УДК 004.9:004.45

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА УЧЕТА КОНТИНГЕНТА СТУДЕНТОВ УНИВЕРСИТЕТА

Д.В. ПОПКОВА
(Представлено: М.В. ДЕКАНОВА)

Рассмотрены вопросы проектирования клиентской части автоматизированной информационной системы учета контингента студентов. Особое внимание уделено разработке графического пользовательского интерфкйса. Разработка клиентского приложения является важнейшим процессом, так как именно посредством него реализуется взаимодействие между пользователем и информационной системой. При разработке клиентского приложения автоматизированной информационной системы учета контингкнта студентов выделены подсистемы для различных видов работы пользователя. Обоснован выбор среды и средств программирования.

В первую очередь следует отметить, что разрабатываемый программный продукт предназначен для обычных пользователей, следовательно, необходимо создать «дружественный» интерфейс, т.е. программу настолько простую, легкую в обращении и интуитивно понятную, что работник деканата сможет использовать ее без специального обучения [1, 2].

Немаловажным является удобство интерфейса. Расположение элементов не должно предоставлять затруднений при работе с приложением, все необходимые элементы и данные должны быть доступны. Для создания удобного интерфейса необходимо придерживаться четкой визуальной иерархии, которая достигается путем расположения элементов на экране в определенном порядке, то есть одни и те же элементы должны отображаться в одном и том же порядке каждый раз. Плохо проработанная визуальная иерархия не приносит никакой пользы и только запутывает пользователей. Грамотная организация элементов интерфейса позволяет придать экрану менее загруженный вид. С помощью продуманной организации элементов можно продемонстрировать связи между ними, и освоить такой интерфейс пользователям будет намного проще. Схожие элементы желательно группировать, располагать их на экране таким образом, чтобы было понятно, как они связаны между собой.

При разработке клиентского приложения «Автоматизированная информационная система учета контингента студентов университета» выделим набор подсистем для различной работы пользователя (подсистема студентов, подсистема справочников, подсистема документов).

Для создания клиентского приложения был выбран язык C#, а среда разработки Microsoft Visual Studio. Данный выбор можно обосновать свободным распространением этой среды разработки, удобством использования, функциональностью. Кроме того, с помощью специальных (также свободно распространяемых) компонентов существует возможность организации полноценной связи Microsoft Visual Studio 2010 с СУБД MySQL [3].

Так как система предназначена для одного пользователя, в ней не предусмотрена авторизация. Поэтому при запуске появляется главная форма, которая представляет собой элемент tabcontrol [4] с несколькими вкладками, отвечающими за работу с основными функционалами программы. Они тематически разбиты на работу со справочниками, студентами, группами, документами, а также добавлением студентов в систему.

Если пользователь пройдет на вкладку «Новый студент», то ему будет предоставлена возможность внести необходимую информацию о студенте для добавления его в базу данных. Поскольку о студентах хранится большое количество информации, целесообразно было ее разбить на категории, которые в свою очередь представлены отдельными вкладками с помощью вложенного элемента tabcontrol на вкладке «Новый студент». Вид главной формы приложения и одной из вкладок добавления студента представлен на рисунке 1.

Таким же образом сформированы и остальные вкладки, где пользователь может осуществить те операции, которые ему необходимы, например, работать с группами, добавлять, изменять и удалять информацию в справочниках и т.д. Справочники необходимы для добавления, изменения или удаления информации требующейся при работе со студентами. Вся введенная информация сохраняется и работнику деканата не придется каждый раз заново ее вводить. Стоит отметить, что редактирование справочников происходит в новых окнах (не на главной форме).

Использование таблиц, представлений и хранимых процедур из сервера происходит с помощью объекта DataSet [4]. Так как достаточно много операций по добавлению различных данных было связано с сервером, добавление проводится при помощи хранимых процедур.

