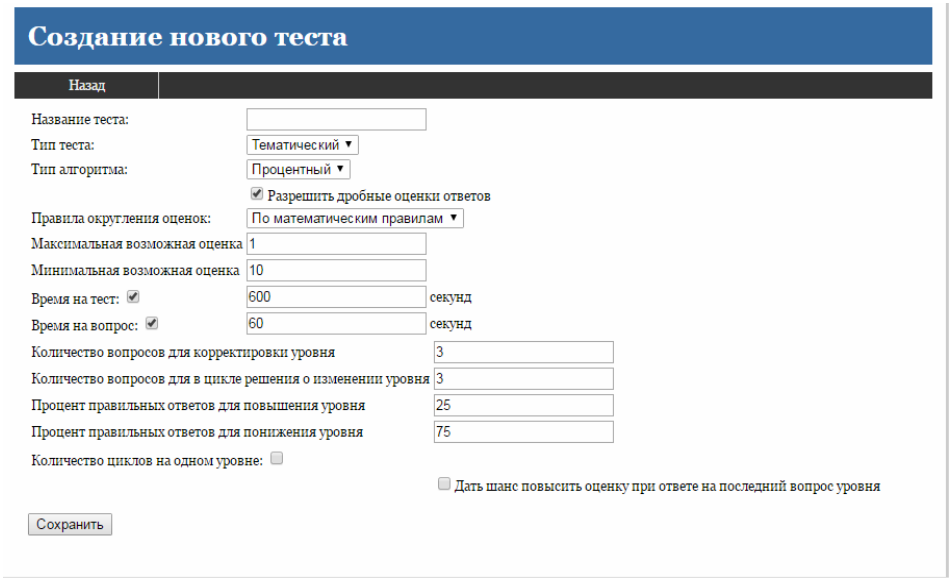


Рисунок 2. – Форма создания нового адаптивного теста

Полученная в итоге система не является самым оптимальным решением для проведения адаптивного тестирования, но она допускает расширение. Работа по дальнейшему улучшению может быть направлена на увеличение количества доступных адаптивных алгоритмов, внедрение методов машинного обучения для регулирования сложности вопросов и увеличение количества типов тестовых заданий. Но как простейший вариант реализации системы адаптивного тестирования разработанное приложение выполняет все поставленные цели.

ЛИТЕРАТУРА

1. Словарь терминов [Электронный ресурс]. – Режим доступа http://www.profcareer.ru/lib/dicty.php. – Дата доступа: 25.09.2016.

УДК 004.021

СОВРЕМЕННЫЕ СТАНДАРТЫ АЛГОРИТМОВ ХЭШИРОВАНИЯ

К.С. ГОЛОВАН
(Представлено: Е.Р. СУХАРЕВ)

Рассматриваются принципы построения хэш-функций. Представлена конструкция Меркла – Дамгарда и её модификации Wide Pipe, Fast Wide Pipe, HAIFA. Описывается последняя разработка в области построения хэш-функций – криптографическая губка. Приведены примеры современных стандартов алгоритмов хэширования и их особенности.

Функция хэширования – это функция, которая принимает на вход строку битов (байтов) произвольной длины и выдаёт результат фиксированной длины. Криптографические функции хэширования играю важную роль в современной криптографии, особенно в области аутентификации сообщений, контроля целостности, цифровой подписи и хранения ключей.

Рассмотрим основные принципы построения хэш-функций.

Структура Меркла – Дамгарда описана в докторской диссертации Ральфа Меркла. Ральф Меркл и Иван Дамгор независимо показали, что если функция сжатия устойчива к коллизиям, то и хеш-функция будет также устойчива. Чтобы доказать устойчивость структуры, Меркл и Дамгор предложили дополнить сообщение блоком, который кодирует длину первоначального сообщения. Это называется упрочнение Меркла – Дамгарда.

Суть конструкции заключается в итеративном процессе последовательных преобразований, когда на вход каждой итерации поступает блок исходного текста и выход предыдущей итерации. Входное сообщение x разбивается на t одинаковые по длине блоки, длина блока x_i равна r. Длина блока r должна соответствовать длине входного блока функции сжатия f. Последний блок дополняется нулями, если его длина не равна r. К этим блокам добавляется дополнительный (t + 1)-й блок, содержащий информацию о длине сообщения x. На каждой итерации к блоку x_i применяется функция сжатия f, выходным значением которой является промежуточное значение H_i. Это значение используется в следующей итерации, поэтому предполагается использование начального вектора H_0. После обработки всех блоков к последнему промежуточному значению применяется функция финализации g. Она используется для уменьшения размера хэша или для лучшего смешивания битов. Структура Меркла – Дамгарда представлена на рисунке 1.

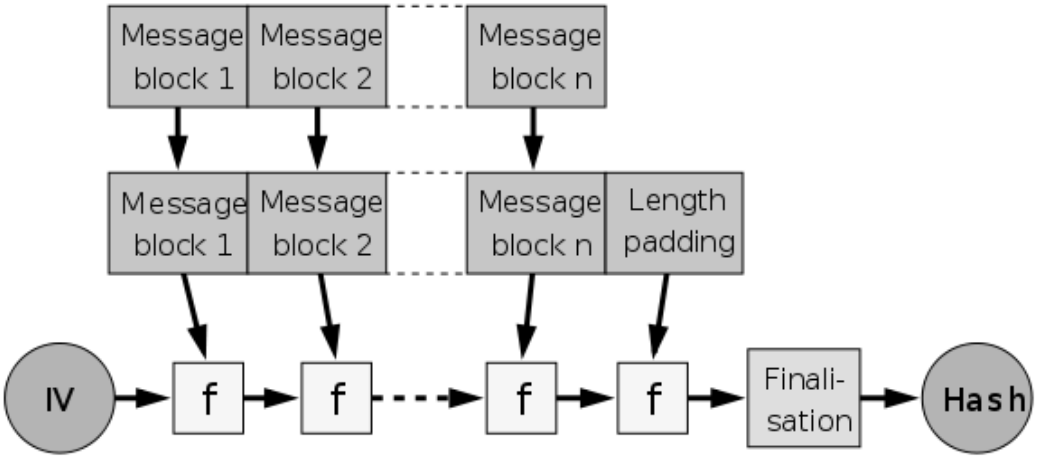


Рисунок 1. – Конструкция Меркла – Дамгарда

В качестве функции сжатия может использоваться блочный шифр. Данное расширение конструкции легло в основу схем Девиса – Мейера, Матиса – Мейера – Осеаса и Миагучи – Пренеля.

Конструкция Меркла – Дамгарда [1] используется в таких широко известных функциях, как MD5, SHA-1, SHA-2. Популярность структуры Меркла – Дамгарда обусловлена тем, что, как было доказано, если односторонняя функция сжатия f устойчива к коллизиям, то и хеш-функция, построенная на её основе, будет также устойчива к коллизиям. Несмотря на популярность схемы Меркла – Дамгарда, ряд работ показал недостатки данной конструкции, связанные с множественными коллизиями [2], дополнением сообщения до нужной длины [3], нахождением второго прообраза [4]. Атаки направлены не на конкретный алгоритм, а применимы к любой хеш-функции, построенной по схеме Меркла – Дамгарда. Атака,

основанная на множественных коллизиях [2], может быть осуществлена злоумышленником, даже если он имеет доступ к функции хеширования в целом, но не имеет возможности контролировать значения функции сжатия. Дополнение сообщения до нужной длины направлено на схему Девиса – Мейера, однако никак не затрагивает особенности самого блочного шифра, что делает эту атаку также достаточно общей. В связи с этим стали появляться предложения по модификации столь популярной конструкции.

Стефан Лакс предложил способ использования wide pipe конструкцию вместо конструкции Меркла – Дамграда. Wide pipe похож на конструкцию Меркла – Дамграда, но в данном методе длина промежуточного состояния в битах больше, чем длина хэша. Поэтому на последнем этапе функция финализации сжимает последнее промежуточное значение до нужного размера. После этого была предложена модификация этой конструкции Fast Wide Pipe. Основная идея этого метода в том, что промежуточное значение делится пополам. Одна половина идет на вход функции сжатия и складывается по модулю 2 со второй половиной. Данный метод работает быстрее, но при этом приходится использовать дополнительную память.

В 2006 году была предложена еще одна известная модификация конструкции Меркла – Дамграда под названием «HAIFA» [5]. Принципиальное отличие заключается в том, что на каждой итерации к блоку добавляется «соль» и количество бит, уже поступавших на вход функции сжатия на предыдущих итерациях. Добавление количества бит, уже поступавших на вход функции сжатия, позволяет противодействовать атакам с использованием неподвижных точек. Использование «соли» обеспечивает стойкость к атакам с предварительными вычислениями (словарная атака, использование радужных таблиц).

В 2007 году была предложена совершенно новая конструкция построения хэш-функций. Криптографическая губка [6] была разработана с целью заменить устаревшую конструкцию Меркла – Дамграда. Функция губки относится к классу алгоритмов с конечным внутренним состоянием, на вход которой поступает двоичная строка произвольной длины и которая возвращает двоичную строку также произвольной длины.

Губка – это итеративная конструкция для создания функции с произвольной длиной на входе и произвольной длиной на выходе на основе преобразований f. Губка имеет внутреннее состояние S с данными фиксированного размера b (бит). При этом данные разделены на 2 части – первая $S1$ размера r, а вторая $S2$ – размера c. Значение r называется битовой скоростью, а значение c – мощностью $b = r + c$.

На фазе инициализации блок данных размера b заполняется нулями, а входные данные M разбиваются на блоки размера r. Дальнейшая работа губки производится в 2 этапа:

1) фаза «впитывания» (absorbing) – выполняется операция XOR очередного блока исходного сообщения с первой частью состояния $S1$ размера r (бит), оставшаяся часть $S2$ состояния ёмкостью c остается незатронутой. Результат помещается в $S1$, а затем состояние S обрабатывается функцией f – многораундовой бесключевой псевдослучайной перестановкой, и так повторяется до исчерпания блоков исходного сообщения;

2) фаза «выжимания» (squeezing) – состояние S подаётся на функцию f, после чего часть $S1$ подаётся на выход. Эти действия повторяются до тех пор, пока не будет получена последовательность нужной длины (например, длины хэша).

Последние биты c зависят от входных блоков лишь опосредованно и не выводятся в ходе фазы «выжимания» (squeezing). Конструкция представлена на рисунке 2.

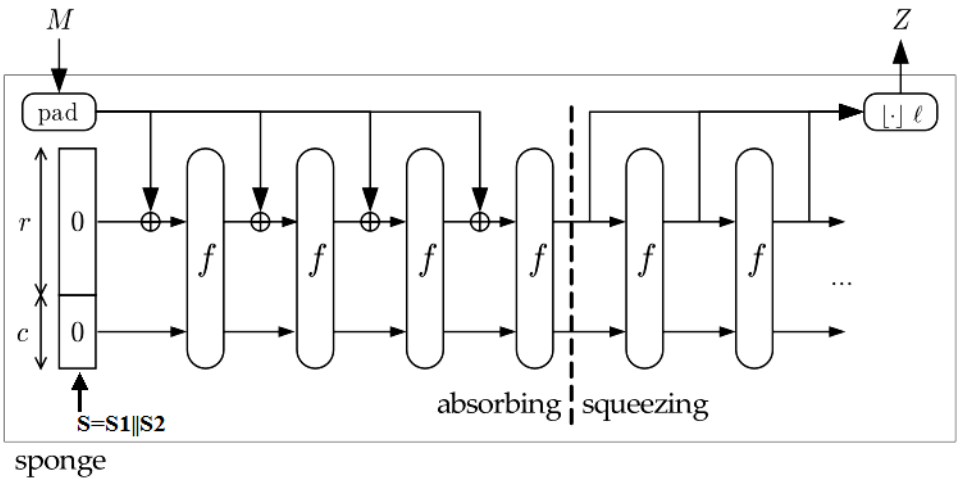


Рисунок 2. – Схема работы криптографической губки (Sponge)

Традиционный дизайн хэш-функций основан на функции сжатия. Такая функция отображает строку бит длиной m в строку длиной $m > n$ псевдослучайным образом, при этом значение n не больше 512 бит. Эта функция повторяется на протяжении нескольких раундов с различными константами, таким образом достигая нужной стойкости. Хэш-функции, построенные таким образом, преследует проблема коллизий 1 и 2-го рода. Казалось бы, многораундовое использование функции сжатия должно скрыть дефекты, но для таких функций легко находятся частичные коллизии, что ставит под сомнение стойкость всей конструкции.

Далее, рассмотрим существующие алгоритмы хэширования, которые построены с использованием конструкции Меркла – Дамграда и её модификаций.

Международный стандарт ISO/IEC 10118-3 описывает специальные функции хэширования. Функции хэширования в данном стандарте основаны на повторном использовании тактовой функции. Указывается семь отдельных тактовых функций, от которых происходят отдельные специальные функции хэширования.

Специальные функции хэширования RIPEMD-160 и SHA-1 имеют длину хэш-кода до 160 бит и начальный вектор длиной 160 бит, состоящий из пяти 32-битных определённых стандартом слов. Специальная функция хэширования RIPEMD-128 имеет длину хэш-кода до 128 бит и начальный вектор длиной 128 бит, состоящий из четырёх 32-битных определённых стандартом слов. Специальная функция хэширования SHA-256 имеет длину хэш-кода до 256 бит и начальный вектор длиной 256 бит, состоящий из восьми 32-битных определённых стандартом слов. Специальная функция хэширования SHA-384 имеет длину хэш-кода до 384 бита и начальный вектор длиной 512 бит, состоящий из восьми 64-битных определённых стандартом слов. Специальные функции хэширования SHA-512 и WHIRLPOOL имеют длину хэш-кода до 512 бит и начальный вектор длиной 512 бит, состоящий из восьми 64-битных определённых стандартом слов.

Исходя из представленных выше данных, можно сделать вывод о практическом применении данных функций хэширования: функции RIPEMD-128, RIPEMD-160 и SHA-1 не обеспечивают достаточный уровень безопасности. Используя парадокс задачи о днях рождения, можно найти реальные коллизии для данных функций за 2^{64}, 2^{80} и 2^{80} шагов соответственно.

Оставшиеся функции хэширования (SHA-256, SHA-384, SHA-512, WHIRLPOOL) являются более стойкими к атакам на основе коллизий, и на данный момент являются пригодными для практического применения. Однако в ходе проведённых исследований были проведены атаки на усечённые варианты приведённых функций. Для WHIRLPOOL сгенерировать коллизию удалось для варианта в 4.5 раунда из 10, при этом сложность данной атаки составила 2^{120}. Для функции SHA-256 были найдены множественные коллизии с вариантом в 21 раунд из 64.

В Беларуси до 2016 года были определены два стандарта: по функции хэширования СТБ 1176.1-99 [9] и СТБ 34.101.31-2011 [10].

В основе алгоритма хэширования по стандарту СТБ 1176.1-99 лежит итеративный механизм на базе четырех односторонних преобразований. Начальный вектор размером 256 бит выбирается произвольным образом, и хэш-код имеет длину 256 бит.

В основе алгоритма хэширования по стандарту СТБ 34.101.31-2011 лежит итеративный механизм на базе двух блочных односторонних преобразований, использующих шифрующую функцию, определённую в этом же стандарте. Начальный вектор размером 256 бит определён заранее, хэш-код имеет длину 256 бит.

После того как для семейства SHA-2 были приведены теоретические атаки, в 2007 году NIST было принято провести конкурс SHA-3 для выбора нового стандарта. С этого момента начался активный отбор кандидатов на использование хэш-функции в качестве нового стандарта SHA-3. Новый стандарт SHA-3 должен поддерживать семейство алгоритмов, реализующих хеш-суммы длиной 224, 256, 384 и 512 бит. Как минимум, одна функция из семейства должна поддерживать хеш-код аутентификации сообщений (HMAC) и рандомизированное хеширование. Кроме того, для всех n бит хеш-суммы алгоритм должен обеспечивать выполнение следующих условий:

- устойчивость к нахождению прообраза для n бит;
- устойчивость к нахождению второго прообраза для $(n - L)$ бит, где первый прообраз имеет длину не более $2L$ блоков;
- устойчивость к коллизиям для $n/2$ бит;
- устойчивость к атакам дополнением сообщения;
- для любого $m \leq n$ любое подмножество из m бит хеш-суммы длиной в n бит должно удовлетворять выше названным условиям.

Финалистами конкурса стали алгоритмы Skein, JH, Grostl, BLAKE и Keccak. Из них самые лучшие показатели продемонстрировал алгоритм Keccak, в 2012 году став новым стандартом хэширования SHA-3. Алгоритм 5 августа 2015 года утверждён и опубликован в качестве стандарта FIPS 202. Алгоритм SHA-3

построен по принципу криптографической губки (данная структура криптографических алгоритмов была предложена авторами алгоритма Keccak ранее). Функция сжатия представляет собой 1600-битовую перестановку. Размер блока сообщения зависит от желаемого размера хеш-суммы: для Keccak-512, -384, -256, -224 блок сообщения имеет размер 576, 832, 1088 и 1152 бит соответственно. Также команда разработчиков Keccak предложила версии с уменьшенной длиной перестановки в 25, 50, 100, 200, 400 и 800 бит вместо 1600 бит. Однако в качестве стандарта была принята только функция Keccak с длиной состояния 1600 бит, и разработчики рекомендуют пользоваться только ей. Функция сжатия f перемешивает внутренне состояние алгоритма, которое имеет вид массива 5×5 с элементами длиной 64 бита, на протяжении 24 раундов.

Атаки на нахождение коллизий (двух произвольных сообщений, дающих одинаковый хэш) и вторых прообразов (сообщения, дающего такой же хэш, что и имеющееся) имеют важное практическое и теоретическое значение, но наряду с неинвертируемостью не обеспечивают полной оценки стойкости хэш-функции. Существует целый ряд экзотических атак – на удлинение сообщения, атаки на частичные коллизии с подобранным префиксом и др. Чтобы доказать стойкость конструкции губки ко всем возможным атакам, авторы приняли ещё одно радикальное решение – считать единственным и достаточным общим критерием стойкости неразличимость хэш-функции от случайного оракула [7].

Случайный оракул (*Random Oracle*) – это идеализированная функция, описывающая работу машины с практически бесконечным объёмом памяти, которая на любой запрос может выдать идеально случайное число и запомнить пару «запрос – ответ». Если такой же запрос будет когда-либо повторён, то ответ будет не генерироваться заново, а взят из запомненного списка. Если перестановка f, лежащая в основе губки идеальна, то и хэш-функция доказуемо неразличима от RO, значит никакие из возможных атак действовать не будут. Конечно, есть оговорка на наличие универсального тривиального различителя для всех возможных хэш-функций: построение случайного оракула на алгоритме с конечным числом состояний невозможно. Например, коллизия внутреннего состояния приведёт к неслучайному выходу, что было бы невозможно в RO, у которого никаких внутренних состояний нет [7].

Авторы Keccak доказали, что стойкость такой конструкции не отличается от случайного оракула с размером выхода равным $c/2$, при условии, что f – идеальная функция перестановки. То есть чтобы получить хэш с доказанной математически стойкостью в 512 бит, нужно сделать размер параметра c (независимой части состояния) 512*2 = 1024 бита.

Данные теоретические результаты позволяют использовать Keccak в качестве хэш-функции с произвольным размером выхода, потокового шифра, функции выработки ключей из пароля, кодов аутентификации сообщений, криптостойкого генератора псевдослучайных чисел с подкачкой энтропии из внешнего источника и с затиранием внутреннего состояния.

В 2014 году появились первые атаки на Keccak [8]. В режиме хэш-функции взлом пока составляет 5 раундов, в ключевых режимах – 9 раундов. Но для полных 24 раундов исследователи оценивают стойкость ключевых режимов Keccak как всё ещё высокую.

В 2016 году в Беларуси появился новый стандарт алгоритмов хэширования СТБ 34.101.77 [12]. Переход от алгоритма СТБ 34.101.31 к алгоритмам этого стандарта позволяет увеличить скорость хэширования, по крайней мере, на паритетном уровне стойкости и на 64-разрядных аппаратных платформах. Кроме этого, алгоритмы хэширования данного стандарта поддерживают все три уровня стойкости алгоритмов ЭЦП, определённых в стандарте СТБ 34.101.45 [11], в то время как алгоритм СТБ 34.101.31 поддерживает только первый уровень. В отличие от Keccak, СТБ 34.101.77 внутренне состояние имеет длины 1536. Состояние представляется в виде массива 3×8 с элементами длиной 64 бита. Преобразование выполняется, так же как и в Keccak, на протяжении 24 раундов, при этом в каждом раунде происходит наложение раундовой константы с помощью операции XOR. В целом пошаговая функция f СТБ 34.101.77 разработана иначе. На каждом раунде в преобразовании дополнительно участвуют 5 переменных.

Алгоритмы этого стандарта отличаются уровнем стойкости l. Это натуральное число, кратное 16 и не превосходящее 256. Алгоритм уровня l вычисляет хэш-значения длины, обрабатывая входные слова блоками длины $1536 – 4l$. Уровни $l = 128$, $l = 192$, $l = 256$ являются стандартными. Разработчики стандарта рекомендуют отдавать им предпочтение.

Заключение. Криптография не стоит на месте и развивается. Функции хэширования используются во многих программных средствах, что повышает требования к их стойкости. Принятие нового стандарта алгоритмов хэширования SHA-3 стимулировало развитие различных областей криптографии, а также послужило толчком к разработке модификаций и новых алгоритмов.

ЛИТЕРАТУРА

1. Merkle, R.C. A Certified Digital Signature / R.C. Merkle // In Advances in Cryptology – CRYPTO’89 Proceedings // Lecture Notes in Computer Science. – Vol. 435; ed. G. Brassard; Springer-Verlag, 1989. – P. 218–238.

2. Joux, A. Multicollisions in iterated hash functions. Application to cascaded construction / Antoine Joux // In Advances in Cryptology – CRYPTO '04 Proceedings, Lecture Notes in Computer Science, Vol. 3152, M. Franklin, ed, Springer-Verlag, 2004. – 306 – 316 p.
3. Kelsey, J. A long-message attack on SHAx, MDx, Tiger, N-Hash, Whirlpool, and Snefru. Draft / John Kelsey. – Unpublished Manuscript.
4. Maurer, U. Indiff erentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology / Ueli Maurer, Renato Renner, Clemens Holenstein // In M. Naor, editor, Theory of Cryptography, First Theory of Cryptography Conference, Vol. 2951 of Lecture Notes in Computer Science. – Springer, 2004. – P. 21–39.
5. Biham E. A Framework for Iterative Hash Functions – HAIFA / Eli Biham, Orr Dunkelman // Cryptology ePrint Archive, 2007 [Electronic resource]. – Mode of access: http://eprint.iacr.org/2007/278. – Дата доступа: 25.09.2016.
6. Cryptographic sponge functions – STMicroelectronics / Guido Bertoni [et al.], 2011. – 93 с.
7. Проект OpenPGP в России [Электронный ресурс] / Хэш-функция Keccak и конструкция Sponge как универсальный криптопримитив. – Россия, 2007. – Режим доступа: https://www.pgpru.com/biblioteka/statji/keccaksponge. – Дата доступа: 25.09.2016.
8. Проект OpenPGP в России [Электронный ресурс] / Низкая алгебраическая сложность Keccak/Keyak облегчает кубические атаки. – Россия, 2007. – Режим доступа: https://www.pgpru.com/novosti/2014/nizkajaalgebraicheskaja slozhnostjkeccakkeyakoblegchaetkubicheskie ataki. – Дата доступа: 25.09.2016.
9. Информационная технология. Защита информации. Функция хэширования: СТБ 1176.1-99. – Минск : Госстандарт, 1999.
10. Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности: СТБ 34.101.31-2011. – Минск : Госстандарт, 2011.
11. Информационные технологии. Алгоритмы электронной цифровой подписи и транспорта ключа на основе эллиптических кривых : СТБ 34.101.45 – 2013. – Минск : Госстандарт, 2013.
12. Информационные технологии. Защита информации. Алгоритмы хэширования: СТБ 34.101.77-2016. – Минск : Госстандарт, 2016.

УДК 004.021

КОНТРОЛЬ ЦЕЛОСТНОСТИ ИНФОРМАЦИИ

К.С. ГОЛОВАН
(Представлено: Е.Р. СУХАРЕВ)

Раскрываются понятия целостности информации, контроля целостности информации и её актуальность. Приводятся методы контроля целостности информации: контрольные суммы, циклические коды, хэш-функции.

В условиях глубокого проникновения информационных технологий во все области жизнедеятельности человека надежность идентификации информации и контроль ее целостности становится важной проблемой, причем как научно-технической, так и социальной. Научно-техническая проблема включает в себя создание математических подходов, алгоритмов, программного и аппаратного обеспечения для решения этой проблемы. Социальный аспект проблемы связан с необходимостью создания общедоступной, удобной и защищенной системы надежной идентификации данных, адекватной степени развития информационных технологий.

Контроль целостности информации имеет огромное значение в различных областях деятельности человека. В финансовой области каждый день осуществляется большое количество финансовых транзакций в Интернете, в которых контроль целостности обрабатываемой, передаваемой и хранимой информации является важной частью.

Хозяйственная и юридическая деятельность также чрезвычайно критична к целостности информации в компьютерных сетях. В Беларуси в связи с утверждением «Инструкции о порядке взаимодействия ведомственных систем электронного документооборота с системой межведомственного электронного документооборота государственных органов» от 27 мая 2013 г. № 33 также возникает проблема контроля целостности сообщений и документов. Важность контроля целостности подтверждает принятие Закона Республики Беларусь от 28 декабря 2009 года № 113-З «Об электронном документе и электронной цифровой подписи».

Создание защищённой системы – задача комплексная, и решается она путём применения программно-технических средств, а также организационных мер. Реальная система защиты строится исходя из возможных угроз и выбранной политики безопасности.

Основными угрозами информационной безопасности по аспекту, на который эти угрозы направлены, являются:

- угрозы конфиденциальности;
- угрозы целостности;
- угрозы доступности;
- угрозы подлинности;
- угрозы сохранности.

Система защиты информации обеспечивает целостность информации, если она обеспечивает достоверность, полноту и защищённость информации от неумышленных и преднамеренных искажений при хранении, передаче и обработке.

Одним из способов обеспечения целостности информации является применение средств контроля целостности программного обеспечения и обрабатываемой информации, включая и её восстановление.

Контроль целостности информации программ и данных – обнаружение их любых несанкционированных изменений, которые могут носить как случайный характер, так и быть вызваны несанкционированными действиями.

Основной задачей средств контроля целостности информации является обеспечение такого состояния системы, когда невозможно скрыть факт любой несанкционированной модификации информации.

Защита информации от модификации требует решения комплекса задач, связанных с надёжность аппаратного и программного обеспечения, организационными вопросами, резервированием данных, защитой от компьютерных вирусов, проверкой целостности данных, оперативностью их восстановления и др. В реальных информационных системах всегда имеется существенная вероятность модифицирования информации. Одной из важнейших задач защиты от модифицирования данных является обнаружение факта искажения данных. Во многих случаях обнаружение этого факта является достаточно сложной задачей. Последствия от использования модифицированных программ и данных в информационных автоматизированных системах могут привести к большому ущербу.

Причинами нарушения целостности информации являются ненадёжность аппаратных средств, используемых в информационных технологиях, воздействие внешних электромагнитных излучений, наличие естественных помех в каналах связи, ошибки операторов и программистов, компьютерные вирусы и действия нарушителей. Многие из этих причин вызывают появление непреднамеренных ошибок, которые имеют случайный характер, а другие связаны с умышленными воздействиями на информацию.

Это может быть осуществлено специально внесённой в компьютерную систему программой или специально разработанным вирусом. Нарушитель, получая возможность несанкционированного доступа к информационным ресурсам, может внести изменения в электронные документы, модифицировать программу, удалить информацию из базы данных или внести дополнительную информацию. Умышленные воздействия имеют целенаправленный характер. При целенаправленном внесении изменений нарушитель может учитывать используемые механизмы обнаружения модификаций с целью осуществления такого модифицирования, которое нельзя было бы обнаружить. В некоторых случаях умышленные воздействия будут приводить к случайному модифицированию данных. Очевидно, что если решается задача обнаружения целенаправленного модифицирования, то одновременно решается задача обнаружения случайных искажений информации.

В общем случае контроль целостности информации реализуется путем предварительного определения характеристики целостности информации, называемой эталонной характеристикой, или эталонным кодом обнаружения модификаций. Эта эталонная характеристика по своему объему значительно меньше контролируемой информации, а ее значение отражает содержимое защищаемых от модификации данных. В зарубежной литературе эталонную характеристику обнаружения модификаций называют MAC-кодом (message authentication code) [3].

В процессе непосредственного контроля целостности информации выполняются следующие действия:

- для контролируемой информации определяется текущая характеристика обнаружения модификаций по тому алгоритму, по которому формировалась эталонная характеристика;
- текущая и эталонная характеристики обнаружения модификаций сравниваются. Если они совпадают, то считается, что контролируемая информация не подвергалась изменению.

К методам определения модификаций информации, вызванных случайным образом, относятся контрольное суммирование и использование циклических кодов.

Использование контрольных сумм и циклических кодов имеет существенный недостаток. Алгоритм получения контрольных характеристик для этих методов хорошо известен, поэтому злоумышленник может произвести модификации таким образом, чтобы контрольная характеристика не изменилась [3].

Задача злоумышленника значительно усложнится, если эталонная характеристика формируется и защищается криптографическими методами. Для этого необходимо использовать так называемую хэш-функцию. Преимущество хэш-функции в том, что она является необратимой функцией, т.е. для любого значения m легко вычислить $h(m)$, но для любого значения x невозможно найти такое m, что $h(m) = x$ [1].

В широком смысле термин «хэш-функция» используется для обозначения преобразований, которые отображают массив данных произвольного размера в блок данных фиксированного размера.

Функция хэширования позволяет осуществлять проверку целостности сообщений (документов), передаваемых в системах обработки информации различного назначения, с гарантированной достоверностью.

Для осуществления возможности контроля целостности данных необходимо вычислить от них значение функции хэширования и хранить его достоверным способом. При необходимости проверки целостности данных значение функции хэширования от этих данных вычисляется заново. В случае если вновь вычисленное значение совпало с достоверно хранимым, целостность данных подтверждается. В противном случае – целостность нарушена.

Функции хэширования применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур электронной цифровой подписи при передаче, обработке и хранении информации.

Хэш-функция должна удовлетворять следующим требованиям [2]:

- аргументом для хэш-функции может быть сообщение произвольной длины;
- значение хэш-функции имеет фиксированный размер;
- хэш-функция является эффективно вычислимой;
- хэш-функция является криптографически стойкой.

С точки зрения криптографической стойкости важным свойством хэш-функций является отсутствие коллизий, т.е. невозможно найти такие значения $x \neq y$, чтобы $h(x) = h(y)$. В криптографических приложениях важным понятием является криптографически стойкая хэш-функция, для которой не существует эффективного алгоритма нахождения значений $x \neq y$, где выполнялось бы условие $h(x) = h(y)$ (функ-

ция стойкая в сильном смысле), или не существует эффективного алгоритма нахождения коллизии при заданном x такого $y \neq x$ $h(x) = h(y)$ (функция стойкая в слабом смысле). Также было показано, что отсутствие коллизий не позволяет судить о практической стойкости хэш-функции. Практически значимым является отсутствие у хэш-функции корреляции. Свободной от корреляции называется хэш-функция, у которой невозможно найти пару $x \neq y$ такую, что вес Хэмминга двоичного вектора $h(x) \oplus h(y)$ будет меньше веса Хэмминга применительно к двоичному вектору $h(M)$ для некоторого малого M. Свобода от корреляции с точки зрения криптографической стойкости является гораздо более сильным свойством хэш-функции, чем свобода от коллизий [2].

Заключение. Угроза целостности является одной из основных угроз информационной безопасности. Несанкционированные изменения информации могут быть вызваны как случайными, так и преднамеренными действиями. Одним из способов обеспечения целостности информации является применение средств контроля целостности ПО и обрабатываемой информации, включая и её восстановление.

В общем случае контроль целостности информации реализуется путем предварительного определения характеристики целостности информации, называемой эталонной характеристикой, или эталонным кодом обнаружения модификаций.

В процессе непосредственного контроля информационной целостности выполняются следующие действия:

- для контролируемой информации определяется текущая характеристика обнаружения модификаций по тому алгоритму, по которому формировалась эталонная характеристика;
- текущая и эталонная характеристики обнаружения модификаций сравниваются. Если они совпадают, то считается, что контролируемая информация не подвергалась изменению.

Для выработки контрольных характеристик применяются криптографические хэш-функции.

ЛИТЕРАТУРА

1. Криптография: скоростные шифры / А.А. Молдовян [и др.]. – СПб. : БХВ-Петербург, 2002. – 496 с.
2. Криптография: от примитивов к синтезу алгоритмов / Н.А. Молдовян [и др.]. – СПб. : БХВ-Петербург, 2004. – 448 с.
3. Петров, А.А. Компьютерная безопасность. Криптографические методы защиты информации / А.А. Петров. – М. : ДМК, 2000. – 448 с.

УДК 004.021

ПРОЕКТИРОВАНИЕ КРИПТОСИСТЕМЫ, ОСНОВАННОЙ НА АУТЕНТИФИКАЦИИ ЛИЧНОСТИ ПО ЛИЦУ И ФЛЭШ КАРТЕ, С ИСПОЛЬЗОВАНИЕМ СЛУЖБЫ МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ

Н.А. ГУРЕЦКИЙ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Представлен практический способ создания надёжной криптосистемы с нестандартным способом хранения пароля и аутентификации. Целью работы явилось написание программы для защиты пользовательских файлов с использованием флэш карты в качестве пароля и аутентификации пользователя по лицу. Задача решалась путём разбиения основной программы на библиотеки и вспомогательные подпрограммы. Программа написана на языке программирования C# с использованием Win32 API и библиотеки распознавания OpenCV.

В данной работе в качестве объекта исследования выступают пользовательские файлы на персональном компьютере с применением шифрования и нестандартного метода хранения пароля, исключающего человеческий фактор. При этом подходе злоумышленник, даже завладев данными, воспользоваться ими без знания ключа и алгоритма шифрования не сможет. Разработанный программный продукт предназначен для быстрого и надёжного шифрования файлов, дешифрования, открытия без возможности внесения изменений, сокрытия этих файлов стандартными средствами Windows; использование надёжного ключа шифрования, который не знает даже сам пользователь.

Принцип разработанного алгоритма защиты информации и его функционал. В качестве алгоритма шифрования использован алгоритм Triple DES (3DES). Этот алгоритм, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES, представляет собой симметричный блочный шифр. Цель создания Triple DES – устранение главного недостатка алгоритма DES: недостаточной длины ключа (56 бит), который можно взломать полным перебором. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше. Время, требуемое для криптоанализа 3DES, может быть в миллиард раз больше, чем время, необходимое для вскрытия DES [1].

Для генерации пароля, а затем и для дальнейшей аутентификации, пользователю необходимо использовать USB флэш-карту – первый уровень защиты. Также в первоначальной настройке программы пользователю необходимо сделать фотографию своего лица для его дальнейшего распознавания – второй уровень защиты. Из логина и данных флэш-карты (каждая флэш-карта обладает уникальным сочетанием данных, вшитых в плату) получаем хеш длиной 1024 бит – он же и есть пароль для шифрования. В алгоритме 3DES в качестве пароля используется лишь 192 бит, поэтому хеш специальным алгоритмом урезаем до 192 бит. При аутентификации сверяться будут не пароли, а одно и то же слово, зашифрованное ими, причем правильный вариант будет храниться в защищенном разделе реестра. Пользователю необходимо лишь один раз после полного включения компьютера вставить нужную флэш-карту и сопоставить лицо и видеокамеру, после чего пароль будет храниться в стэке приложения.

Для удобства использования этой криптосистемы процесс аутентификации пользователя и хранение пароля вынесен в отдельную службу Windows. Службы операционной системы Windows (англ. Windows Service, службы) – приложения, автоматически (если настроено) запускаемые системой при запуске Windows и выполняющиеся вне зависимости от статуса пользователя. Имеет общие черты с концепцией демонов в Unix [2].

Для общения службы с приложением дешифрования и открытия файлов необходимо использовать межпроцессорное взаимодействие. Межпроцессорное взаимодействие (англ. Inter-Process Communication, IPC) – набор способов обмена данными между множеством потоков в одном или более процессах.

Процессы могут быть запущены на одном или более компьютерах, связанных между собой сетью. IPC-способы делятся на методы обмена сообщениями, синхронизации, разделяемой памяти и удаленных вызовов (RPC). Методы IPC зависят от пропускной способности и задержки взаимодействия между потоками и типа передаваемых данных.

IPC также может упоминаться как межпотоковое взаимодействие (англ. inter-thread communication), межпоточное взаимодействие и межпрограммное взаимодействие (англ. inter-application communication). IPC наряду с концепцией адресного пространства является основой для разграничения адресного пространства [3].

Аутентификация по лицу будет производиться с помощью технологии OpenCV [4]. OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++/C#, также разрабатывается для Python, Java, Ruby, Matlab, Lua. Может свободно использоваться в академических и коммерческих целях – распространяется на условиях лицензии BSD [5].

Логика работы приложения представлена на рисунке 1.

Рисунок 1. – Логика работы приложения

Для экономии ресурсов и удобства работы вся программа разбита на несколько отдельных подпрограмм:

- служба Windows CryptoServiceGurezkiy.exe для мониторинга подключенных флэшек, аутентификации пользователя, мониторинга появления новых файлов на виртуальном диске и шифрования этих файлов, а также для выдачи пароля другим программам по IPC;
- программа cryper.exe для расшифровывания файлов и их копирования;
- CryptoSystem.exe для настройки приложения.

Так как эти программы используют схожий функционал, то целесообразно вынести все классы и методы в общую библиотеку классов, причём в несколько разных.

Библиотеки:

- библиотека Crypto3DES.dll для шифрования и расшифровки файлов;
- библиотека InterComunication.dll для общения службы CryptoServiceGurezkiy.exe и cryper.exe;
- библиотека LogicApp.dll для хранения общей логики программы и классов для хеширования данных, распознавания флэшек в системе, работы с реестром, сохранения изображения лица пользовате-

ля, распознавания лица пользователя, монтирования папки в качестве виртуального диска, отслеживания изменений в файловой системе виртуального диска.

Общая схема зависимостей показана на рисунке 2.

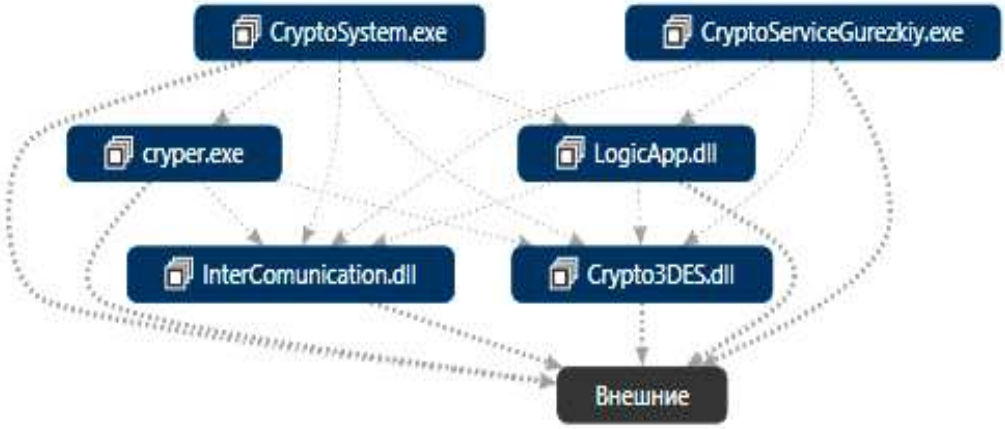


Рисунок 2. – Схема зависимостей проекта

Заключение. Авторами спроектирован программный продукт для надёжного хранения конфиденциальной, приватной, секретной информации на персональном компьютере, позволяющий пользователю комфортно проходить процедуру аутентификации. Для этого достаточно подключить нужную флэш-карту к компьютеру и показать лицо видеокамере. На данный момент ведутся работы по увеличению функционала программы.

ЛИТЕРАТУРА

1. Triple DES [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/Triple_DES. – Дата доступа: 08.12.2015.
2. Службы Windows [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/Службы_Windows. – Дата доступа: 19.12.2015.
3. Межпроцессное взаимодействие [Электронный ресурс]. – Режим доступа http://dic.academic.ru/dic.nsf/ruwiki/658474. – Дата доступа: 10.01.2016.
4. OpenCV [Электронный ресурс]. – Режим доступа http://opencv.org . – Дата доступа: 12.01.2016.
5. OpenCV [Электронный ресурс]. – Режим доступа https://ru.wikipedia.org/wiki/OpenCV. – Дата доступа: 13.01.2016.

УДК 004.021

РАЗРАБОТКА ИНТЕРФЕЙСА КРИПТОСИСТЕМЫ, ОСНОВАННОЙ НА АУТЕНТИФИКАЦИИ ЛИЧНОСТИ ПО ЛИЦУ И ФЛЭШ-КАРТЕ, С ИСПОЛЬЗОВАНИЕМ СЛУЖБЫ МЕЖПРОЦЕССНОГО ВЗАИМОДЕЙСТВИЯ

Н.А. ГУРЕЦКИЙ
(Представлено: канд. физ.-мат. наук, доц. О.В. ГОЛУБЕВА)

Представлен практический способ создания интерфейса для криптосистемы с использованием флэш-карты как пароля и распознаванием лиц. Целью работы явилось написание программы для защиты пользовательских файлов с использованием флэш карты в качестве пароля и аутентификации пользователя по лицу, а также создание интуитивно понятного пользовательского интерфейса. Задача реализации интерфейса решалась с использованием Windows Form и языка программирования C#.

В настоящий момент самым распространённым способом хранения информации является запись её на цифровые носители. Часть сохраняемой информации может быть конфиденциальной, приватной, секретной и нуждаться в защите. У такой информации есть законные пользователи. Но всегда существует вероятность появления пользователей незаконных, стремящихся захватить секретную информацию с целью обращения её себе во благо. Хакеры ежедневно воруют номера кредитных карт, банковские счета и даже личность человека. Опасаясь этого, законные пользователи принимают меры по защите информации.

На сегодняшний день известны несколько подходов к проблеме защиты информации, хранящейся в персональном компьютере. Один из них – шифрование данных.

Принцип разработанного алгоритма защиты информации и его функционал. В качестве алгоритма шифрования использован алгоритм Triple DES (3DES). Этот алгоритм, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES, представляет собой симметричный блочный шифр. Цель создания Triple DES – устранение главного недостатка алгоритма DES: недостаточной длины ключа (56 бит), который можно взломать полным перебором. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше. Время, требуемое для криптоанализа 3DES, может быть в миллиард раз больше, чем время, нужное для вскрытия DES [1].

Для генерации пароля, а затем и для дальнейшей аутентификации пользователю необходимо использовать USB флэш-карту – первый уровень защиты. Также в первоначальной настройке программы пользователю необходимо сделать фотографию своего лица для его дальнейшего распознавания – второй уровень защиты. Из логина и данных флэш-карты (каждая флэш-карта обладает уникальным сочетанием данных, вшитых в плату) получаем хеш длиной 1024 бит – он же и есть пароль для шифрования. В алгоритме 3DES в качестве пароля используется лишь 192 бит, поэтому хеш специальным алгоритмом урезаем до 192 бит. При аутентификации сверяться будут не пароли, а одно и то же слово, зашифрованное ими, причем правильный вариант будет храниться в защищенном разделе реестра. Пользователю необходимо лишь один раз после полного включения компьютера вставить нужную флэш-карту и поставить в соответствие лицо и видеокамеру, после чего пароль будет храниться в стэке приложения.

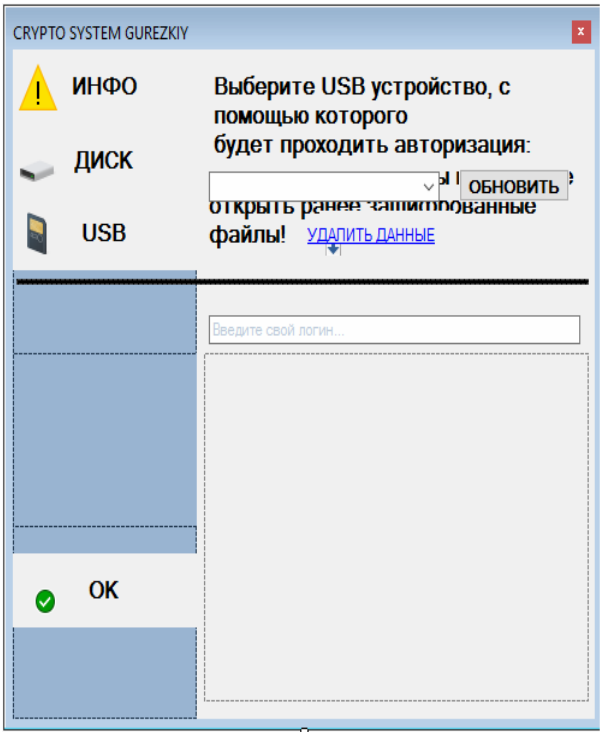


Рисунок 1. – Создание интерфейса CryproSystem.exe

Интерфейс программы CryptoSystem.exe создан с помощью Windows Form на языке программирования C# (рис. 1). На первый взгляд интерфейс кажется непонятным, однако не все эти элементы будут отображаться сразу. После некоторых настроек интерфейс будет выглядеть, как показано на рисунках 2–5. После настройки программы нужно нажать «OK». Изменения программы вступят в силу после перезагрузки компьютера. Если пользователь уже однажды настраивал эту программу, то перед настройкой необходимо удалить данные, нажав на строку «Удалить данные на вкладке ИНФО».

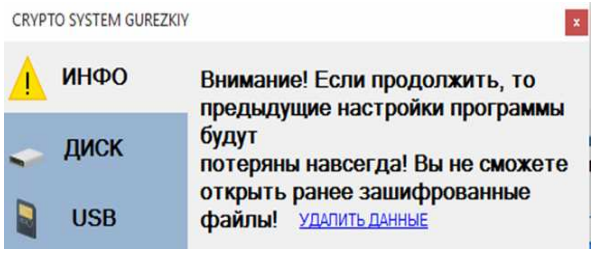


Рисунок 2. – Инфо

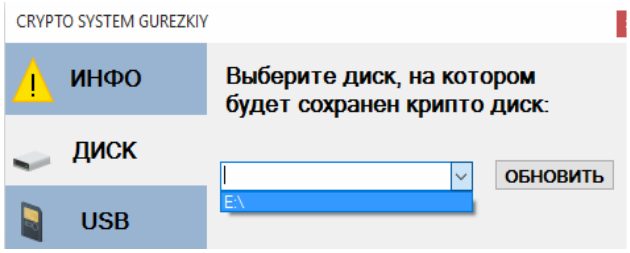


Рисунок 3. – Выбор диска для хранения папки с зашифрованными файлами

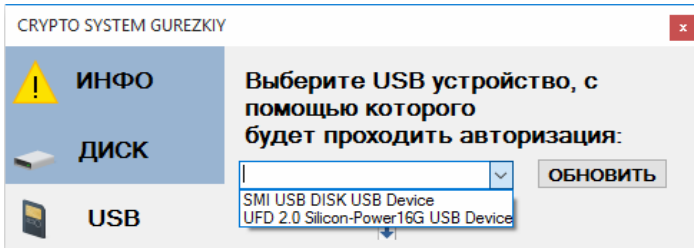


Рисунок 4. – Выбор флэш-карты

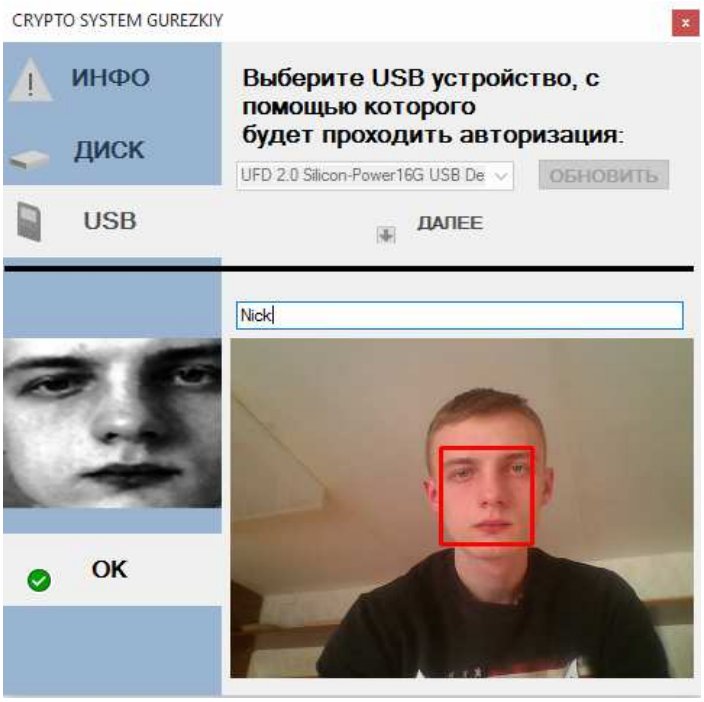


Рисунок 5. – Снимок лица пользователя

Заключение. Авторами разработан программный продукт для надёжного хранения конфиденциальной, приватной, секретной информации на персональном компьютере, позволяющий пользователю комфортно проходить процедуру аутентификации. Для этого достаточно подключить нужную флэш-карту к компьютеру и показать лицо видеокамере. На данный момент ведутся работы по увеличению функционала программы.

ЛИТЕРАТУРА

1. Triple DES. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Triple_DES. – Дата доступа: 08.12.2015.
2. Службы Windows [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Службы_Windows. – Дата доступа: 19.12.2015.
3. Межпроцессное взаимодействие [Электронный ресурс]. – Режим доступа http://dic.academic.ru/dic.nsf/ruwiki/658474. – Дата доступа: 10.01.2016.
4. OpenCV [Электронный ресурс]. – Режим доступа: http://opencv.org. – Дата доступа: 12.01.2016.
5. OpenCV [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/OpenCV. – Дата доступа: 3.01.2016.

УДК 004.89

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ: ИСТОРИЯ, КОНЦЕПЦИИ И МЕТОДИКА ПРОЕКТИРОВАНИЯ

А.И. ДУНЧЕНКО
(Представлено: Д.В. ПЯТКИН)

Представлена история возникновения и развития интеллектуальных информационных систем. Показаны их разновидности, классы решаемых ими задач. Раскрываются особенности архитектуры, описаны основные принципы проектирования.

В действительности, термин *интеллектуальная информационная система* (ИИС) [1] может быть применен к довольно широкому кругу компьютерных программ, объединенных общей характерной особенностью – использованием базы знаний как главного критерия в вычислениях при решении различных типов сложных задач. Это означает, что вместо жесткого кодирования всех возможных ситуаций и их исходов при выполнении программы с помощью конструкций *if-then-else*, в ИИС применяется другой подход: они формализуются и с помощью какой-либо декларативной нотации представляются в виде фактов и правил. По сути ИИС представляет собой гибрид интеллектуальной и информационной систем [2].

Первыми ИИС были в основном базирующиеся на правилах *экспертные системы* (ЭС) [3]. Иногда по ошибке оба этих термина применяются как взаимозаменяемые. Но на самом деле между ними существует значительная семантическая разница. Интеллектуальная информационная система – это общий принцип построения архитектуры программных систем. Если выражаться в терминах объектно-ориентированного программирования, то ИИС следует рассматривать как абстрактный базовый класс. С этой точки зрения экспертная система будет классом-потомком, специфицирующим применяемый подход для решения уже более узкого класса проблем.

Каждая ЭС дублирует способности к принятию решений реального человека – эксперта в какой-либо конкретной области (отсюда и название таких систем) [4]. Другими словами, ЭС может самостоятельно выполнять анализ предложенных фактов и параметров и на их основании выдавать высокоточные заключения (давая, при необходимости, пояснения к своим выводам). Таким образом, одна ЭС способна заменить целую команду экспертов, обычно необходимую для выполнения этой задачи. Использование таких программных систем предоставляет (в теории) широкие возможности по оптимизации и автоматизации бизнес-процессов (проводить реинжиниринг).

История экспертных систем берет свое начало в 70-х годах пршлого столетия, когда они были предложены руководителем проекта Stanford Heuristic Programming Project Эдвардом Фейгенбаумом. В то время основной платформой для их реализации были Lisp-машины, выпускаемые компаниями Symbolics, Xerox и Texas Instruments. Экспертные системы стали одними из первых технически и коммерчески успешных форм *программного обеспечения* (ПО), использующего *искусственный интеллект* (ИИ) и спроектированного с учетом необходимости решения сложных задач путем оперирования знаниями.

В 80-х годах вместе с появлением на рынке IBM PC, всеобщим растущим интересом к более новой *клиент-серверной* парадигме организации вычислений, с увеличением финансирования для исследований в университетах, последние начали предлагать специализированные обучающие курсы, в результате чего ЭС получили необходимый толчок в развитии.

После 90-х, когда программисты уже полностью освоили все тонкости разработки подобных систем и основанные на правилах алгоритмы логического вывода стали еще одним инструментом в арсенале IT-специалистов, множество ведущих компаний, специализирующихся на разработке программных бизнес-комплексов, начали интегрировать в свои продукты возможности ЭС как один из способов описания бизнес-логики.

Не стоит, однако, делать поспешный вывод о том, что ЭС являются единственными представителями ИИС. На самом деле возможности по применению ИИС довольно широкие. В круг наиболее распространенных задач, с которыми они способны справляться, входят: интерпретация данных, диагностика, мониторинг, проектирование, прогнозирование, планирование, обучение, управление, принятие решений.

Примером самой популярной и успешной ИИС в наши дни служит Google Translate [5]. Этот сервис реализует концепцию *статистического машинного перевода* [6], где база знаний формируется на основе статистических моделей, полученных в результате анализа больших корпусов текстов.

Несмотря на кажущуюся сложность, ядро ИИС состоит всего из двух компонентов: *базы знаний* (БЗ) и *машины вывода* (МВ).

Говоря простыми словами, БЗ – это технология представления и хранения данных – фактов об окружающем мире [7]. Ранние системы имели довольно примитивную БЗ, которая представляла собой

обычные присваивания значений переменным. Само понятие возникло для того, чтобы подчеркнуть отличие таких хранилищ от традиционных *баз данных* (БД), поскольку ИИС оперируют структурированной информацией (иногда связанной с помощью указателей и перекрестных ссылок), а не просто таблицами с числами и строками. Тем не менее по мере развития информационных технологий БЗ стали приобретать черты, характерные для объектных БД (классы и экземпляры), в результате чего различия стали менее очевидными.

Вся хранящаяся в БЗ информация должна удовлетворять двум важным требованиям: достоверности и релевантности, т.е. она должна быть объективной и принципиально доказуемой [8].

Стоит отдельно отметить, что в связи с растущей значимостью сети Internet и количеством web-контента, понятие «база знаний» часто используется для обозначения большого репозитория с документами, которые, в отличие от ИИС, предназначены для использования людьми, а не какой-либо программной системой.

Машина вывода – второй важный элемент ИИС – служит для получения логических выводов путем применения правил к имеющимся в БЗ фактам [9]. Новые данные, полученные в результате этих вычислений, могут привести к срабатыванию других правил, что, в свою очередь, инициирует новую итерацию.

Обычно МВ представляет собой совокупность *if-then* правил. Исследователями в области ИИ была предпринята попытка использования вместо таких выражений более унифицированного исчисления предикатов (логики первого порядка), но эта идея не прижилась, так как любое обобщающее утверждение касательно бесконечного множества вводит программу в нескончаемый цикл. Кроме того, очевидно, что психологически люди склонны к более простой и понятной *if-then* схеме для описания более сложной логики.

Раньше для написания МВ в основном применялись такие языки программирования (ЯП), как Lisp (Common Lisp, Scheme) и Prolog. Сейчас большой популярностью пользуются современные динамически интерпретируемые скриптовые ЯП (например, Python, Ruby), так как они позволяют вносить поправки в конфигурацию и менять логику работы приложения «на лету», без необходимости перекомпиляции модулей программы.

Машина вывода может работать в одном из двух возможных режимов: прямое связывание и обратное связывание.

Прямое связывание является более популярной и легко реализуемой стратегией логического вывода [10]. В этом режиме алгоритм начинает анализ с имеющихся данных и использует правила для выведения новых фактов. При этом выполняется просмотр всех правил и среди них ищется такое, для которого предикат (условие) принимает значение *истина*. Если поиск был успешным, то его следствие добавляется к уже известной информации и процесс повторяется до тех пор, пока не будет достигнута цель. Важным преимуществом этого метода является то, что он лучше всего подходит для работы в динамических системах, где в зависимости от ситуации могут меняться условия.

Обратное связывание, как и следует из названия, работает в строго противоположном направлении [11]. Для указанной цели осуществляется выборка правил, которые ведут к ней. Затем осуществляется анализ фактов, соответствующих этим правилам. В конечном счете, будет получено множество всех возможных комбинаций начальных условий, необходимых для достижения цели. Этот алгоритм часто применяется для доказательства теорем и для поиска решений в теории игр.

Как и для любого программного продукта, техника проектирования ИИС тесно связана с выбранной моделью разработки ПО. Но наиболее важным аспектом этого процесса является разработка БЗ, поскольку качество исполнения данного компонента (достоверность знаний) определяет функциональную пригодность получаемого на выходе программного продукта.

В процессе создания БЗ принимают непосредственное участие три вида специалистов: эксперты проблемной области, инженеры по знаниям и программисты, на которых возлагается практическая реализация алгоритмической составляющей.

Разработка БЗ включает несколько последовательных этапов. Вначале ставится цель и выявляется круг задач, подлежащих решению, определяется тип конечного пользователя. Затем следует этап извлечения знаний, когда с помощью экспертов выполняется тщательный анализ предметной области, который подразумевает выделение используемых понятий и связей между ними. Полученные таким образом знания после формализации подлежат дальнейшей структуризации с определением способов их представления и интерпретации, т.е. того, как система будет манипулировать ими. Когда методы решения задач ясны, можно приступить к моделированию работы системы и оценить ее адекватность. Затем следует наиболее трудоемкий этап, заключающийся в наполнении БЗ информацией, полученной инженерами по знаниям на основе анализа опыта экспертов, накопленного ими при решении реальных проблем. После этого создается один или несколько прототипов системы и осуществляется тестирование. По его результатам делается вывод о применимости выбранного способа представления знаний.

Проблемным местом при проектировании ИИС является то обстоятельство, что даже на сегодняшний день не найдено какое-либо универсальное логико-математическое и программное решение,

способное удовлетворить запросы разработчиков любой разновидности ИИС. В отличие от БД, фактически каждый раз приходится либо искать способы комбинирования и применения уже готовых наработок (технологий и алгоритмов), либо пытаться решить проблему самостоятельно.

Однако ИИС продолжают активно разрабатываться и успешно применяются в различных областях человеческой деятельности, начиная с автоматизации отдельных бизнес-процессов и заканчивая управлением деятельностью целых организаций (авиакомпании, железная дорога, NASA и т.п.). Если говорить об ЭС в отдельности, то, конечно же, они пока не способны полностью заменить реального человека. Потому что, во-первых, создание этих систем требует участие экспертов (значит, спрос на специалистов по-прежнему остается актуальным) и, во-вторых, в любом случае решающее слово всегда остается за человеком (директором, менеджером и т.п.). В то время как компьютерная программа, даже превосходя человека по скорости получения логического вывода, способна работать только с объективными данными и не обладает таким качеством, как *интуиция*, а значит, не может делать заключения с учетом эмпирической оценки ситуации. Поэтому данным системам обычно отводится второстепенная роль как вспомогательный инструмент для самопроверки.

ЛИТЕРАТУРА

1. Knowledge-based systems [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https: //en. wikipedia. org/wiki/Knowledge-based_systems. – Date of access: 17.09.2016.
2. Интеллектуальная информационная система [Электронный ресурс] / Википедия – Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Интеллектуальная_информационная_система. – Дата доступа: 17.09.2016.
3. Expert systems [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https: //en.wikipedia.org/ wiki/Expert_system. – Date of access: 18.09.2016.
4. Экспертная система [Электронный ресурс] / Википедия – Свободная энциклопедия. – Режим доступа: https://ru. wikipedia.org/wiki/Экспертная_система. – Дата доступа: 18.09.2016.
5. Google Translate [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/ wiki/Google_Translate. – Date of access: 19.09.2016.
6. Statistical machine translation [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en. wikipedia.org/ wiki/Statistical_machine_translation. – Date of access: 19.09.2016.
7. Knowledge base [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia. org/wiki/Knowledge_base. – Date of access: 20.09.2016.
8. База знаний [Электронный ресурс] / Википедия – Свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/ wiki/База_знаний. – Дата доступа: 20.09.2016.
9. Inference engine [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/ wiki/Inference_engine. – Date of access: 20.09.2016.
10. Forward chaining [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/ wiki/Forward_chaining. – Date of access: 20.09.2016.
11. Backward chaining [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/ wiki/ Backward_chaining. – Date of access: 20.09.2016.

УДК 004.89

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ОБРАБОТКА ЕСТЕСТВЕННЫХ ЯЗЫКОВ

А.И. ДУНЧЕНКО
(Представлено: Д.В. ПЯТКИН)

Раскрывается суть концепции искусственного интеллекта. Показано, как исследования в данной области привели к возникновению компьютерной лингвистики, представлены наиболее важные задачи, решением которых занимается эта научная дисциплина.

В отличие от животных, человек обладает такой специфической чертой, как стремление к познанию природы окружающей действительности: выявлению структурных особенностей материальных объектов, а также причин и механизмов взаимодействия между ними. И, как следствие этой особенности сознания, стремлением к самопознанию. Там, где философские рассуждения заходят в тупик ввиду информационной недостаточности, а реальные физические эксперименты невозможны из-за этических соображений, последним оплотом исследователей становится *моделирование* [1].

Моделирование есть процесс построения информационной (математической, логической, компьютерной) или физической модели изучаемого объекта, процесса или явления – прототипа, который по своим характеристикам максимально приближен к оригиналу, что позволяет, используя научный подход, получить фактические данные о нем.

Таким образом, если исходить из материалистического взгляда на мир, согласно которому любое понятие, в том числе и разум, возможно объяснить физически, есть вероятность того, что возможно создание модели человеческого сознания – *искусственного интеллекта* (ИИ) [2], базирующегося на другом, отличном от биологического, носителе.

В связи с появлением такой перспективы возникает вполне закономерный вопрос: как определить, что такая форма разума действительно обладает сознанием (в том числе, самосознанием) и является свободно мыслящей (т.е. не привязана жестко к базовым алгоритмам, способна к обучению и самостоятельному построению логических выводов).

Первые рассуждения на эту тему уходят своими корнями в далекий 1637 год и ведут к труду французского философа и математика Рене Декарта «Discourse on the Method» [3]. Таким образом, еще задолго до появления современных ЭВМ и возможности практической реализации ИИ уже рассматривались теоретические предпосылки к этому.

В 1936 году в своей книге «Language, Truth and Logic» философ Альфред Айер, который задался вопросом «*Как узнать, что другие люди имеют тот же сознательный опыт?*», сделал следующий вывод: «*Единственным основанием, на котором я могу утверждать, что объект, который кажется разумным, на самом деле не разумное существо, а просто глупая машина, является то, что он не может пройти один из эмпирических тестов, согласно которым определяется наличие или отсутствие сознания*».

Затем произошло событие, которое инициировало и определило направление в дальнейших исследованиях в области ИИ на несколько десятилетий вперед. В 1950 году Алан Тьюринг опубликовал статью «Computing Machinery and Intelligence», в которой предложил свой чрезвычайно простой тест как критерий для определения наличия интеллекта [4]. Суть теста довольно проста: человек общается с компьютером и другим человеком посредством текстовой переписки (участники не видят друг друга); его задача – на основании ответов определить, кто есть кто. Обмен текстовыми сообщениями осуществляется через фиксированные промежутки времени для того, чтобы нельзя было сделать вывод исходя из скорости реакции. Если по окончании испытания судья не может дать определенного ответа – считается, что ЭВМ прошла тест.

Данная идея встретила массу возражений. В частности, критики утверждали, что такой способ определения наличия интеллекта нельзя считать доказательным, поскольку он всего лишь демонстрирует сложность используемого программного обеспечения, которым управляется ЭВМ. На это Тьюринг дал поистине философский ответ: *«А с чего вы взяли, что человеческий разум не является лишь комбинацией сложных алгоритмов?»*

Вместе со своей концептуальной простотой тест Тьюринга обладает и рядом довольно серьезных недостатков. *Во-первых*, он ставит перед ЭВМ заведомо ложную задачу: вместо того, чтобы доказать свою разумность, машина должна ввести экзаменатора в заблуждение. А для этого ей достаточно преднамеренно допустить несколько грамматических или синтаксических ошибок, характерных для человеческой речи (как это однажды произошло во время ежегодного конкурса на получение премии Лёбнера). То есть определяющим фактором для прохождения теста является не демонстрация выдающихся интеллектуальных способностей, а простое подражание человеку. Но ведь нет никаких объективных фактов,

доказывающих, что наше сознание является самой совершенной формой разума во Вселенной. *Во-вторых*, тест затрагивает только внешнее проявление интеллекта – способность к общению, при этом не решенным остается вопрос: действительно ли машина *думает* или только симулирует данный процесс?

Поэтому в наше время специалисты по ИИ предпочитают сосредоточить свое внимание на изучении основополагающих принципов интеллекта, вместо того чтобы решать задачу прохождения теста Тьюринга, который, как многие считают, только отвлекает от реальных (и важных) исследований.

Несмотря на то, что тест Тьюринга стал чем-то вроде «спортивной олимпиады», он не потерял своей актуальности. Более того, он стал одной из предпосылок к возникновению *компьютерной лингвистики* [5].

Компьютерная лингвистика – это научное направление в области математического и компьютерного моделирования интеллектуальных процессов при создании систем ИИ, которое ставит своей целью использование математических моделей для описания *естественных языков* (ЕЯ). Основная задача компьютерных лингвистов заключается в разработке алгоритмов и прикладного программного обеспечения для обработки языковой информации.

Компьютерная лингвистика непосредственно связана с *обработкой естественных языков* [6], которая изучает проблемы машинного анализа и синтеза ЕЯ. Применительно к ИИ анализ означает понимание языка, а синтез – генерацию грамотного текста. Решение этих проблем ведет к созданию более удобной формы взаимодействия ЭВМ и человека. Теоретически построение естественно-языкового интерфейса для компьютеров – весьма привлекательная цель.

Четыре года спустя после выхода статьи Алана Тьюринга в штаб-квартире корпорации IBM состоялся *Джорджтаунский эксперимент*, который продемонстрировал полностью автоматический перевод более 60 предложений с русского языка на английский. Презентация положительно повлияла на развитие машинного перевода в последующие 12 лет. Демонстрация была широко освещена в СМИ и воспринята как успех. Она повлияла на решение правительств некоторых государств, в первую очередь США, направить инвестиции в область вычислительной лингвистики. Организаторы эксперимента уверяли, что в течение 3–5 лет проблема машинного перевода будет решена. Однако в действительности всё оказалось сложнее.

Ранние системы, такие как SHRDLU, работая с ограниченным «миром кубиков» и используя минимальный словарный запас, выглядели чрезвычайно хорошо, вдохновляя этим своих создателей. Однако оптимизм быстро иссяк, когда эти системы столкнулись со сложностью и неоднозначностью реального мира. Поэтому в последнее время акцент делается не на абстрактные модели, а на прикладные методы описания и обработки языка для компьютерных систем.

Понимание естественного языка считается AI-полной задачей, потому как распознавание живого языка требует огромных знаний системы об окружающем мире и возможности с ним взаимодействовать. Само определение смысла слова *понимать* – одна из главных задач ИИ.

Современные алгоритмы основаны на машинном обучении, в частности *статистическом машинном обучении* [7]. Данная парадигма отличается от большинства предыдущих попыток. Для реализации задач языковой обработки, как правило, выполняется прямое кодирование множества правил. Парадигма машинного обучения вместо общих алгоритмов использует статистические выводы, которые позволяют автоматически распознавать правила на основе анализа больших корпусов типичных примеров из реального мира (корпус – это набор документов, которые были вручную аннотированы, что позволяет извлечь из них правильные значения).

Обработка естественных языков имеет важное как теоретическое, так и практическое значение и состоит из множества отдельных исследуемых задач. Некоторые из них имеют прямое применение в реальном мире, другие служат в качестве подзадач, решение которых позволяет справляться с более сложными проблемами:

- оптическое распознавание символов (определение соответствия между графическим и информационным представлениями);
- распознавание речи (определение соответствия между звуковым и информационным представлениями); сегментация речи (разделение произносимой речи на отдельные слова);
- определение границ предложения;
- синтаксический разбор (построение синтаксического дерева предложения);
- определение частей речи (многие слова, особенно наиболее употребительные, могут выступать в роли различных частей речи в зависимости от контекста);
- разрешение кореферентности (определить в предложении или тексте слова, которые указывают на одни и те же объекты);
- распознавание именованных сущностей (определить, какие элементы в тексте соответствуют именам собственным и их тип: лицо, местонахождение, организация);
- выделение отношений (определение характера отношений между объектами, указанными в предложении);

- понимание естественного языка (преобразование текста в более формальное представление – логические структуры первого порядка, которыми легче оперировать на программном уровне);

- анализ речи (включает в себя определение характера отношений между предложениями;

-распознавание и классификацию речевых актов), анализ настроений (извлечение субъективной информации);

- тематическая сегментация (разделение текста на куски, посвященные отдельным темам);

- разделение сплошного текста на отдельные слова, разрешение неоднозначностей (для слов-омонимов);

- машинный перевод (является членом класса AI-полных задач, т.е. для решения надлежащим образом требует всех различных типов знаний, которыми обладают люди: грамматика, семантика, факты о реальном мире), ответ на вопрос (дать осмысленный ответ на поставленный вопрос);

- морфологическая сегментация (разделение слов на отдельные морфемы и определение их класса; трудность этой задачи во многом зависит от сложности структуры слов рассматриваемого языка);

- автоматическое реферирование (часто используются для предоставления резюме текста известного типа);

- синтез естественного языка (преобразование информации из компьютерных баз данных в читаемую форму), поиск информации.

Все эти задачи требуют большого объема исследований, для каждой из них есть, как правило, стандартная метрика оценки и стандартные корпусы, на котором задача может быть оценена.

Напоследок хотелось бы поделиться личным опытом, полученным автором этих строк в процессе работы над проектом «SayWhat?». При проектировании и реализации данной интеллектуальной информационной системы, которая представляет собой орфографический, лексико-грамматический и синтаксический анализатор простых предложений на английском языке, пришлось решать следующие задачи: выделение слов в предложении, проверка правописания, морфологический анализ, определение лексических и грамматических категорий, построение синтаксического дерева. Так как предполагалось написание всей основной логики «с нуля» – без использования сторонних библиотек – ключевым моментом оказался выбор языка программирования (ЯП) и декларативной нотации для базы знаний. Дело в том, что именно от используемого ЯП зависит то, насколько разработчик будет раскрепощен (или, наоборот, ограничен) в своих возможностях и архитектурных решениях в процессе кодирования. Поскольку предпочтение было отдано языку Ruby, который можно охарактеризовать как симбиоз самых лучших черт из Common Lisp, Smalltalk, Python, Perl и Bourne Shell, удалось добиться высокой степени повторного использования кода и закончить проект в срок.

Прогресс в области ИИ и компьютерной лингвистики тесно связаны друг с другом. Естественные языки являются продуктом деятельности разума и средством коммуникации между мыслящими существами, благодаря чему становиться возможным обмен знаниями и опытом. Исследования ихтиологов и орнитологов явно доказывают, что животным также свойственно осмысленное общение, причем речь идет не только об одном языке в рамках конкретного вида, – у них, как и у людей, существуют диалекты. Язык позволяет описать, охарактеризовать и классифицировать поступающую от органов чувств информацию, придает ей осмысленный оттенок. Изучение механизмов восприятия ЕЯ позволит лучше понять мыслительные процессы, протекающие в нашем сознании, и, возможно, получить ответы на интересующие нас вопросы философского характера. Кроме того, это открывает перспективы к созданию прогрессивных естественно-языковых интерфейсов между человеком и ЭВМ, т.е. переходу от командного типа управления (с помощью клавиатуры, мыши и других манипуляторов) к естественному общению. Правда, стоит задать себе вопрос: действительно ли мы хотим общения с компьютерами, как с себе подобными, или же лучше чтобы они оставались бездушными машинами, послушно выполняющими возложенную на них работу?

ЛИТЕРАТУРА

1. Scientific modelling [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Scientific_modelling. – Date of access: 23.09.2016.
2. Artificial intelligence [Electronic resource] / Wikipedia – The Free Encyclopedia. Mode of access: https://en.wikipedia.org/wiki/Artificial_intelligence. – Date of access: 24.09.2016.
3. Rene Descartes [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Rene_Descartes. – Date of access: 24.09.2016.
4. Turing test [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Turing_test. – Date of access: 25.09.2016.
5. Computational linguistics [Electronic resource] / Wikipedia – The Free Encyclopedia. Mode of access: https://en.wikipedia.org/wiki/Computational_linguistics. Date of access: 26.09.2016.
6. Natural language processing [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Natural_language_processing. – Date of access: 25.09.2016.
7. Machine learning [Electronic resource] / Wikipedia – The Free Encyclopedia. – Mode of access: https://en.wikipedia.org/wiki/Machine_learning. Date of access: 26.09.2016.

УДК 004.031.43

МНОГОКОМПОНЕНТНЫЕ СИСТЕМЫ ОБНАРУЖЕНИЯ СЕТЕВЫХ АТАК

П.А. ЖУРАВКОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассматривается подход к построению многокомпонентных систем обнаружения сетевых атак. Анализируются два базовых вида систем обнаружения атак: система поиска по паттернам и система, выявляющая аномалии в функционировании системы.

В настоящее время актуальна разработка и внедрение систем обнаружения сетевых атак для минимизации рисков информационной безопасности в информационных корпоративных сетях. Данные системы представляют собой специализированные программные или программно-аппаратные средства, которые позволяют вести активный аудит и управление безопасностью (прогнозировать, обнаруживать, предупреждать, контролировать, реагировать в режиме реального времени на риски безопасности) в корпоративной сети. Решение задачи разработки эффективной защиты информации от сетевых атак требует разработки новых методов, способных противостоять распределенным сетевым атакам различного происхождения.

Обнаружение атак злоумышленного поведения. Выделяются два базовых вида систем обнаружения сетевых атак: системы поиска заранее известных признаков атаки и системы, выявляющие аномалии в функционировании системы.

Если для обнаружения атаки требуется понимание ожидаемого поведения контролируемого нарушителя информации, то это технология обнаружения злоумышленного поведения. Работа систем обнаружения злоупотреблений базируется на создании шаблонов атак. Защитные системы такого типа эффективно показывают себя на известных схемах атак, однако в случае новой, ранее неизученной атаки или отклонения хода атаки от шаблона возникают проблемы [1]. Поэтому приходится постоянно поддерживать базу данных, включающую сигнатуру каждой атаки и ее вариации, непрерывно пополнять базы шаблонов. Немаловажным является правильно определить выборку параметров, контролируемых методом обнаружения сетевой атаки, основанной на злоумышленном поведении. Их малое количество или неправильно выбранные параметры могут привести к неполноте модели описания поведения атак, поэтому многие атаки могут быть не обнаружены [2]. С другой стороны, если взять слишком много параметров, учитываемых методом, это вызовет снижение производительности наблюдаемого узла за счет увеличения вычислительных операций.

Обнаружение атак аномальной активности. Метод обнаружения сетевых атак, основанный на методах обнаружения аномальной (подозрительной) активности, в отличие от рассмотренной выше, более гибок и позволяет обнаруживать неизвестные атаки. Системы обнаружения аномалий основаны на предположении, что все действия злоумышленника обязательно чем-то отличаются от поведения обычного пользователя, т.е. они аномальны.

Выявления атак, обусловленных аномальной активностью, основано на сравнении текущих значений параметров активности с нормальными значениями [3]. В качестве таких параметров могут выступать, например, количественные показатели использования системных ресурсов, интенсивности обращений к ресурсам или системным сервисам. Текущими значениями параметров активности являются средние значения, которые вычислены за короткий промежуток времени (от нескольких секунд до нескольких минут). Нормальными считаются средние значения этих параметров, вычисленные за большой период времени, относительно текущих значений параметров (от суток до нескольких недель).

Данный метод основан на том, что аномальное поведение субъекта проявляется как превышение нормального поведения. В частности, аномальным поведением может быть большое количество соединений за короткий промежуток времени. Однако аномальное поведение не всегда является атакой. Например, увеличение активности пользования социальными сервисами в момент празднования какого-либо мероприятия не является атакой.

Перед тем как система сможет начать работать, ей необходим период накопления информации, когда создается профиль нормальной активности системы, процесса или пользователя. Он становится эталоном, по которому оцениваются последующие данные.

Эта технология требует постоянной регистрации всех действий контролируемого субъекта, необходимых для обнаружения аномальной активности, что ведет к существенному снижению производительности защищаемого узла. Подобные системы сильно загружают центральный процессор, требуют больших объемов дискового пространства для хранения собираемых данных и в принципе не могут применяться в системах, которые работают в режиме реального времени, то есть систем критичных к быстродействию [4].

Многокомпонентные системы обнаружения аномальной сетевой активности. Принимая во внимание текущие и перспективные тенденции развития систем информационных технологий, а также объективные недостатки описанных выше двух подходов к обнаружению сетевых атак, можно сделать

вывод о необходимости смещения усилий на разработку и внедрение комплексных концепций построения систем защиты на основе распределенных вычислительных систем с использованием механизмов защиты на основе активного аудита [5]. Компоненты таких систем должны быть специализированы по типам решаемых задач, взаимодействовать друг с другом с целью обмена информацией и принятия согласованных решений, адаптироваться к реконфигурации аппаратного и программного обеспечения сети, изменению трафика, новым видам атак. Среди возможных технологий реализации такого подхода в качестве наиболее перспективного рассматривается технология многокомпонентных систем.

Основные положения предлагаемого подхода состоят в следующем. Компоненты системы защиты информации (агенты защиты) представляют собой автономные программы, реализующие определенные функции защиты для обеспечения требуемого уровня защищенности. Они позволяют реализовать комплексную надстройку над механизмами безопасности используемых сетевых программных средств, операционных систем и приложений, повышая защищенность системы до требуемого уровня. Предполагается, что агенты распределены по всем узлам защищаемой информационной системы и способны собирать и обрабатывать собранную информацию независимо от остальных агентов, работающих на других узлах. Они должны адаптироваться к реконфигурации сети, то есть изменению топологии сети в информационной системе, либо установке или удалению сетевых устройств на узлах. Очевидно, что собирать и анализировать данные агенты должны постоянно. Предлагаемый подход на основе многокомпонентных систем позволит использовать новые подходы, повышающие эффективность системы обнаружения сетевых аномалий, такие как мобильность (сбор и обработка сетевых данных построены на мобильных агентах, что позволяет сделать систему гибкой), адаптация к реконфигурированию аппаратного и программного обеспечения системы или изменению трафика, активность (система не только фиксирует факты удаленных сетевых атак, но и проводит действия направленные на оповещение специалистов информационной безопасности).

Заключение. Обнаружение сетевых атак на ресурсы информационных систем – весьма сложный технологический процесс, который связан со сбором больших объемов информации о функционировании защищаемой системы, анализом этих данных и, наконец, выявлением факта атаки. Для эффективного обнаружения атак требуется комплексное применение различных методов обнаружения аномальной сетевой активности.

ЛИТЕРАТУРА

1. Основы информационной безопасности : учеб. пособие для вузов / Е.Б. Белов [и др.]. – М. : Горячая линия – Телеком, 2006. – 544 с.
2. Щеглов, А.Ю. Защита компьютерной информации от несанкционированного доступа / А.Ю. Щеглов. – СПб. : Наука и техника, 2004. – 384 с.
3. Мониторинг состояния автоматизированной системы и обеспечение стабильности / А.А. Сапожников [и др.]. // Информационно-телекоммуникационые системы : сб. материалов Всерос. конкурса инновационных проектов аспирантов и студентов по приоритетному направлению развития науки и техники. – М. : ГНИИ ИТТ «Информика», 2005. – С. 123–124.
4. Сапожников, А.А. Практика централизованного мониторинга сетей / А.А. Сапожников // Проблемы функционирования информационных сетей : материалы междунар. конф. – Новосибирск : ЗАО РИЦ Прайс Курьер, 2006. – С. 257–260.
5. Сапожников, А.А. Обнаружение аномальной сетевой активности / А.А. Сапожников // Докл. Томского гос. ун-та систем управления и радиоэлектроники. – 2009. – № 1. – С. 79–80.
6. Емельянова, Ю.Г. Современный уровень и тенденции развития средств обеспечения сетевой безопасности систем облачных вычислений / Ю.Г. Емельянова, Э. Мбайкоджи, И.В. Соченков // Вестн. Рос. ун-та дружбы народов. Серия Математика. Информатика. Физика. – М. : РУДН, 2012. – № 2. – С. 116–126.
7. Шипулин, А. Мониторинг аномалий сетевой активности в промышленных система / А. Шипулин // Безопасность деловой информации. – 2015. – С. 32.
8. Aydin, M. A hybrid intrusion detection system for computer netwrok security Text / M. Ali Aydin, A. Halim Zaim, K. Gokhan Ceylan // Computer & Electrical Engineering. – 2009. – Vol. 35, Iss. 3, May. – P. 517–526.
9. Performance Measurement and Analysis of H. 323 Traffic Text / P. Calyam [et al.] // Passive and Active Network Measurement. – 2004. – P. 137–146.
10. Garfinkel, S. Practical Unix & Internet Security, O'Reilly / S. Garfinkel, A. Schwartz, G. Spafford, 2003. – 984 p.
11. Hoglund, G. Exploiting Software. How to Break Code / G. Hoglund, G. McGraw, A. Wesley, 2004. – 512 p.
12. Levenberg, K. A Method for the Solution of Certain Problems in Least Squares / G. Hoglund // Quart. Appl. Math. – 1944. – Vol. 2. – P. 164–168.
13. Sperotto, A. Anomaly characterization in flow-based traffic time series Text / A. Sperotto, R. Sadre, A. Pras // Proceedings of the 8th IEEE International Workshop on IP Operations and Management, IPOM 2008, Samos, Greece, 2008.

УДК 004.415.2.031.43

ПОДХОД К ПРОЕКТИРОВАНИЮ МНОГОКОМПОНЕНТНОЙ СИСТЕМЫ ОБНАРУЖЕНИЯ СЕТЕВЫХ АТАК

П.А. ЖУРАВКОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассматриваются походы к проектированию служб, осуществляющих сбор и обработку сетевых пакетов, поступающих в корпоративную систему. Анализируется принцип разработки современных веб-приложений. Представлены два компонента системы обнаружения атак: агент, выполняющий сбор и обработку сетевых пакетов, и клиентское веб-приложение.