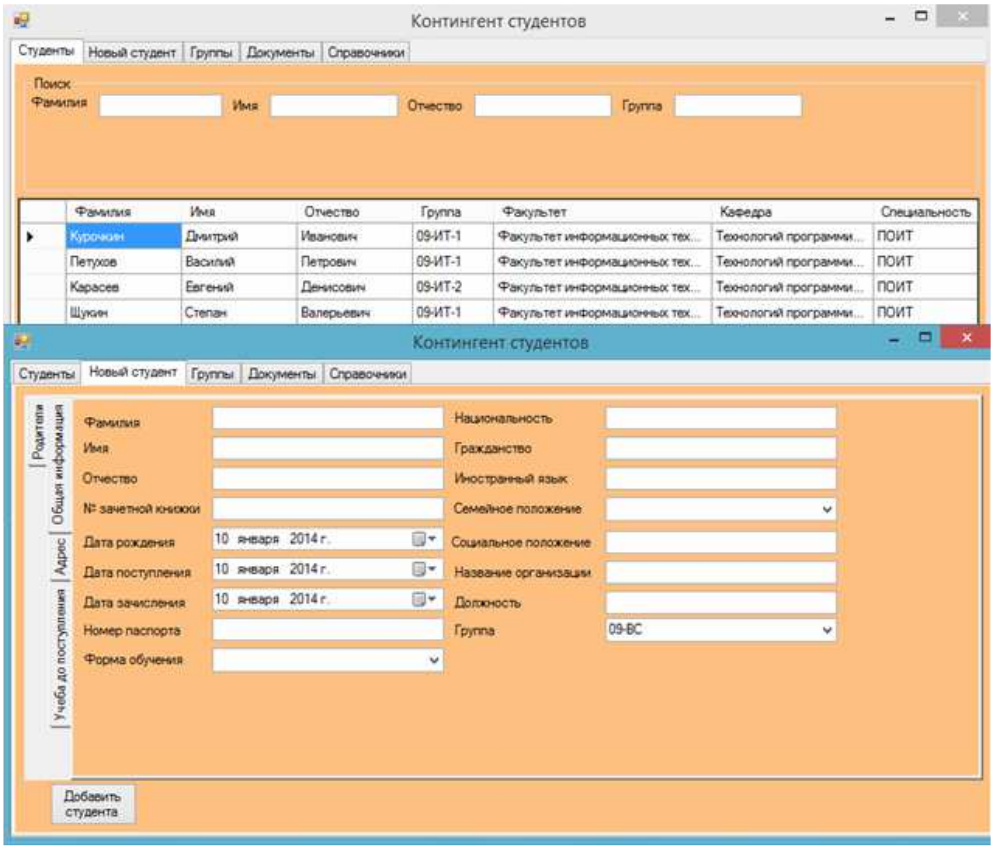

Рис. 1. Вид форм «Студент» и «Новый студент»

Основной функционал приложения открывается при выборе вкладки «Документы», при этом появляется окно, предоставляющее возможность ввода необходимой информации для создания приказов, распоряжений и справок.

Все виды документов можно вывести на печать с помощью Microsoft Word. На рисунке 2 приведен пример формы для справок для студентов по месту требования. Пропущенные поля заполняются автоматически в зависимости от выбранных пользователем параметров.

«Полоцкий государственный университет»
факультет _____
Исх № _____
Дата _____

СПРАВКА

Дана _____ Ф.И.О. студента _____ в том, что он(а) является студентом Учреждения образования «Полоцкий государственный университет» Курс курса факультета _____ Название факультета _____ группы Номер группы _____ Форма обучения _____ формы обучения и получает первое высшее образование.
Период обучения с __.__.____г. по __.__.____г.
Справка дана для предоставления в _____ Место требования _____.
Срок действия справки – 6 (шесть) месяцев.

Декан Название факультета _____ Ф.И.О. декана _____

Рис. 2. Форма для формирования справки студентам по месту требования

Таким образом, разработана автоматизированная информационная система учета контингента студентов университета с удобным и понятным пользователю интерфейсом. В справочниках вся информация разграничена по блокам, при ошибочных действиях появляются окна с информацией об ошибке. Кроме этого возможен быстрый переход между подсистемами. Использование разработанной автоматизированной информационной системы приведет к повышению производительности и качества труда сотрудников, оперативности и достоверности данных, исключению многих ошибок, предоставлению современной отчетности