В связи с проникновением новых сетевых технологий в различные сферы деятельности человека становятся всё более актуальными вопросы защиты компьютерных сетей от постороннего вмешательства. Ведущими IT-компаниями мира в настоящее время разрабатываются различные принципы организации защиты облачных вычислений. Актуальность подобных исследований следует из современных возможностей подключения вычислительных ресурсов к сетям Интернет и предоставления пользователям облачных услуг, что, в свою очередь, создает реальную угрозу их безопасности. Обратим внимание на два компонента системы обнаружения атак: агент, выполняющий (Windows-служба) сбор и обработку сетевых пакетов, работающий на узлах корпоративной системы, и клиентское веб-приложение.

Агенты сбора и обработки сетевых пакетов

Основная цель работы агента заключается в сборе и анализе количества, типов, IP-адрес, откуда пакет был послан, а также размера принимаемых и передаваемых пакетов на узле. Разрабатываемый компонент должен быть реализован в виде Windows-сервиса, который после загрузки операционной системы способен автоматически запуститься.

Фундаментальная задача разрабатываемой службы – бесперебойная работа 24/7 основных компонент, отвечающих за сбор и анализ данных. В противном случае система перестанет анализировать данные, что может вылиться в серьезные убытки для компании, содержащей атакуемый узел. По данным Лаборатории Касперского за 2015 год, потери компаний пострадавших от сетевых атак, основанных на небезграничности ресурсов атакуемой системы, в среднем составляют до 400$ тыс. В среднем 61% пострадавших временно теряли доступ к критичной информации, 38% не могли осуществлять свою непосредственную деятельность, а 33% компаний сообщили об упущенных сделках и контрактах. В связи с этими фактами система анализа и мониторинга обязана быть доступна в любое время.

Для достижения этой цели необходимо продумать и реализовать механизм проверки работы агента. На основании одного из пункта в настройках служба должна отправлять подобие ping-запроса в базу данных, сообщая тем самым информацию о том, что она корректно функционирует [2].

Необходимо разработать такую инфраструктуру, которая позволяла бы продолжить работу какой-либо подсистемы даже после ошибки в ней. В качестве основы для такой системы необходимо взять подход на основе потоков, когда для единицы работы создается отдельный поток (Task) операционной системы. Потоки, они же задачи, в операционных системах позволяют абстрагироваться от текущего последовательного выполнения программы. Единица работы начинает выполняться в отдельной нити, позволяя полностью отстраниться от главного контекста выполнения программы. Главным плюсом данного выбора в данной ситуации является то, что если в потоке происходит ошибка выполнения, она никак не влияет на работу остальной системы. Система продолжит работу.

В случае если какая-то подсистема службы вышла из строя (произошла ошибка выполнения программы), эта же подсистема должна корректно обработать произошедшую ошибку и продолжить корректно работать. Одним из паттернов потокового проектирования является реализация так называемых менеджеров задач, которые отвечают на создание, выполнение задач, а также обработку ошибок в этих задачах. Весь процесс создания задачи берет на себя менеджер, автоматически обертывая единицу работы в поток. Такая реализация позволяет значительно сократить объем кода, а также улучшить читаемость кода. После того как поток создан, его необходимо запустить на выполнение в отдельной нити. Менеджер автоматически запускает задачу после ее создания. Наконец, если в потоке произошла ошибка выполнения, менеджер обработает эту ошибку, тем самым предотвратив аварийное завершение работы всей системы [4].

Одной из задач проектирования агента является возможность удаленного конфигурирования службы. Данный функционал необходимо реализовать как часть клиентского веб-приложения. Также одним из требований является незаметное для работы агента изменение текущих настроек агента во вре-

мя его работы, то есть служба должна автоматически подгружать необходимые настройки из базы данных и применять их, не нарушая и не останавливая свою работу. Данный функционал необходимо реализовать с помощью кеш-механизма, который должен через какой-то промежуток времени проверять актуальные настройки. Основной критерий выбора данного подхода – уменьшение нагрузки на базу данных.

Основной задачей агента, исходя из требований, является сбор данных для последующего анализа. Единицей данных, передаваемой по сети, служит сетевой пакет – это блок данных, который сформирован определенным образом. Основной задачей злоумышленника, который хочет нарушить работу какого-либо узла в системе, является многократное увеличение передаваемых на узел пакетов, тем самым узел не в состоянии справиться с таким количество информации. В итоге, легальные пользователи не могут частично или в полной мере пользоваться корпоративной системой. Сбор информации о количестве пакетов, а также их содержимого является первостепенной задачей.

Захват пакетов должен быть реализован на уровне драйвера сетевой карты, который использует NDIS для чтения пакетов, которые получает сетевая карта. Существует библиотека Pcap, реализующая всю низкоуровневую логику по работе с данными драйвера, тем самым позволяя программисту миновать работу напрямую с драйвером. Рассматриваемая библиотека написана на языке программирования C, что не всегда удобно при работе с языками более высокого уровня, такими как Java и C#. Поэтому в разработке необходимо использовать обертку для Pcap на языке разработки дипломной работы – SharpPcap. SharpPcap реализует весь функционал библиотеки Pcap, дополняя ее специфическими для языка C# особенностями, а именно событийной моделью обработки поступающих пакетов. При разработке также необходимо учитывать размер и тип пакетов. Низкоуровневая библиотека Pcap реализует обработку следующих типов пакетов: ARP, IPv4, IPv6, ICMP, ICMPv6, IGMP, UDP, TCP. Все перечисленные типы пакетов разрабатываемая система должна собирать и подсчитывать. Для этого в режиме реального времени отдельный поток (задача) выполняет количественный подсчет каждого пакета, а также его размер и тип [4].

Подсистема фильтров должна осуществлять фильтрацию всех вышеперечисленных типов пакетов. На стадии проектирования необходимо продумать концепцию переиспользования фильтров, так называемых многоразовых фильтров, которые могут быть применены для разных сетевых устройств на разных узлах в корпоративной системе, что позволит удобно управлять процессом настройки подсистемы фильтров. На основе кеш-механизма подгрузки настроек агента нужно реализовать кешированную подгрузку фильтров из базы данных для сетевых устройств на данном узле, где работает агент.

Веб-приложение

Основной задачей клиентской веб-части является возможность настройки агентов, а также показ информации об их текущей нагрузки. Настройка агентов включает в себя отображение информации о работе агентов на узлах; вывод ошибок агентов, позволяющий специалисту по информационной безопасности наблюдать за корректностью работы агентов; вывод срабатываний различных фильтров на соответствующих узлах. Также веб-приложение должно показывать в режиме реального времени данные, собранные с помощью агентов сбора и обработки на узлах, а также строить наглядный график.

Веб-приложение должно быть разработано с применением паттерна MVC. MVC – паттерн проектирования, с помощью которого модель приложения, интерфейс пользователя и взаимодействие с пользователем разделены на три отдельных компонента так, чтобы изменение одного из компонентов минимально влияло на работу остальных. Контроллер перехватывает событие извне и в соответствии с заложенной в него логикой реагирует на это событие, изменяя модель посредством вызова соответствующего действия. Затем модель использует событие о том, что она изменилась, и все подписанные на это события отображения, получив его, обращаются к модели за обновленными данными и отображают их. Таким образом, придерживаясь такого подхода на протяжении всего цикла разработки ПО, код становится масштабируемым и легко читаемым. Схема работы MVC подхода показана на рисунке 1.

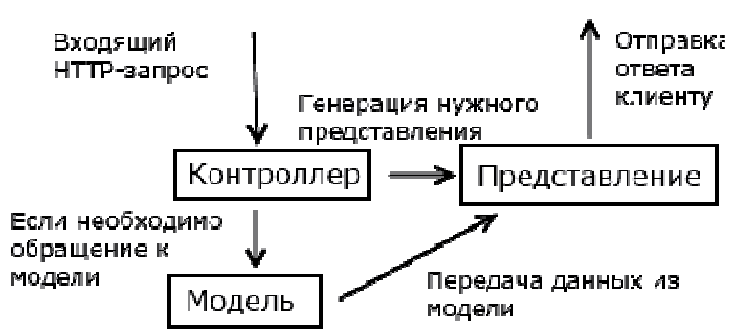


Рисунок 1. – Функциональная схема устройства MVC патттерна

Рассмотрим реализацию пользовательского интерфейса. На главной странице должна быть представлена общая сводка статистики со всех узлов. Для более наглядного восприятия на странице должны располагаться график нагрузки на узлах, график процентного соотношения типов пакетов, график количества пакетов на всех узлах и график количества трафика на узлах. Макет описываемой страницы представлен на рисунке 2.

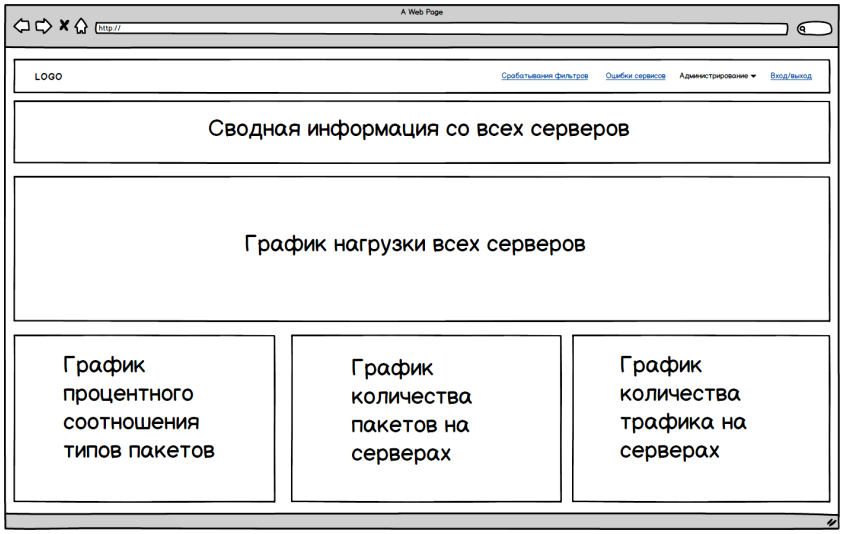


Рисунок 2. – Макет главной страницы веб-приложения

ЛИТЕРАТУРА

1. Система обнаружения вторжений – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikibooks.org/wiki/Система_обнаружения_вторжений. – Дата доступа: 26.09.2016.
2. Ping – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https: //ru.wikibooks.org/wiki/Ping. – Дата доступа: 25.09.2016.
3. Обработка исключений (библиотека параллельных задач) – MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/dd321409(v=vs.110).aspx. – Дата доступа: 25.09.2016.
4. Pcap – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https: //ru.wikibooks.org/wiki/Pcap. – Дата доступа: 27.09.2016.
5. Model-View-Controller – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikibooks.org/wiki/Model-View-Controller. – Дата доступа: 27.09.2016.

УДК 004.051

ЦЕЛЕСООБРАЗНОСТЬ РАЗРАБОТКИ ИГР ДЛЯ ПЛАТФОРМЫ ANDROID

Е.В. ЛАПУНОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассматривается целесообразность разработки мобильных игр для платформы Android, а также возможность заработка на них. Показано, что разработка игр для платформы Android целесообразна, так как платформа занимает ведущие позиции на рынке мобильных приложений.

Мобильные телефоны, а сейчас уже и смартфоны настолько прочно вошли в повседневную жизнь, что сложно представить, как можно обойтись без интернета или телефона. Возможно, есть люди, не пользующиеся этими услугами, но таких людей меньшинство.

Разработчики игр под мобильные устройства, хорошо понимая существующую тенденцию, разрабатывают огромное их количество.

Но существует множество операционных систем и их версий. Игра, написанная для одной системы, не будет работать на другой, если не предусмотреть совместимость с ней. Поэтому, как правило, приложение пишется для какой-то определённой платформы и, если оно заработало популярность, портируется на остальные.

Поэтому есть смысл изучить популярность платформ и их версий перед тем, как заниматься разработкой, а также ознакомиться с возможными способами заработка от игры.

Основной раздел

Согласно данным сайта tagline.ruhttp://www.3dnews.ru/ [1] по итогам 2016 года, лидирующей на данный момент платформой является iOS, однако Android ей практически не уступает, при этом у него открытый исходный код, что позволяет вести разработку продукта более глубоко. Рейтинг мобильных платформ был сформирован на основе анкетирования, которое пришлось на конец 2015-го и начало 2016 года. Респондентам предлагалось выбрать один или несколько вариантов ответа на следующий вопрос: «На каких платформах вы разрабатываете приложения и другие решения для мобильных устройств?».

На рисунке представлен рейтинг мобильных платформ на 2016 год.

#	Название	Год	Языки программирования	Открытый исходный код	Устройства
1 0	iOS	2007	Objective C, Swift, C / C++ и др.	×	смартфоны, планшеты, часы (watchOS), TV (tvOS), auto
2 0	Android	2008	Java, C / C++ и др.	✓	смартфоны, планшеты, часы, TV, auto
3 0	Windows	2010	C#, C / C++ и др.	×	смартфоны, планшеты, TV, auto
4 0	BlackBerry	2007	C / C++, Java и др.	×	смартфоны, планшеты
5 +1	Tizen	2012	C++	✓	смартфоны, планшеты, часы, TV, auto, фотокамеры

Рейтинг мобильных платформ на 2016 год

Компания Google регулярно отслеживает информацию о том, насколько популярными являются различные версии операционной системы Android [2].

У мобильной платформы Android есть более весомые плюсы, чем у iOS, для разработки мобильных приложений. Рассмотрим их:

1. *Открытый исходный код*. В операционной системе от гугл разработчику предоставляется больше свободы, чем в iOS. С таким подходом в итоговом продукте можно сделать значительно больше.

2. *Кастомизация*. Если разработчик желает изменить систему внутренне или внешне, например, для рекламы своей игры, то он сделает это без особых проблем.

3. *Поддержка нескольких аккаунтов*. Если смартфоном пользуется ваш ребенок и вы, то это не вызовет особых проблем, чего нельзя сказать о системе iOS.

4. *Доступ к файловой системе*. В iOS она закрыта, в Android – открыта всем желающим, что очень важно для многих. И сегодня в отзывах можно прочесть о том, что на «яблоке» этой функции нет, и это немалая проблема.

Игра, разработанная под Android, всегда найдёт своего пользователя. И чем актуальнее и интереснее она будет, чем меньше рекламы будет в ней, тем больше вероятность того, что в нее будет играть большое количество людей.

В сервисе Google Play заработать можно тремя способами:

- рекламные баннеры в бесплатной игре;
- продажа игры пользователю за деньги, как правило, в самой игре, тогда рекламу не размещают;
- реализация платных функций в бесплатных приложениях [3].

Наличие рекламы в игре сильно влияет на ее популярность, количество скачиваний и, соответственно, на окупаемость игры, так как чем больше людей будет в нее играть, тем больше прибыли она будет приносить. Большинство современных игр получает основной доход именно с рекламы.

Как и любая реклама, реклама в Android-играх показывается ради прибыли. Разработчики бесплатных игр и приложений встраивают специальный код, который отвечает за показ рекламы. Иными словами: встраивая рекламу, разработчики получают прибыль.

Но иногда реклама бывает слишком агрессивной (показывается каждые несколько секунд или после каждого уровня игры). Порой кнопка закрытия рекламы намеренно скрыта, что вынуждает пользователя нажимать на рекламу, и, соответственно, после этого пользователь перенаправляется на сайт рекламодателя. Именно из-за таких способов и ухищрений реклама отталкивает пользовательскую аудиторию.

Для того чтобы игра имела спрос и покупалась пользователями, она должно быть уникальной в своём роде. Также, если пользователь готов платить за игру деньги, она должна быть реализована на довольно высоком уровне, не содержать дефектов, полностью выполнять поставленные перед пользователем задачи. Как правило, стоимость игр относительно невысокая, что привлекает пользователей. С каждой продажи приложения в сервис Google Play идёт процент от стоимости приложения.

Платные функции в бесплатных играх – также довольно популярный способ заработка, к тому же он является одним из основных. Обычно в таких случаях игра не требует вложений со стороны пользователя и распространяется бесплатно, но требует внесения денежных средств за дополнительную функциональность. В качестве примера можно рассмотреть любую из игр, основанную на бизнес-модели free2play.

Free2play – это способ распространения компьютерных игр, позволяющий пользователю играть без внесения денежных средств. Авторы игр получают прибыль путём внесения небольших денежных сумм в игру. Часто для внутриигровых покупок используется игровая валюта с нетривиальными правилами конвертации из реальных денег (например, скидками при обмене крупных сумм). Многие Free-to-play игры являются способом завлечь игроков, не желающих тратить своё время на прохождение игры, и желающих получить игровые преимущества более быстрым способом. Траты не ограничиваются и могут достигать сотен и даже тысяч долларов. Одна из целевых групп пользователей – дети, которые могут играть и без родительского контроля.

Таким образом, разработка игр для платформы Android целесообразна, так как платформа занимает ведущие позиции на рынке мобильных приложений.

Если есть желание получать доход от игр, то есть смысл размещать в нём небольшие, ненавязчивые рекламные баннеры или реализовывать внутриигровые покупки.

ЛИТЕРАТУРА

1. Рейтинг и обзоры digital-рынка [Электронный ресурс] / Рейтинг мобильных платформ 2016. – Режим доступа: http://tagline.ru/mobile-platforms-os-rating/. – Дата доступа: 29.09.2016.
2. Специализированный российский информационно-аналитический сайт с самыми актуальными новостями из сферы IT [Электронный ресурс] / Статистика Google о популярности различных версий Android. – Режим доступа: http://www.ixbt.com/news/it/188256. – Дата доступа: 29.09.2016.
3. Способы заработка в Интернете [Электронный ресурс] / Как зарабатывать тысячи долларов на Android Market. – Режим доступа: http://max1net.com/kak-zarabatyvat-tysyachi-dollarov-na-android-market/. – Дата доступа: 29.09.2016.

УДК 004.51

КАК РАЗРАБАТЫВАЕТСЯ ИНТЕРФЕЙС В МОБИЛЬНЫХ ИГРАХ

Е.В. ЛАПУНОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассматриваются основные подходы к разработке интерфейса в мобильных играх, которых должны придерживаться разработчики мобильных игр. Сделан вывод, чем больше внимания разработчик будет уделять степени удобства продукта на всех этапах его разработки, тем более качественным продукт будет на выходе.

Каждый разработчик понимает, чтобы его игра пользовалось популярностью, она должна отличаться не только уникальной идеей, но и привлекательным и удобным интерфейсом. И даже если уникальная идея есть, то все может закончиться, еще не начавшись, если реализация этой идеи имеет много недочетов. Специально, чтобы такого не случилось, а игра обязательно смогла обратить на себя пристальное внимание, необходимо разработать удобный, красивый и функциональный интерфейс.

Основная часть. Различные приложения требуют совершенно разных подходов к своему воплощению. Тем не менее абсолютно всем приложениям присущи несколько глобальных принципов, то есть тех правил, которые нужно соблюдать всегда, независимо от того, какая игра разрабатывается [1]:

Реагирование. Независимо от того, что пользователь хочет сделать внутри игры, она должна реагировать на всего его действия мгновенно. И никак иначе. Нужно обратить внимание, что отзывчивость и скорость – это разные вещи. Разумеется, что разные операции требуют разное время для своего выполнения. Но пользователь обязательно должен понимать, что он делает, и знать, что игра про него не забыла и не зависла, она работает, просто для выполнения конкретной задачи нужно немного подождать, например, если это игра типа фермы, то пользователь должен понимать, что какое-то здание еще строится, и доступа к нему пока что нет.

Внимание к деталям. Даже самые незначительные детали в игре должны быть видны. Возможно, что воспользоваться какой-нибудь из этих мелких деталей пользователю никогда не потребуется, но если такая ситуация возникнет, то он не должен тратить свое время на ее поиск. Надо помнить, что внимание к деталям важно. Именно внимание к деталям поднимает приложение на уровень выше. В данном случае будет к месту аналогия с автомобилями: у автомобиля может быть мощный двигатель и превосходный дизайн, но если во время движения внутри салона будет шумно, то это уже будет не самый лучший автомобиль.

Принцип большого пальца. Этот принцип в своей английской версии звучит как thumbs. Принцип очень прост: пользователь должен без всяких проблем управлять игрой одним большим пальцем. Если пользователь взаимодействует с игрой двумя руками, то это уже неудобно. Конечно, это совсем не значит, что приложением нужно управлять только одним большим пальцем, но это должно быть возможно. Собственно говоря, принцип thumbs – это принцип по умолчанию.

Расположение и размер. Для того чтобы определить оптимальный размер кнопки в игре, вернемся к принципу thumbs. Достаточно посмотреть на большой палец правой руки, на ту часть, которой производятся нажатия на экран смартфона. Вот это и есть примерный размер кнопки, которая будет удобна для любого пользователя. Не нужно делать маленькие кнопки: во-первых, по ним сложно попасть, во-вторых, возвращаемся к пункту о внимании к деталям. Также нужно осознавать, где будет происходить размещение элементов управления в игре. Так, например, расположение кнопки «Delete» по соседству с кнопкой «Send» – это плохая идея [2].

Содержание. Революция, которую совершил сенсорный экран, заключается в том, что он позволяет пользователю напрямую взаимодействовать с содержанием игры. Все это устраняет лишние проблемы, такие как, например, «мышь», клавиатура, трекпад и т.д. Практически любой маленький ребенок без особых проблем играет во что-нибудь на iPad или iPhone, в то время как ноутбук остается для него еще сложной и непонятной загадкой. Именно поэтому все содержание игры должно быть интуитивно понятно и делать это нужно путем минимизации интерфейса.

Управление. Если нужно добавить в игру какие-нибудь элементы, то лучше всего поместить их в нижнюю часть экрана (другими словами, под контент). Сравните это с компьютерным интерфейсом: там, как правило, меню управления находится вверху, но не стоит забывать, курсор, которым ведется управление с помощью мышки, очень маленьких размеров, в отличие от того, что, например, играя на телефоне, можно загородить своим пальцем весь обзор, если потянуться им в самый верх экрана. Поэтому нижняя часть экрана – самое лучшее место для кнопок управления.

Прокрутка. По поводу прокрутки в играх можно сказать только одно: ее нужно стараться избегать, так как это неудобно и путает пользователей. Например, при выборе уровней в игре лучше сделать первые 9 на одной странице, а последующие на других, нежели все делать на одной странице с прокруткой.

Навигация по игре. Есть огромное количество моделей для навигации в играх, однако если была выбрана одна из уже существующих моделей, а не придумывается что-то уникальное, то стоит потратить немного времени, чтобы выбрать именно ту модель, которая наиболее подходит для разрабатываемой игры [3]. Одними из самых распространённых моделей являются:

- *панель вкладок.* Хорошая модель при использовании от трех до шести вкладок, чтобы разделить между собой содержание приложения. Яркий пример: Twitter для iPhone (Для игр эта модель также используется);

- *список.* Подробный список (перечень) всего того, что содержит в себе конкретная игра или приложение. Примером может послужить стандартное приложение «Настройки»;

- *None*. В данном случае, пожалуй, переводить тоже не стоит, иначе будет не так красиво. Эта модель заключается в том, что игра содержит только один экран, без всевозможных кнопок, вкладок, списков и т.д. Например, игра Fruit Ninja.

Ввод информации. Необходимо сделать все возможное, чтобы ввод информации был как можно проще и удобнее. Существует около десятка различных клавиатур (Default, Email, URL, Phone и т.п.), нужно убедиться, что когда пользователь хочет ввести информацию в то или иное поле в разрабатываемой игре, то перед ним автоматически появится нужная ему клавиатура. Пользователь не должен тратить время на то, чтобы переключать и искать более удобный способ ввода – это должно происходить само собой. Если в разрабатываемой игре нужно много печатать, то нужно убедиться, что игра поддерживает альбомную ориентацию, иначе это будет неудобно и сопровождаться определенными трудностями.

Ориентация. На сегодняшний день портретная ориентация – это самая популярная ориентация, поэтому именно под нее нужно оптимизировать свою игру в первую очередь. Как уже было сказано, если игра подразумевает, что пользователь будет много печатать, то она обязательно должна поддерживать альбомную ориентацию и, само собой, доступ к клавиатуре должен быть максимально быстрым.

Жесты должны быть невидимы. Жесты, в отличие от кнопок, являются невидимыми, что становится некоторой проблемой. Управление жестами – это хорошо, но нужно решить, как раскрыть своим пользователям существование этих самых жестов. Но сделать это надо так аккуратно, чтобы это не мешало пользователям, но в то же время они должны это запомнить.

Multi-touch. Яркий пример того, что такие жесты очень удобны – это любое картографическое приложение на iOS. Однако если одна рука у пользователя чем-то занята, то другой свободной рукой он не сможет и держать смартфон, и выполнять жест (например, для увеличения масштаба), поэтому об этой проблеме нужно также серьезно подумать. Должна быть какая-то альтернатива, так как ситуации бывают разные.

Жесты нужны, но не принципиальны. Приятно, когда с помощью жестов можно управлять игрой. Но если их нет – это совершенно не критично. Жесты – это как сочетания клавиш: опытным пользователям они нравятся, а большинство людей даже не знают об их существовании.

Должна быть альтернатива жестам. К сожалению, так называемого «словаря жестов» еще не существует. Еще не пришло время, чтобы во всех приложениях и играх было управление только лишь с помощью одних жестов. В любом случае, в игре должны быть такие элементы управления, которыми можно воспользоваться с помощью одного пальца.

Заключение. Подходов для разработки пользовательского интерфейса существует множество, возможностей изучать поведение пользователя становится с каждым днем все больше. И чем больше внимания разработчик будет уделять степени удобства продукта на всех этапах его разработки, тем более качественным он будет на выходе.

ЛИТЕРАТУРА

1. Информационный ресурс [Электронный ресурс] / Особенности проектирования интерфейсов в мобильных играх. – Режим доступа: http://app2top.ru/game_development/osobennosti-proektirovaniya-interfejsov-v-mobil-ny-h-igrah-81668.html. – Дата доступа: 29.09.2016.
2. Информационный ресур [Электронный ресурс] / Навигация в игре: Как организовать систему подсказок и не отвлечь пользователя от сюжета. – Режим доступа: https://vc.ru/p/environmental-narratives/. – Дата доступа: 29.09.2016.
3. Виктор Корсун, ZeptoLab. Принципы построения игровых интерфейсов в мобильных играх [Электронный ресурс]. – Режим доступа: http://www.slideshare.net/Apps4All/ss-20009701/. – Дата доступа: 29.09.2016.

УДК 004.658

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ КАРТЫ ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

В.В. МАГЕРОВ
(*Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН*)

Представлен объект разработки – административное приложение для управления информационным содержимым карты Полоцкого государственного университета. Рассмотрены вопросы проектирования базы данных для информационной системы карты рассматриваемого университета.

Цель работы – создание информационной системы Полоцкого государственного университета с упором на отображение информации на карте.

Для создания административного приложения, работающего с базой данных, необходимо сначала спроектировать саму базу. Для удобства последующей разработки проектируемая база должна быть максимально проста для понимания и нормализована.

Для проектирования базы данных необходимо выделить сущности, которые будут использоваться в информационной системе. В ходе анализа предметной области, которая определяется широтой всей информации об университете, которая может понадобиться кому-нибудь для ознакомления, были выделены следующие основные сущности:

- корпуса университета (housing);
- общежития университета (hostel);
- города, в которых расположены общежития и корпуса (city);
- факультеты университета (faculty);
- кафедры университета (department);
- общие отделы и службы университета (univ_services);
- общественные организации университета (public_organizations);
- люди, стоящие во главе университета, факультета, кафедры, службы или общественной организации (people);
- места для досуга студентов, такие как столовая, тренажёрный зал, библиотека и другие (leisure);
- места, имеющие историческую или культурную ценность (history_culture_places).

Соответственно, администратор должен редактировать всю информацию об этих сущностях, хранящуюся в базе данных. Исходя из этого перечислим функции администратора:

- добавление/удаление/редактирование информации о городах, в которых расположены здания университета;
- добавление/удаление/редактирование информации об общежитиях;
- добавление/удаление/редактирование информации о корпусах университета;
- добавление/удаление/редактирование информации о факультетах университета;
- добавление/удаление/редактирование информации о кафедрах университета;
- добавление/удаление/редактирование информации об общих отделах и службах университета;
- добавление/удаление/редактирование информации об общественных организациях университета;
- добавление/удаление/редактирование информации о людях, связанных с университетом и его подразделениями;
- добавление/удаление/редактирование информации о местах досуга, расположенных в университете;
- добавление/удаление/редактирование информации о местах, которые имеют историческую или культурную ценность и связаны с университетом.

Для построения схемы базы данных необходимо определить совокупность отношений между сущностями, которые будут использоваться в базе. Также важно, чтобы в отношениях между сущностями выполнялись условие целостности сущностей и условие целостности ссылок.

Условие целостности сущностей заключается в следующем: каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения, то есть любое отношение должно обладать первичным ключом, а условие целостности ссылок, в свою очередь, заключается в том, что для каждого значения внешнего ключа, появляющегося в дочернем отношении, в родительском отношении должен найтись кортеж с таким же значением первичного ключа [2].

Схема спроектированной базы данных представлена на рисунке.

Выбор системы управления базами данных (СУБД) является одним из самых важных этапов разработки системы. Среди всего множества доступных СУБД выбор был сделан в пользу SQLite – компактной встраиваемой реляционной базы данных, так как её возможностей будет вполне достаточно для

разрабатываемой системы: нет необходимости использовать клиент-серверные системы, ибо разрабатываемая система используется как самостоятельное приложение со встроенной базой данных.

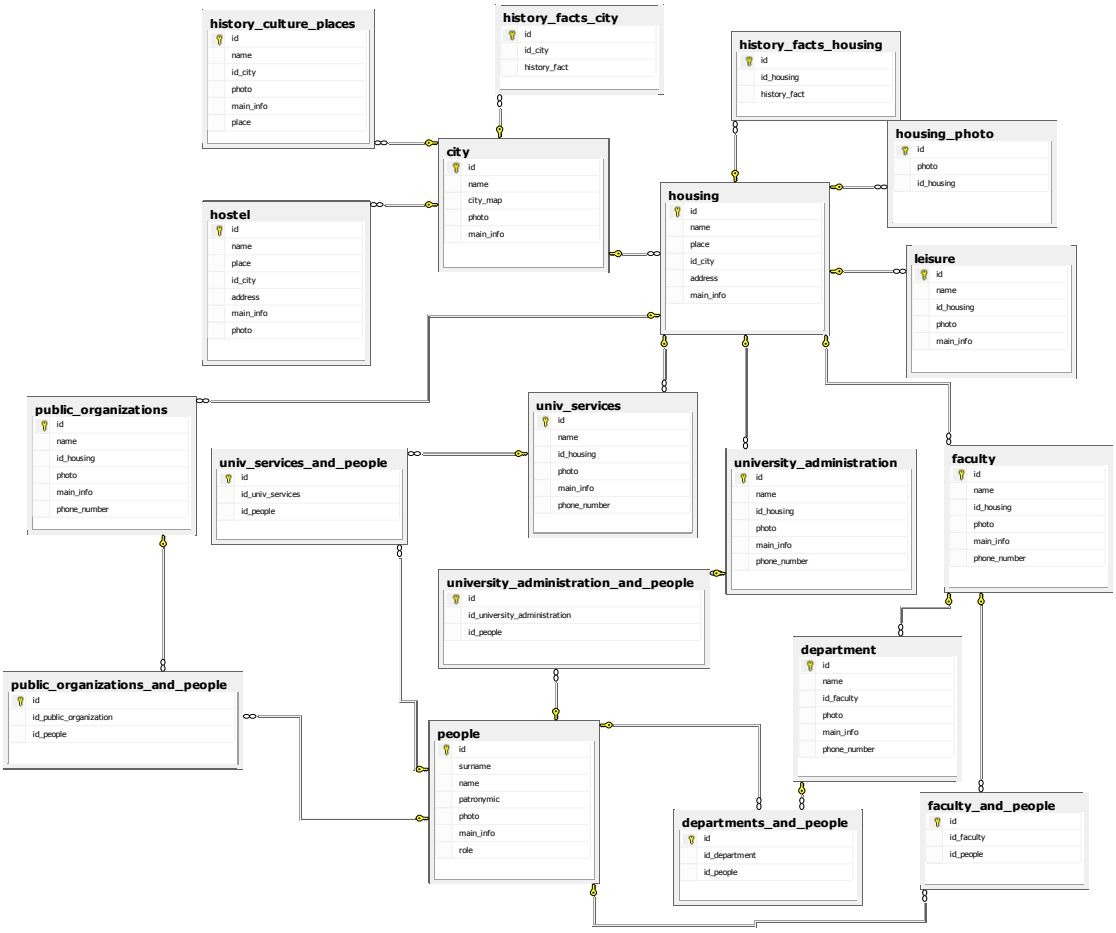


Схема спроектированной базы данных

Среди основных преимуществ SQLite можно выделить такие, как надёжность, производительность, возможность обработки и хранения больших объёмов данных, а также отсутствие необходимости настройки сервера СУБД [3].

Для правильной работы и сохранения ссылочной целостности для сущностей в базе были разработаны триггеры на добавление/удаление/редактирование, которые предотвращают добавление в базу одинаковых записей и организуют каскадное удаление, не оставляя неактуальные ссылки.

Заключение. В результате проделанной работы спроектирована информационная база данных Полоцкого государственного университета. Спроектированная база может основой для создания информационной системы для карты Полоцкого государственного университета.

ЛИТЕРАТУРА

1. Официальный сайт СУБД SQLite [Электронный ресурс]. – Режим доступа: https://www.sqlite.org. – Дата доступа: 29.09.2016.
2. Сайт Мурманского государственного технического университета [Электронный ресурс]. – Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/ch_4_3.html. – Дата доступа: 29.09.2016.
3. Что такое SQLite. Плюсы и минусы [Электронный ресурс] / Заметки WEB разработчика. – Режим доступа: http://webnotes.by/docs/sql/259. – Дата доступа: 29.09.2016.

УДК 004.451.83

РАЗРАБОТКА АДМИНИСТРАТИВНОГО ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ КАРТОЙ ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

В.В. МАГЕРОВ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматривается административное приложение для управления информационным содержимым карты Полоцкого государственного университета. Создание данной информационной системы осуществлено с упором на отображение информации на карте. Пранализированы вопросы проектирования и разработки административного приложения для работы с базой данных.

Целью разработки административного приложения для карты Полоцкого государственного университета является создание удобного приложения с компактным интерфейсом, с помощью которого можно легко работать с базой данных, обрабатывать возможные ошибки или совсем избегать их.

Для написания приложения была выбрана платформа .NET Framework версии 4.5. такой выбор обусловливается тем, что эта платформа является основной платформой для операционных систем семейства Windows, самого распространённого семейства операционных систем (ОС) в мире. К тому же большинство людей визуально привыкли к ОС Windows, и им будет проще воспринимать приложение, написанное для этой ОС.

Для непосредственной разработки приложения выбрана интегрированная среда разработки Microsoft Visual Studio 2013. Она позволяет создавать различного рода приложения, а также применять различные современные технологии данной платформы. Среда имеет удобный редактор кода и визуальную среду для разработки графического интерфейса.

Среди всех языков программирования выбран язык программирования C#. Выбор был сделан исходя из его преимуществ, а именно: подлинная объектная ориентированность; безопасный (по сравнению с языками C и C++) код; поддержка событийно-ориентированного программирования; «родной» язык для приложения на платформе .NET; объединение лучших идей современных языков [1].

Для разработки интерфейса приложения использована графическая подсистема в составе .NET Framework – технология Windows Presentation Foundation – система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, использующая язык XAML. В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом современного графического оборудования [2].

WPF обладает следующими преимуществами: независимость от разрешения экрана; хорошее взаимодействие с WinForms; аппаратное ускорение графики; возможность декларативного определения графического интерфейса с помощью XAML [3].

Для взаимодействия приложения с базой данных используется специальный драйвер для платформы .NET для работы с SQLite, который поставляется вместе с самой базой данных от её разработчиков.

Для работы с базой данных были реализованы классы, которые отражают сущности, хранящиеся в базе данных. В них присутствуют необходимые поля и методы, благодаря которым осуществляется «общение» с базой.

Создание графических форм приложения начинается с определения их размера и компонентного содержимого формы, затем осуществляется создание обработчиков событий, которые могут произойти при взаимодействии администратора с различными компонентами формы.

В разрабатываемом приложении имеются следующие компоненты: *TabControl* – связанный набор страниц вкладок; *Button* – командная кнопка; *ComboBox* – представляет собой комбинацию поля редактирования и списка, что дает возможность ввести данные путем набора на клавиатуре или выбором из списка; *StackPanel* – представляет собой контейнер для других компонентов; *Label* – предназначен для отображения текстовой информации; *TextBox* – предназначен для ввода данных с клавиатуры; *Grid* – контейнер для других компонентов; другие компоненты.

При разработке был сделан упор на удобство и простоту для будущего пользователя. Элементы расположены так, чтобы пользователю было сразу понятно, для чего они там и как с ними работать. Например, элементы типа *Label* размещены рядом с соответствующими им компонентами, такими как *TextBox* и *ComboBox*, и пользователю сразу становится понятно, как эти элементы связаны и как нужно с ними работать.

Приложение состоит из главного окна и набора дочерних окон. На главном окне располагаются вкладки для работы с наиболее часто используемыми сущностями, хранящимися в базе данных: персонал, факультеты, кафедры, общественные организации, общие отделы и службы. Внешний вид главного окна можно увидеть на рисунке 1.

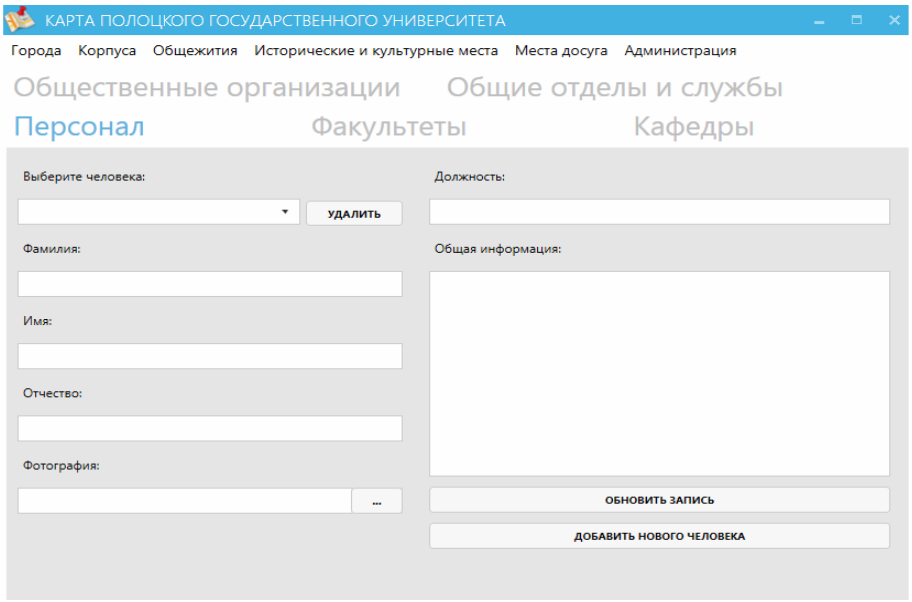


Рисунок 1. – Главное окно административного приложения

Окна для работы с остальными сущностями можно вызвать посредством их выбора в верхнем меню. Там расположены: город, корпуса, общежития, исторические и культурные места, места досуга и администрация. Такой подход упрощает работу с приложением, делая более важные и часто используемые сущности более доступными. К примеру, внешний вид окна для работы с корпусами представлен на рисунке 2.

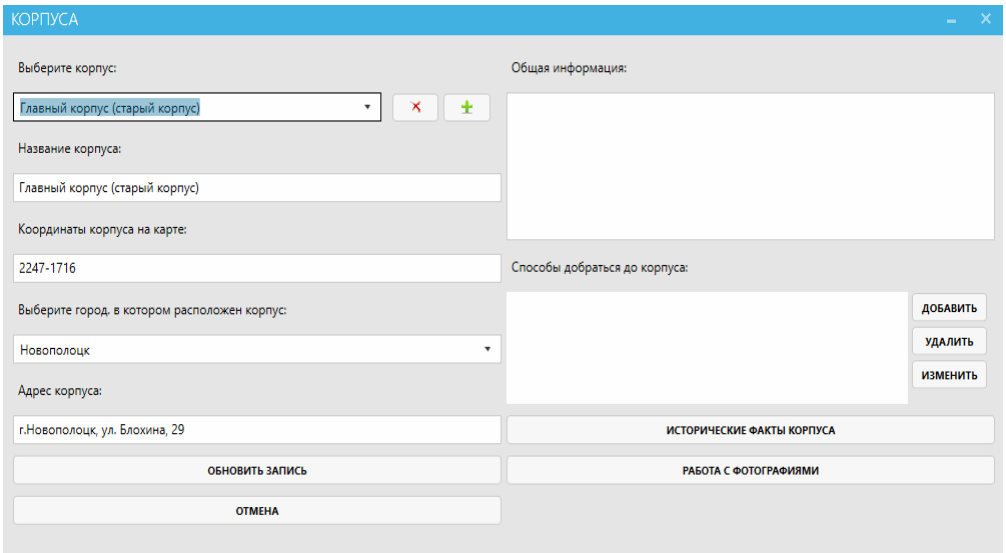


Рисунок 2. – Окно работы с корпусами университета

Заключение. Спроектирована и реализована административная часть для карты Полоцкого государственного университета. Для административного приложения разработан удобный для работы интерфейс, который не требует дополнительного обучения и интуитивно понятен, а также реализована система информирования об ошибках или нестандартных ситуациях, которые помогают администратору в работе.