ЛИТЕРАТУРА

1. Головач, В. Искусство мыть слона [Электронный ресурс] / В. Головач // Дизайн пользовательского интерфейса. – Режим доступа: http://uibook2.usethics.ru/uibookII. – Дата доступа: 06.05.2015.
2. Тидвелл, Дж. Разработка пользовательских интерфейсов / Дж. Тидвелл. – СПб. : Питер, 2008. – 395 с.
3. Шилдт, Г. C# 3.0 : полное руководство / Г. Шилдт. – М. : И.Д. Вильямс, 2010. – 992 с.
4. Windows Forms [Электронный ресурс] / Википедия. – Режим доступа: http://ru.wikipedia.org/wiki/Windows_Forms. – Дата доступа: 26.05.2015.

УДК 004.45.001.63

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЕТА КОНТИНГЕНТА СТУДЕНТОВ УНИВЕРСИТЕТА

Д.В. ПОПКОВА
(Представлено: М.В. ДЕКАНОВА)

Рассмотрены необходимые сущности для построения базы данных, с их описанием применения и характеристик. Построена концептуальная схема, выделены необходимые отношения и связи между таблицами. Все таблицы нормализованы и приведены к третьей формальной форме. Обоснован выбор системы управления базами данных для хранения информации о студентах. Описаны способы поддержания целостности базы данных, а также ошибочного и некорректного ввода сведений при заполнении карточек и документации студентов.

Учет и ведение документации в учреждениях образования становится все сложнее [1, 2]. В деканатах высших учебных заведений (ВУЗ) осуществляется работа с личными данными студентов, распределение студентов по группам, заполнение учебных карт студентов, ведение общего списка студенческих групп, учет аттестационных, экзаменационных ведомостей, движения контингента студентов, учет и оформление приказов о зачислении; разработка и исполнение различных видов приказов, распоряжений и справок.

Всей этой вышеперечисленной работой занимается один человек – секретарь деканата. Это очень трудоемкая работа, если выполнять ее вручную. К тому же объем информации, который нужно хранить в бумажном варианте, достаточно велик, и поиск нужной информации значительно затрудняется [2]. Создание автоматизированной информационной системы, основанной на современных технологиях сбора, обработки, анализа, передачи и хранения информации, значительно облегчит труд работника деканата.

Важная роль при разработке автоматизированной информационной системы отводится проектированию базы данных, в которой хранится большая по объему информация о какой-либо области человеческих знаний. Важно, что для пользователя эта база представляется, как единое хранилище информации, куда он может обратиться с запросом. В данной работе рассматривается вопрос проектирования базы данных для автоматизированной информационной системы учета контингента студентов университета.

Выделим следующие сущности проектируемой базы данных:

– студент (описывает студента, обучающего в университете; характеризуется фамилией, именем, отчеством, номером зачетной книжки, условиями обучения, контингентом и группой);

– группа (представляет собой список всех учебных групп студентов факультета; характеризуется названием, специализацией и формой обучения);

– специализация (описывает все имеющиеся в университете специализации; характеризуется кодом, названием и специальностью, за которой закреплена);

– специальность (хранит все имеющиеся в университете специальности; характеризуется кодом, названием, кафедрой, за которой закреплена, может разделяться на специализации);

– кафедра (описывает кафедры, имеющиеся в университете; характеризуется названием и факультетом, к которому относится);

– факультет (хранит информацию о факультетах университета; характеризуется названием и деканом);

– предмет (описывает изучаемые дисциплины; характеризуется названием, видом отчетности, семестром);

– успеваемость (содержит информацию об успеваемости студента; характеризуется предметом, изучаемым студентом, первой и второй аттестацией, а также полученной отметкой);

– документ (описывает всевозможные документы на студентов; характеризуется студентом и оформленным документом);

– вид документов (представляет собой справочники типов документа; характеризуется типом документа);

– приказ, распоряжение, справка (хранит дату формирования документа для всех видов);

– приказ (описывает приказы по всем его видам и основания назначения);

– распоряжение (описывает распоряжения по всем его видам и основания назначения данного документа);

– справка (описывает выдаваемую справку по месту требования);

– контингент (хранит информацию о студентах на момент поступления в университет. Информация следующего характера: гражданство, национальность, адрес проживания и регистрации, иностранный язык, изучаемый ранее, информация о родителях, семейное и социальное положение, место учебы до поступления).

Чтобы построить схему реляционной базы данных необходимо определить связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между уровнями. Одним из основных требований к организации базы данных является обеспечение возможности поиска одних сущностей по значениям других, для чего необходимо установить между ними определенные связи [3].

Для реализации информационной системы учета контингента студентов установим все связи между объектами. А именно, рассмотрим всю информационную систему в совокупности и определим взаимное влияние объектов, составляющих систему.

В проектируемой базе данных все таблицы имеют первичные ключи, присутствуют связи двух типов «один ко многим» и «один к одному» [4].

Концептуальная схема базы данных, в первой нормальной форме, представлена на рисунке 1. Определим совокупность отношений, которые составляют базу данных. Данная совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

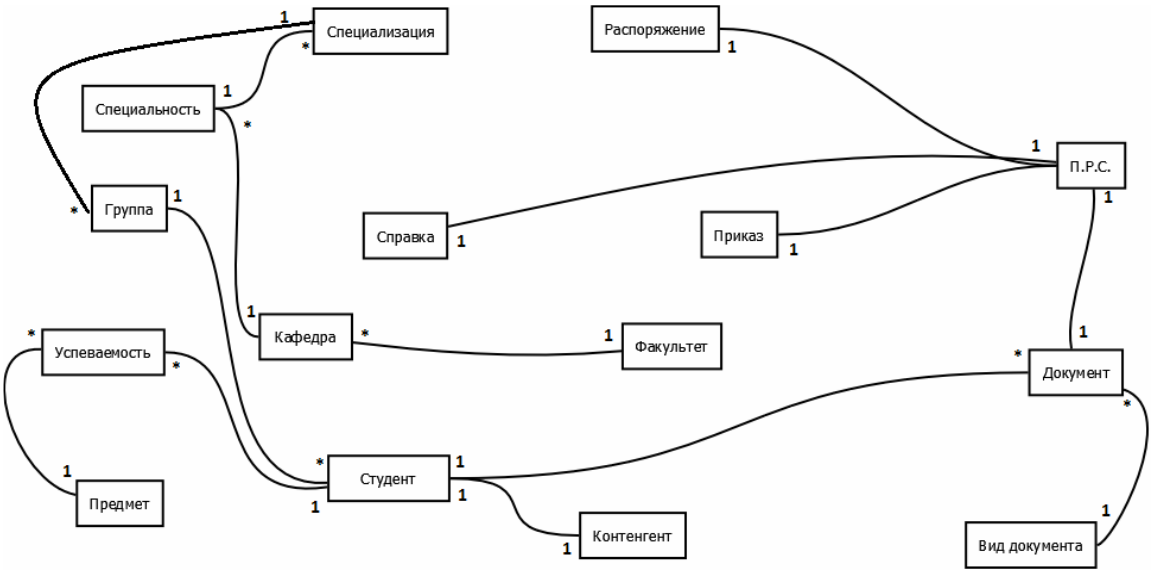

Рис. 1. Концептуальная схема базы данных

На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2 представлены отношения для базы данных контингента студентов университета.

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждое отношение должно обладать хотя бы одним ключом [5].