ЛИТЕРАТУРА

1. Языки программирования [Электронный ресурс]. – Режим доступа: http://www.uzluga.ru/potrd/Программа+последовательность+инструкцийd/part-7.html. – Дата доступа: 29.09.2016.
2. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org. – Дата доступа: 29.09.2016.
3. WPF и C# [Электронный ресурс]. – Режим доступа: http://metanit.com/sharp/wpf/. – Дата доступа: 29.09.2016.

УДК 004.457

ПОДХОДЫ К РАЗРАБОТКЕ HTTP-СЕРВЕРА ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ АНДРОИД С НЕБОЛЬШИМ ПОТРЕБЛЕНИЕМ РЕСУРСОВ

А.А. ЮШКЕВИЧ
(Представлено: Е.Р. СУХАРЕВ)

Рассматривается подход к проектированию серверной части андроид-приложения для взаимодействия с компьютером по протоколу HTTP. Рассмотрена легковесная библиотека NanoHTTPD для создания HTTP-сервера, а также описаны важные моменты проектирования архитектуры приложения.

В настоящее время компьютеры получили широкое распространение. Причём у большинства людей есть как персональные компьютеры или ноутбуки, так и различные смартфоны, планшеты, использующие операционную систему Android. И здесь появляется вопрос, как можно легко осуществить какое-либо взаимодействие (передать файлы, экспортировать контакты и т.д.) между различными устройствами. В данной ситуации существует огромное множество способов, которыми можно решить поставленную задачу. Например, данные можно передать с помощью технологии bluetooth, которую поддерживает большинство современных смартфонов. Также необходимая информация может быть перенесена с помощью специального кабеля. Такой способ передачи обеспечит максимальную скорость.

Ещё файлы можно переписать с помощью протокола FTP или специальных программ, которые включают в себя клиентскую и серверную часть.

Все эти способы способны решить поставленную задачу. Однако под рукой может оказаться кабель с неподходящим разъёмом, bluetooth слишком медленный, а с помощью FTP можно передавать только файлы и при попытке расширения функционала можно столкнуться с проблемами. Поэтому стоит задуматься об универсальном аналоге, который сможет решать задачу быстро, без дополнительного оборудования и помимо передачи файлов выполнять другие функции.

Средства решения задач. Широкое распространение таких приложений, как браузеры, позволяет говорить о веб-приложениях как о кроссплатформенных. Например, данный подход можно использовать с целью создания интерфейса между компьютером и мобильным телефоном на базе андроид. Однако в связи с тем, что на телефоне ресурс существенно ограничен, попытка переноса полноценного сервера не всегда может оказаться эффективной. Поэтому в данной работе описана такая небольшая библиотека, как NanoHTTPD. Её главным отличием от многих других является минимализм, что важно при разработке под мобильные платформы. Библиотека распространяется в виде открытых исходных кодов. Причём лицензия позволяет её свободно изменять. Это означает, что она может быть оптимизирована под какую-либо конкретную задачу, что может стать огромным плюсом при разработке под устройство, обладающее небольшим количеством ресурсов.

Для использования библиотеки, можно воспользоваться исходными кодами с github, которые распространяются согласно модифицированной лицензии BSD.

Чтобы запустить сам сервер, необходимо создать класс и унаследовать его от NanoHTTPD и вызвать метод start. Для обработки запросов, необходимо переопределить метод *public Response serve (IHTTPSession session).*

IHTTPSession – интерфейс, который включает в себя методы для работы с cookie, получения параметров запроса, заголовка запроса. Также присутствует метод для разбора тела запроса. В библиотеке уже есть готовая реализация данного интерфейса, которая позволяет передавать и разбирать заголовки, параметры. Причём последние могут быть как в закодированном, так и в раскодированном виде. Кроме этого присутствует возможность передавать файлы.

При обработке POST-запросов следует учесть, что разбор параметров происходит не автоматически, а в результате явного выхода метода *void parse Body (Map<String, String> files) throws IOException, Response Exception.*

Как можно было заметить, метод serve возвращает такой класс, как Response. Он предназначен для передачи ответа клиенту и может быть создан различными способами, в зависимости от передаваемого содержимого. Чтобы создать класс, можно воспользоваться одним из его статических методов. Например, если необходимо передать поток данных указанного типа, можно воспользоваться методом *static Response new Chunk end Response (IStatus status, String mime Type, Input Stream data).* Это может оказаться эффективным для передачи файлов или иных данных неизвестного размера. Если последний известен,

можно воспользоваться методом *static Response new Fixed Length Response (IStatus status, String mime Type, byte [] data)* для передачи массива байтов, либо *static Response new Fixed Length Response (IStatus status, String mimeType, Input Stream data, long total Bytes)* для пересылки данных из потока.

Если необходимо переслать строку, можно воспользоваться методом *static Response new Fixed Length Response (IStatus status, String mime Type, String txt)* для передачи данных произвольного типа, либо *static Response new Fixed Length Response (String msg)* для отправки информации, которая будет интерпретирована как html-содержимое.

Из важных особенностей NanoHTTPD следует отметить, что данная технология также позволяет работать с cookie. Для того чтобы их получить, необходимо вызвать метод *CookieHandler get Cookies()* у класса, реализующего интерфейс IHTTPSession. Как видно, метод возвращает CookieHandler, который реализует интерфейс Iterable, а сделовательно все куки можно проитерировать с помощью конструкции foreach. Но итерироваться будут имена. Чтобы прочитать значение, необходимо воспользоваться методом *String read (String name).* Для удаления используется метод *void delete (String name),* а для добавления или изменения значения, метод *void set (Cookie cookie),* либо *void set (String name, String value, int expires)* [1].

Кроме описанных возможностей, библиотека может также создавать соединение как по протоколу HTTP, так и по защищенному HTTPS [2].

Во избежание переполнения оперативной памяти, запросы большого размера кэшируются в память устройства. При передаче файлов в виде multipart data запрос кэшируется в память устройства, затем выполняется его разбор, после чего формируется временный файл, который позже можно переместить туда, где он нужен.

Этот момент реализации может оказаться не совсем удачным при использовании мобильной операционной системы потому, что память на устройстве ограничена и создавать много копий файлов не стоит. В связи с этим при реализации мобильного приложения было решено внести изменения в исходный код библиотеки. В частности, модифицирован код, касающийся загрузки и разбора запроса. Теперь файл вместо того, чтобы записываться во временные, записывается сразу по необходимому адресу.

Методы решения задач. Очевидно, что при реализации приложения важно сформировать некоторую архитектуру. Здесь отлично подходит модель MVC. Она представляет собой 3 элемента: модель, представление и контроллер [3].

Модель является некоторыми данными, которые описывают предметную область и над ними могут быть выполнены некоторые операции. Также эта информация может быть предоставлена пользователю на графическом интерфейсе.

Представление является этим самым графическим интерфейсом.

Контроллер реализует некоторые алгоритмы для получения данных с представления, обработки их и отправки обратно.

Для работы с моделью удобным может оказаться создание некоторого базового класса или интерфейса, что и было сделано при разработке андроид-приложения. В классы, выполняющие функции моделей, также были помещены методы для работы с JSON [4]. Таким образом, JSON-содержимое может быть легко преобразовано в класс-модель и наоборот.

Для реализации представления используется язык разметки html, содержимое на котором генерируется с помощью javascript и ajax [5].

Учитывая, что для каждой реализуемой функции приложения необходимо разработать свой собственный контроллер, необходимо создать фабрику контроллеров. Входным параметром может быть URI-запроса. Таким образом, контроллер – это класс, наследующий некоторый базовый класс, отвечающий за контроллеры. И все классы-контроллеры лежат в коллекции Map, ключами которой являются URI.

Однако в сами контроллеры помещать весь функционал не стоит, потому что это может привести к загромождению кода, а следовательно и к ухудшению его читабельности. Поэтому при разработке приложения был выделен уровень сервисов, который должен отвечать за выполнение операций на телефоне, таких как получение сведений, модификация и др. Контроллеры же только преобразуют данные для взаимодействия с клиентом и вызывают необходимый метод из сервиса [6].

Исходя из всего сказанного, краткий алгоритм работы разработанного алгоритма работы приложения можно описать следующим образом:

- в методе NanoHTTPD.serve захватывается запрос;
- для get-запроса выполняется попытка найти и отправить подходящий файл;
- для всех остальных запросов выполняется поиск контроллера по URI;
- передача запроса в контроллер;
- разбор запроса;
- преобразование данных из JSON в класс модель;

- выполнение действия, за которое отвечает контроллер;
- преобразование выходных данных из класса-модели в формат JSON;
- создание ответа;
- отправка ответа клиенту.

Заключение. Библиотека NanoHTTPD зарекомендовала себя как минималистичная, легковесная, экономная в плане расходования ресурсов устройства, что важно при разработке программных продуктов для мобильных устройств. К тому же объектно-ориентированный подход к программированию может существенно упростить процесс разработки. В частности, он позволяет избегать повторов в коде. В разработанном приложении это видно в механизмах работы с моделями и контроллерами. Кроме всего прочего, улучшить читабельность кода, а значит облегчить разработку и исправление ошибок, позволяет использование модели MVC, а также паттерна service layer.

ЛИТЕРАТУРА

1. NanoHttpd/nanohttpd: Tiny, easily embeddable HTTP server in Java / Jarno Elonen [Electronic resource]. – 2010. – Mode of access: https://github.com/NanoHttpd/nanohttpd. – Date of access: 21.09.2016.
2. Rescorla, E. HTTP Over TLS / E. Rescorla // RTFM Inc. [Electronic resource]. – 2010. – Mode of access: https://tools. ietf.org/ html/rfc7159. – Date of access: 21.09.2016.
3. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – Москва, Санкт-Петербург, Киев, 2006. – 544 с.
4. Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format / Bray T. // Google Inc. [Electronic resource]. – 2014. – Mode of access: https://tools.ietf.org/html/rfc7159. – Date of access: 21.09.2016.
5. Мархвид, И.В. Создание WEB-страниц: HTML, CSS, JavaScript / Мархвид И.В. – Минск : ООО «Новое знание», 2002. – 350 с.
6. Хоп, Г. Шаблоны интеграции корпоративных приложений / Г. Хоп, Б.Б. Вульф. – М. : Вильямс, 2006. – 672 с.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ ГРАФИЧЕСКОГО ВЕБ-ИНТЕРФЕЙСА НА ПРИМЕРЕ АНДРОИД-ПРИЛОЖЕНИЯ ДЛЯ ВЗАИМОДЕЙСТВИЯ С КОМПЬЮТЕРОМ ПО ПРОТОКОЛУ HTTP

А.А. ЮШКЕВИЧ
(Представлено: Е.Р. СУХАРЕВ)

Анализируются технологии, которые часто используются при проектировании веб-интерфейсов. Приведен пример построения веб-интерфейса андроид-приложения, с помощью которого можно выполнять некоторые действия на мобильном телефоне.

Развитие информационных технологий принесло в наш мир огромное количество электронных устройств. Это компьютеры, ноутбуки, мобильные телефоны, планшеты и т.д. Поэтому при разработке приложений важным является вопрос кроссплатформенности. Часто он решается разработкой одного и того же приложения под разные архитектуры с помощью компиляции в байт-код (например, в языке программирования Java). Однако стоит отметить, что сейчас практически для каждого устройства и операционной системы существует веб-браузер. И самое главное, им пользуются практически все. Поэтому приложение с веб-интерфейсом может быть запущено на большинстве устройств и его можно считать кроссплатформенным.

Средства решения задач. Рассмотрим подходы к проектированию интерфейса на примере разработанного андроид-приложения, которое позволяет установить взаимодействие между мобильным телефоном и компьютером.

Разработка андроид-приложения для взаимодействия с компьютером по протоколу HTTP является актуальной в связи с высокой популярностью мобильных устройств. А большую долю рынка занимает операционная система Андроид.

Взаимодействие между устройствами стоит рассматривать с точки зрения обмена файлами, просмотра и редактирования сообщений и контактов и прочих возможностей. Многие существующие в настоящее время решения могут оказаться платформенно-зависимыми. Даже с использованием кабеля решение задачи может оказаться проблемным при попытке взаимодействия с Linux-дистрибутивом по причине того, что драйвера, необходимые для какого-либо взаимодействия, за исключением обмена файлами, могут оказаться только для операционной системы Windows.

Разработка клиентской части с помощью веб-интерфейса может сделать этот элемент платформенно независимым. То есть для работы с приложением понадобится всего лишь соединение, например, с помощью Wi-Fi, а также веб-браузеры, которых сейчас огромное количество, и они разработаны практически для каждой операционной системы.

Современные технологии позволяют легко сделать интерфейс дружелюбным. При его разработке было решено использовать jQuery, Bootstrap и ajax.

При разработке интерфейса было решено, что бóльшая часть информации будет загружаться не при загрузке страницы, а по мере необходимости. Это необходимо для уменьшения объёма передаваемых данных.

Для облегчения работы с javascript была применена библиотека jQuery [1]. Главное её преимущество состоит в том, что разрабатывать с её помощью очень просто, так как можно использовать селекторы для выбора интересующих элементов, события. Также код получается намного короче, что позволяет повысить его читабельность, и в случае возникновения ошибок они могут быть быстрее обнаружены. А самое главное то, что разработчик может разрабатывать не задумываясь, под каким браузером код будет исполняться. Ведь кто-то использует Microsoft Edge, а кто-то браузеры на WebKit, такие как Opera, Google Chrome и т.д. У каждого из них есть свои особенности, которые должен учитывать разработчик, разрабатывая на «голом» javascript [2].

Bootstrap – это фреймворк, который разработала Twitter Inc. Он предназначен для облегчения построения графического интерфейса. Сама библиотека включает в себя огромное количество элементов, таких как кнопки, веб-формы, блоки навигации, и многое другое. Конечно, если изначально ничего не менять, то полученная форма будет похожа на то, что получается у многих других разработчиков, использующих аналогичный фреймворк, но следует учитывать, что на ранних этапах разработки может быть

получен достаточно аккуратный интерфейс, и в итоге его всё равно можно настроить. Исходные коды распространяются и лицензией MIT, что позволяет свободно использовать и изменять технологию [3].

Технология ajax предназначена для выполнения запросов на сервере с целью получения динамического контента, не перезагружая всю страницу. Это приводит к тому, что запросы выполняются быстрее, а соответственно, улучшается реакция на действия пользователя. В настоящее время эта технология применяется практически везде. Многие популярные приложения используют XMLHttpRequest [4]. Очень часто в качестве примера приводят сервисы Google. Например, Google Maps получает необходимую информацию налету, едва пользователь куда-либо нажмёт [5].

Проектирование интерфейса. При проектировании интерфейса решено сформировать несколько страниц, на каждой из которых реализуется какой-то конкретный функционал, а весь контент генерируется динамически посредством ajax-запросов.

Таким образом, сформированы страницы для работы с файлами, контактами, сообщениями, приложениями и фотографиями. При необходимости вывода какой-либо вспомогательной информации использовались модальные окна. Код самого окна заранее помещается на веб-страницу, но оно скрыто. При необходимости просмотра какой-либо информации, например списка сообщений какого-либо пользователя или подробной информации о контакте, выполняется ajax-запрос на сервер, где можно получить необходимую информацию. Затем она помещается в поле модального окна, после чего окно становится видимым [6].

Следует отметить, что в подобный элемент можно помещать progress bar для отображения состояния загрузки файлов на сервер.

Чтобы пользователь мог сразу видеть все возможности приложения, все кнопки действий расположили в левом столбце, а переключение между вкладками поместили в верхней части. Во вкладках было решено разделять глобальные действия, такие как работа с файлами, работа с сообщениями, работа с контактами, а также возможность работать с приложениями и фотографиями. Причём за каждую вкладку отвечает своя страница.

На рисунке 1 представлен пример формы для работы с файлами.

Рисунок 1. – Разработанный интерфейс на примере формы для работы с файлами

Как было сказано выше, располагаются вкладки для переключения между основными функционалами, а слева – кнопки для выполнения действий над текущей страницей (выбрать всё, снять выделение, обновить) или над файлами (загрузка файла, загрузка папки, скачивание папки, скачивание файла, создание папки, удаление файла или папки).

В основной части вверху находится строка, указывающая, в каком каталоге сейчас находится пользователь. Ниже изображена таблица, в которой содержится перечень файлов и папок в текущем каталоге, а также информация о них (тип, дата последнего изменения, размер файла). При загрузке файлов на сервер появляется модальное окно, отображающее прогресс.

При щелчке по строке, выполняется переход в папку, либо скачивание файла. При щелчке по checkbox-у, выполняется выделение или снятие выделения элемента. Выделенные элементы подсвечиваются.

Для построения панели с именами вкладок, функциональных кнопок в левой части, таблицы и для реализации эффекта чередующегося цвета строк в ней применялся фреймворк Bootstrap. Для подсвечивания отмеченных строк были использованы функции jQuery.

Для перехода по папкам информация собиралась с сервера без перезагрузки страницы с использованием ajax. Также с его помощью вызывается выполнение функций, возложенных на кнопки.

Остальные страницы разработаны в аналогичном стиле.

Заключение. Проведенное исследование позволяет сделать вывод, что правильным подходом при разработке веб-интерфейса является использование фреймворков, которые позволяют выполнять свои задачи во всех браузерах. Это значительно экономит время при разработке интерфейса, которое можно потратить на реализацию функциональной части.

Предложен процесс построения интерфейса и динамическая обработка запросов без перезагрузки страницы с помощью фреймворков jQuery, Bootstrap и технологии ajax.

ЛИТЕРАТУРА

1. jQuery / The jQuery Foundation [Electronic resource]. – 2016. – Mode of access: https://jquery.com. – Date of access: 20.09.2016.
2. W3Schools Online Web Tutorials / Refsnes [Electronic resource]. – 1999–2016. – Mode of access: http://www.w3schools.com. – Date of access: 20.09.2016.
3. Bootstrap / Twitter Inc. [Electronic resource]. – 2016. – Mode of access: http://www.w3schools.com. – Date of access: 20.09.2016.
4. XMLHttpRequest Level 1 / W3C [Electronic resource]. – 2014. – Mode of access: http://www.w3schools.com. – Date of access: 20.09.2016.
5. AJAX – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikibooks.org/wiki/AJAX. – Дата доступа: 20.09.2016.
6. Модальное окно Bootstrap Modal / htmlhook.ru [Электронный ресурс]. – 2016. – Режим доступа: http://htmlhook.ru/modal-popup-bootstrap-3.html. – Дата доступа: 20.09.2016.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ ГРАФИЧЕСКОГОВЕБ-ИНТЕРФЕЙСА НА ПРИМЕРЕ АНДРОИД-ПРИЛОЖЕНИЯ ИВЕБ-САЙТА КУЛИНАРНЫХ РЕЦЕПТОВ

А.В. ОСИПОВА
(Представлено: А.Ф. ОСЬКИН)

Рассматривается технология построения интерфейса при проектировании мобильного приложения. Приведен пример построения многоэкранного интерфейса Андроид-приложения для синхронизации с веб-сайтом.

Рассмотрим процесс построения интерфейса для веб-сайта и мобильного приложения с синхронизацией контента из базы данных MySQL.

Интернет технологии плотно вошли в повседневную жизнь каждого человека. Каждый день мы используем различные девайсы, такие как персональные компьютеры, ноутбуки, смартфоны и т.д. Они развиваются с каждым днем все больше, охватывая разные стороны нашей жизни: социальные сети, разнообразные приложения, календари, напоминания, журналы контроля, веб-сайты и т.д. При таком ежедневном использовании возникает необходимость в приятном и дружественном интерфейсе, который не требует дополнительных специальных знаний от пользователя. Ранее пользователи могли работать с Интернет-ресурсами только через персональные компьютеры или ноутбуки, но с развитием смартфонов и планшетов все необходимое остается под рукой 24 часа в сутки. Большинство девайсов работает на операционной системе Android.

Данная операционная система дает разработчику возможность создания разнообразных интерфейсов для мобильных приложений, а специальные платформы для разработки и настройки веб-сайтов позволяют не только создавать свои уникальные интерфейсы веб-сайтов, но и использовать или настраивать готовые шаблоны.

С уровнем современных сред разработки интерфейсов приложения также могут быть адаптивными, т.е. автоматически подстраивать интерфейс одного и того же приложения для работы как на смартфоне, так и на планшете либо другом устройстве, работающем на этой же операционной системе.

Средства решения задач. При выборе средства разработки следует учитывать такие факторы, как клиент-серверный вариант работы, защита данных от несанкционированного доступа, способ хранения данных, наличие единого набора средств, предназначенного для разработки новых прикладных решений и изменения прикладных решений при внедрении программ, универсальность функциональности, производительность системы при одновременной работе пользователей и др. В качестве технологии разработки определена платформа WordPress 4.4.2, за серверную часть отвечает OpenServer 5.2.2.0 для веб-сайта, и AndroidStudio 2.0 для разработки мобильного приложения.

WordPress – система управления содержимым веб-сайта с открытым исходным кодом; написана на PHP; сервер базы данных – MySQL; выпущена под лицензией GNUGPL версии 2. Сфера применения – от блогов до достаточно сложных новостных ресурсов и интернет-магазинов. Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать проекты широкой функциональной сложности [1].

WordPress предназначен для установки на локальном веб-сервере, что предоставляет полный контроль над блогом, не стоит забывать и о простоте его установки. Также WordPress использует контроль на уровне пользователей при определении пользовательских прав. WordPress использует шаблоны для генерации динамических страниц. Что позволяет управлять отображением содержания путем редактирования шаблонов с помощью инструмента «Редактор шаблонов» и «Теги шаблонов». В систему встроен редактор файлов, который можно использовать для редактирования шаблонов и связанных с ним файлов, прямо в браузере, без необходимости скачивать их для редактирования или иметь к ним доступ по FTP. Предоставляется возможность расширить функциональность ядра блога или веб-сайта. Большое количество плагинов доступно для скачивания в разделе Plugins. После установки и активации плагина в панели администратора, он сразу же начинает взаимодействовать с системой, выполняя только те задачи, которые на него возложены.

OpenServe – это портативная серверная платформа и программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администриро-

ванию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

Хотя изначально программные продукты, входящие в состав комплекса, не разрабатывались специально для работы друг с другом, такая связка стала весьма популярной среди пользователей OCWindows [2].

Прежде всего, OpenServer – это полностью портативный сервер. Его можно переносить на другие информационные ресурсы без дополнительной установки программного обеспечения. В случае отсутствия на компьютере нужных системных компонентов OpenServer установит их сам, достаточно выбрать в меню «Инструменты» – «Первый запуск» если сервер запускается на компьютере впервые.

AndroidStudio – это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции GoogleI/O.AndroidStudio, основанная на программном обеспечении IntelliJIDEA от компании JetBrains, официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OSX и Linux [3].

Проектирование интерфейса. При проектированииинтерфейсамобильного приложения и веб-сайта было решено сформировать несколько страниц, на каждой из которых реализуется свой функционал, а контент загружается с базы данных после отправки соответствующего запроса.

При разработке веб-сайта был выбран подходящий по теме шаблон. Который впоследствии был полностью настроен при помощи плагинов и дополнительных модификаций кода в редакторе WordPress. Данный способ значительно сокращает время на разработку веб-сайта, что позволяет акцентировать все внимание на мобильно приложении.

Для разработки мобильного приложения с разнообразными функциональными характеристиками, лучшим решением является разработка многоэкранного Андроид-приложения. Каждый новый экран (activity) содержит свой функционал, если есть необходимость в дублировании, то используются разные фрагменты на одном activity.

Таким образом, для веб-сайта были сформированы страницы для работы с рецептами, записями, коллекциями, пользовательскими формами и настройками. При просмотре контента веб-сайта сначала выполняется запрос на сервер, где получается необходимая информация и идентификатор записи в базе и ссылка для изображения. Полученные данные подгружаются в отведенных для них metabox, после чего запрашиваемая информация отображается на странице веб-сайта. Андроид-приложение состоит из «экранов» рецептов, списков, настроек, авторизации и меню. Для подгрузки контента из базы данных веб-сайта используются запросы-обращения к .php файлу конфигураций с Json-объектами.

Чтобы интерфейс был интуитивно понятным и дружественным, решено использовати минимальное, но информативное количество функциональных элементов, что даст пользователю возможность не путаться в расположении информации и конкретно определит возможности используемого программного приложения.

Например, главная страница веб-сайта содержит:

- баннер;
- главное меню;
- строку поиска; небольшое количество «свежих» рецептов;
- список «свежих» статей;
- footer.

Главное меню представляет собой стационарную строку меню, находящуюся над баннером, которая содержит основные пункты меню (главная страница, рецепты, статьи, избранное, обратная связь, о сервисе, вход), второе пользовательское меню размещено в виджетах. «Свежие» записи и рецепты представляют собой последние добавленные записи для веб-сайта.

В мобильном приложении после авторизации пользователь перейдет на страницу главного меню приложения. В нём были выделены следующий пункты меню: избранные рецепты, поиск, проверенные рецепты, купить, выход, настройки. Над колонкой меню будет располагаться баннер с названием веб-сайта. На страницу меню при необходимости можно свернуть с любой страницы приложения.

На рисунках 1 и 2 представлен пример главной страницы веб-сайта и меню для мобильного приложения.

Для веб-сайта при щелчке по пункту меню, выполняется переход на следующую и задействованная вкладка подсвечивается. Для Андроид-приложения в кладке «Купить» при щелчке по checkbox-у, выполняется выделение или снятие выделения элемента списка покупок, ранее занесенных пользователем. Выделенные элементы подсвечиваются.

Остальные страницы разработаны в аналогичном стиле и принципом действия.

Таким образом, можно сделать *вывод*, что для более успешной реализации интерфейса программных приложений стоит строго разграничивать страницы/экраны и их функционал. Это позволит сделать интерфейс более понятным и дружественным для любого пользователя, вне зависимости от

его познаний в области веб-программирования. Также значительно сократит время при разработке интерфейса, поиске ошибок и дальнейшем его развитии.

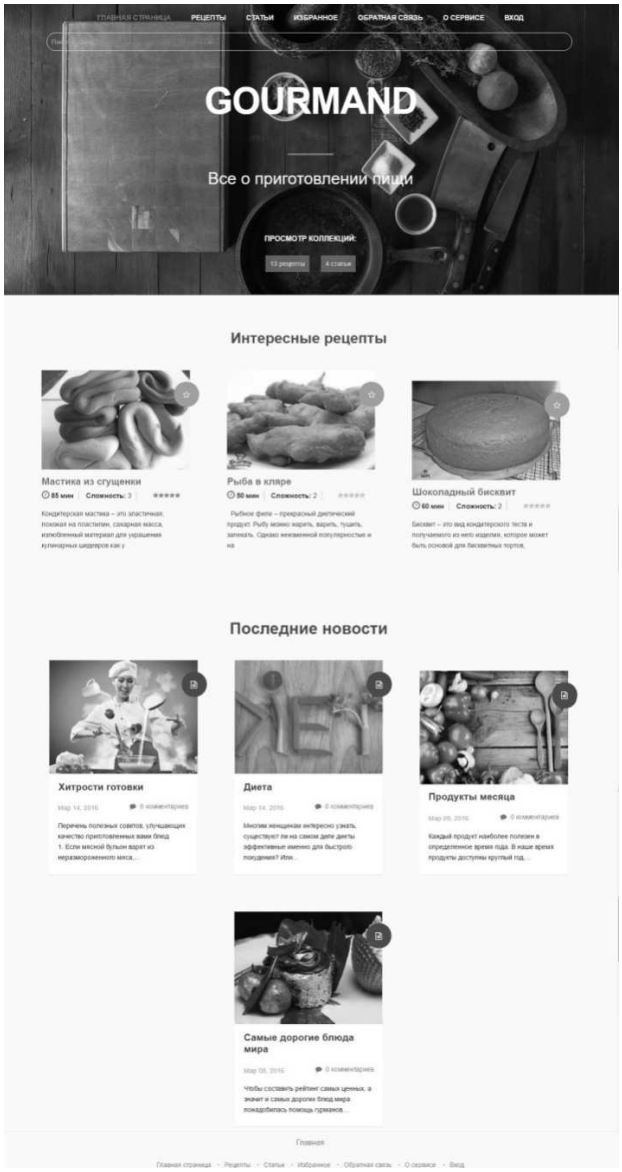


Рисунок 1. – Разработанный интерфейс на примере главной страницы веб сайта

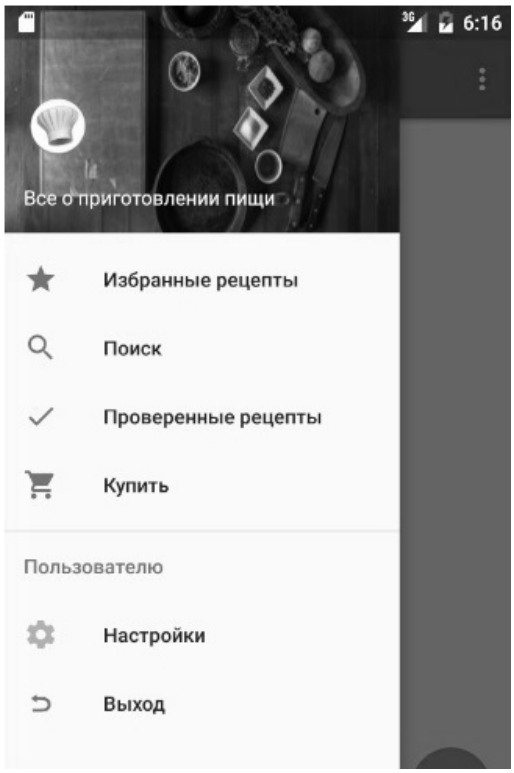


Рисунок 2. – Разработанный интерфейс на примере меню для мобильного приложения

ЛИТЕРАТУРА

1. WordPress [Электронный ресурс] – Википедия, свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/WordPress. – Дата доступа: 29.09.2016.
2. OpenServer [Электронный ресурс]. – Официальный веб-сайт. – Режим доступа: http://open-server.ru/. – Дата доступа: 29.09.2016.
3. AndroidStudio [Электронный ресурс] / Википедия, свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/Android_Studio. – Дата доступа: 30.09.2016.

УДК 004.457

ПОДХОД К ВЗАИМОДЕЙСТВИЮ АНДРОИД-ПРИЛОЖЕНИЯ С БАЗОЙ ДАННЫХ ВЕБ-САЙТА

А.В. ОСИПОВА
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматривается подход взаимодействия серверной части Андроид-приложения с базой данных MySQL веб-сайта. Представлен один из вариантов подключения Андроид-приложения к базе данных путем комбинирования JSON + PHP + MySQL для создания соединения.

Интернет технологии стали неотъемлемой частью нашей повседневной жизни. Они развиваются с каждым днем все больше, охватывая разные её стороны: социальные сети, разнообразные приложения, календари, напоминания, журналы контроля, веб-сайты и т.д. Ранее пользователи могли работать с Интернет-ресурсами только через персональные компьютеры или ноутбуки, но с развитием смартфонов и планшетов все необходимое остается под рукой 24 часа в сутки. Большинство девайсов работает на операционной системе Android.

При использовании сети Интернет любой пользователь может посещать разнообразные сайты и по желанию регистрироваться там. Важным показателем «современного» веб-сайта является возможность поддержки его мобильной версией или приложением. Но из этого вытекает вопрос, как синхронизировать данные базы веб-сайта и мобильного приложения.

Большинство веб-сайтов написаны на языке PHP и работают с базой данных MySQL, а мобильное приложение может быть разработано на языке Java через среду разработки AndroidStudio. И для того чтобы синхронизировать данные веб-сайта с мобильным приложением можно применить связку передачи данных: из MySQL через JSONобъекты в AndroidStudio.

Решение задачи. Очень часто возникает ситуация, когда для разработанного веб-сайта нужно создать мобильное приложение. В основном мобильные приложения, синхронизированные с веб-сайтами, нуждаются в авторизации пользователей и минимальном, но достаточном количестве данных для работы, так как характеристики смартфоном ниже, чем у персональных компьютеров или ноутбуков. Для этого разработчику необходимо продумать функциональную часть приложения, его интерфейс и способ получения данных из базы данных.

Рассмотрим способ подключения базы данных MySQL через PHP файл JSON запросами с использованием библиотеки Volley.

PHP (*PHP: Hypertext Preprocessor* – «PHP: препроцессор гипертекста»; первоначально *Personal Home Page Tools* – «Инструменты для создания персональных веб-страниц») – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов [1].

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой SunMicrosystems, которая ранее приобрела шведскую компанию MySQLAB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией [2].

JSON (англ. *Java Script Object Notation*) – текстовый формат обмена данными, основанный на Java Script. Как и многие другие текстовые форматы, JSON легко читается. За счёт своей лаконичности по сравнению с XML, формат JSON может быть более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, то в таком ключе он уместен в задачах обмена данными как между браузером и сервером (AJAX), так и между самими серверами (программныеHTTP-сопряжения).

Поскольку формат JSON является подмножеством синтаксиса языка JavaScript, то он может быть быстро десериализован встроенной функцией eval(). Кроме того, возможна вставка вполне работоспособных JavaScript-функций. В языке PHP, начиная с версии 5.2.0, поддержка JSON включена в ядро в виде функций json_decode()и json_encode(), которые сами преобразуют типы данных JSON в соответствующие типы PHP и наоборот [3].

Volley это HTTP библиотека, которая делает сетевой обмен Android приложений более легким, и самое главное, более быстрым. Volley доступен для использования через открытое AOSP хранилище.

Библиотека Volley обладает следующиит преимуществами:

- автоматическое планирование сетевых запросов;
- несколько параллельных сетевых подключений;

- прозрачное кэширование на диске и в памяти ответов, используя стандартную HTTP когерентность кэша;

- поддержка приоритетов запросов;

- API для отмены запроса. Можно отменить один запрос, или можно установить блокировки или область запросов для отмены;

- простота настройки, например, для повторных попыток и отсрочки;

- строгий порядок, который позволяет легко и правильно заполнить пользовательский интерфейс данными, полученными из сети асинхронно.

- инструменты для отладки и трассировки.

Данная библиотека легко интегрируется с любым протоколами и поставляется с поддержкой сырых строк, изображений и JSON [4].

Чтобы реализовать взаимодействие Андроид-приложения и веб-сайта, необходимо подключить библиотеку *com.mcxiaoke.volley:library:1.0.19* в AndroidStudio в файле *build.grable*. После того как библиотеку подключили, будут доступны для работы два класса *Request Queue* и *Request*, и необходимо подключить использование интернета в файле *Android Manifest.xml*. *Request Queue* используется для отправки сетевых запросов, этот класс можно создавать в любой момент, но, как правило, его создают во время запуска и используют как Singleton. *Request* содержит все необходимые детали для создания вызова Web API. Например, какой метод использовать (GET или POST), данные запроса, responselistener, errorlistener.

Для примера можно рассмотреть способ авторизации пользователя веб-сайта черз мобильное приложение с получением доступа к данным учетной записи.

Чтобы подключиться к базе данных, предварительно нужно создать .php файли в нем прописать функционал получения данных из выбранной БД и вывод их в формате JSON. Впоследствии Андроид-приложение будет обращаться к этому файлу для подключения к базе веб-сайта, отправки и получения данных через URL прописанный в Java коде приложения.

При заполнении полей ввода логина/пароля и отправке их методом *request* в качестве ответа пользователь получает Json объект с оповещением удачной или ошибочной авторизации *jsonObject.*get().

При обработке POST-запросов необходимо помнить то, что разбор параметров происходит не автоматически, а в результате явного выхода метода *protected Map<String, String>getParams() throws Auth Failure Error*.

После успешной авторизации система аналогичным способом обращается к базе данных непосредственно по идентификатору пользователя, предоставляя тем самым доступ к учетной записи. Кроме этого, данная библиотека обеспечивает прозрачность дискового кэширования и кэширования в памяти.

Методы решения задач. В качестве модели архитектуры мобильного приложения и веб-сайта лучше всего использовать модель «пользователь – оболочка – БД – администратор». При такой архитектуре можно производить преобразования одной из частей модели, и изменения для других ее частей будут не столь глобальны. Каждая часть данной модели обладает своими характеристиками, которые задают предметную область, которая может быть отражена в графическом интерфейсе, т.е. в оболочке.

Пользователь реализует некоторые методы для отправки и получения данных. Часть архитектуры как БД – администратор управляет всем содержимым контентом.

Заключение. Для синхронизации данных веб-сайта и мобильного приложения с операционной системой Android удобно использовать связку: MySQL база данных, конфигурационный PHP файл и Json объекты. Со стороны Андроид приложения в качестве коннектора лучше использовать библиотеку Volley, это даст возможность напрямую подключиться к .php файлу с настройками подключения базы данных и позволит напрямую обмениваться данными между мобильным приложением и веб-сайтом. Кроме всего прочего, данный способ позволяет легко ориентироваться в коде программы, что значительно упрощает ее дальнейшую модернизацию, расширение функционала и исправление ошибок, возникших в ходе разработки. Архитектура приложений «пользователь – оболочка – БД – администратор» позволит производить модернизацию одной из частей модели, что, в свою очередь, не приведет к глобальным изменениям для других ее частей.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа:https://ru.wikipedia.org/wiki/PHP. – Дата доступа: 22.09.2016.
2. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/MySQL. – Дата доступа: 22.09.2016.
3. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/JSON. – Дата доступа: 23.09.2016.
4. ОС для многоэкранных устройств [Электронный ресурс]. – Режим доступа:http://developer-android.unlimited-translate.org/training/volley/index.html. – Дата доступа: 23.09.2016.

УДК 004.932

АНАЛИЗ АЛГОРИТМОВ ОБРАБОТКИ ДИНАМИЧЕСКИХ ИЗОБРАЖЕНИЙ ДЛЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ МОНИТОРИНГА АВТОМОБИЛЬНЫХ СТОЯНОК

П.В. ЯРОШЕВИЧ
(Представлено: канд. техн. наук, доц. Р.П. БОГУШ)

Рассматривается классификация современных методов обнаружения свободных мест на автостоянках. Представлена общая модель построения алгоритмов классификации парковочных мест на свободные и занятые, приведены характеристики современных алгоритмов мониторинга статусов парковочных мест.

Системы видеонаблюдения приобретают все большую значимость в различных сферах деятельности человека [5; 8; 14]. Благодаря развитию технологий компьютерного зрения предложен ряд систем видеонаблюдения с интеллектуальной обработкой входных видеоданных, включая системы управления автомобильными парковками. Такие видеосистемы могут автоматически выполнять многие функции. Например, определение занятости парковочного места, подсчет количества автомобилей на стоянке, определение расположения или типа транспортного средства, анализ активности людей, предоставление справочной информации в виде маршрута к парковочному месту, удаленное наблюдение владельца за автомобилем с помощью смартфона и оповещение в случае возникновения нештатной ситуации и т.д.

Очевидно, что для таких систем необходимо алгоритмическое обеспечение по обработке входных видеоданных, основным требованием к которому является высокая эффективность классификации парковочных мест, в том числе при наличии различных шумовых факторов: теней, световых пятен в солнечную погоду, изменение общей освещенности автомобильной парковки в течение суток, изменение погодных условий и т.д. Таким образом, сложность задачи предопределило разработку и развитие ряда методов и алгоритмов для её решения. В настоящее время задача автоматизированного обнаружения свободных парковочных мест автостоянки в системах видеонаблюдения не решена в полной мере, следовательно, исследования в данном направлении являются актуальными и необходимыми.

Использование видеоданных, полученных с камер наружного видеонаблюдения, сопряжено со следующими проблемами:

- требуется установить камеру видеонаблюдения достаточно высоко;
- при удалении от точки съемки места парковки характеризуются меньшим количеством информации;
- бросаемые на сцену тени автомобилей, столбов, ограждений пешеходов и др.;
- изменяемость освещения в зависимости от времени суток;
- влияние изменяемости погодных условий и времен года;
- структурирование парковочных мест на изображениях парковки;
- использование парадигмы «получение изображения» – «сжатие» – «передача» – «анализ» работы системы.

В ряде публикаций [2; 12] авторы отмечают то, что в зависимости от используемых методов системы мониторинга парковочных мест можно разделить на три группы:

- основанные на обнаружении автомобилей (car-driven methods, car-oriented methods – CO);
- основанные на сравнении со свободными парковочными местами (space-driven methods, space-oriented methods – SO);
- смешанные/гибридные методы (hybrid methods – H).

Основной компонентой любого предложенного метода является алгоритм классификации. Общая модель алгоритма классификации парковочных мест может быть представлена в виде последовательности следующих шагов, на первый из которых передается изображение парковки:

- извлечение регионов интереса парковочных мест;
- вычисление признаки каждого отдельного региона интереса парковочного места;
- классификация вычисленных признаков – определение статусов соответствующих парковочных мест;
- интерпретация результатов, которая может включать в себя как визуализацию статусов на исходном изображении парковки, так и их представление в цифровом виде для дальнейшей передачи или обработки.

Под регионом интереса (Region Of Interest – ROI) парковочного места (РИ) подразумевается область на исходном изображении, характеризующая данное место парковки. Например, в работе [11] каж-

дое парковочное место характеризуется четырьмя областями: областью земли и тремя областями предполагаемого припаркованного автомобиля (боковая, передняя стороны и крыша).