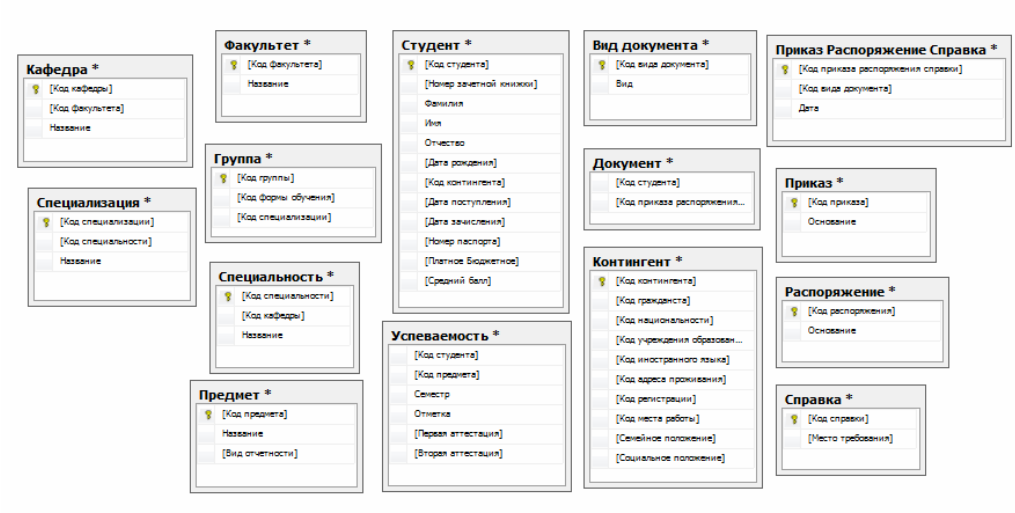

Рис. 2. Набор необходимых отношений базы данных

После определения связей между таблицами, назначения ключей и построения реляционной базы данных, остается привести ее к третьей нормальной форме. Например, имеем таблицу контингент, у которой атрибуты гражданство, национальность, иностранный язык, учреждение образования, место работы, адрес проживания до поступления, адрес регистрации, семейное положение, социальное положение, родственные связи определяют информацию о студенте. Эту таблицу логичнее разбить на несколько отдельных.

Поэтому необходимо привести эту таблицу к третьей нормальной форме. Результатом приведения будут таблицы: место работы, иностранный язык, гражданство, национальность, регистрация, адрес проживания до поступления, учреждение образования (рис. 3).

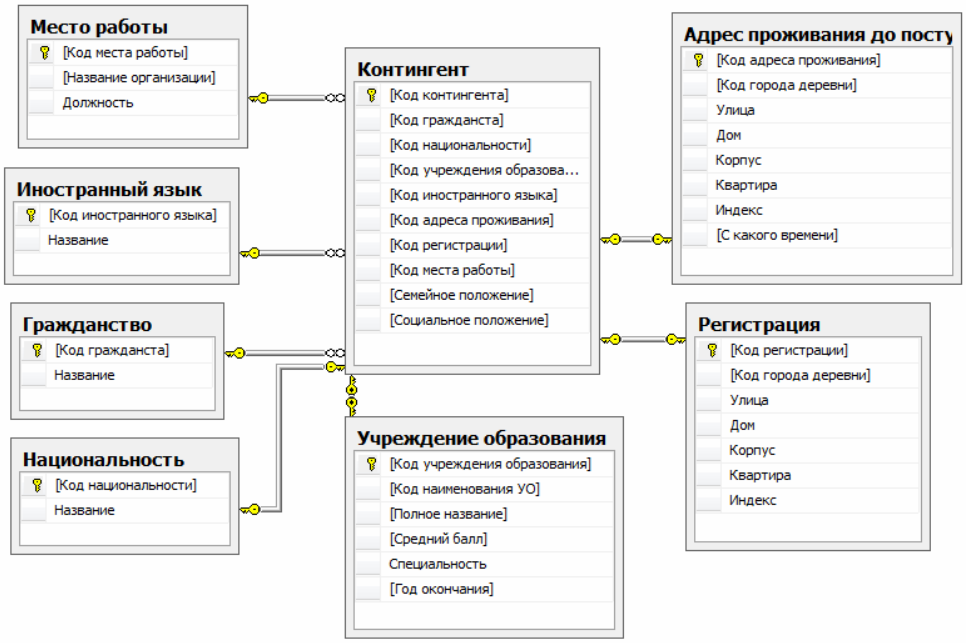

Рис. 3. Результат приведения таблиц к третьей нормальной форме

Процесс нормализации применяем также к таблицам распоряжение, приказ, родственные связи, адрес прописки до поступления, адрес регистрации, группа, учреждение образования. В итоге нормализации таблиц получаем новые таблицы перевод в другую группу – для оформления документа на перевод в другую группу, отчисление, восстановление – для оформления документов на отчисление, либо восстановление, перевод – для оформления документов на перевод, академический отпуск – для оформле-

ния документов на академический отпуск, смена студенческого билета, зачетной книжки – для оформления документов на смену студенческого билета, либо зачетной книжки, начисление стипендии – для оформления документов на начисление стипендии, смена фамилии – для оформления документов на смену фамилии, выпуск – для оформления документов на получение квалификации, наименование учреждения образования – для хранения всех видов учреждений образования, информация о родственниках – для хранения информации о каждом родственнике студента, город, деревня – для хранения информации о городах и деревнях, район – для хранения информации о районах, область – для хранения информации об областях, страна – для хранения информации о странах, состав группы – функциональная таблица для формирования групп.