В работах, посвященных системам мониторинга парковочных мест и определениям их статусов, основное внимание уделено описанию и оценке эффективности используемых дескрипторов. Обычно используется один алгоритм классификации, хотя некоторые работы сравнивают эффективность нескольких.

Общая модель алгоритма классификации парковочных мест, является скелетом большинства рассмотренных методов, в которых также можно выделить алгоритмы, предшествующие ей и выполняющиеся после неё. Алгоритмы предобработки предназначены для того, чтобы подготовить исходные данные в приемлемый для метода вид. К алгоритмам предобработки можно отнести: преобразование перспективы; определение регионов парковочных мест; преобразование в требуемое цветовое пространство; извлечение изображений региона интересов парковочного места. Алгоритмы постобработки нацелены на улучшение результатов этапа классификации признаков, и представляют собой корректировку и объединение результатов классификации.

В таблице приводится сравнительная характеристика современных алгоритмов мониторинга статусов парковочных мест для организации интеллектуальных систем управления автопарковками.

Таблица – Современные алгоритмы мониторинга статусов парковочных мест

Источник	Тип метода	Цветовое пространство	Алгоритм обучения	Условия работы						Число парковочных мест	Точность метода, %
				искажение перспективы	межобъектное перекрытие	эффекты теней	изменяемость освещенности	различные погодные условия	работа в ночное время суток		
[6]	CO	RGB	Thrshld	–	–	+	–	–	–	10	–
[7]	CO	RGB	Thrshld	–	–	+	+	+	+	10	93
[15]	SO	Chrom	SVM	–	+	+	–	–	–	46	93.5
[4]	SO	YUV	–	–	+	–	–	–	–	52	90
[1]	SO	Gray	SVM	–	–	+	+	+	–	28	99.8
[2]	SO	Gray	SVM	+	–	+	+	+	–	28 37 100	99.6 99.3 99.6
[3]	SO	HSV	SVM	+	–	+	+	+	–	28 37 100	96 93 87
[9]	H	RGB	Bayes	+	+	–	–	–	–	46	99
[10]	H	RGB	Bayes	+	+	+	+	–	–	46	99
[11]	H	RGB	Bayes	+	+	+	+	+	+	72	98.5
[13]	H	RGB	Network	–	–	+	+	–	+	126	97.9
[12]	H	RGB	Network	–	–	+	+	–	+	126	97.9

В публикации [6] D.B.L. Bong и др. был предложен проект «Информационная система занятости парковки» (Car-Park Occupancy Information System – COINS). В этой работе рассматривается структурирование парковочных мест на автостоянке на основе комбинированного метода, включающего два шага. Причем первый шаг выполняет оператор, который выделяет границы каждого места парковки белым цветом и присваивает центральному пикселю желтый цвет. На втором шаге на основе программной реализации алгоритма выполняется автоматический поиск преобразованных пикселей желтого цвета и определение координат границ обнаруженного региона интереса. Функция обнаружения состоит в процентном содержании белых пикселей в разнице бинаризированных изображений места парковки текущего кадра и изображения свободного парковочного места. Если количество белых пикселей больше 30%,

парковочное место считается занятым. В экспериментах демонстрируется работа системы на примере парковки, состоящей из 10 мест, при различных сценариях парковки одного или двух автомобилей на различных местах парковки. В следующей работе [7] авторы переопределили порог для функций обнаружения объекта до 40%. Приведены результаты работы системы для симуляционной модели, представленной уменьшенной копией парковки и игрушечными машинками. Работа системы также была оценена и в реальных сценариях: при различных состояниях освещения (утреннее, дневное, ночное, облачное) и при различных помехах, вызванных дождем различной степени силы. Точность определения занятости парковочного места составляет 93%.

В [15] авторы предлагают осуществлять классификацию трех мест парковки за один раз. Так как каждое парковочное место может быть свободным или занятым, то для классификации используется 8-классовый классификатор метода опорных векторов. Эффективность подобного рода классификации составляет 84%. При использовании системы случайных полей Маркова (Markov Random Field – MRF) для корреляции результатов классификации методом опорных векторов, улучшается вероятность правильной классификации до 94%. Предложенная система работает при наличии значительных перекрытий между автомобилями и наличии теней на снимках.

В [4] авторы разработали систему, которая использует одну камеру видеонаблюдения для наблюдения за автомобильной парковкой, состоящей более чем из 100 парковочных мест. Для определения вакантности места парковки используется комбинация двух типов алгоритмов: статистический анализ изображений регионов интереса парковочного места и динамического анализа изображений. Точность предложенного метода составляет 90%.

В работе [1] для задачи классификации парковочных мест авторы оценивают эффективность использования в качестве дескрипторов LBP и LPQ. Авторами достигается показателя коэффициента ошибки в 0.16% при использовании комбинаций дескрипторов и различных классификаторов. Отличительной чертой этой работы является то, что в качестве данных для исследования было использовано большое количество снимков парковки, содержащих 105837 изображений регионов интереса мест парковки. В своей следующей работе [2] авторы продолжили сравнивать дескрипторы LBP и LPQ при использовании в качестве классификатора метода опорных векторов. База данных, изображений регионов интереса мест парковки была расширена до 695899. Новую версию базы снимков парковки формируют изображения двух парковок, снятых на три камеры, снимки которых осуществлялись в течение дня, на протяжении нескольких суток. Эти изображения были получены при различных погодных условиях, тенях бросаемых на сцену, условиях освещенности [16]. Точность предложенного метода для изображений, полученных с трех камер: UFPR04 (28 мест), UFPR05 (37 мест) и PUCPR (100 мест), составляет 99% для каждй из них.

В предложенной в работе [3] системе в качестве дескриптора парковочного места используется гистограмма цвета компоненты тона, а классификация данных осуществляется методом опорных векторов с линейным ядром. Авторами предлагается передавать с камеры видеонаблюдения не снимки парковки, а вычисленные для изображений регионов интереса мест парковки дескрипторы. В качестве данных для исследования использовалась база данных PkLot. Точность предложенного метода для изображений, полученных с трех камер: UFPR04 (28 мест), UFPR05 (37 мест) и PUCPR (100 мест), составляет 96, 93 и 87% соответственно.

В работах [9–11] авторы предлагают рассматривать парковку в виде иерархии изображений: изображения регионов интереса парковочного места формируют место парковки, парковочные места, идущие друг за другом, формируют парковочные ряды, которые, в свою очередь, составляют парковку. Авторы строят 3D-модель каждого места парковки с помощью четырех изображений регионов интереса парковочных мест, представляющих переднюю, боковую, верхнюю плоскости автомобиля и поверхности земли. Каждая из поверхностей отдельно классифицируется как свободная или занятая, на основе полученных результатов определяется статус места парковки. Также при классификации учитываются регионы интереса и не принадлежащие месту парковки, если оно перекрыто поверхностями тех регионов интереса. Предлагаемая система обеспечивает надежное обнаружение свободных парковочных мест с точностью, равной 99.37%, при различных погодных условиях, сильных изменениях освещения, теневых эффектах, межобъектном перекрытии, искажении перспективы, недостаточности освещения в ночное время суток.

Авторы в [13] предложили использовать обучаемые нейронные сети для определения статусов парковочных мест. Метод адаптивен к изменению интенсивности света с помощью определения эталонного пикселя тротуара. Отдельно рассматривается определение статусов парковочных мест в ночное время. Точность составила 99% для занятых мест парковки и 97% – для свободных парковочных мест, для исследуемого 24-часового видео.

В работе [12] авторы производят анализ снимков парковки, полученных в течение суток. Для определения статуса парковочного места используются особенности, связанные с освещением, статистические значения пикселей, гистограмма цветов, особенности краев, которые классифицируются нейронными сетями. Для повышения точности предложенного метода используется метод обнаружения движения. Точность составляет 99% для занятых парковочных мест и 97.9% для свободных.

Заключение. Представлена классификация современных методов обнаружения свободных мест на автостоянках. Рассмотрена общая модель построения алгоритмов классификации парковочных мест, которая предполагает извлечение регионов интереса парковочных мест, вычисление признаков каждого отдельного региона интереса парковочного места, классификацию вычисленных признаков и определение статусов соответствующих парковочных мест, а также интерпретацию результатов. В работе также приведены характеристики современных алгоритмов мониторинга статусов парковочных мест.

ЛИТЕРАТУРА

1. Parking space detection using textural descriptors / P.R. de Almeida [et al.] // In IEEE international conference on systems, man, and cybernetics (SMC). – 2013. – P. 3603–3608.
2. Pklot – a robust dataset for parking lot classification / P.R. de Almeida [et al.] // Expert Systems with Applications. – 2015. – Vol. 42, no. 11. – P. 4937–4949.
3. A visual sensor network for parking lot occupancy detection in Smart Cities / L. Baroffio [et al.] // wf-iot. – 2015.
4. Cost-effective single-camera multi-car parking monitoring and vacancy detection towards real-world parking statistics and real-time reporting / K. Blumer [et al.] // [J]. Neural Information Processing, Lecture Notes in Computer Science. – 2012. – P. 506–515.
5. Motion Detection and Tracking Algorithms in Video Streams / R. Bogush [et al.] // VNU Journal of Science, Mathematics – Physics. – 2009. – Vol. 25, no. 3. – P. 143–151.
6. Bong, D.B.L. Car-Park Occupancy Information System / D.B.L. Bong, K.C. Ting, N. Rajaee // Third Real-Time Technology and applications symposium, RENTAS 2006, Serdang, Selangor, December 2006.
7. Bong, D.B.L. Integrated approach in the design of car park occupancy information system / D.B.L. Bong, K.C. Ting, K.C. Lai // IAENG International Journal of Computer Science. – 2008. – 35. P. 1–8.
8. Brovko, N. Smoke detection algorithm for intelligent video surveillance system / N. Brovko, R. Bogush, S. Ablameyko // Computer Science Journal of Moldova. – 2013. – Vol. 21, no. 1(61). – P. 142–156.
9. A bayesian hierarchical detection framework for parking space detection / C.-C. Huang [et al.] // In Processing, 2008. ICASSP 2008. IEEE international conference on acoustics, speech and signal. – 2008. – P. 2097–2100.
10. Huang, C.-C. A hierarchical bayesian generation framework for vacant parking space detection / C.-C. Huang, S.-J. Wang // IEEE Transactions on Circuits and Systems for Video Technology. – 2010. – 20. – P. 1770–1785.
11. Huang, C.-C. Vacant parking space detection based on plane-based bayesian hierarchical framework / C.-C. Huang, Y.-S Tai, S.-J Wang // IEEE Transactions on Circuits and Systems for Video Technology. – 2013. 23. – P. 1598–1610.
12. Car park system: A review of smart parking system and its technology / M.Y.I. Idris [et al.] // Information Technology Journal. – 2009. – 8(2). – P. 101–113.
13. One-day long statistical analysis of parking demand by using single-camera vacancy detection / [et al.] // Journal of Transportation Systems Engineering and Information Technology. – 2014. – 14, 33.
14. Ngan, K. Video Segmentation and Its Applications / K. Ngan, H. Li // Springer; 2011. DOI 10.1007/978-1-4419-94182-0.
15. Robust parking space detection considering inter-space correlation / Q. Wu // In IEEE international conference on multimedia and expo. – 2007. – P. 659–662.
16. PKLot – A robust dataset for parking lot classification [Электронный ресурс]. – 2015. – Режим доступа: http://www.inf. ufpr.br/vri/databases/PKLot.tar.gz, свободный. – Дата доступа: 15.12.2015.

УДК 004

ПРИМЕНЕНИЕ КРИВЫХ БЕЗЬЕ ДЛЯ УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ В ТЕХНОЛОГИИ РЕДАКТОРА ВИДЕОШАБЛОНОВ

В.А. ПЛЯСОВ
(Представлено: канд. техн. наук, доц. Д.О. ГЛУХОВ)

Рассматриваются различные виды кривых Безье. Анализируются вопросы их определений и построений, а также свойства этих кривых. Показано, как их можно использовать в технологии редактора в виде ошаблонов.

В настоящий момент кривые Безье широко используются в программных продуктах для построения кривых различной формы. Данный вид кривых является простым для построения, так как имеет довольно простой алгоритм и является наиболее гибким, чем другие виды построения кривых по 3 и более опорным точкам.

Кривая Безье представляет собой параметрическую кривую, которая задается выражением (1):

$$B(t) = \sum_{i=0}^{n} P_i b_{i,n}(t), \quad 0 \le t \le 1, \tag{1}$$

где P_i – функция компонент векторов опорных вершин (т.е. координаты ключевых вершин) $b_{i,n}(t)$ – базисные функции кривой Безье, которые записываются следующим выражением:

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{2}$$

где $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ – число сочетаний из n по i (n – степень полинома; i – порядковый номер опорной вершины).

Рассмотрим типы кривых Безье:

1. Линейные кривые:

В данном случае $n = 1$, то есть говорит о наличии 2-х опорных вершин для построения кривой Безье. После попытки построения кривой получим обыкновенную прямую между двумя точками. Кривая задается выражением (3):

$$B(t) = (1-t)P_0 + tP_1, \quad 0 \le t \le 1. \tag{3}$$

2. Квадратичные кривые:

Здесь $n = 2$, то есть кривая строится по 3-м опорным точкам P_0, P_1, P_2. Кривая задается выражением (4), произвольная точка на кривой задается выражениями (5) и (6).

$$B(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + tP_2, \quad 0 \le t \le 1 \tag{4}$$

$$x = (1-t)^2 x_0 + 2t(1-t)x_1 + tx_2, \quad 0 \le t \le 1 \tag{5}$$

$$y = (1-t)^2 y_0 + 2t(1-t)y_1 + ty_2, \quad 0 \le t \le 1, \tag{6}$$

где x_0, x_1, x_2, y_0, y_1, y_2 – координаты опорных вершин P_0, P_1, P_2.

Пример данной кривой представлен на рисунке 1.

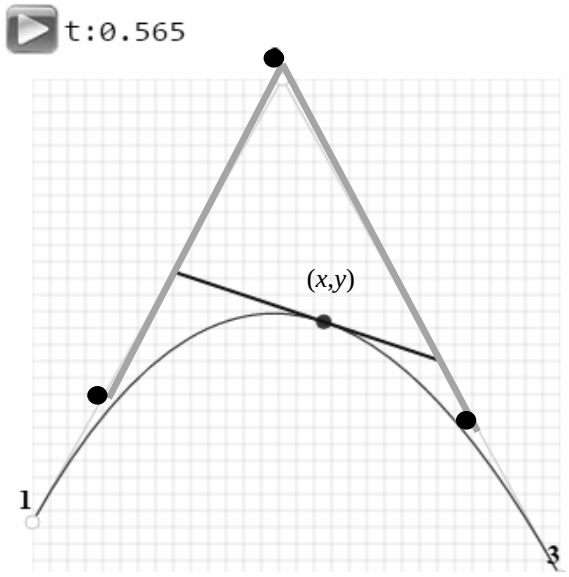


Рисунок 1. – Пример квадратичной кривой Безье

3. Кубические кривые:

При $n = 3$ кривая строится по 4-м опорным точкам. Данная кривая описывается следующим выражением (7):

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, \quad 0 \le t \le 1. \tag{7}$$

Линия берёт начало из точки P_0, направляясь к P_1, и заканчивается в точке P_3, подходя к ней со стороны P_2. То есть кривая не проходит через точки P_1 и P_2, они используются для указания её направления. Длина отрезка между P_0 и P_1 определяет, как скоро кривая повернёт к P_3 (1).

4. Кривые Безье высших порядков:

В данном случае $n > 3$, что непосредственно реализует форму записи (1). Здесь уже количество опорных точек определяется как $n + 1$.

В современных графических системах и форматах, таких как PostScript (а также основанные на нём форматы Adobe Illustrator и Portable Document Format (PDF)), Scalable Vector Graphics (SVG)[1], Metafont, CorelDraw и GIMP, для представления криволинейных форм используются кривые Безье, составленные из кубических кривых (1).

Рассмотрим основные свойства кривых Безье:

1) непрерывность заполнения сегмента между начальной и конечной точками;

2) кривая всегда располагается внутри фигуры, образованной линиями, соединяющими контрольные точки;

3) при наличии только двух контрольных точек сегмент представляет собой прямую линию;

4) прямая линия образуется при коллинеарном (на одной прямой) размещении управляющих точек;

5) кривая Безье симметрична, то есть обмен местами между начальной и конечной точками (изменение направления траектории) не влияет на форму кривой;

6) масштабирование и изменение пропорций кривой Безье не нарушает её стабильности, так как она с математической точки зрения «аффинно инвариантна»;

7) изменение координат хотя бы одной из точек ведет к изменению формы всей кривой Безье;

8) любой частичный отрезок кривой Безье также является кривой Безье;

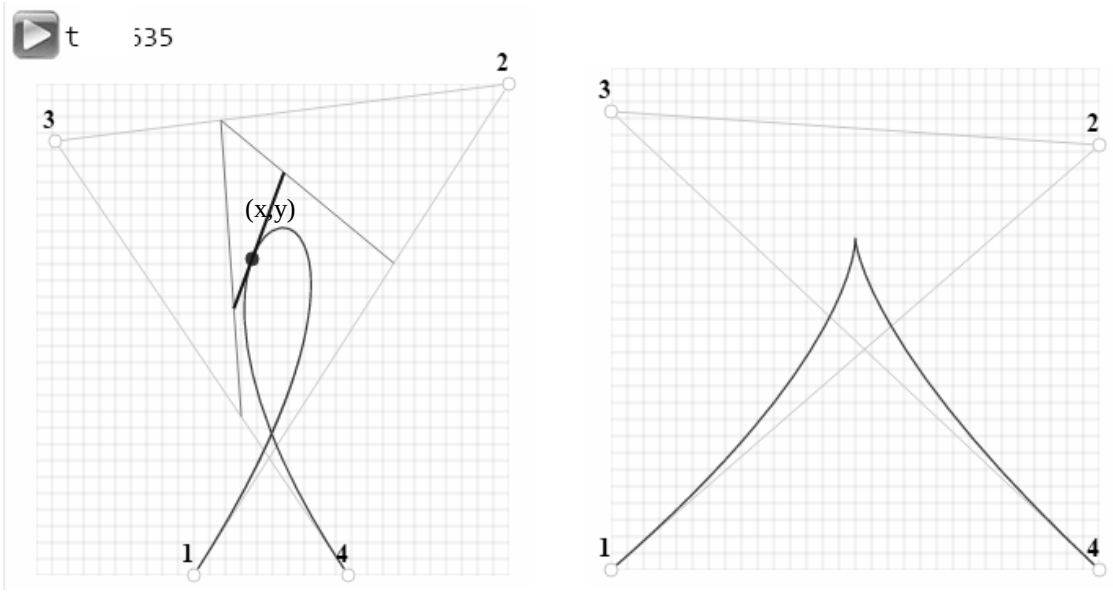
9) степень кривой всегда на одну ступень ниже числа контрольных точек. Например, при трех контрольных точках форма кривой – парабола;

10) окружность не может быть описана параметрическим уравнением кривой Безье;

11) невозможно создать параллельные кривые Безье, за исключением тривиальных случаев (прямые линии и совпадающие кривые), хотя существуют алгоритмы, строящие приближённую параллельную кривую Безье с приемлемой для практики точностью.

Применение кривых Безье при построении траектории движения динамических объектов дает: повышение быстродействия программы (за счет простых алгоритмов реализации), параметризацию опорных точек (легко меняем координаты опорных точек, не замедляя работы программы), гибкость построения (возможность построения различных кривых с различным расположением точек).

На рисунке 2 (*а*, *б*) показана кривая Безье, которая имеет форму петли, а также негладкая форма, что довольно сложно реализовать, используя другие типы построения кривых по данному примеру.



а *б*

Рисунок 2. – Кривая Безье в виде петли (*а*); негладкая кривая Безье (*б*)

Из вышесказанного можно сделать *вывод*, что кривые Безье хорошо подходят для решения проблемы в движении динамических объектов между кадрами видео в технологии редактора видеошаблонов за счет своей простоты в реализации, параметризации опорных вершин, гибкости в построении.

ЛИТЕРАТУРА

1. Кривая Безье [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikipedia.org/wiki. – Дата доступа: 20.09.2016.
2. Кривые Безье [Электронный ресурс]. – 2016. – Режим доступа: https://learn.javascript.ru/bezier. – Дата доступа: 20.09.2016.
3. Простые в использовании кривые Безье [Электронный ресурс]. – 2016. – Режим доступа: http://www. cyberguru.ru/ algorithms/algorithms-theory/curves-bezier.html. – Дата доступа: 20.09.2016.

УДК 004

АРХИТЕКТУРА ПРОГРАММНОГО КОМПЛЕКСА РЕДАКТОРА ВИДЕОШАБЛОНОВ

В.А. ПЛЯСОВ
(Представлено: канд. техн. наук, доц. Д.О. ГЛУХОВ)

Рассматриваются основные моменты в архитектуре данного приложения: из каких основных частей будет построен проект, как и где будет храниться информация о видеошаблоне. Показаны система слоев в шаблоне, различные системы редактирования динамических объектов; рендеринг основного видеопотока.

Редактор видеошаблонов будет представлять приложение из нескольких модулей, т.е. клиентская часть приложения (то, что видит пользователь в браузере) серверная часть (основная архитектура приложения, база данных всех видеошаблонов) и сам редактор видеошаблонов.

На рисунке 1 представлена вся архитектура приложения.

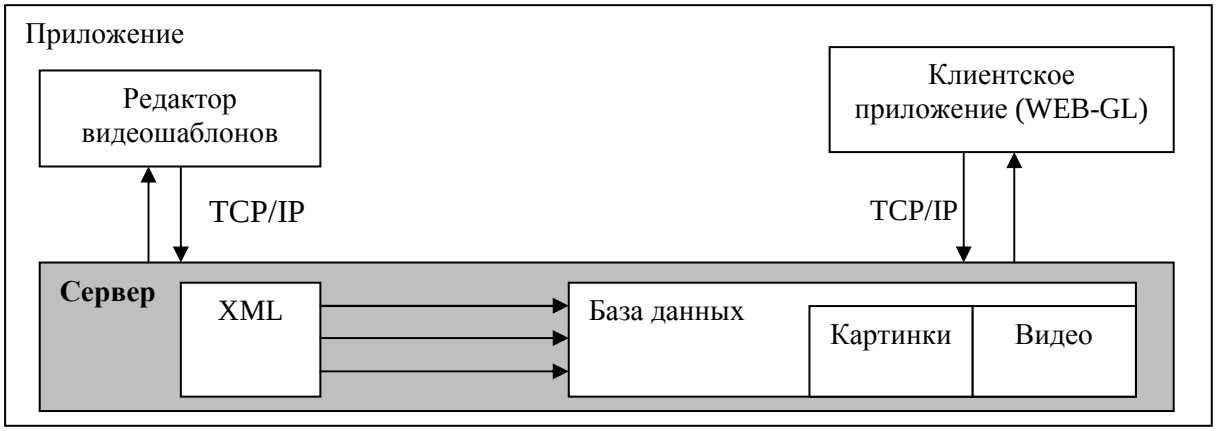


Рисунок 1. – Архитектура приложения

Сервер представляется в виде базы данных всех файлов (картинки и видео) и XML файла, который хранит в себе ссылки на данные объекты, а также список доступных видеошаблонов. Клиентское приложение представляет собой обычное WEB-приложение, которое будет через сервер работать с самой программой редактора видеошаблонов.

Основная идея заключается в том, чтобы в видео добавить различные движущиеся динамические объекты (анимация, видео и т.д.), создавая тем самым видео шаблон, т.е. есть какое-либо видео, которое загружено на сервер и есть отдельно картинки, которые необходимо добавить на данное видео. После добавления на него различных объектов, которые будут двигаться, искажаться, вращаться по мере просмотра видео, все эти данные будут сохраняться в отдельный файл типа XML – расширяемый язык разметки.

Отсюда возникла проблема, где и как на начальном этапе хранить эти данные. Чтобы упростить систему в приложении, т.е. не сразу всё загружать в XML файл, а только конечный результат, было принято решение создать отдельный класс, который будет заниматься хранением промежуточных данных, чтобы в дальнейшем всю собранную и отформатированную информацию интерпретировать в XML и тем самым создать файл видеошаблона.

Для создания более сложного шаблона необходима поддержка системы слоев, чтобы под каждый слой создавался отдельный экземпляр данного класса, который пользователь может настраивать под себя с помощью клиентского приложения.

Также возникла проблема – по какому правилу лучше всего задавать движение объектов и как их искажать между кадрами в видео. Чтобы облегчить решение данной проблемы, необходимо сдвигать и искажать не весь слой, что довольно сложно в реализации, а сдвигать вершины слоя по мере просмотра видео, т.е. сразу решается проблема с искажением (сдвинув несколько вершин слоя, – достигается искажение).

Задание ключевых кадров на видео, на которых пользователь может изменять траекторию движения объекта, искажение, вращение. На промежуточных кадрах пользователь может просмотреть, как объект выполняет все действия, если ему хочется немного изменить на данном кадре объект, то данный кадр определяется как ключевой и появляется доступ к редактированию.

Класс слоя будет хранить в себе следующую информацию:
- ссылку на объект (URL или из базы данных);
- массив из ключевых кадров, которых есть объект;
- массив из координат всех вершин объекта на ключевых кадрах;
- угол вращения объекта на ключевых кадрах;
- прозрачность объекта.

То есть в конечном итоге получаем массив объектов на видео, ссылка на видео и картинки. Затем данная информация интерполируется в XML файл (конечный видеошаблон) и загружается на сервер.

Что касается клиентского приложения, то здесь задача сводится к чтению XML-файла с видеошаблоном, создание плеера, который будет воспроизводить видео и отрисовывать все объекты на видео и функционал по редактированию объектов. Рендеринг основного видеопотока будет осуществляться параллельно с воспроизведением видео, в настоящий момент в WEB индустрии есть мощные плееры и движки, которые дают возможность осуществлять рендеринг «налету» без потерь производительности, т.е. поддерживать частоту смены кадров 30 и выше.

На рисунке 2 представлен простейший плейер с частично реалированным функционалом видеошаблонов.

Рисунок 2. – Редактор видеошаблонов

Как только видеошаблон будет создан или был загружен с сервера, пользователь может сохранить его в новое видео, в котором уже будет присутствовать данный шаблон.

В заключение проведенного исследования можно сделать **вывод**, что архитектура данного приложения является довольно сложной для реализации «в лоб».

Проанализировав всю задачу, найдены легкие способы реализации той или иной части модуля программы. Основная сложность при реализации – взаимодействие базы данных с XML файлом, который хранит список файлов и видеошаблонов, а затем выгрузка на клиентскую часть приложения.

ЛИТЕРАТУРА

1. XML формат [Электронный ресурс]. – 2016. – Режим доступа: http://okitgo.ru/xml/xml-format.html. – Дата доступа: 20.09.2016.

УДК 004.42

ЯДРО УНИФИЦИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ОТДЕЛ КАДРОВ»

Д.И. САЛАЙ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассмотрены вопросы проектирования функциональной структуры ядра унифицированной информационной системы «Отдел кадров». Обоснован выбор технологий, среды и средств программирования. Выполнен сравнительный анализ и изучены возможности существующих аналогичных проектов, в результате чего сформулированы требования к концепции клиентской части информационной системы.

Ядро унифицированной информационной системы (ИС) «Отдел кадров» предназначено для ведения кадрового учета на коммерческих предприятиях малого и среднего бизнеса. Система должна предоставлять возможность обрабатывать, просматривать и редактировать информацию обо всех сотрудниках предприятия, составлять штатное расписание, формировать основную кадровую документацию.

Помимо предоставления информации система должна обладать удобным и интуитивно понятным интерфейсом, предоставлять возможность поиска информации по фамилии сотрудника, должна обеспечивать предоставление и редактирование информации о сотрудниках, предоставлять необходимый набор данных о штатном расписании [1].

Исходя из приведенных характеристик и требований была разработана функциональная структура данной информационной системы. Функциональная структура системы представлена на рисунке 1.

Рисунок 1 – Функционально-организационная структура проектируемой системы

Ядро унифицированной ИС «Отдел кадров» состоит из следующих подсистем:

- *подсистема администрирования* – в данной подсистеме происходит назначение прав на доступ к определенной информации и регистрация пользователя, а также осуществление управления системой в целом;
- *подсистема аутентификации* – данная подсистема предназначена для авторизации пользователей;
- *подсистема генерации отчетов* – в данной подсистеме происходит формирование необходимых отчетов для печати или просмотра на основе данных о сотруднике или другой справочной информации;
- *подсистема справочников* – обеспечивает ведение справочной информации о сотрудниках и штатном расписании.

На предприятии сотрудники выполняют свою работу и обладают разной квалификацией. У них могут быть определенные ограничения на доступ к конфиденциальной информации, находящейся в базе программы отдела кадров. Сотрудникам должно быть удобно работать с системой. Соответственно, определенные категории сотрудников могут иметь возможность работы с ограниченным количеством информации [2].

При разработке программного продукта целесообразно выделить такие вкладки: «*Сотрудники*», «*Вакансии*», «*Приказы*», «*Все даты*», «*Отпуска*». Для конфиденциальности информации, а также для

удобства работы с системой функции и возможности, предоставляемые пользователям, отличаются от возможностей Администратора.

Вкладка «*Сотрудники*» содержит следующие панели:

- добавление нового сотрудника – позволяет внести всю необходимую информацию о новом сотруднике предприятия;
- подробно – предоставляет возможность просмотра и редактирования подробной информации о сотруднике;
- список сотрудников предприятия.

Панель «*Подробно*» вкладки «*Сотрудники*» содержит следующие вкладки:

- личные данные – просмотр общих сведений о сотруднике;
- контракты – просмотр контрактов и текущих назначений сотрудника;
- история отпусков – просмотр истории отпусков сотрудника;
- сведения о воинском учете – просмотр сведений о воинском учете сотрудника;
- образование – просмотр информации об образовании сотрудника;
- командировки – просмотр командировок сотрудника;
- поощрения и взыскания – просмотр поощрений и взысканий сотрудника.

Вкладка «*Найм*» позволяет работать с документами, относящимися непосредственно к приему на работу или увольнению с работы сотрудника:

- приказ о приеме на работу – позволяет сформировать приказ о приеме на работу нового сотрудника предприятия;
- приказ об увольнении – позволяет сформировать приказ об увольнении сотрудника с работы.

Вкладка «*Вакансии*» позволяет просматривать и создавать штатное расписание предприятия.

Вкладка «*Отпуска*» позволяет создавать график отпусков и назначать отпуск выбранному сотруднику.

Администратор имеет возможность просмотра всех зарегистрированных пользователей системы, добавлять нового пользователя в систему с определенной ролью, редактировать информацию о пользователях, удалять пользователя из системы.

Для реализации задачи по разработке ядра унифицированной информационной системы «Отдел кадров» была выбрана СУБД MySQL Server. Данная СУБД является одной из наиболее популярных систем управления и обслуживания баз данных. В ней применяются новейшие разработки и последние достижения в области проектирования, построения и обслуживания баз данных. Данная СУБД распространяется по модели СПО (свободное программное обеспечение). Для создания клиентской части решено использовать среду разработки Visual Studio, а языком написания приложения был выбран C#. Данное решение можно обосновать широким распространением этой среды разработки, удобством ее использования и функциональностью [3].

Таким образом, разработанное ядро унифицированной информационной системы «Отдел кадров» предназначено для ведения кадрового учета в рамках отделов кадров коммерческих предприятий малого и среднего бизнеса, для автоматизации бизнес-процессов в области управления персоналом: ведение штатного расписания, личных карточек, командировок, отпусков, табельного учета рабочего времени, формирования и ведения приказов по личному составу. Разработанный программный продукт предназначен для ведения кадрового учета в отделе кадров коммерческого предприятия малого или среднего бизнеса.

ЛИТЕРАТУРА

1. Кадровая служба [Электронный ресурс] // Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Кадровая_служба. – Дата доступа: 25.09.2016.
2. Тидвелл, Дж. Разработка пользовательских интерфейсов / Дж. Тидвелл. – СПб. : Питер, 2008. – 395 с.
3. Пелланд, П. Переход к Microsoft Visual Studio 2010 / П. Пелланд, П. Паре, К. Хайнс, 2011. – 256 с.

УДК 004.65

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ ЯДРА УНИФИЦИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ «ОТДЕЛ КАДРОВ»

Д.И. САЛАЙ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассмотрены необходимые сущности для построения базы данных информационной системы. Построена диаграмма потоков данных для предметной области «Отдел кадров». Все таблицы нормализованы и приведены к третьей нормальной форме.

На сегодняшний день существует множество систем управления базами данных, которые позволяют хранить и обрабатывать большие объемы информации. Однако актуальной проблемой на многих предприятиях остается проблема оформления и хранения больших объемов бумажной документации и информации. Это связано с тем, что хотя и существует ряд предприятий с похожими предметными областями, но у каждого из них присутствует своя индивидуальная специфика [1].

Другой актуальной проблемой является то, что вся информация, обрабатываемая на предприятии, представляет собой большую предметную область, и автоматизировать работу с ней как с целостной системой в совокупности оказывается непросто, так как отдельные части области, также требуют глубокого анализа и изучения, а это, в свою очередь, требует много времени.

Описанные проблемы явно прослеживаются в работе отдела кадров предприятия, поскольку обрабатываемый там объем информации очень велик и требуется хранение и оборот большого количества информации в форме отчетов, приказов и другой документации, постоянное отслеживание изменения данных о рабочих, отделах и вакансиях, анализ и подбор требуемых кадров. В данной работе рассматривается вопрос проектирования базы данных для ядра унифицированной информационной системы «Отдел кадров».

В ходе анализа знаний и разработки базы данных были выявлены следующие **основные** сущности:

- сущность *Сотрудник* – описывает сотрудника, который был связан с организацией в тот или иной момент времени контрактом. Характеризуется личной информацией: имя, фамилия, отчество, телефон, адрес прописки и пр.;
- сущность *Отдел* – описывает имеющие место в составе организации отделы. Характеризуется наименованием отдела, описанием и его должностным составом;
- сущность *Должность* – описывает должности и вакансии, существующие на предприятии. Характеризуется своим названием и описанием, отделом, к которому она относится, и числом рабочих единиц относящихся к данному отделу;
- сущность *Контракт* – описывает взаимоотношения организации и сотрудника. Является связующей для сущностей сотрудника и должности. Характеризуется описанием, временным интервалом действия договора и номером приказа, его подтверждающего;
- сущность *Приказ* – основополагающая, подтверждающая истинность событий организации, регистрируемых в базе. Характеризуется уникальным номером и лицом его подписавшим, а также датой подписания;
- сущности Отпуск, Командировка, Поощрение, Наказание – сущности описания события относительно сотрудника. Характеризуются временным интервалом, номером приказа, утвердившего событие, кодом сотрудника и номером приказа прерывания или приказа замещения события. Один приказ может описывать несколько однородных событий отличных сотрудников;
- сущности *Повышение квалификации, Больничный* – сущности описания события относительно сотрудника. Характеризуются, помимо вышеописанного, документом подтверждения отличного от приказа;
- сущность *Документ* – сущность, подтверждающая истинность некоторых событий, описывающих сотрудника. Характеризуется уникальным номером, серией, датой и содержанием.

При начальном анализе полезно построить диаграмму потоков данных (*англ.* – DataFlowDiagram, DFD-diagram). Такая диаграмма описывает внешние по отношению к системе источники и адресата данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. По правилам построения DFD диаграмм прямоугольником обозначаются внешние источники и адресаты данных, прямоугольником с чертой – хранилища данных, кругом – функции, стрелками – потоки данных, подписями на потоках обозначаются перемещаемые объекты [2].

Диаграмма потоков данных для предметной области «Отдел кадров» представлена на рисунке 1.

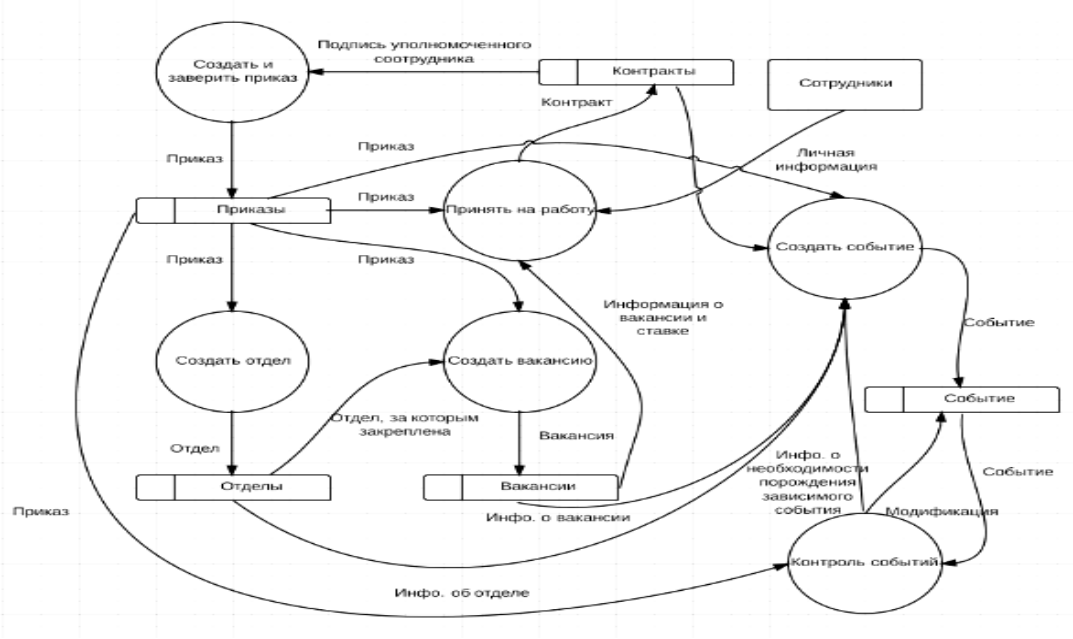


Рисунок 1. – Диаграмма потоков данных

Чтобы построить схему реляционной базы данных, необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных [2].

На рисунке 2 представлены отношения для проектируемой базы данных.

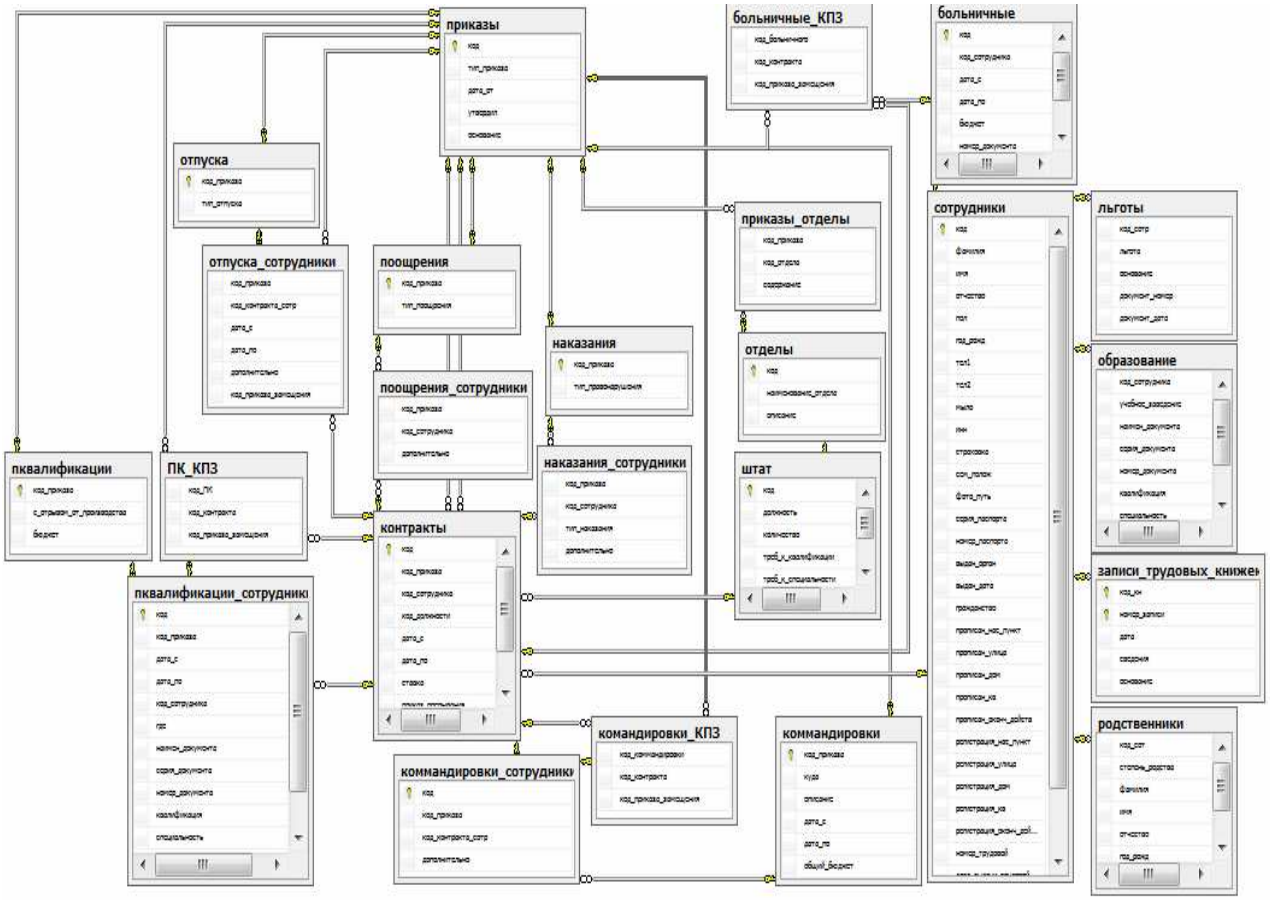


Рисунок 2. – Набор необходимых отношений базы данных

После определения связей между таблицами, назначения ключей и построения реляционной базы данных, было выполнено её приведение к третьей нормальной форме.

Полностью или частично к ЗНФ были приведены таблицы:

- образование;
- больничные;
- штат;
- наказания;
- наказания – сотрудники;
- поощрения;
- отпуска;
- квалификации – сотрудники;
- приказ.

Были выделены в самостоятельные сущности и представлены в виде таблиц следующие объекты (далее перечисляются таблицы объектов):

- тип приказа;
- тип отпуска;
- документ об образовании;
- квалификация;
- специальность;
- подтверждающий документ и др.

По итогу было произведено неполное приведение к ЗНФ. Неполное приведение к ЗНФ объясняется тем, что, увеличивая число отношений, мы тем самым увеличиваем объем запросов к базе и размер кода хранимых процедур. Помимо того, как было уже сказано, увеличение размера кода запросов и хранимых процедур имеет негативное влияние за счет увеличения потоков трафика при генерации запросов на клиенте и сложностью программирования, заключающейся в требовании проявления излишней внимательности.

В результате проделанной работы спроектирована база данных, при помощи которой можно создать автоматизированную информационную систему, направленную на сокращение временных затрат работника отдела кадров, занимающегося ведением различной документации и учетом контингента, минимизировать появление ошибок при составлении документов и автоматизировать формирование отчетов и документации.

ЛИТЕРАТУРА

1. Кадровая служба [Электронный ресурс] // Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Кадровая_ служба. – Дата доступа: 25.09.2016.
2. Малыхина, М. Базы данных: основы, проектирование, использование / М. Малыхина. – М. : BHV, 2004. – 512 с.
3. Кузнецов, М.В. MySQL 5 / М.В. Кузнецов, И.В. Симдянов. – СПб. : БХВ-Петербург, 2010. – 1024 с.

УДК 004.514

ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «ИНТЕРАКТИВНАЯ КАРТА ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА»

П.А. СИЛУКОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассматривается в качестве объекта разработки программное обеспечение «Интерактивная карта Полоцкого государственного университета». Цель разработки – создание программного обеспечения для поиска информации об объектах университета на интерактивной карте. Проанализированы вопросы проектирования пользовательского интерфейса программного обеспечения «Интерактивной карты Полоцкого государственного университета».

Предоставление пользователям возможности легко и удобно работать с функциональными возможностями программного обеспечения – цель разработки пользовательского интерфейса программного обеспечения.

Для начала необходимо определить требования к формам пользовательского интерфейса.

Для этого были выделены следующие требования к пользовательскому интерфейсу:

- функциональность;
- соответствие технологии;
- понятность и логичность;
- обеспечение высокой скорости работы пользователя;
- обеспечение защиты от человеческих ошибок;
- быстрое обучение пользователя;
- субъективное удовлетворение пользователя [1].

К функциональности относятся требования, которые должны обеспечить выполнение функциональных возможностей, которые были определены при проектировании функциональной части программного обеспечения:

- отображение карты с объектами на ней;
- возможность перемещаться по карте и масштабировать карту;
- отображение меток на карте с названиями объектов;
- отображение информации об объектах, включая связанные объекты;
- возможность выполнить поиск объектов.

Учитывая выбранные требования, необходимо определить виды форм пользовательского интерфейса. Были выделены следующие формы:

- форма, отображающая карту с элементами управления: изменение масштаба, включение поиска;
- форма, отображающая информацию об объекте: корпусе, общежитии, факультете, кафедре, сотруднике, службе или организации;
- форма поиска объектов на карте.

Далее необходимо выбрать технологии и инструменты проектирования пользовательского интерфейса. Для разработки пользовательского интерфейса была использована система Windows Presentation Foundation [2]. Данная система обладает рядом преимуществ:

- веб-подобная модель компоновки – поддерживает гибкий поток, размещающий элементы управления на основе их содержимого;
- развитая текстовая модель – отображение расширенного стилизованного текста в любом месте пользовательского интерфейса;
- стили и шаблоны – позволяют стандартизировать форматирование и многократно использовать его по всему приложению;
- декларативный пользовательский интерфейс – содержимое каждого окна сериализуется в виде XML-дескрипторов в документе XAML [3].

Также была использована библиотека MahApps.Metro для создания более привлекательного пользовательского интерфейса [4].

Далее будут описаны основные формы, доступные пользователю.

Главной формой выступает форма, отображающая карту с элементами управления. На данной форме располагается карта. В правом нижнем углу находятся кнопки увеличения/уменьшения масштаба карты. В правом верхнем углу располагается элемент, отображающий текущий населенный пункт. В левом верхнем углу – кнопка для отображения формы поиска объектов. На карте отображаются метки

корпусов и общежитий, по нажатию на которые открывается форма с информацией об объекте. Форма представлена на рисунке 1.

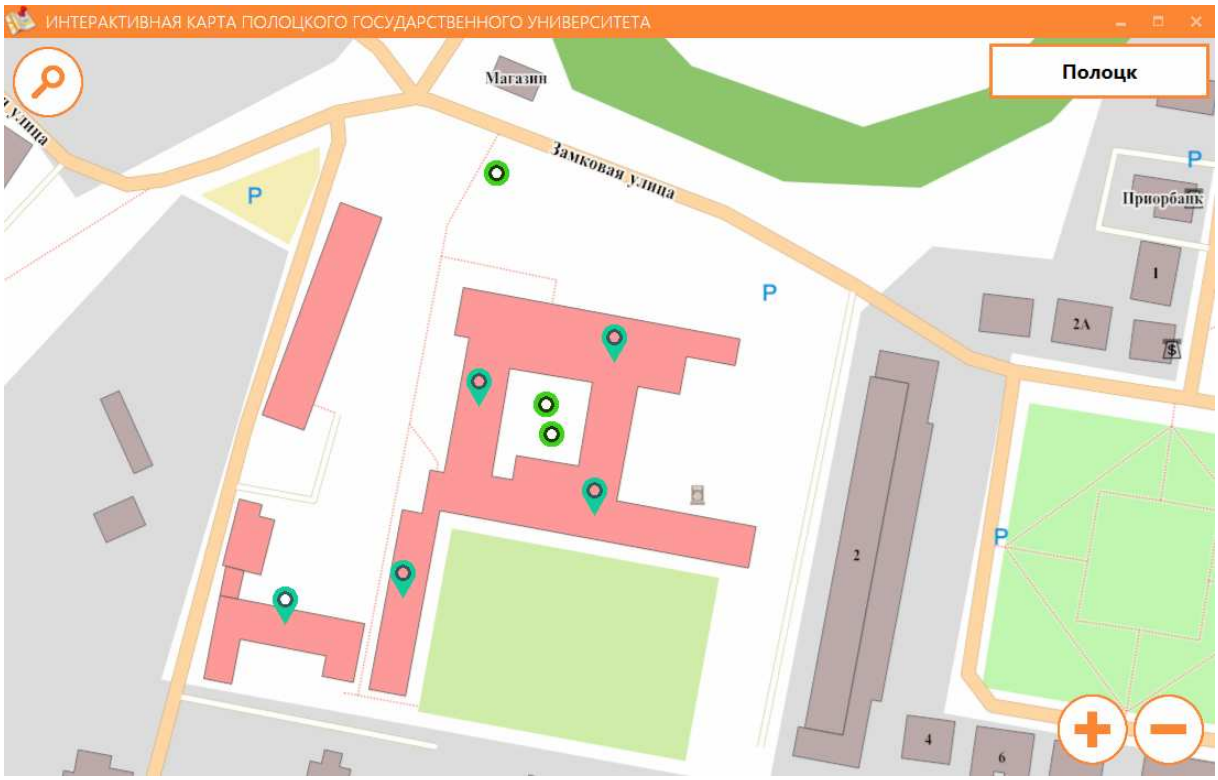


Рисунок 1. – Форма, отображающая карту и элементы управления

Следующей важной формой является форма отображения информации об объектах. Для каждого объекта разработана своя форма, но все формы имеют общий вид. На форме отображаются: название объекта; изображение (если имеется); основная информация; исторические факты (если имеются); списки других объектов, а именно: факультетов, кафедр, служб, организаций и работников.

На рисунке 2 представлена форма с информацией о факультете ФИТ.

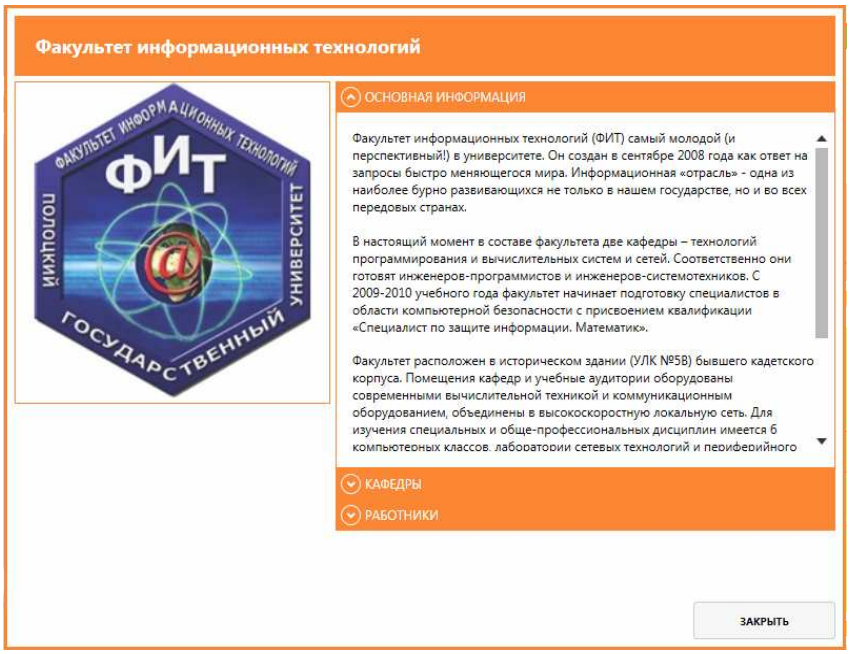


Рисунок 2. – Форма с информацией о факультете ФИТ

Последней формой пользовательского интерфейса является форма поиска объектов на карте.

Форма состоит из элементов выбора категории поиска и объекта поиска. На выбор представляются следующие категории:

- города;
- корпуса;
- общежития;
- исторические достопримечательности;
- администрация;
- организации;
- службы;
- факультеты;
- кафедры;
- работники.

Заключение. В результате проведенной работы был спроектирован пользовательский интерфейс программного обеспечения «Интерактивная карта Полоцкого государственного университета». Интерфейс состоит из 3 видов форм. Формы позволяют пользователю легко и удобно использовать функциональные возможности программного обеспечения.

ЛИТЕРАТУРА

1. Требования к дизайну пользовательского интерфейса [Электронный ресурс]. – Режим доступа: http://www.okd.mdk.ksue.edu.ua/index.php?option=com_content&view=article&id=174&Itemid=90. – Дата доступа: 28.09.2016.
2. Windows Presentation Foundation [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/ Windows_Presentation_Foundation. – Дата доступа: 28.09.2016.
3. Преимущества WPF [Электронный ресурс]. – Режим доступа: http://professorweb.ru/my/WPF/base_WPF/level1/1_3.php. – Дата доступа: 28.09.2016.
4. Mahapps.Metro [Электронный ресурс]. – Режим доступа: http://mahapps.com/. – Дата доступа: 28.09.2016.

УДК 004.031.42

ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНОЙ ЧАСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «ИНТЕРАКТИВНАЯ КАРТА ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА»

П.А. СИЛУКОВ
(Представлено: Е.Р. СУХАРЕВ)

Представлен объект разработки – программное обеспечение «Интерактивная карта Полоцкого государственного университета». Показана цель разработки – создание программного обеспечения для поиска информации об объектах университета на интерактивной карте. Рассмотрены вопросы проектирования функциональной части программного обеспечения «Интерактивной карты Полоцкого государственного университета».

Инфраструктура Полоцкого государственного университета представлена 4 корпусами, расположенными в 3 населенных пунктах, 5 общежитиями, большим числом служб, организаций, факультетов и кафедр. Для абитуриентов и студентов начальных курсов порой трудно найти, в каком кабинете, каком корпусе располагается та или иная служба или кафедра. Для поиска такой информации студенты могут использовать официальный сайт, но сайт не позволяет определить точное расположение требуемых объектов. Для определения точного месторасположения объектов с выводом полной информации о них и будет предназначено программное обеспечение «Интерактивная карта Полоцкого государственного университета».

Первым этапом проектирования программного обеспечения является *определение функциональных и других возможностей, которым программное обеспечение должно соответствовать*. Таким образом, были выделены следующие функциональные возможности:

- перемещение по карте с помощью мыши и клавиатуры;
- увеличение и уменьшение масштаба карты с помощью мыши и клавиатуры;
- отображение меток на карте с названиями объектов, к которым они относятся: корпуса, общежития;
- отображение информации об объектах: корпусах, общежитиях, факультетах, кафедрах, службах и организациях;
- отображение объектов, связанных с выбранным, например, списком работников на определенной кафедре;
- глобальный поиск объектов на карте.

Также были выделены следующие общие требования к программному обеспечению:

- надежность – программное обеспечение должно быть устойчиво к различного рода отказам системы;
- переносимость – возможность перенести программное обеспечение с одного носителя информации на другой;
- ресурсоемкость – программное обеспечение должно эффективно расходовать предоставленные мощности компьютера;
- ориентированность на пользователя – удобный и понятный графический интерфейс ПО;
- программное обеспечение должно быть реализовано как standalone приложение для компьютеров с ОС Windows 7 и старше;
- динамическая работа с данными – добавление или удаление данных не должно влиять на работоспособность программы.

Следующим этапом проектирования является *разработка функциональной структуры программного обеспечения*. Были выделены 4 подсистемы программного обеспечения:

- подсистема для работы с интерактивной картой - предоставляет возможности по управлению картой, отображению объектов на карте;
- подсистема вывода информации об объектах - предоставляет возможности по выводу всей информации о корпусах, общежитиях, факультетах и кафедрах и т.д.;
- подсистема поиска объектов на карте - предоставляет возможности по поиску объектов на карте по разным критериям;
- подсистема доступа к информации в БД.

Связь подсистем представлена на рисунке.

Следующим этапом проектирования программного обеспечения является *создание диаграммы вариантов использования* [1] *и диаграммы классов* [2].

В диаграмме классов были выделены следующие основные классы:

- MapControl – класс для отрисовки и управления картой;
- HousingControl – класс для работы с объектами «корпус»;
- HostelControl – класс для работы с объектами «общежитие»;
- PeopleControl – класс для работы с объектами «сотрудник»;
- FacultyControl и DepartmentControl – классы для работы с объектами «факультет» и «кафедра» соответственно;
- ServiceControl и OrgControl – классы для работы с объектами «служба» и «организация» соответственно;
- SearchControl – класс для работы с поиском объектов на карте;
- DBConnect – класс для подключения к базе данных и выполнения запросов.

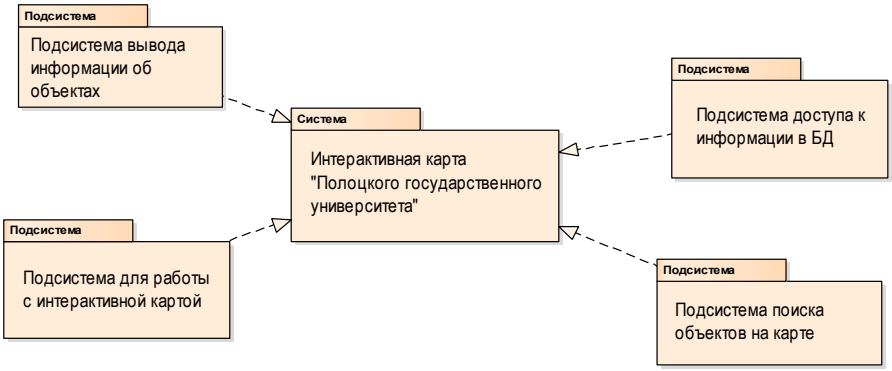


Схема связи подсистем программного обеспечения

На диаграмме вариантов использования был выделен один актер – «Пользователь», который совершает все действия, выделенные при проектировании.

Последним этапом проектирования функциональной части программного обеспечения является *выбор инструментов*.

В качестве языка программирования был выбран CSharp. Данный язык имеет ряд преимуществ перед другими языками:

- подлинная объектная ориентированность;
- компонентно-ориентированное программирование;
- безопасный (по сравнению с языками C и C++) код;
- унифицированная система типизации;
- поддержка событийно - ориентированного программирования;
- «родной» язык для создания приложений в среде .NET [3].

В качестве СУБД был выбран SQLite, обладающий следующими преимуществами:

- надежность;
- производительность – в несколько раз быстрее других СУБД [4];
- занимает мало места на диске;
- возможность копировать базу данных вместе с программой и использовать ее без предварительной установки и настройки.

Для работы программного обеспечения была взята готовая база данных, поэтому процесс проектирования базы данных не рассматривался.

В результате работы было спроектировано программного обеспечение, позволяющее выполнять поиск объектов Полоцкого государственного университета на интерактивной карте с возможностью вывода информации о найденных объектах.

ЛИТЕРАТУРА

1. Лекция 3: Элементы графической нотации диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/32/32/lecture/1004?page=2. – Дата доступа: 28.09.2016.
2. Лекция 4: Диаграмма классов: крупным планом [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/1007/229/lecture/5956. – Дата доступа: 28.09.2016.
3. Язык C#. Преимущества, недостатки. Интерфейсы, делегаты [Электронный ресурс]. – Режим доступа: http://www.uzluga.ru/potrd/. – Дата доступа: 28.09.2016.
4. Что такое SQLite. Плюсы и минусы [Электронный ресурс]. – Режим доступа: http://webnotes.by/docs/sql/259. – Дата доступа: 28.09.2016.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ ГРАФИЧЕСКОГО ВЕБ-ИНТЕРФЕЙСА НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ «ВЕБ-ПОРТАЛ ЛЮБИТЕЛЬСКОГО ГАНДБОЛА РЕСПУБЛИКИ БЕЛАРУСЬ»

В.О. ТРАПЕЗНИКОВ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Анализируются технологии, используемые при реализации интерфейсов веб-приложений. Приведен пример реализации веб-интерфейса одной из страниц веб-приложения. Показано, что при разработке веб-интерфейсов необходимо учитывать такие факторы, как «юзабельность», простота использования, понятность для конечного пользователя, независимость от браузера, кроссплатформенность, адаптивность для различных размеров экранов, динамичность.

Веб-приложениями или иногда также веб-системами называются на сегодняшний день различные программные продукты, доступ к которым осуществляется через веб-интерфейс. Веб-приложение клиент – серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется преимущественно на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.

Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создаётся один раз для произвольно выбранной платформы и на ней разворачивается.

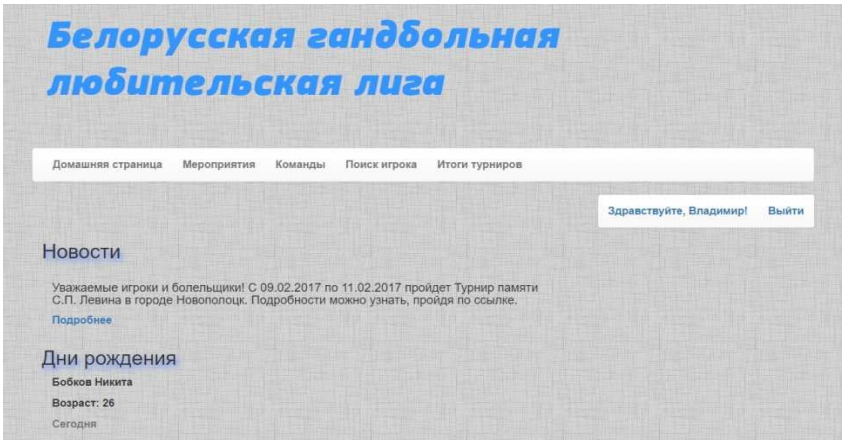
Актуальность создания веб-приложения обоснована еще и тем, что с развитием портативных компьютеров и мобильных устройств, клиентская часть веб-приложения может быть использована не только на стационарных компьютерах и ноутбуках, но и на упомянутых мобильных устройствах.

Средства решения задачи. Рассмотрим основные подходы и средства реализации интерфейса разработанного веб-приложения, освещающего любительский гандбол Республики Беларусь. В качестве системы авторизации и аутентификации будет использоваться система ASP.NET Identity, которая позволяет авторизоваться через внешние сервисы, управлять ролями для разграничения доступа к данным, производить подтверждение email и телефона посредством SMS, производить валидацию паролей [1].

Для оптимизации внешнего вида сайта под различные виды устройств и упрощения задачи графического оформления сайта использовались Twitter Bootstrap, jQuery, AJAX. *jQuery* – javascript библиотека, использование которой делает разработку на javascript кода намного быстрее и проще. В недалеком прошлом эта библиотека позволяла быстро разрабатывать скрипты. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX [2]. *AJAX*, или *Asynchronous Javascript And Xml*, – технология для взаимодействия с сервером без перезагрузки страниц. За счет этого уменьшается время отклика и веб-приложение по интерактивности больше напоминает десктоп. Например, вы можете оставлять комментарии на странице подведения итогов проведения соревнований и сразу видеть результат добавления комментария без перезагрузки страницы. Для обмена данными с сервером используется специальный объект XmlHttpRequest, который умеет отправлять запрос и получать ответ с сервера. При обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее [3].

Bootstrap – это фреймворк, который разработала Twitter Inc. Он предназначен для облегчения построения графического интерфейса. Сама библиотека включает в себя огромное количество элементов, таких как кнопки, веб-формы, блоки навигации и многое другое. Конечно, если изначально ничего не менять, то полученная форма будет похожа на то, что получается у многих других разработчиков, использующих аналогичный фреймворк, но следует учитывать, что на ранних этапах разработки может быть получен достаточно аккуратный интерфейс и в итоге его всё равно можно настроить. Исходные коды распространяются и лицензией MIT, что позволяет свободно использовать и изменять технологию [4].

Проектирование интерфейса. На главной странице сайта можно увидеть новости из мира гандбола, внизу экрана – информацию о ближайших Днях рождения игроков. Вверху каждой страницы сайта с помощью мастер-страницы выводится навигационная панель и кнопки регистрации, входа, выхода из личного кабинета (рисунок). При нажатии на ссылку «Регистрация» пользователь будет перенаправлен на страницу регистрации. На данной странице необходимо ввести имя, уникальный email, пароль и подтверждение пароля. При правильном заполнении форм пользователь будет перенаправлен на страницу входа в аккаунт.



Главная страница сайта

В личном кабинете пользователь может регистрировать новые спортивные мероприятия, но при условии, что его email подтвержден. Для подтверждения email необходимо нажать ссылку «Подтвердить email», после чего на email пользователя будет отправлено письмо с дальнейшей инструкцией. Также в правой части экрана на странице личного кабинета пользователь может изменить аватар, загрузив изображение размером до 4 Мб. В левой части экрана на странице личного кабинета пользователь может просмотреть список всех мероприятий, которые он зарегистрировал ранее.

При выборе пункта «Итоги турниров» на навигационной панели пользователь попадет на страницу с перечнем прошедших спортивных мероприятий. Выбрав из списка мероприятий один из турниров, пользователь сможет просмотреть итоги проведения турнира с фото- и видеоотчётом, а также, если пользователь авторизован, поделиться комментариями внизу страницы. Динамичность добавления комментариев без перезагрузки страницы получена с помощью использования веб-технологии AJAX для формы добавления новых комментариев. При редактировании спортивного мероприятия администратор помимо редактирования основной информации о турнире может прикрепить фото и видео к выбранному мероприятию, а также с помощью встроенного в страницу wysiwyg-редактора [5] красиво оформить текстовое описание проведения мероприятия.

При выборе пункта «Поиск игрока» на навигационной панели пользователь попадет на страницу, содержащую таблицу со списком всех игроков, принимающих участие в любительских турнирах.

Пользователю предоставлен следующий функционал на данной странице:

1) выбор количества игроков, отображаемых на странице;
2) сортировка игроков по любому полю;
3) постраничный вывод списка игроков;
4) динамический поиск игрока по любому из полей.

Данный функционал достигнут с использованием jQuery-плагина DataTables [6], который помимо данных функций предоставляет широкий ряд настроек и кастомизаций для таблиц.

При выборе пункта «Команды» на навигационной панели пользователь попадет на страницу, содержащую перечень команд. При нажатии на название одной из команд происходит переход на страницу, содержащую подробную информацию о выбранной команде.

Заключение. При разработке веб-интерфейсов необходимо учитывать не только красивую внешнюю составляющую, но и такие важные факторы, как «юзабельность», простота использования, понятность для конечного пользователя, независимость от браузера, кроссплатформенность, адаптивность для различных размеров экранов, динамичность. Всех этих требований можно достичь, используя абсолютно небольшой набор фреймворков и библиотек, в данном случае – это Twitter Bootstrap, AJAX, jQuery.

ЛИТЕРАТУРА

1. Metanit-сайт о программировании [Электронный ресурс]. – Режим доступа: http://metanit.com. – Дата доступа: 27.09.2016.
2. jQuery – Описание библиотеки [Электронный ресурс]. – Режим доступа: https://htmlweb.ru/java/jquery.php. – Дата доступа: 27.09.2016.
3. Введение в AJAX [Электронный ресурс]. – Режим доступа: http://javascript.ru/ajax/intro. – Дата доступа: 27.09.2016.
4. Bootstrap [Электронный ресурс]. – Режим доступа: http://getbootstrap.com. – Дата доступа: 27.09.2016.
5. Bootstrap-wysiwyg-редактор [Электронный ресурс]. – Режим доступа: https://mindmup.github.io/bootstrap-wysiwyg. – Дата доступа: 27.09.2016.
6. jQuery-плагин DataTables [Электронный ресурс]. – Режим доступа: https://datatables.net. – Дата доступа: 27.09.2016.

УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ АРХИТЕКТУРЫ СЕРВЕРНОЙ ЧАСТИ НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ «ВЕБ-ПОРТАЛ ЛЮБИТЕЛЬСКОГО ГАНДБОЛА РЕСПУБЛИКИ БЕЛАРУСЬ»

В.О. ТРАПЕЗНИКОВ
(Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН)

Рассматриваются технологии, используемые при реализации серверной части веб-приложений. Анализируется фрейморк ADO.NET Entity Framework, подход CodeFirst. Фреймворк ADO.NET Entity Framework позволяет создать базу данных, содержащую всю информацию и функционал, необходимый для комфортной работы веб-приложения; подход CodeFirst предоставляет возможность сгенерировать базу данных с нуля лишь на основе написанных классов моделей.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент – сервер». Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Серверная часть веб-приложения – это программа или скрипт на сервере, обрабатывающая запросы пользователя. При каждом переходе пользователя по ссылке браузер отправляет запрос к серверу. Сервер обрабатывает этот запрос, вызывая некоторый скрипт, который формирует веб-страницу, описанную языком HTML, и отсылает клиенту по сети. Браузер тут же отображает полученный результат в виде очередной веб-страницы.

Также практически ни одно веб-приложение не может обойтись без использования базы данных. База данных (БД, или система управления базами данных, СУБД) – программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент. В данном случае в базе данных хранится информация о пользователях, игроках, результатах проведения соревнований, командах. База данных располагается на сервере. Серверная часть веб-приложения обращается к базе данных по необходимости, извлекая данные, которые необходимы для формирования страницы, запрошенной пользователем.

Средства решения задачи. Рассмотрим основные средства и технологии реализации серверной части веб-приложения «Веб-портал любительского гандбола Республики Беларусь».

В работе был использован ASP.NET MVC Framework – фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller. Данный фреймворк добавлен Microsoft в ASP.NET. Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления. Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Модель представляет слой, описывающий логику организации данных в приложении. Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации. При обработке запросов фреймворк ASP.NET MVC опирается на систему маршрутизации, которая сопоставляет все входящие запросы с определенными в системе маршрутами, которые указывают, какой контроллер и метод должен обработать данный запрос. Встроенный маршрут по умолчанию предполагает трехзвенную структуру: контроллер/действие/параметр [1; 2].

В качестве IoC-контейнера будет использоваться Ninject. Собственно IoC (Inversion of Control (инверсия управления)) – это некий абстрактный принцип, набор рекомендаций для написания слабо связанного кода, суть которого в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов. Dependency Injection (внедрение зависимостей) – это одна из реализаций этого принципа, а IoC-контейнер – это фреймворк, который позволит упростить и автоматизировать написание кода с использованием данного подхода настолько, насколько это возможно [1; 2].

При реализации архитектуры приложения будет использоваться «луковая» архитектура. Onion-архитектура представляет собой разделение приложения на уровни. Причем есть один независимый уровень, который находится в центре архитектуры. От этого уровня зависит второй уровень, от второго – третий, и так далее. То есть получается, что вокруг первого независимого уровня наслаивается второй – зависимый. Вокруг второго наслаивается третий, который также может зависеть и от первого. Образно это может быть выражено в виде луковицы, в которой также есть сердцевина, вокруг которй наслаиваются все остальные слои, вплоть до шелухи.

Количество уровней может отличаться, но в центре всегда находится модель домена (Domain Model), то есть те классы моделей, которые используются в приложении и объекты которых хранятся в базе дан-

ных. Первый уровень вокруг модели домена образуют интерфейсы, которые управляют работой с моделью домена. Обычно это интерфейсы репозиториев, через которые мы взаимодействуем с базой данных.

Внешний уровень представляет такие компоненты, которые часто изменяются. Обычно внешний уровень образуют пользовательский интерфейс, тесты, какие-то вспомогательные классы инфраструктуры приложения. К этому уровню также относятся конкретные реализации интерфейсов, объявленных на нижележащих уровнях. Например, реализация интерфейса репозитория, который объявлен на уровне Domain Services. Вообще все внутрение уровни, которые можно объединить в Application Core, определяют только интерфейсы, а конкретная реализация этих интерфейсов располагается на внешнем уровне.

Также стоит отметить, что все внешние хранилища, как базы данных, файлы, внешние веб-сервисы, от которых мы можем получать данные, – все это является внешним по отношению к архитектуре [1; 2].

Для работы с базой данных будет использоваться фреймворк ADO.NET Entity Framework, подход CodeFirst. ADO.NET Entity Framework (EF) – объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM [1; 2].

При проектировании приложений с подходом CodeFirst, сначала создаются классы модели данных, не обращая никакого внимания на Entity Framework. После того как нам понадобилось работать с базой данных, используем различные инструменты, которые проецируют структуру базы данных из созданной модели классов. После этого можем вернуться к этой модели в коде и, например, изменить её. Эти изменения затем можно будет отразить в базе данных, используя все те же инструменты.

Методы решения задач. В соответствии с выполняемыми функциями, веб-приложение можно разбить на несколько модулей.

1 *Модуль работы с аккаунтом пользователя реализует функции*:
а) регистрация нового пользователя;
б) вход в аккаунт;
в) редактирование персональных данных;
г) подтверждение email;
д) восстановление пароля.
2 *Модуль работы с информацией о тренерах реализует функции*:
а) создание нового тренера;
б) удаление тренера;
в) редактирование информации о тренере.
3 *Модуль работы с информацией о комментариях реализует функции*:
а) создание комментариев;
б) удаление комментариев.
4 *Модуль работы с информацией об играх реализует функции*:
а) создание результатов игры;
б) удаление результатов игры;
в) редактирование информации об игре.
5 *Модуль работы с информацией о спортивных мероприятиях реализует функции*:
а) регистрация нового мероприятия;
б) редактирование мероприятия;
в) удаление мероприятия;
г) добавление результатов проведения мероприятия;
д) просмотр результатов проведения спортивного мероприятия.
6 *Модуль работы с информацией о новостях реализует функции*:
а) создание новостей;
б) удаление новостей;
в) редактирование новостей.
7 *Модуль работы с информацией об игроках реализует функции*:
а) создание нового игрока;
б) удаление игрока;
в) редактирование информации об игроке;
г) поиск игрока.

8 *Модуль работы с информацией о судьях реализует функции*:
а) создание нового судьи;
б) удаление судьи;
в) редактирование информации о судье.
9 *Модуль работы с информацией о командах реализует функции*:
а) создание новой команды;
б) удаление команды;
в) редактирование информации о команде;
г) просмотр информации о команде.

Основной функционал серверной части содержится в следующих классах:

1 AccountController – класс контроллера для работы с пользовательскими аккаунтами.
2 CoachController – класс контроллера для работы с информацией о тренерах.
3 CommentController – класс контроллера для работы с комментариями.
4 EventController – класс контроллера для работы с информацией о спортивных мероприятиях.
5 GameController – класс контроллера для работы с информацией о прошедших играх.
6 HomeController – класс контроллера для работы с отображением информации на главной странице.
7 NewsController – класс контроллера для работы с информацией о новостях.
8 PlayerController – класс контроллера для работы с информацией об игроках.
9 RefereeController – класс контроллера для работы с информацией о судьях.
10 ResultController – класс контроллера для работы с информацией о результатах проведения спортивных мероприятий.
11 TeamController – класс контроллера для работы с информацией о командах.

Заключение. Использование модели MVC позволило четко разграничить функционал серверной части, не смешивая его составные части, что привело к простоте понимания и последующего изменения кода.

Фреймворк ADO.NET Entity Framework позволил создать базу данных, содержащую всю информацию и функционал, необходимый для комфортной работы веб-приложения, без вникания в тонкости SQL.

Подход CodeFirst дал возможность сгенерировать базу данных с нуля лишь на основе написанных классов моделей.

Использование IoC-контейнера Ninject сделал веб-приложение гибким для изменений. Надо лишь реализовать интерфейс нужным нам образом на верхнем слое приложения, не углубляясь в нижние слои.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/. Дата доступа: 27.09.2016.
2. Metanit – сайт о программировании [Электронный ресурс]. – Режим доступа: http://metanit.com. – Дата доступа: 27.09.2016.

УДК 621.375.026

РАЗРАБОТКА ВЫСОКОЧАСТОТНОЙ ЧАСТИ УСТРОЙСТВА УПРАВЛЕНИЯ УДАЛЕННОЙ СВЧ АППАРАТУРОЙ ПО КОАКСИАЛЬНОЙ ЛИНИИ

А.В. КАРАСЬ
(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Представлена высокочастотная часть устройства управления удаленной СВЧ аппаратурой по коаксиальной линии. Данная схема способна компенсировать усилением затухание в коаксиальном кабеле и передавать данные по ней, способные управлять СВЧ конвертером либо активной антенной.

Для передачи сигнала и его преобразования используются следующие элементы, способные увеличить амплитуду передаваемого сигнала в режимах усиления 12, 16, 20 и 24 Дб, а также пропускать сигнал без преобразований в режиме «обход». Высокочастотная часть (ВЧ) часть электрической схемы показана на рисунке 1.

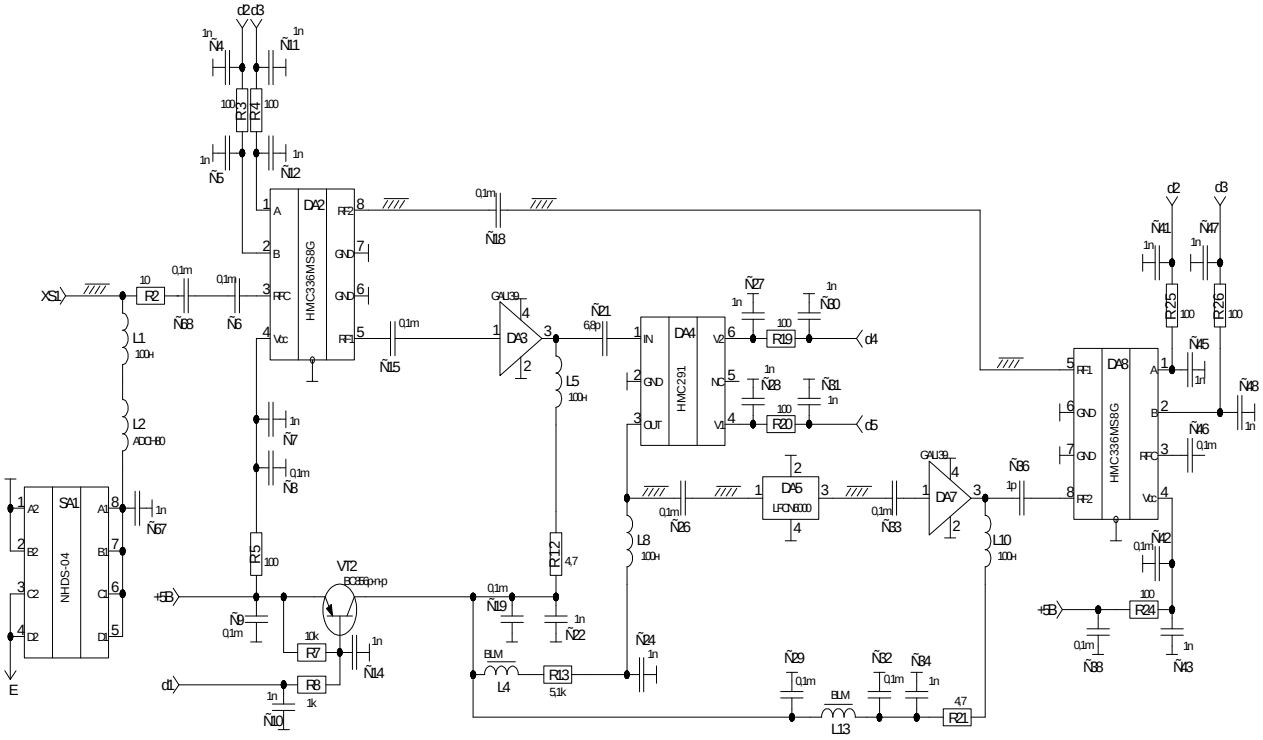


Рисунок 1. – Высокочастотная часть

Для усиления передаваемого по линии сигнала используются микросхемы DA3, DA7. Усилители включены постоянно, поэтому режимами усиления управляет аттенюатор, ослабляющий выходное значение усиления.

Переключатель NHDS-04 включает необходимую цепь, таким образом можно сделать XS1 и XS2 как входом, так и выходом. В качестве переключателя режимов «обход» и «усиление» был выбран коммутатор HMC336MS8G. Этот переключатель обеспечивают высокую изоляцию и маленькие потери. Предназначен для переключения 50 Ом линий. Схема обвязки изображена на рисунке 2.

Элементы обвязки жестко заданы даташитом [1]:

$$C1 - C3 = 100 \text{ пФ}, \tag{1}$$

$$C4 = 10 \text{ пФ}, \tag{2}$$

$$R1 - R2 = 100 \text{ Ом}. \tag{3}$$

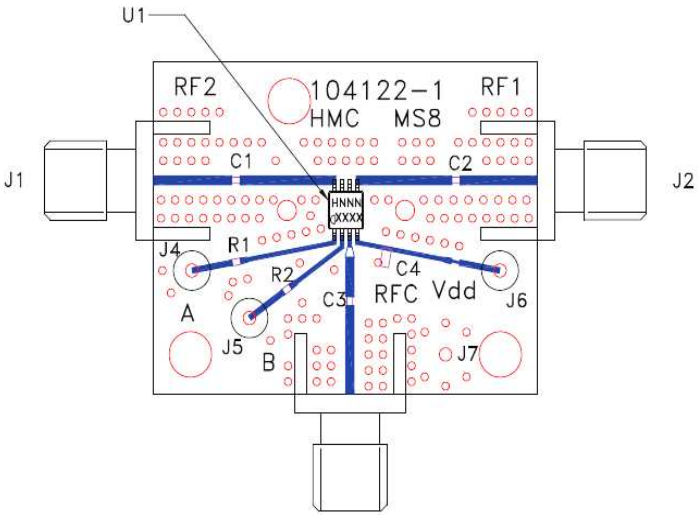


Рисунок 2. – Схема обвязки HMC336MS8G

Для усиления ВЧ-сигнала в схеме используется два операционных усилителя GALI39[2]. Схема обвязки изображена на рисунке 3.

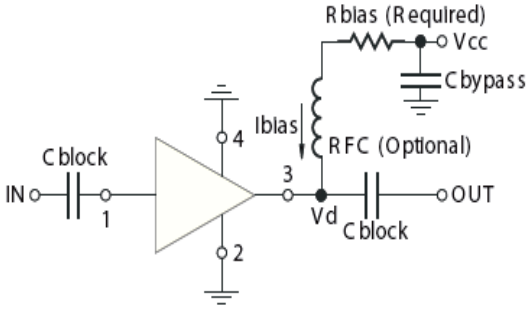


Рисунок 3. – Рекомендуемая схема обвязки GALI39

В качестве аттенюатора была выбрана микросхема HMC291, способная ослаблять сигнал до 12 Дб. Структурная схема изображена на рисунке 4.

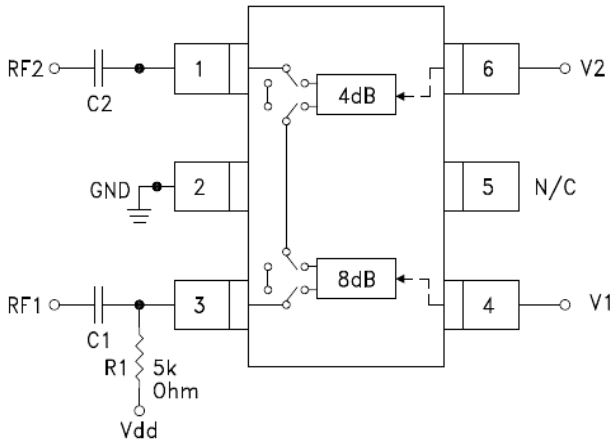


Рисунок 4. – Структурная схема HMC291

Конденсаторы *C*1 и *C*2 выбираются с тем условием, чтобы сигнал мог пройти с минимальными потерями. Даташит рекомендует емкости 100 ~ 300 пФ [3].