Для организации хранения данных была выбрана система управления базами данных (СУБД) MySQL Server 5.5, так как она является удобной в использовании, многопоточной, свободно распространяемой и мультиплатформенной [6].

Для добавления, изменения и удаления данных написаны пользовательские функции и процедуры, многие из которых организуют поддержание целостности системы. Самое важное ограничение целостности на уровне отдельных полей – это тип данных. Механизм баз данных MySQL обеспечивает богатый спектр типов данных.

Целостность системы поддерживается также при помощи триггеров и ограничений в виде Check [5, 7].Созданы триггеры, которые проверяют регистр букв в именах, автоматически формируют название группы, полное и сокращенное имя студента, проверяют отметки студента (помимо цифр от одного до десяти, можно вводить «зачтено»/«не зачтено») и т.д. Для безошибочной работы с данными вводим следующие ограничения: проверка форм обучения на имеющиеся в базе, проверка семестров (нельзя ввести число меньше единицы и больше 12), проверка всевозможных дат (например, чтобы дата рождения не была больше текущей или, чтобы студенту не превышало 60), проверка среднего балла на вхождение в отрезок [4.0, 10.0], проверка типа учреждения образования (выбор только из существующих в базе) и т.д.

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

ЛИТЕРАТУРА

1. Деканат [Электронный ресурс] // Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Декан/. – Дата доступа: 10.04.2015.
2. Что такое деканат в университете? [Электронный ресурс] // Студенческая жизнь. – Режим доступа: http://life-students.ru/chto-takoe-dekanat-v-universitete/. – Дата доступа: 10.04.2015.
3. Малыхина, М. Базы данных: основы, проектирование, использование / М. Малыхина. – М. : BHV, 2004. – 512 с.
4. Дейт, К.Д. Введение в системы баз данных / К.Д. Дейт. – М. : Вильямс, 2005. – 1328 с.
5. Боуман, Дж. Практическое руководство по SQL / Дж. Боуман, С. Эмерсон, М. Дарновски. – Изд. 4-е. – М. : Вильямс, 2002. – 352 с.
6. Кузнецов, М.В. MySQL 5 / М.В. Кузнецов, И.В. Симдянов. – СПб. : БХВ-Петербург, 2010. – 1024 с.
7. Грофф, Дж. Полное руководство по SQL / Дж. Грофф, П. Вайнберг. – М. : BHV, 2001. – 816 с.

УДК 004

РАЗРАБОТКА ФОРМАЛЬНОГО ЯЗЫКА ОПИСАНИЯ ПОВЕДЕНИЯ АГЕНТА НА БАЗЕ НЕЧЕТКОЙ ЛОГИКИ

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

Рассмотрены основные моменты создания формального языка для искусственного интеллекта на базе нечеткой логики.

Нечеткая логика – это логика, в которой мы можем оперировать не только с лог. «0» или лог. «1», а так же со значениями в интервале [0;1]. Рассмотрим классический пример: расстояние от машины до препятствия в обыкновенной логике: мы бы оперировали такими значениями: 1(150м) – далеко, 0(0м) – близко, а нечеткая логика подразумевает нечеткие понятия, такие как очень близко, близко, средняя, далеко и т.д., такие понятия характерны для мышления человека (рис. 1.) [1].