Таким образом, в схеме обвязки аттенюатора *C*1 = C2 = 100 пФ.

Во избежание наводок цепь должна быть сформирована, как показано на рисунке 5.

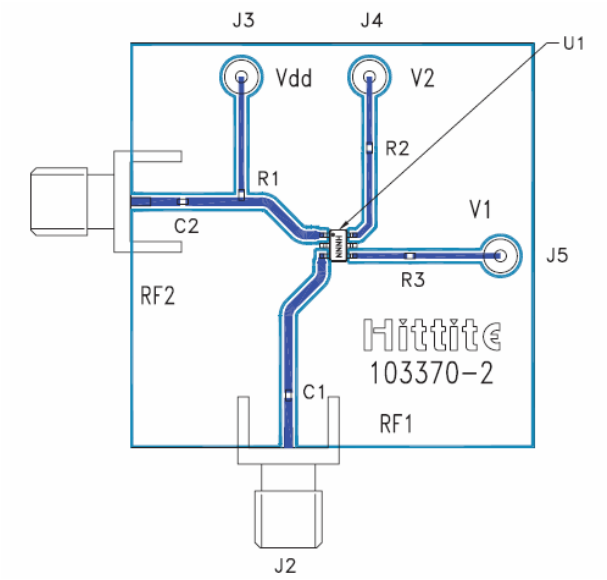


Рисунок 5. – Трассировка печатной платы элемента HMC291

Резисторы *R*2 и *R*3 используются для лучшей развязки сигналов RF от управляющих входов.
*R*2 = *R*3 = 100 Ом.

Сигнальные линии должны иметь 50 Ом импеданс, в то время как земельный вывод должен быть подключен непосредственно к земле. Достаточное количество переходных отверстий должны быть использованы для соединения слоев платы.

ЛИТЕРАТУРА

1. Positive Control Switch [Электронный ресурс]: Datasheet / Hittite. – Режим доступа: http://www.hittite.com/content/documents/data_sheet/hmc336ms8g.pdf. – Дата доступа: 12.09.2016.
2. Monolithic Amplifier [Электронный ресурс]: Datasheet / Surface Mount. – Режим доступа: http://www.minicircuits.com/pdfs/GALI-39+.pdf. – Дата доступа: 12.09.2016
4. Attenuator [Электронный ресурс]: Datasheet / Hittite. – Режим доступа: http://www.hittite.com/content/ documents/data_sheet/hmc291.pdf. – Дата доступа: 12.09.2016

УДК 621.3

ДИЭЛЕКТРИЧЕСКАЯ ПРОНИЦАЕМОСТЬ СРЕДЫ НАД УГЛЕВОДОРОДНЫМИ ЗАЛЕЖАМИ В РЕЖИМЕ ЧАСТОТНО-МОДУЛИРОВАННЫХ СИГНАЛОВ

Е.Р. АДАМОВСКИЙ
(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Рассматриваются вопросы взаимодействия частотно-модулированных сигналов с анизотропными средами над углеводородными залежами. Используется вертикальная поляризация электромагнитных волн. Даны рекомендации по использованию оптимальных характеристик зондирующих сигналов.

Вопросы поиска, выделения и оконтуривания анизотропных сред плазмоподобного типа (АСПТ) представляют интерес во многих областях науки и техники. Одним из приоритетных направлений является георазведка углеводородных залежей (УВЗ). Подобие многих процессов, происходящих над месторождениями нефти и газа, со свойствами анизотропных сред (АС) позволяет использовать теоретические и практические наработки в области исследования плазмы и плазмоподобных сред при разработке современных электромагнитных методов (ЭММ) георазведки углеводородов [1].

Исследование взаимодействия электромагнитных волн (ЭМВ) с УВЗ может быть использовано в поисковой геофизике для повышения точности и уровня достоверности ЭММ обнаружения залежей нефти и газа. Применение частотно-модулированных (ЧМ) сигналов с вариацией частот позволяет проводить точные оценки трансформации фазовых характеристик отраженных ЭМВ и интерпретировать всевозможные эффекты взаимодействия ЭМВ с АС [2–6].

Объекты и методы исследования

В данной работе проведен анализ компонентов тензора среды над УВЗ в режиме ЧМ-сигналов [4]:

$$\begin{cases} \dot{\varepsilon}_1 = \varepsilon_r(1+\beta\cdot k_\omega\cos\omega_1 t)+\sum\limits_{i=1}^{2}\left\{\begin{array}{l}\dfrac{\omega_{\Pi i}^2\tilde{\omega}_3}{\omega_2}\dfrac{\omega_{\Gamma i}^2-\tilde{\omega}_3^2-\nu_i^2}{(\nu_i^2+\omega_{\Gamma i}^2-\tilde{\omega}_3^2)^2+4\tilde{\omega}_3^2\nu_i^2}- \\ -j\left[\dfrac{\sigma_r}{\omega_2\varepsilon_0}+\dfrac{\omega_{\Pi i}^2\nu_i}{\omega_2}\dfrac{\tilde{\omega}_3^2+\nu_i^2+\omega_{\Gamma i}^2}{(\nu_i^2+\omega_{\Gamma i}^2-\tilde{\omega}_3^2)^2+4\tilde{\omega}_3^2\nu_i^2}\right]\end{array}\right\}, \\ \dot{\varepsilon}_2 = \sum\limits_{i=1}^{2}\left\{\dfrac{\omega_{\Pi i}^2\tilde{\omega}_3}{\omega_2}\dfrac{\omega_{\Gamma i}^2-\tilde{\omega}_3^2+\nu_i^2}{(\nu_i^2+\omega_{\Gamma i}^2-\tilde{\omega}_3^2)^2+4\tilde{\omega}_3^2\nu_i^2}-\dfrac{2j\tilde{\omega}_3\nu_i\omega_{\Pi i}^2\omega_{\Gamma i}}{\left[(\nu_i^2+\omega_{\Gamma i}^2-\tilde{\omega}_3^2)^2+4\tilde{\omega}_3^2\nu_i^2\right]\omega^2}\right\}, \\ \dot{\varepsilon}_3 = \varepsilon_r(1+\beta\cdot k_\omega\cos\omega_1 t)+\sum\limits_{i=1}^{2}\left\{\dfrac{\omega_{\Pi i}^2\tilde{\omega}_3}{\omega_2}\dfrac{1}{\nu_i^2+\tilde{\omega}_3^2}-j\left[\dfrac{\sigma_r}{\omega_2\varepsilon_0}+\dfrac{\omega_{\Pi i}^2\nu_i}{\omega_2}\dfrac{1}{\tilde{\omega}_3^2+\nu_i^2}\right]\right\}. \end{cases} \quad (1)$$

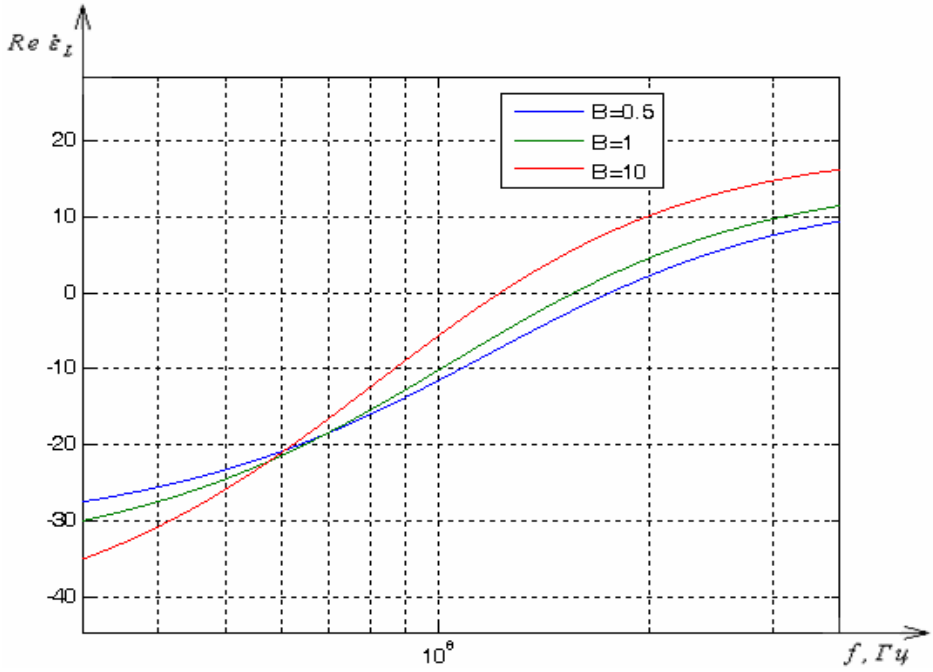
Было проведено исследование частотных зависимостей компонентов тензоров диэлектрической проницаемости среды над УВЗ в различных режимах взаимодействия ЭМВ для оптимизации параметров сигналов при поиске, оконтуривании и выделении залежей углеводородов.

Результаты исследования

Анализ тензоров проведем по методике, приведенной в работе [2], заключающейся в исследовании частотных характеристик комбинационных составляющих:

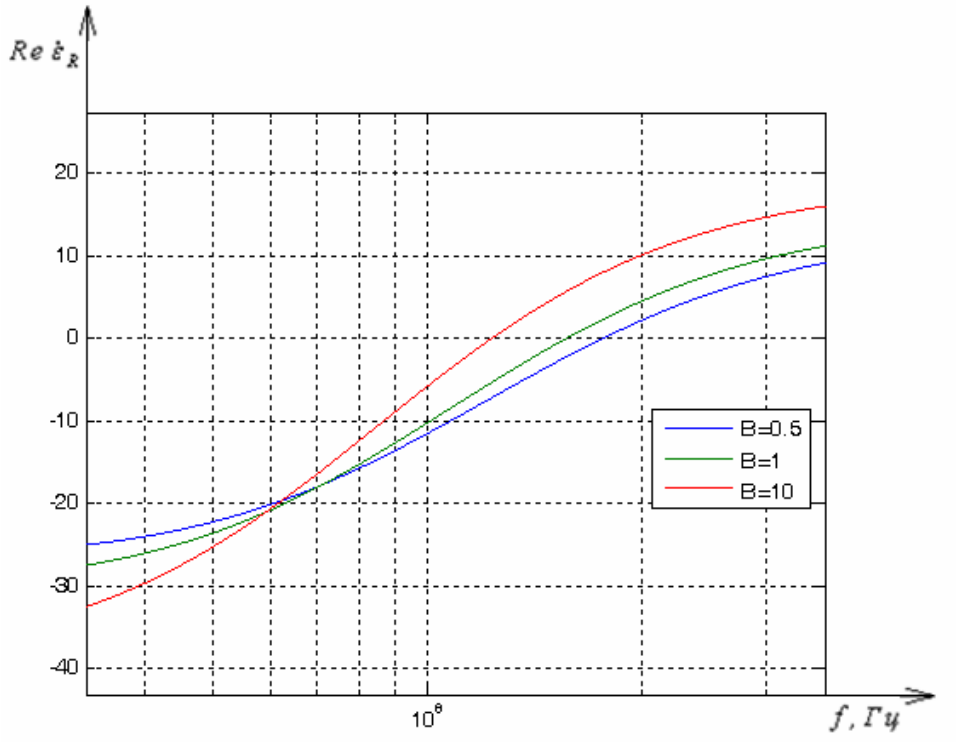
$$\begin{aligned} b_2\dot{\varepsilon}_R(\omega) &= \dot{\varepsilon}_1+\dot{\varepsilon}_2 = \mathrm{Re}\,\varepsilon_R + j\,\mathrm{Im}\,\varepsilon_R, \\ \dot{\varepsilon}_L(\omega) &= \dot{\varepsilon}_1-\dot{\varepsilon}_2 = \mathrm{Re}\,\varepsilon_L + j\,\mathrm{Im}\,\varepsilon_L. \end{aligned} \quad (2)$$

Приведенные зависимости показывают, что увеличение индекса модуляции приводит к несущественному увеличению частоты циклотронного электронного резонанса и резкому уменьшению частоты плазменного резонанса (рис. 1). Это же характерно и для $\mathrm{Re}\,\dot{\varepsilon}_L$ (рис. 2).



1 – для β = 0,5; 2 – для β = 1; 3 – для β = 10

Рисунок 1. – Зависимости Re $\dot{\varepsilon}_R\varphi(f_2)$:



1 – для β = 0,5; 2 – для β = 1; 3 – для β = 10

Рисунок 2. – Зависимости Re $\dot{\varepsilon}_R\varphi(f_2)$

Заключение

С целью получения полной картины взаимодействия ЭМВ и исследуемой среды использован широкий диапазон соотношений частот и амплитуд сигналов.

На основе проведенного анализа приводятся рекомендации по использованию наиболее эффективных соотношений частот и амплитуд ЭМВ, обеспечивающих точное выделение границ УВЗ на фоне окружающей среды.

Полученные результаты могут быть использованы при разработке радиотехнических систем для обнаружения локальных плазмоподобных неоднородностей, а также создания оптимальных методов поиска и оконтуривания залежей нефти и газа.

ЛИТЕРАТУРА

1. Гололобов, Д.В. Радиотехнические системы поиска и идентификации углеводородных залежей в режиме двухчастотного взаимодействия / Д.В. Гололобов, В.Ф. Янушкевич // Весці НАН Беларусі. Сер. фіз.-тэхн. – 2002. – № 1. – С. 49–54.
2. Moskvichew, V.N. Interraction of electromagnetic waves (EMW) with anisotropic inclusion in communication line / V.N. Moskvichew // 9-th Microw. Conf. NICON-91, Rydzyna, May 20–22, 1991. – Vol. 1. – P. 240–244.
3. Гололобов, Д.В. Поверхностный импеданс углеводородной залежи в режиме двухчастотного взаимодействия / Д.В. Гололобов, Н.В. Цывис, В.Ф. Янушкевич // Изв. Беларус. инж. акад. – Минск, 2001. – № 1(11). – С. 101–104.
4. Гололобов, Д.В. Поверхностный импеданс среды над углеводородными залежами в режиме частотно-модулированных сигналов / Д.В. Гололобов, С.В. Калинцев, В.Ф. Янушкевич // Весці НАН Беларусі. Сер.фіз. тэхн. – 2010. – № 4. – С. 98–101.
5. Москвичев, В.Н. Исследование взаимодействий электромагнитных волн с углеводородной залежью / В.Н. Москвичев // Радиотехника и электроника. – Минск, 1988. – Вып. 18. – С. 91–96.
6. Финкельштейн, М.И. Применение радиолокационного подповерхностного зондирования в инженерной геологии / М.И. Финкельштейн, В.А. Кутев, В.П. Золоторев. – М. : Недра, 1986. – 182 с.

УДК 621.3

ФАЗОВЫЕ ХАРАКТЕРИСТИКИ ДИЭЛЕКТРИЧЕСКОЙ ПРОНИЦАЕМОСТИ СРЕДЫ НАД УГЛЕВОДОРОДНЫМИ ЗАЛЕЖАМИ В РЕЖИМЕ ЧАСТОТНО-МОДУЛИРОВАННЫХ СИГНАЛОВ

Е.Р. АДАМОВСКИЙ
(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Рассматриваются фазовые характеристики диэлектрической проницаемости среды над углеводородными залежами в режиме частотно-модулированных сигналов. Анализируются взаимодействия анизотропного слоя плазмоподобного типа с ЭМВ в режиме ЧМ-сигналов.

Актуальность рассматриваемых в настоящей работе задач заключается в усовершенствовании существующих электромагнитных методов (ЭММ) георазведки и разработке новых методов поиска, идентификации месторождений нефти и газа (углеводородов), являющихся стратегическим видом полезных ископаемых и определяющих широкий спектр глобальных экологических вопросов современного общества [1–3]. Исследование фазовых характеристик взаимодействия электромагнитных волн (ЭМВ) с углеводородными залежами (УВЗ) в режиме частотно-модулированных (ЧМ) сигналов может быть использовано в поисковой геофизике для повышения точности и уровня достоверности ЭММ обнаружения залежей нефти и газа.

Объекты и методы исследования. Модель многослойной среды включает слой 2 толщиной h с плоскими границами раздела, имеющий относительную диэлектрическую проницаемость $\dot{\varepsilon}_2$, расположенный между полубесконечными средами 1 и 3 с относительными диэлектрическими проницаемостями $\dot{\varepsilon}_1 = 1$ и $\dot{\varepsilon}_3$ (рис. 1).

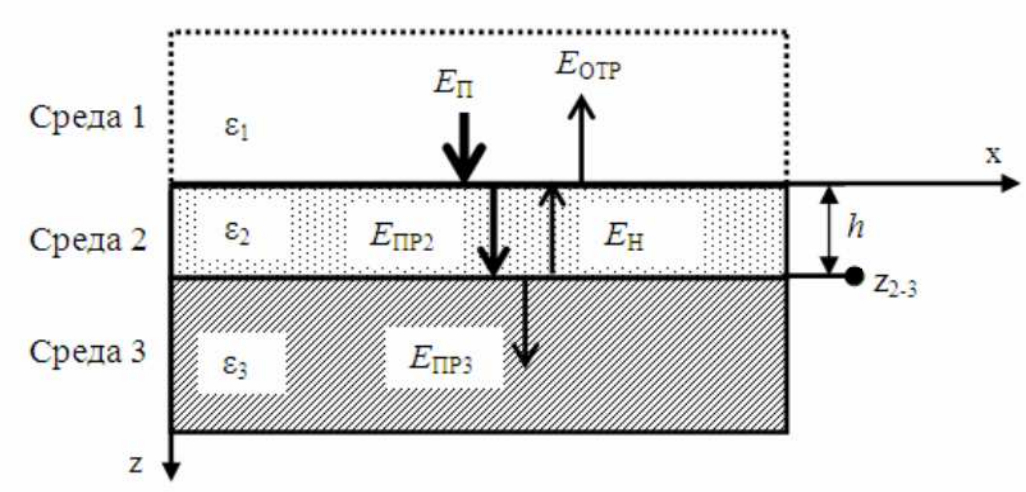


Рисунок 1. – Фрагмент слоистой среды и отражения электромагнитной волны

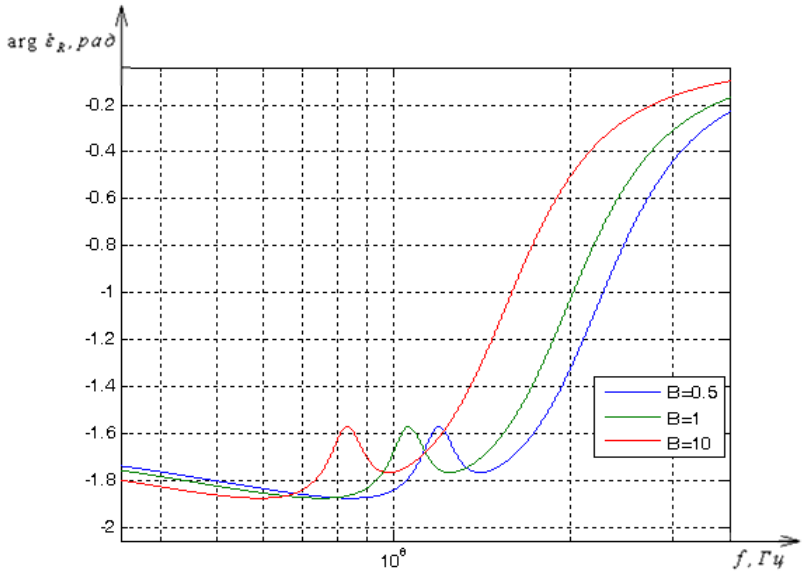
На границу раздела сред 1 и 2 нормально падает ЭМВ с напряженностью поля $E_П$. Во вторую среду проникает волна $E_{ПР2}$, в третью среду проникает волна $E_{ПР3}$, от нижней границы слоя 2 отражается волна $E_Н$, от границы слоя 1 отражается волна $E_{ОТР}$. На границе 2 и 3 сред образуется импеданс $Z_{2\text{-}3}$.

Результаты исследования

Проведен анализ фазовых характеристик диэлектрической проницаемости среды над УВЗ (рис. 2, 3) для радиосигнала с тональной ЧМ вида

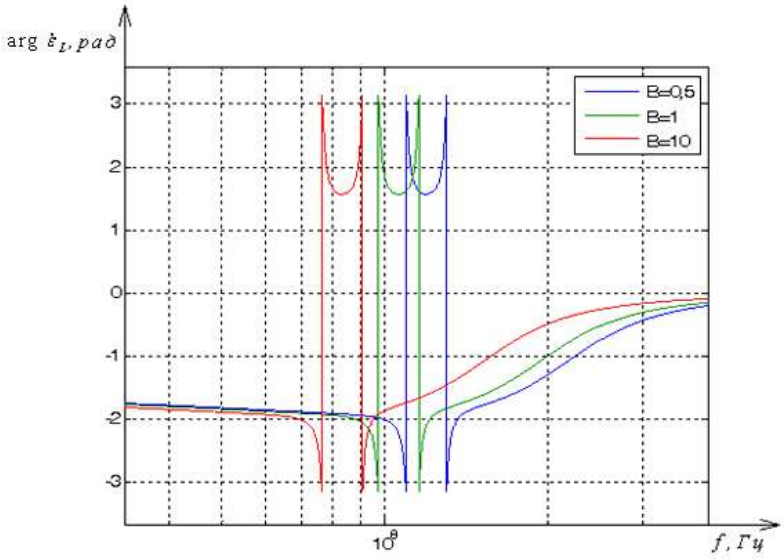
$$e(t) = E_2 \cos(\omega_2 t + \beta \sin \omega_1 t), \quad (1)$$

где E_2 и ω_2 – соответственно амплитуда и частота несущего колебания; ω – модулирующая частота; $\beta = \dfrac{\Delta\omega}{\omega_1}$ $\dot{\varepsilon}_2$ – индекс модуляции; $\Delta\omega$ – девиация частоты.



1 – для β = 0,5; 2 – для β = 1; 3 – для β = 10

Рисунок 2. – Зависимости arg $\dot{\varepsilon}_R\varphi(f_2)$



1 – для β = 0,5; 2 – для β=1; 3 – для β = 10

Рисунок 3. – Зависимости arg $\dot{\varepsilon}_R\varphi(f_2)$

Характер фазочастотных характеристик $\dot{\varepsilon}_R$ и $\dot{\varepsilon}_L$ для различных β не изменяется . При этом с ростом индекса модуляции частота, на которой $\dot{\varepsilon}_{R,L} = 0$, соответствующая ВЧ-резонансу, уменьшается.

Заключение. Проведен анализ взаимодействия анизотропного слоя плазмоподобного типа с ЭМВ в режиме ЧМ-сигналов. Вариация характеристик зондирующих сигналов позволяет повысить информативность методов оконтуривания и выделения УВЗ. Исследована трансформация фазовых характеристик отраженных ЭМВ от АС. Результаты исследования могут быть исследованы в поисковой геофизике.

ЛИТЕРАТУРА

1. Moskvichew, V.N. Interraction of electromagnetic waves (EMW) with anisotropic inclusion in communication line / V.N. Moskvichew // 9-th Microw. Conf. NICON-91, Rydzyna, May 20–22, 1991. – Vol. 1. – P. 240–244.
2. Финкельштейн, М.И. Применение радиолокационного подповерхностного зондирования в инженерной геологии / М.И. Финкельштейн, В.А. Кутев, В.П. Золоторев. – М. : Недра, 1986. – 182 с.
3. Москвичев, В.Н. Исследование взаимодействия электромагнитных волн с углеводородной залежью / В.Н. Москвичев // Радиотехника и электроника. – Минск : Высш. школа, 1989. – Вып. 18. – С. 91–96.

537.872.2

ВОЗМОЖНЫЕ ОБЛАСТИ ПРИМЕНЕНИЯ СВЕРХВЫСОКОЧУВСТВИТЕЛЬНЫХ РАДИОПРИЁМНЫХ УСТРОЙСТВ

Д.И. ШИШКОВ
(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Рассматриваются возможные области применения сверхвысокочувствительных радиоприёмных устройств. Показана актуальность данной темы – для приёма маломощных сигналов на больших расстояниях требуются малошумящие приёмники, особенно для космической сферы.

Введение 1. Чувствительность – способность объекта реагировать определённым образом на определённое малое воздействие, а также количественная характеристика этой способности. Чувствительность радиоприёмника – способность радиоприёмника принимать слабые по интенсивности радиосигналы и количественная мера этой способности. Чувствительность, ограниченная шумами, – чувствительность радиоприёмника, определяемая минимальным уровнем радиосигнала на его входе при заданном отношении уровней полезного сигнала и шума и заданном уровне полезного сигнала на выходе радиоприёмника. Источники шума могут находиться как вне, так и внутри самой системы передачи.

В общем виде влияние помехи k на передаваемый сигнал s может быть выражено оператором

$$x = f(s,k). \tag{1}$$

В том частном случае, когда этот оператор вырождается в сумму

$$x = (s+k). \tag{2}$$

Помеха k называется аддитивной. Аддитивную помеху часто называют шумом.

Если же x может быть представлен в виде

$$x = \nu s, \tag{3}$$

где случайный процесс $\nu(t)$ неотрицателен, то помеху ν называют мультипликативной. Если ν – медленный (по сравнению с s) процесс, то явление, вызываемое мультипликативной помехой, носит название замирание (федингг).

В более общем случае оператор f не может быть приведён к основным формам (2) и (3). При одновременном наличии шума и мультипликативной помехи удобно ввести два случайных процесса, выражающих оба вида помехи:

$$x = (\nu s+k). \tag{4}$$

С физической точки зрения случайные помехи порождаются различного рода флуктуациями. Флуктуациями в физике называют случайные отклонения тех или иных физических величин от их средних значений. Так, источником шума в электрических цепях постоянного тока могут являться флуктуации тока около среднего значения, обусловленные дискретной природой носителей заряда (ионов и электронов). Это явление носит название дробового эффекта

Наиболее универсальной причиной шума являются флуктуации, обусловленные тепловым движением. Случайное тепловое движение носителей заряда в любом проводнике вызывает случайную разность потенциалов на его концах. Эта разность потенциалов флуктуирует около среднего значения, равного нулю; её средний квадрат пропорционален абсолютной температуре. Возникающая помеха называется тепловым шумом.

Из сказанного видно, что флуктуации и обусловленные ими помехи заложены глубоко в природе вещей. Флуктуации есть результат дискретного строения вещества и статистической природы ряда физических величин. Многие физические величины представляют результат усреднения по большому числу индивидуальных частиц, поведение и действие которых подчиняется законам случая. Поэтому флуктуации этих физических величин принципиально неустранимы, и можно лишь ставить вопрос о том, какова относительная величина флуктуации и каким образом мы можем на неё повлиять находящимися в нашем распоряжении средствами.

Имеется ещё один источник принципиально неустранимого шума, возникающего из-за дискретной природы электромагнитного излучения. Согласно современным воззрениям излучение совершается дис-

кретными порциями – квантами, энергия которых равна hv, где h – постоянная Планка, v – частота. Квант электромагнитного излучения называется фотоном.

В настоящее время в технике имеются две ясные тенденции: к увеличению расстояний и к повышению частоты. Увеличение расстояний означает уменьшение потока энергии, а повышение частоты – укрупнение фотонов.

Таким образом, при определённых условиях не только начинает ощущаться дискретная фотонная структура излучения, но обусловленный этой причиной шум может превзойти все остальные помехи. Канал, работающий при таких условиях, получил название фотонного канала.

Выше перечисленные шумы являются аддитивными, но имеется обширный класс мультипликативных помех.

Области применения сверхчувствительных приёмников 2. Исходя из вышесказанного понятно, что для достижения высоких скоростей передачи данных на больших расстояниях, в условиях большой зашумлённости требуется повышенная чувствительность приёмника.

Данные условия характерны для космических аппаратов, для них же эта тема является весьма актуальной, так как радиокоммуникация в данных аппаратах служит многим системам, таким как: система дистанционного управления курсом и действиями космического аппарата, система приёма полученных аппаратом данных на земле.

Сверхчувствительный приём также актуален и для наземных систем, которые в силу некоторых особенностей обладают довольно малой мощностью передатчика, например, автономные метеозонды, установленные в труднодоступных местах, таких как открытый океан, тропосфера, вершины гор, Арктика.

Такие приёмники благодаря высокой энергоэффективности и низкому энергопотреблению могли бы применься и в бытовой сфере.

Выводы

На данный момент рассматриваемая тема является весьма актуальной.

Несмотря на сложность реализации, данное направление стоит считать одним из наиболее перспективных в области проектирования радиоприёмных устройств – нахождение решения данной проблемы экономически выгодно и способствует развитию техники радиоприёма, особенно в космической сфере.

ЛИТЕРАТУРА

1. Бойерле, Х. Коммуникация в технике автоматизации: пер. с нем. / Х. Бойерле, П. Беценар, Г. Бах. – Берлин, Мюнхен : АО Siemens, 1991. – 155 с.
2. Варакин, Л.Е. Теория систем сигналов / Л.Е. Варакин. – М. : Советское радио, 1978. – 375 с.
3. Гоноровский, И.С. Радиотехнические цепи и сигналы : учебник для вузов / И.С. Гоноровский. – 4-е изд., перераб. и доп. – М. : Радио и связь, 1986. – 512 с.
4. Гук, М. Аппаратные средства локальных сетей. Энциклопедия / М. Гук. – СПб. : Питер, 2000. – 576 с.

537.872.31

ВАЖНОСТЬ ПРИЁМА СИГНАЛА С МИНИМИЗАЦИЕЙ ПОМЕХ

Д.И. ШИШКОВ
(Представлено: канд. техн. наук, доц. В.Ф. ЯНУШКЕВИЧ)

Исследуется влияние помех на приём сигналов. Рассматривается минимизация помех – одна из основных задач при разработке электроники, в частности радиоприемных устройств. Анализируется отношение сигнал/шум на количество ошибок при декодировании сигналов.

Введение 1. Отношение сигнал/шум – безразмерная величина, равная отношению мощности полезного сигнала к мощности шума:

$$\frac{S}{N} = \frac{P_{сигнала}}{P_{шума}}, \tag{1}$$

где $P_{сигнала}$ – средняя мощность сигнала; $P_{шума}$ – средняя мощность шума. Оба сигнала измеряются в полосе пропускания системы.

Обычно отношение сигнал/шум выражается в децибелах (дБ). Чем больше это отношение, тем меньше шум влияет на характеристики системы:

$$\frac{S}{N(dB)} = 10\log_{10}\left(\frac{P_{сигнала}}{P_{шума}}\right). \tag{2}$$

Влияние помех 2. Шумы определяют емкость канала и задают частоту ошибок при передаче цифровых данных. Шум по своей природе нестабилен и можно говорить лишь о том, что его величина с некоторой вероятностью лежит в определенном интервале значений. Плотность вероятности $p(x)$ определяет вероятность того, что случайный сигнал x имеет значение амплитуды в интервале между x и $x + Dx$. При этом вероятность того, что значение x лежит в интервале между x_1 и $x2$ определяется равенством:

$$\frac{S}{N} = \frac{сигнала}{P_{шума}} P\{x_1 < x_2\} = \int_{x_1}^{x_2} p(x)dx. \tag{3}$$

Условием нормировки при этом является равенство:

$$\int_{-\infty}^{\infty} p(x)dx = 1, \tag{4}$$

Здесь $P(x)$ – вероятность, $p(x)$ – плотность вероятности.

Вероятность того, что x меньше некоторой величины y равна:

$$\int_{-\infty}^{y} p(x)dx, \tag{5}$$

откуда следует, что $P\{x_{1,2}\} = P(x_2) - P\{x_1\}$, а $P(-\infty) = 1, \;\; P(-\infty) = 0.$

Так называемый белый шум подчиняется непрерывному нормальному (Гауссову) распределению:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-a)^2}{2\sigma^2}}, \tag{6}$$

где a – среднее значение x, а σ – среднеквадратичное отклонение x от a. В случае шумов среднее значение x с учетом полярности часто принимает нулевое значение $(a = 0)$.

В этом случае, если мы хотим знать вероятность того, что амплитуда шумового сигнала лежит в пределах $\pm v$, можно воспользоваться выражением:

$$P\{\nu < x < -\nu\} = \int_{-\nu}^{\nu} p(x)dx. \tag{7}$$

Шум определяет вероятность ошибки при передаче сообщения по каналу связи и, в конечном итоге, – пропускную способность канала.

Теорема Шеннона ограничивает предельную пропускную способность канала I с заданной полосой пропускания F и отношением сигнал/шум S/N:

$$I = F \cdot \log_2\left(1+\frac{S}{N}\right); \tag{8}$$

$$\frac{I}{F} \approx \cdot 1,44\frac{S}{N}. \tag{9}$$

Для стандартного телефонного канала F = 3 *кГц*, N/S = 30 *db*, следовательно, теоретический предел для публичной коммутируемой телефонной сети равен примерно 30 кбит/с. Ослабление для телефонных скрученных пар составляет около 15 дБ/км, дополнительные ограничения возникают из-за перекрестных наводок.

Если рассмотреть сигнал с полосой F, то согласно теореме Найквиста частота стробирования должна быть равна или больше 2F. При использовании больших частот стробирования можно получить при воспроизведении более высокие гармоники, но они при заданной полосе пропускания все равно будут подавлены. При K дискретных уровнях преобразования максимальный поток данных составит 2F log2(K) бит/с, что при F = *4* кГц/с и K = 256 даст 64 кбит/с. Практически при F = *4* кГц даже в отсутствие шума нельзя получить скорость передачи более 8 кбит/с (если передается один бит за такт).

Из теоремы Шеннона следует, что при нулевом уровне шума можно получить сколь угодно высокую скорость передачи при сколь угодно низкой полосе пропускания канала.

По существу, К. Шеннон развил идеи Найквиста.

Если используется двоичное представление сигнала, то согласно теореме Найквиста, максимальная скорость передачи данных I по каналу без шума составит:

$$I = F \cdot \log_2 V \text{ (бит/с)}, \tag{10}$$

где F – полоса пропускания канала в Гц, а V – число дискретных уровней сигнала на выходе цифрового преобразователя.

Суть теоремы Найквиста – Котельникова заключается в том, что при полосе сигнала F частота стробирования должна быть больше 2F, чтобы принимающая сторона могла корректно восстановить форму исходного сигнала. По этой причине для стандартного телефонного канала с полосой F = 3 кГц, при отсутствии шумов и при V = 2 нельзя получить скорость передачи более 6 кбит/с. Здесь нет противоречия с теоремой Шеннона. Ведь в отсутствие шумов значение V не будет иметь ограничения сверху. Здесь не имеется в виду, что максимальная амплитуда сигнала может достигнуть киловольтов. Но в отсутствие шумов можно и в пределах одного вольта представить себе любое число уровней сигнала. Фактически теорема Шеннона проясняет то, как уровень шумов ограничивает предельное значение V при заданной максимальной амплитуде сигнала.

Проанализировав влияние шумов и помех, можно сделать вывод, что приём чистого незашумлённого сигнала крайне важен, особенно в условиях маломощного источника или большой дальности приёма, так как в данных условиях в подавляющем большинстве случаев отношение уровня сигнала к уровню шума составит небольшую величину.

ЛИТЕРАТУРА

1. Сэломон, Д. Сжатие данных, изображения и звука / Д. Сэломон. – М. : Техносфера, 2004. – 368 с.
2. Конхейм, А.Г. Основы криптографии / А.Г. Конхейм. – М. : Радио и связь, 1987.
3. Морелос-Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса ; пер. с англ. В.Б. Афанасьева. – М. : Техносфера, 2006. – 320 с. – (Мир связи).

УДК 681.586.773:624.072.233.5

ОСОБЕННОСТИ И ПРИНЦИПЫ ВЫЯВЛЕНИЯ ДЕФЕКТОВ ГОЛОВКИ ЖЕЛЕЗНОДОРОЖНОГО РЕЛЬСА

А.В. ШЛЯХТЕНОК
(Представлено: канд. техн. наук, доц. Д.А. ДОВГЯЛО)

Рассматриваются датчики, используемые в дефектоскопе РДМ-22 для выявления дефектов головки железнодорожного рельса. Определены их основные достоинства и недостатки с точки зрения практического выявления дефектов. Практическая часть работы – комплексный анализ дефектограмм, содержащих сигналы, отраженные от реальных дефектов рельсов.

Дефекты головки рельса являются самыми распространенными среди всех дефектов, входящих в классификацию НТД/ЦП 1-2-3-93 [1]. Они представляют наибольшую опасность, так как в процессе роста поперечных трещин (развиваются в виде окружности с характерными «кольцами роста»), в результате взаимодействия их плоскостей, последние приобретают зеркальную поверхность, что существенно усложняет обнаружение. Еще одним фактором, усложняющим поиск и определение места залегания дефектов, являются поверхностные повреждения (горизонтальные расслоения, выкрашивания металла на поверхности катания), которые маскируют поперечные трещины, препятствуя доступу ультразвуковой волны к ним. Для решения этих проблем применяются различные схемы контроля, а также пьезоэлектрические преобразователи (ПЭП) с различными углами ввода ультразвука.

Схемы контроля головки рельса на базе дефектоскопа РДМ-22 и основные принципы выявления дефектов. В общем случае схемой контроля (прозвучивания) можно назвать группу ПЭП, реализованных на базе конкретного дефектоскопа, которые обеспечивают наиболее эффективное выявление дефектов в соответствии с поставленными целями и задачами.

На сегодняшний день существует большое разнообразие схем, применяемых для контроля головки рельса. Каждая из них имеет свои преимущества и недостатки. Наибольшую эффективность демонстрируют схемы, реализующие более одного метода (пример, схема «РОМБ», реализующая ЭХО- и зеркальный методы, каждый из которых активен в определенные моменты времени так, что они не работают одновременно). Данные схемы позволяют наряду с дефектами произвольной ориентации выявлять дефекты с различными типами отражающей поверхности.

В дефектоскопах РДМ-22, для контроля всего объема головки рельса, применяются следующие ПЭП:

- *канал № 1* – канал реализован посредством использования РС (раздельно-совмещенного) ПЭП с углом ввода ультразвука в контролируемое изделие 0°. На практике данный канал показывает высокую эффективность при выявлении различного рода горизонтальных расслоений и трещин. Зона контроля в зависимости от типа рельса составляет от 0 до 190 мм;

- *канал № 2(7)* – ПЭП с углом ввода 70°, развернутый вдоль продольной оси рельса по ходу движения ультразвукового дефектоскопа (7 канал – против хода). Практическая полезность данного канала заключается в возможности обнаружения трещин, развивающихся под протяженными горизонтальными расслоениями, которые в силу особенностей отражения ультразвуковых волн от различно ориентированных поверхностей невозможно выявить ПЭП с другими углами ввода. Канал № 2 уверенно выявляет трещины, развивающиеся под углом 18…20° от нормали. Зона контроля в зависимости от типа рельса составляет от 3 до 45 мм. Основным недостатком данного преобразователя является наибольшая среди всех используемых в дефектоскопе РДМ-22 зависимость размеров, снятых с дефектного места, от температуры. Так, например, в диапазоне температур от минус 40 до плюс 50 ºС номинальный угол ввода изменяется от 60 до 74º. Изменение угла ввода до 74º приводит к образованию поверхностной волны, которая чувствительна даже к небольшим поверхностным дефектам;

- *Каналы № 3; 6; 8* – ПЭП с углом ввода 55°, развернутые на 34° относительно продольной оси рельса и направленные в рабочую грань против хода ультразвукового дефектоскопа, в нерабочую грань против хода ультразвукового дефектоскопа, в рабочую грань по ходу движения дефектоскопа соответственно. Разворот вставок на 34° позволяет контролировать весь объем головки рельса. Единственным недостатком данных ПЭП является высокая чувствительность к поверхностным дефектам. В отдельных случаях даже при небольших и неглубоких расслоениях вследствие многократного переотражения ультразвуковой волны (УЗВ) между расслоением и поверхностью, оператор может выдать ложное заключение о наличии дефекта в рельсе. Зона контроля данных ПЭП составляет 144 мм [2].

Пьезоэлектрические преобразователи с углом ввода 55°, развернутые на 34° относительно продольной оси рельса реализуют схему «ЗМЕЙКА». Данная схема является наиболее сложной и с точки зрения процесса отражения ультразвуковых лучей от различных граней головки рельса, и для понимания принципа отображения сигналов на В-развертке. Это вызвано тем, что для обнаружения дефектов головки в основном используются одно- , двукратно- и трехкратно переотраженные от ее граней лучи [3].

На рисунке 1 проиллюстрирован путь распространения ультразвука в головке рельса при использовании схемы «ЗМЕЙКА» и фрагмент дефектограммы, на котором присутствуют сигналы от типового отражателя (к примеру, торца рельса).

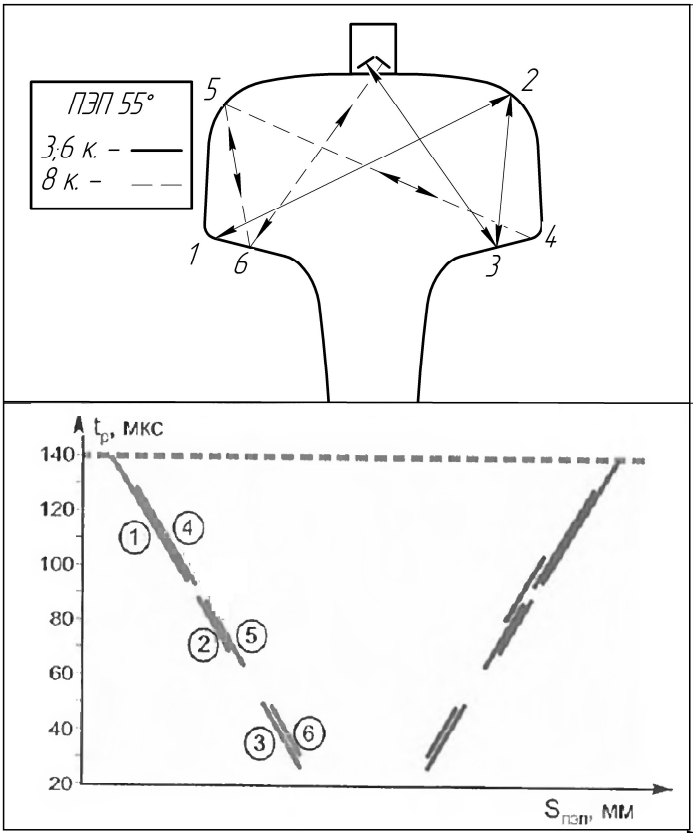


Рисунок 1. – Путь распространения ультразвука при использовании схемы «ЗМЕЙКА» и фрагмент дефектограммы типовых отражателей

Всю зону контроля при использовании данной схемы условно можно разделить на ближнюю (3; 6), среднюю (2; 5) и дальнюю (1; 4). Вследствие этого, проводя анализ принятых датчиками сигналов, можно с большой долей вероятности определить расположение дефекта относительно поперечного сечения рельса. Так, при наличии сигналов в ближней и средней зонах для каналов 3 и 6 можно говорить о дефекте, расположенном в рабочей грани рельса. Если данные каналы зарегистрировали сигнал в дальней зоне, речь идет о дефекте, развивающемся с нерабочей грани. Эти утверждения справедливы и для 8-го канала с поправкой на его разворот относительно продольной оси рельса.

Однако при всей кажущейся простоте проведения подобного анализа, существует ряд факторов, способных существенно исказить результаты контроля, что может привести к перебраковке рельса. Их можно условно разделить на несколько групп: факторы, обусловленные некачественной настройкой дефектоскопа; климатические, выражающиеся экстремально высокими и низкими температурами; факторы, обусловленные наличием на поверхности рельсов различного рода повреждений. Последние ввиду многократных переотражений ультразвука между ними и поверхностью сканирования нередко являются причиной регистрации сигнала достаточно большой протяженности, который может быть интерпретирован, как сигнал от развитого дефекта.

С целью исключения перебраковки необходимо проводить комплексный анализ с сопоставлением сигналов от различных ПЭП для определения общей картины дефектности рельса. Анализ начинается с рассмотрения сигналов, способных помешать выявлению поперечных трещин, т.е. с поиска горизонтально-

ориентированных дефектов. В случае если горизонтальное расслоение по своим параметрам превышает нормы, допускающие возможность эксплуатации рельса, он изымается из пути как остродефектный. Исходя из фрагмента дефектограммы (рис. 2) на рельсе имеется небольшое расслоение металла глубиной 3…5 мм. Данное расслоение классифицируется как дефект кода 10.2 – отслоение и выкрашивание металла на поверхности катания головки рельса вследствие нарушения технологии изготовления рельса и допускает его дальнейшую эксплуатацию.

Далее анализируются сигналы, зарегистрированные наклонными (с углом ввода, отличным от 0º) ПЭП. Согласно дефектограмме, преобразователем с углом ввода 70º зарегистрирован сигнал протяженностью от 13 до 27 мм. Исходя из того, что в режиме сведения сигналов, показания зарегистрированы до горизонтального расслоения, можно сделать вывод о наличии дефекта в виде поперечной трещины.

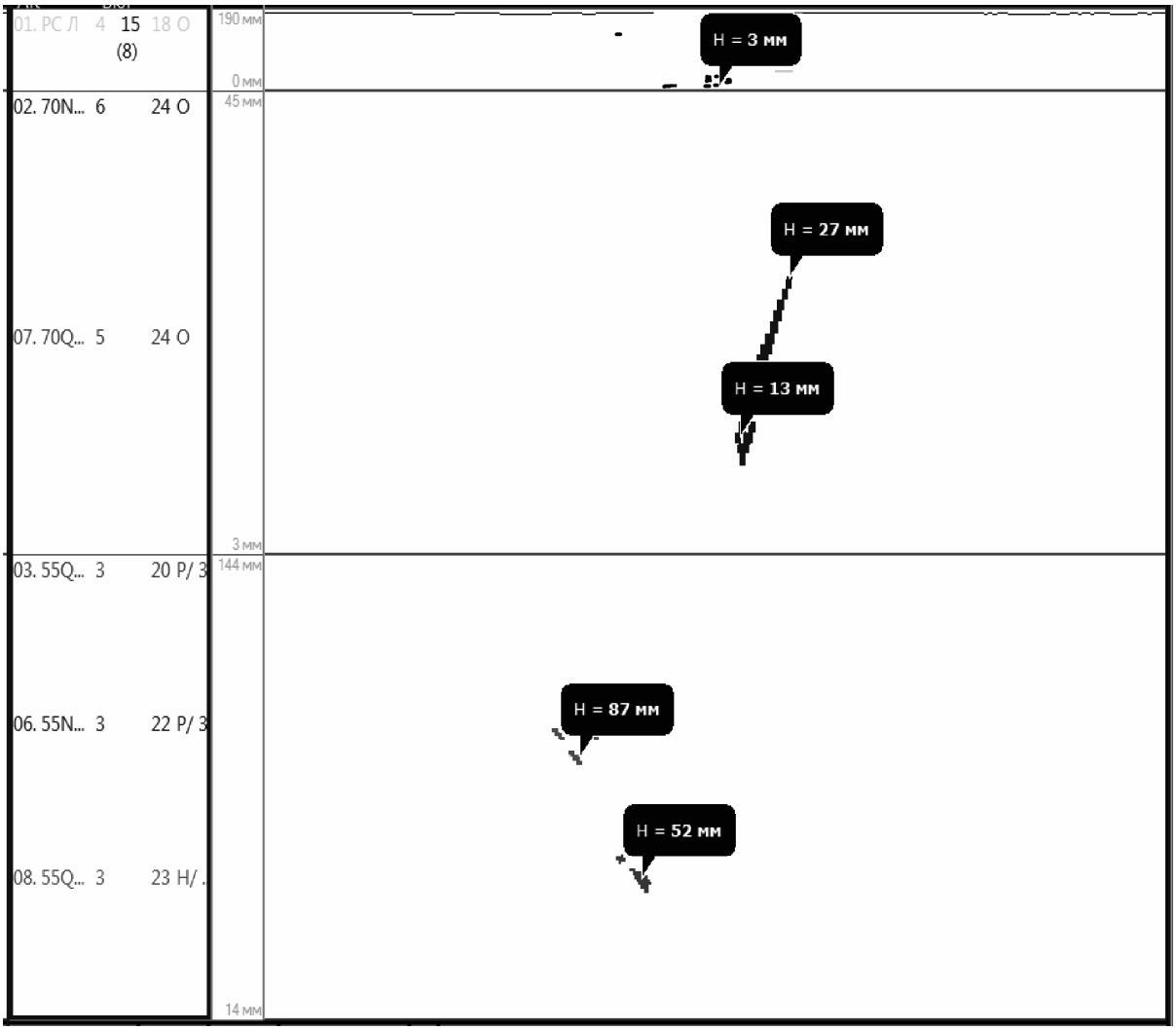


Рисунок 2. – Фрагмент дефектограммы

Для подтверждения наличия дефекта дальнейший анализ производится на основании показаний ПЭП с углом ввода 55º. Так, 55-градусными ПЭП зарегистрированы две пачки сигналов – восьмым каналом в начале дальней зоны (87 мм) и третьим каналом в конце средней зоны (52 мм). Данное обстоятельство указывает на вероятное наличие дефекта, расположенного в рабочей грани и центре головки рельса.

На основании данных комплексного анализа можно сделать вывод о наличии под горизонтальным расслоением развитого дефекта в нерабочей грани рельса, заходящего в центр головки.

Распространенной практикой в рельсовой дефектоскопии являются контрольные доломы рельсов, выявленных в результате контроля остродефектных рельсов. В ходе проведения контрольного долома был подтвержден дефект, изображенный на рисунке 3. Долом производился на специализированном оборудовании и произошел при нагрузке 64 тонны.

Рисунок 3 – Результат контрольного долома

Вывод. Эффективность комплексного анализа сигналов зависит от многих факторов, однако данный подход позволяет с достаточно высокой вероятностью определять не только наличие дефекта, но и определять его ориентацию и положение относительно поперечного сечения рельса. Данное обстоятельство является определяющим при контроле, т.к. на сегодняшний день дефекты головки рельса являются наиболее сложными с точки зрения их выявления.

ЛИТЕРАТУРА

1. Национальная система подтверждения соответсвия Республики Беларусь. Порядок декларирования соотвествия продукции. Основные положения: НТД/ЦП 1-2-3-93 Классификация дефектов рельсов. Каталог дефектов рельсов. Признаки дефектных и остродефектных рельсов.
2. Национальная система подтверждения соответствия Республики Беларусь. Порядок декларирования соответствия продукции. Основные положения: Дефектоскоп ультразвуковой ДС2-РДМ-22. Руководство по эксплуатации.
3. Марков, А.А. Дефектоскопия рельсов. Формирование и анализ сигналов / А.А. Марков, Е.А. Кузнецова. – СПб.: Ультрапринт, 2014. – 332 с.
4. Александрова, С.Б. Ультразвуковая дефектоскопия: учеб.-метод. компл. / С.Б. Александрова. – Барановичи: ДЦППК-2, 2008. – 205 с.

УДК 004.925.84

СИНТЕЗ 3D-МОДЕЛИРОВАНИЯ И ГЕОИНФОРМАЦИОННЫХ СИСТЕМ

М.С. СПИРИДЕНКО, К.А. КАТАШОВА
(Представлено: П.Ф. ПАРАДНЯ)

Рассматривается возможность объединения технологии 3D-моделирования и геоинформационных систем на примере геоинформационной системы ArcGis и программного продукта Google SkeatcUp. Показано, что соединить воедино качество, легкодоступность и скорость получения необходимой пространственной информации об объектах можно благодаря концепции BIM и её связи с геоинформационными системами.

К настоящему времени возможности информационных технологий возрастают во всех сферах деятельности. Поэтому неудивительным можно считать возникновение принципиально нового подхода в архитектурно-строительном проектировании, заключающемся в создании компьютерной модели нового здания, несущей в себе все сведения о будущем объекте. Новый подход к проектированию объектов получил название **информационное моделирование зданий** или сокращенно **BIM** (от принятого в английском языке термина Building Information Modeling) [2].

Когда новый объект вступает в стадию эксплуатации, происходит его взаимодействие с другими объектами и окружающей средой, то есть начинается активная фаза «жизненного цикла» здания. Подход к проектированию зданий через их информационное моделирование предполагает прежде всего сбор и комплексную обработку в процессе проектирования всей архитектурно-конструкторской, технологической, экономической и иной информации о здании со всеми ее взаимосвязями и зависимостями, когда здание и все, что имеет к нему отношение, рассматриваются как единый объект.

Между тем, анализируя основное направление использования геоинформационных систем, отметим, что ГИС – это основная технология для сбора, организации, обмена и анализа информации. Она позволяет увидеть, где и что происходит, как взаимосвязаны различные события, как на них влияет местоположение, какие географические, социальные и экономические факторы определяют работу организации, и каким образом лучше всего использовать местные особенности и пространственные закономерности. Таким образом, сходство концепции BIM и ГИС очевидно.

Информационная модель существует в течение всего жизненного цикла здания, и даже дольше. Содержащаяся в ней информация может изменяться, дополняться, заменяться, отражая текущее состояние здания.

Важно подчеркнуть, что информационная модель здания – это виртуальная модель, результат применения компьютерных технологий. В идеале BIM – это виртуальная копия здания. На начальном этапе создания модели мы имеем некоторый набор информации, почти всегда неполный, но достаточный для начала работы в первом приближении. Затем введенная в модель информация пополняется по мере ее поступления, и модель становится более насыщенной [2].

С помощью специализированной платформы ArcGis есть возможность создать карту, где можно отобразить все элементы инфраструктуры, что для успешной эксплуатации объекта будет полезным дополнением.

Пока такого типа синтез – лишь перспектива. Но примером этому может служить проект по созданию 3D-модели студенческого городка Полоцкого государственного университета. Известно, что существуют различные технологии создания 3D-карт. Реализовать подобное можно, например, используя геоинформационную систему ArcGis, в частности ее модуль 3D-Analyst и приложение ArcScene, предназначенные для построения 3D-модели рельефа местности, а также создания 3D-карт городской застройки, совмещённой с интерполированным растром (поверхностью), и специализированное приложение для 3D-моделирования городской среды Esri CityEngine.

Однако это не единственный способ создания такого проекта, к тому же он считается достаточно узким, но вполне приемлемым для отображения внешнего мира. Но, если возникает задача моделирования объектов, например, жилого дома и отображения его фасада и внутреннего интерьера, показывая, при этом всю окружающую ситуацию (дорожную сеть, земельное покрытие, элементы инфраструктуры и др.), можно использовать платформу ArcGis и какой-либо программный продукт для трёхмерного моделирования, в данной работе – Google SkeatcUp.

Для визуализации трехмерных объектов в ArcGis использовалось приложение ArcScene. Для их размещения в модели приложение берет данные о высоте объекта, полученные из его геометрии, атрибу-

ты объекта, свойства слоя или заданную 3D-поверхность. При этом каждый слой обрабатывается отдельно от других.

Модуль ArcGIS 3D Analyst позволяет драпировать изображения или векторные данные на поверхность, а также вытягивать векторные объекты в направлении от поверхности, создавая, таким образом, линии, стены и объемные фигуры. Использование 3D-символов позволяет сделать отображение данных более реалистичным и создать высококачественную анимацию для демонстрации или распространения.

Проект основан на создании трехмерной модели учебных корпусов Полоцкого государственного университета.

Для создания 3D-моделей зданий использовался программный продукт Google SketchUp 2015, так как приложение ArcScene ограничивает использование пользовательских текстур. SketchUp позволяет создавать объекты необходимой формы в соответствии с реальными размерами, накладывать на них текстуры, чтобы сделать здания реалистичными и импортировать созданные объекты в среду ArcGis через обменные форматы [3].

SketchUp непосредственно связан с GoogleEarth, откуда можно импортировать снимки земной поверхности в качестве подложки уже в заданной системе координат, в нашем случае – в WGS-84.

После того как карта-подложка получена, производится оцифровка здания линиями для получения плоскости, а затем с помощью инструмента выдавливания здание вытягивается на его действительную высоту.

Следующий этап – наложение текстур на внешнюю часть здания, а затем оформление интерьера корпуса – одна из главных частей работы.

Для этого используются фотографии граней здания, окон, дверей и других мелких деталей. С помощью инструмента «Материалы» накладывается текстура для создания реалистичного отображения. Данный инструмент удобен в использовании, так как он имеет возможности редактирования наложения текстуры на поверхность, в частности, искажение, изменение масштаба, поворот и зеркальное отображение текстуры, а также установления действительных размеров в параметрах. Огромным плюсом является изменение цвета текстуры без посторонних программ. Особое внимание стоит обратить на возможность оформления здания внутри.

В программном продукте SketchUp можно создавать 3D-модели объектов интерьера. В проекте эта возможность была реализована на примере холла нового корпуса Полоцкого государственного университета с применением рендера V-Ray для создания реалистичного отображения (рис. 1).

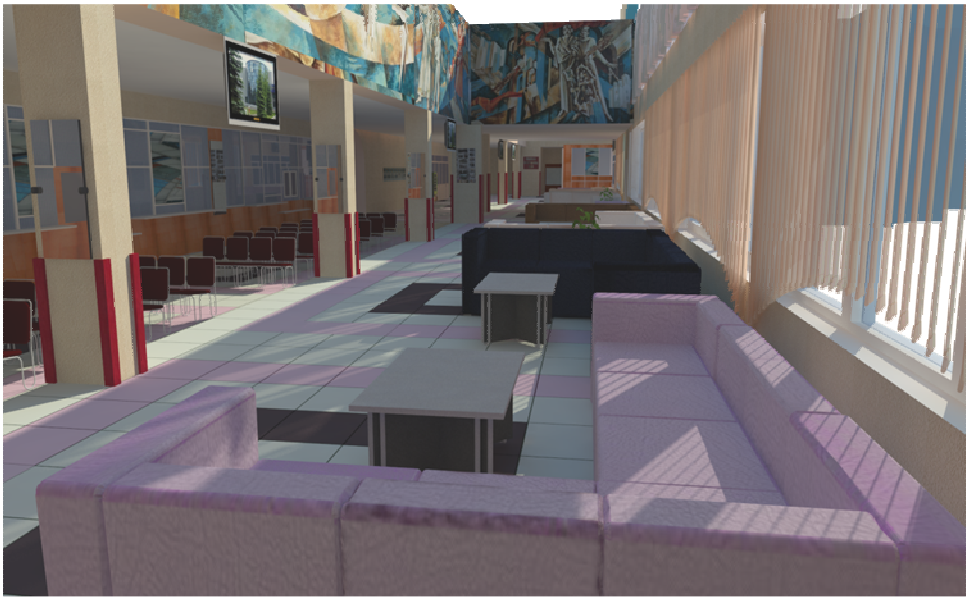


Рисунок 1. – 3D-модель холла нового корпуса университета, созданная в SketchUp

В ArcMap была создана файловая база геоданных, куда сохранялись классы пространственных объектов: деревья, газоны, дороги и др., которые были созданы в результате оцифровки карты-подложки.

После того как все объекты созданы и отредактированы, они импортировались в ArcScene вместе с картой-подложкой. Объекты, созданные в SketchUp, загружались в 3D-Analyst посредством конвертации из файла формата .dae.

В приложение ArcScene подгружался растр высот на обрабатываемый участок местности, чтобы относительно его выровнять уровень высот всех 3D-объектов.

Для отображения точечных объектов в ArcScene в атрибутивной таблице для них создавалось поле с названием типа объекта, а затем сопоставлялось значение этого поля с определенным символом выбранного стиля (например, для деревьев использовался стиль 3D Trees).

Чтобы полигональные объекты, такие как дороги, тротуары и т.п., показать в приближенном к действительности виде, использовались текстуры ArcScene и обыкновенная заливка (рис. 2).

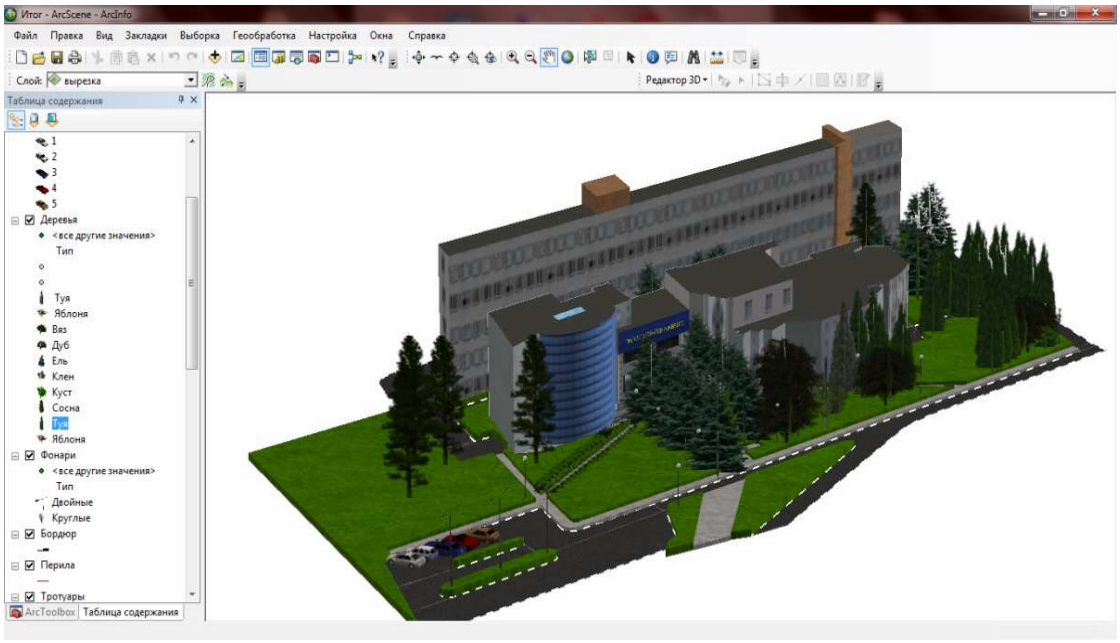


Рисунок 2. – 3D-модель здания нового корпуса Полоцкого государственного университета в среде ArcGis

Программный продукт Google SketchUp наряду с простотой в освоении обладает мощным функционалом для создания 3D-моделей. За счет использования стандартных форматов, поддерживаемых большинством программных продуктов 3D-моделирования, в том числе и ArcGis, возможна их интеграция как на программном уровне, так и на уровне обмена данными. В частности, применительно к геоинформационным системам актуальна задача создания моделей для GoogleEarth. При этом вместо ввода значений координат можно использовать многофункциональную систему управления местоположением объекта с помощью меток, текстовых подсказок, линий различных цветов. Полный набор простых инструментов и применение интеллектуальной системы рисования позволяют легко создавать и редактировать модели, экспортировать их, создавать видеофильмы или распечатывать результаты работы .

ЛИТЕРАТУРА

1. Меженин, А.В. Технологии 3D-моделирования для создания образовательных ресурсов : учеб. пособие / А.В. Меженин. – СПб., 2008. – 112 с.
2. Isicad – Ваше окно в мир САПР [Электронный ресурс] / BIM: что под этим обычно понимают. – 2011. – Режим доступа: https://dwg.ru/pub/42. – Дата доступа: 11.04.2016.
3. 3D-modeling for everyone [Электронный ресурс]. – SketchUp. – Режим доступа: http://www.sketchup.com. – Дата доступа: 3.02.2016.

УДК 004.925.84

АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 3D-МОДЕЛИРОВАНИЯ С ЦЕЛЬЮ СОЗДАНИЯ ТРЕХМЕРНЫХ МОДЕЛЕЙ ОБЪЕКТОВ ПОЛОЦКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

К.А. КАТАШОВА, М.С. СПИРИДЕНКО
(Представлено: П.Ф. ПАРАДНЯ)

Представлен анализ возможностей программного обеспечения для трехмерного моделирования с целью создания 3D-модели зданий Полоцкого государственного университета. Сделан вывод, что при выборе программного продукта, реализующего задачи 3D-моделирования, необходимо разобраться со всеми функциями, достоинствами и недостатками каждого из них.

Визуальные эффекты, отражение реального мира посредством компьютерной графики вызывают у многих интерес к трехмерному моделированию и анимации. 3D-моделирование – современный способ создания трехмерной графики и ее визуализации в век стремительно развивающихся технологий. Основная цель 3D-моделирования заключается в разработке объемного вида объекта. Путем использования современного программного обеспечения сделать это не составит особого труда.

Программы для 3D-моделирования помогают воссоздать любой сложности идеи в первоклассные модели, которые впоследствии возможно реализовать в самых разных планах. В такой тип программных продуктов включено множество инструментов, которые позволяют создавать модели с нуля, а также разрабатывать сложнейшие элементы и сооружения. На сегодняшний день трехмерные модели применяются в самых разных областях деятельности: строительство и архитектура, дизайн, игровая и киноиндустрия, полиграфия и многое другое.

Редакторов трехмерной графики множество, но и представленных в них возможностей довольно много. В общих чертах все 3D-редакторы похожи, но каждый из них имеет нечто особенное и уникальное.

Для того чтобы из большого количества разнообразных программных продуктов выбрать оптимально удовлетворяющий профессиональным запросам специалистов, необходимо разработать критерии сравнения и рекомендации по выбору.

Основные функции и возможности, необходимые программным средствам для 3D-моделирования:

- моделирование трёхмерной графики – создание трёхмерной модели сцены и 3D-объектов в ней;
- система рендеринга (визуализация) – построение проекции модели;
- обработка и редактирование изображений;
- вывод полученного изображения на устройство вывода: принтер, дисплей.

Наиболее распространенными программными средствами создания 3D-моделей являются: 3D Studio MAX, Autodesk Maya, Autodesk Softimage, Blender 3D, GoogleSketchUp.

Программный продукт, который занимает ведущие позиции в сфере дизайна и архитектурной визуализации, – **3D Studio MAX** – программное обеспечение для 3D-моделирования, анимации и визуализации от фирмы Autodesk. В данном 3D-редакторе присутствуют средства для анализа. В программу интегрирован фотореалистичный визуализатор, дающий возможность добиться высокой правдоподобности просчитываемого изображения. 3Ds Max дает возможность очень гибко управлять частицами, создавая самые разнообразные эффекты – от моделирования анимированных массивов объектов до имитации всевозможных природных явлений. При всей своей сложности 3Ds Max легко изучается, любая нехватка какого-либо специфического инструмента компенсируется большой базой дополнений – плагинов, что значительно расширяет стандартные возможности приложения. Редактор 3Ds Max поддерживает множество файлов с различными форматами для импорта и экспорта, что дает возможность хорошей интеграции с другими программными средствами.

Преимущества этого программного обеспечения (ПО):

- удобный интерфейс;
- большое количество инструментов, моделей;
- поддержка векторных карт;
- поддержка V-ray (мощнейшего рендера);
- моделирование сетей и поверхностей;
- наложение и редактирование текстур;
- высокая степень реалистичности изображений;

- возможность создания анимации;
- моделирование на основе полигонов, сплайнов и NURBS (Неоднородный рациональный B-сплайн);
- русификация;
- взаимодействие с другими пакетами Autodesk.

Недостатками выступают:

- сложность в освоении;
- высокая стоимость ПО.

Следующий, достаточно широко известный продукт – **Autodesk Maya** – комплексный программный продукт для 3D-анимации от той же фирмы. Maya позволяет пройти все стадии создания 3D – от моделирования и анимации до текстурирования, композитинга и послойного рендеринга. Этот трехмерный редактор может моделировать физику твердых и мягких тел, просчитывать поведение ткани, эмулировать текучие эффекты, позволяет детально настраивать прическу персонажей, создавать сухой и мокрый мех, анимировать волосы и т.д. Визитной карточкой программы является модуль PaintEffects, который дает возможность рисовать виртуальной кистью такие трехмерные объекты, как цветы, трава, объемные узоры и прочее. Программа довольно сложна в освоении, но это компенсируется большим количеством уроков по данному редактору и удобным интерфейсом.

Основными преимуществами выступают удобный интерфейс, усовершенствованная система визуализации, встроенный язык программирования, полный набор инструментов для NURBS- и полигонального моделирования, UV-текстуры, нормали и цветовое кодирование, широкий набор средств создания динамических спецэффектов, взаимодействие с другими пакетами Autodesk.

В свою очередь, ПО имеет небольшие недостатки: длительное и сложное обучение работе в ПО, высокие требования к системе, высокая стоимость программного продукта.

Следующий пакет **Autodesk Softimage** – многофункциональный редактор трёхмерной компьютерной графики, принадлежащий компании Autodesk, включающий в себя возможности 3D-моделирования, анимации и создания спецэффектов. Это программное обеспечение главным образом используется при создании кино, видеоигр, а также в рекламной индустрии для создания персонажей, объектов и окружения. В Softimage реализована одна из самых лучших систем анимации, которая позволяла «программировать» без знаний какого-либо из языков программирования просто составляя диаграммы из функций, соединённых узлами. Благодаря уникальной системе ICE (Interactive Creative Environment – платформе визуального программирования, основанной на кодах) пакет предлагал широкую функциональность, гибкость, высокую производительность и качество. Благодаря новым функциям, улучшенному взаимодействию с продуктами из состава Autodesk Entertainment Creation Suite Premium и поддержке современных технологий пользователи Autodesk Softimage без труда справляются с самыми сложными задачами на стадиях разработки 3D-игр и постпроизводства. Программное обеспечение может поставляться в виде локальных или сетевых лицензий.

Среди преимуществ невозможно не отметить мощный и удобный полигональный моделинг, процедурное моделирование в среде ICE, удобный и простой в использовании интерфейс, встроенный язык программировани, поддержка JScript, VBScript, Python для написания скриптов, УФ-текстурирование, а также полное взаимодействие с другими пакетами Autodesk.

Стоит обратить внимание, что при всех плюсах программный пакет платный и сложен в освоении [2].

Бесплатный **Blender 3D** – профессиональный продукт для создания трехмерной компьютерной графики, включающий в себя инструменты моделирования, анимации, рендеринга, постобработки видео, а также создания игр. Программа имеет большое количество кистей, возможность создания слоёв, простота работы с анимацией и еще множество других премудростей присущи Blender 3D. В данном ПО продвинутый интерфейс, который позволяет подстроить все элементы и инструменты под желания пользователей. Программа позволяет оперировать системами частиц, контролировать веса отдельных частиц при текстурировании, применять направляющие при анимации и использовать внешние силы.

Заметим, что некоторые форматы можно подключить с помощью установки плагина. Также есть возможность самостоятельно создать плагин, чтобы интегрировать необходимый формат в программный продукт Blender 3D.

Таким образом, главные преимущества:

- бесплатное ПО с открытым исходным кодом;
- наличие русификатора;
- возможность подключения плагинов, в том числе собственных;
- кроссплатформенность;
- небольшой размер программного продукта [3].

Однако высокая сложность освоения считается главным недостатком.

Профессиональное ПО **Google SketchUp** – программная среда, в которой можно выполнять практически любые действия, связанные с 3D-моделированием. В SketchUp встроено большое количество стандартных шаблонов моделей, а также генератор эффектов, что значительно позволяет облегчить работу. Также программа поддерживает макросы и плагины. Основными видами работ, которые выполняются в SketchUp, является построение относительно простых трёхмерных объектов моделирования – мебели, интерьера или архитектурных сооружений. Существуют два варианта программного средства – бесплатная для некомерческого использования, ограниченная по функциональным возможностям SketchUp Make (в первую очередь это касается экспортирования в другие форматы), и платная SketchUp Pro. Есть возможность установления географически достоверных теней в соответствии с заданными широтой, долготой, временем суток и года, а также возможность добавления в модель поверхность земли и регулирования её формы – ландшафт.

Несмотря на весомые преимущества у программы все же имеются недостатки, один из которых – отсутствие собственного мощного рендера. Этот недостаток легко компенсировать внешними популярными рендерами V-ray, Artlantis, которые отлично работает в совокупности [4].

3D-представление данных и возможность полноценного построения и визуализации трехмерных моделей территорий становится необходимой не только для сферы развлечений или традиционных областей 3D, но и для геоинформационных систем (ГИС).

ГИС – это многофункциональная информационная система, предназначенная для сбора, обработки, обмена, моделирования и анализа пространственных данных, их отображения и использования при решении расчетных задач. ГИС имеют прямое отношение к абсолютной и относительной локализации особенностей рельефа местности, также как к свойствам и признакам этих особенностей. ГИС помогает анализировать и отображать эти пространственные зависимости.

Наибольшее распространение, конечно же, получили классические двухмерные геоинформационные системы. Но развивающиеся высокими темпами трехмерное моделирование и возможности компьютерной техники позволяют увидеть наглядную карту с привязанными к ней 3D-объектами (моделями), такими, какими их можно наблюдать в жизни. Именно поэтому некоторые ГИС-платформы стали поддерживать трехмерную графику, а именно: Esri ArcGIS, MapInfo Professional и отечественная Панорама (Карта 2011) и другие [1].

Esri ArcGIS – это полнофункциональная геоинформационная платформа корпоративного уровня, используемая ГИС-специалистами. ПО ArcGIS содержит в себе уникальный набор инструментов, который так необходим для решения различных технологических задач, сопряженных со сбором, управлением, анализом, хранением и визуализацией любых пространственных данных. Вспомогательные модули ArcGIS for Desktop позволяют обеспечивать дополнительные функциональные возможности, например, 3D-визуализация, расширяющая области применения ГИС. Вспомогательный модуль ArcGIS 3D Analyst включает в себя набор инструментов для трехмерного отображения данных, построения поверхностей, редактирования данных в 3D-режиме, также он содержит совокупность инструментов геообработки, которые производят массу разнообразных операций по анализу, управлению и преобразованию пространственных данных для моделей поверхностей и трехмерных векторных данных.

В модуле ArcGIS 3D Analyst реализуется возможность быстрого создания реалистичной виртуальной 3D-сцены на основе пространственных данных, как локального уровня (приложение ArcScene), так и в масштабе всей Земли (приложение ArcGlobe), с использованием цифровых моделей рельефа, космических и аэроснимков, любых векторных данных и фотореалистичных моделей объектов. Эти приложения 3D-визуализации выступают в ролях составных частей модуля ArcGIS 3D Analyst. ArcGlobe беспрепятственно обрабатывает обширные данные, при том что вся информация, содержащаяся в них, сохраняется. Другое приложение ArcScene позволяет управлять трехмерными ГИС данными, производить 3D-анализ, создавать 3D пространственные объекты, а также отображать слои в планиметрическом 3D-виде. Модуль ArcGIS 3D Analyst широко применяют специалисты в областях градостроительства, геологии, бизнеса и других. Геоинформационная платформа ArcGIS полностью адаптирована к русскому языку [5].

ГИС «Панорама» – коммерческая геоинформационная платформа, предлагающая обширный список программных продуктов универсального и специализированного назначений для решения для задач в той или иной области. Данное ПО включает в себя средства создания и редактирования цифровых карт и планов в многопользовательском режиме, обработки данных дистанционного зондирования, выполнения различных измерений и расчетов, оверлейных операций, построения трехмерных моделей, обработки растровых данных, построения ортофотопланов, создания матриц высот, многослойных матриц, средства тематического картографирования, подготовки карт к изданию, работы с GPS-приемниками, обеспечения удаленного доступа к картографическим данным, а также инструментальные средства для работы с базами данных. В программном продукте есть возможность построения трехмерной модели не только местности, но и внутренних помещений, что немаловажно.

3D-модель местности в ГИС «Панорама» – это поверхность, построенная с учетом рельефа местности, на которую накладывается изображение векторной, растровой или матричной карты, и расположенные на ней трехмерные объекты, которые соответствуют объектам двухмерной карты. Типовые 3D-модели создаются по планам городов, топографическим или обзорным картам. Построение типовой модели принят за самый быстрый способ получения качественной трехмерной модели местности. Эти модели применяются для визуальной оценки и анализа взаимного расположения объектов с учетом особенностей рельефа и их высоты.

Трехмерные модели детального вида описывают местность с объектами, содержащие поверхность рельефа местности, типовые объекты и объекты, объемное изображение которых приближается к их реальному виду на местности. 3D-модели внутренних помещений позволяют описывать объемный вид интерьера и создаются на основе поэтажных планов. При отображении трехмерных моделей внутренних помещений также могут быть использованы отдельные объекты и целые интерьеры, созданные в различных программах редактирования трехмерных изображений в VRML-формате и импортированные в библиотеку трехмерных изображений классификатора векторной карты ГИС «Панорама» [6].

MapInfo Professional – полнофункциональная инструментальная географическая информационная платформа цифрового картографирования, основанная компанией Mapping Information Systems Corporation, и предназначенная для создания, редактирования, визуализации и дизайна тематических карт, пространственного и статистического анализа графической и семантической информации, а также их хранения и отображения. Данное ПО предоставляет своим пользователям обширные функциональные возможности по визуализации и анализу пространственных данных. Программный продукт может интегрировать с внешними СУБД, имеет возможность создания запросов на языке SQL, подключения картографических функций MapInfo к приложениям, написанным на других языках программирования: Delphi, Visual Basic, C++, и др., а также позволяет хранить и обрабатывать пространственные объекты в базе данных Oracle без использования дополнительного ПО. Геоинформационная платформа MapInfo Professional поддерживает работу с табличными, с растровыми изображениями, а также позволяет импортировать и экспортировать данные в другие ГИС системы. В геоинформационной платформе MapInfo есть полезный инструмент визуализации пространственных данных – это построение 3D-карт. Наличие в карте слоя-поверхности (файл .mig – MapInfo Grid) – необходимое условие для создания трехмерной карты.

Также есть возможность использования дополнительного модуля MapInfo Vertical Mapper, инструменты которого гораздо шире, и они имеют более детальные настройки. MapInfo Vertical Mapper – модуль трёхмерного анализа для MapInfo Professional, с помощью которого можно создавать, отображать и анализировать трёхмерные данные. Этот модуль обладает широким диапазоном аналитических инструментов и инструментов визуализации, позволяющих выявлять важные тенденции пространственных данных. В модуле Vertical Mapper метод анализа растровых поверхностей не привязан к определённым границам, как это реализовано в стандартных методах. Цветовые настройки поверхностей в MapInfo Vertical Mapper дают возможность отображения вариативности данных, в то же время как динамический 3D-рендеринг позволяет представлять данные в трёхмерном виде. Vertical Mapper расширяет стандартные возможности MapInfo. Он прост в изучении, у него понятный интерфейс, содержащий весь необходимый функционал для работы с поверхностями.

В дополнение к трехмерному моделированию создан Engage3D модуль, позволяющий представление пространственной информации, такой как рельеф местности, статистические данные в виде интерполированных поверхностей, что создаёт новые возможности для анализа и выявления трендов. С помощью Engage3D строятся модели, отображаются данные в 3D и создаются эффектные и информативные видео материалы. Благодаря встроенному в Engage-мастеру создания отчётов легко создать профессионально оформленные карты. Данный модуль представляет набор прогрессивных аналитических функций, позволяющих рассмотреть разнообразные наборы данных в статистическом, пространственном, временном и графическом представлении. Сочетание MapInfo, Vertical Mapper и Engage3D образует прекрасный инструментарий для проведения пространственного анализа. ГИС MapInfo Professional имеет полностью локализованную версию на русском языке [7].

Так как мы живем в трехмерном пространстве, безусловно, хотелось бы представлять реальные объекты жизнедеятельности человека не только на плоскости, но и в их действительных размерах и действительном окружении. Эта возможность стала достаточно востребована и интересна, поэтому появилась идея о создании проекта, основанного на разработке трехмерной модели учебных корпусов Полоцкого государственного университета.

Проанализировав перечисленные 3D-редакторы и геоинформационные платформы, реализующие функцию 3D-моделирования, оценив все их достоинства и недостатки, наиболее рациональными для поставленных задач оказались ArcGIS и SketchUp.

Геоинформационной платформы ArcGIS, содержащей модуль ArcGIS 3D Analyst и приложение ArcScene, оказалось недостаточно для осуществления поставленных задач, но это компенсируется дополнением одного из ПП, предназначенного для трехмерного моделирования – Google SketchUp, у которого в запасе множество функций, отлично справляющихся с задачами 3D-моделирования.

ПО SketchUp предоставляет возможность создания трехмерных объектов необходимой формы в соответствии с реальными размерами, наложения на них пользовательских текстур (что в приложении ArcScene не реализуется), чтобы приблизить к реальной действительности созданные сооружения и здания, и импортировать созданные объекты в геоинформационную среду ArcGis через обменные форматы.

Программное средство Google Earth и 3D-редактор SketchUp представляют собой составные элементы единой семьи программных продуктов, что позволяет пользователю легко переносить информацию из одного пакета в другой. В нашем случае для лучшего восприятия трехмерной модели «в виртуальной жизни» на рельефе из Google Earth был импортирован снимок на территорию университета в системе координат WGS-84 в качестве растра – подложки, вследствие чего создается виртуальное сооружение – модель с помощью оцифровки и инструмента «Тяни-толкай».

Одним из этапов работы стало создание дизайна внешней и внутренней частей корпусов, т.е. наложение текстур на грани здания, окон, дверей и т.п.

В процессе создания проекта было выяснено, что программные средства ArcGIS и Google SketchUp отлично интегрируют между собой, предоставляя возможность обмениваться файлами различных форматов.

Простотой в освоении 3D-редактор Google SketchUp обладает мощными инструментами для создания трехмерных моделей, а возможности приложения ArcScene и модуля ArcGIS 3D Analyst отлично дополняют друг друга, позволяя создавать сложные конструкции в геопространстве.

ЛИТЕРАТУРА

1. Самардак, А.С. Геоинформационные системы : учеб. пособие / А.С. Самардак. – Владивосток, 2005. – 123 с.
2. Программы для 3D-проектирования, дизайна, анимации [Электронный ресурс]. – Autodesk. – Режим доступа: http://www.autodesk.ru. – Дата доступа: 13.04.2016.
4. Home of the Blender project [Электронный ресурс]. – blender.org. – Режим доступа: https://www.blender.org/. – Дата доступа: 21.04.2016.
5. 3D modeling for everyone [Электронный ресурс]. – SketchUp. – Режим доступа: http://www.sketchup.com. – Дата доступа: 02.03.2016.
6. Безлимитная картография [Электронный ресурс]. – ArcGIS. – Режим доступа: https://www.arcgis.com/ features/ index.html. – Дата доступа: 02.03.2016.
7. КБ Панорама. Геоинформационные технологии [Электронный ресурс]. – ГИС «Панорама». – Режим доступа: http://www.gisinfo.ru/download/download.htm/. – Дата доступа: 04.05.2016.
8. Esti-map [Электронный ресурс]. – MapInfo. – Режим доступа: http://esti-map.ru/. – Дата доступа: 13.05.2016.