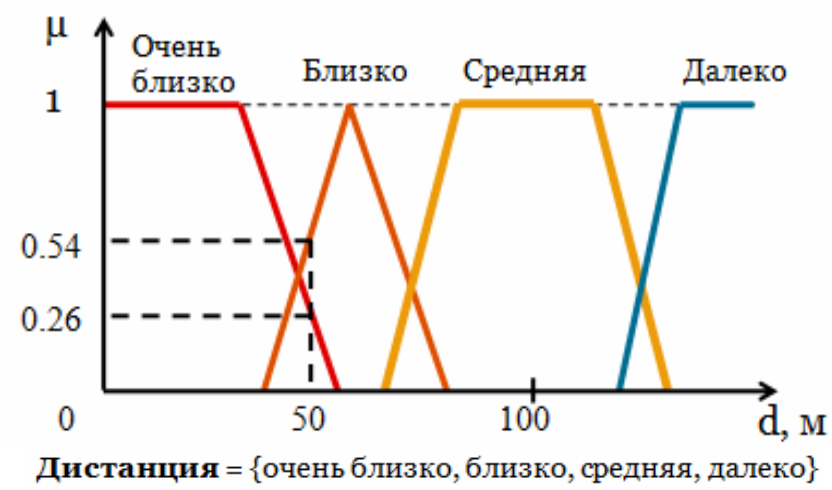

Рис. 1. Графическое изображение дистанции от машины до препятствия

Все основные понятия (например, скорость, дистанция, уровень жизней и т.д.) в нечеткой логике называются лингвистическими переменными, а их возможные значения (например: скорость (маленькая, большая, ...), дистанция (близко, далеко, ...) и т.д.) называются лингвистическими термами. Каждый терм охватывает какой-либо промежуток из линг. переменной, которая называется функция линг. терма. На представленном графике ось y(μ) показывает степень принадлежности для каждого значения дистанции.

Расширим предыдущий пример. Добавим еще 2 переменных: «направление», которая отвечает, в какую сторону движется машина, и «рулевой угол» – куда поворачивать в зависимости от дистанции и направления. Данные переменные будут иметь следующие термы:

Направление = {левое, прямое, правое}

Рулевой угол = {резко влево, влево, прямо, вправо, резко вправо}

Чтобы определить в какую сторону необходимо повернуть «Рулевой угол», воспользуемся различными условиями, представленными в таблице.

Таблица

Условия для определения «рулевой угол»

	Дистанция				
Направление		Очень близко	Близко	Средняя	Далеко
	Правое	Резко влево	Резко влево	Влево	Прямо
	Прямо	Резко влево	Влево	Влево	Прямо
	Левое	Резко вправо	Резко вправо	Вправо	Прямо

$\mu 1 = 0.89$ $\mu 2 = 0.60$

Допустим: if «направление» = «правое» and «дистанция» = «средняя» then «рулевой угол» = «влево». В нечеткой логике в условиях нет понятия истинно или ложно, здесь оперируют понятием вероятность срабатывания. Причем она для этого условия будет определяться по формуле [2]

$$\min(\mu 1, \mu 2) \quad (1)$$

для операции «И».

Для операции «ИЛИ» по формуле:

$$\max(\mu 1, \mu 2), \quad (2)$$

для отрицания по формулам:

$$1 - \mu 1, \quad (3)$$
$$1 - \mu 2. \quad (4)$$

Выходной параметр для данного примера будет вычисляться по формуле (введя следующие замены A = «рулевой угол»; T1 = «влево»):

$$A = \min(\mu 1, \mu 2) \cdot T1. \quad (5)$$

Если рассматривать более сложные конструкции (вложенные условия), то выходной параметр будет находиться следующим образом:

Допустим, у нас имеется конструкция следующего вида:

```
if (cond1)
{
   A is T1
   if (cond2) A is T2
}
if (cond3) A is T3
```

где cond1, cond2, cond3 – какие-либо условия, у которых степени принадлежности $\mu 1$, $\mu 2$, $\mu 3$соответственно.

A – лингвистическая переменная.

T1, T2, T3 – лингвистические термы переменной *A*.

‘if’ – условный оператор.

‘is’ – оператор присвоения.

Из данного выражения выходной параметр *A* будет находится по следующей формуле:

$$A = \frac{T1 \cdot \mu_1 + T2 \cdot \min(\mu_1, \mu_2) + T3 \cdot \mu_3}{\mu_1 + \min(\mu_1, \mu_2) + \mu_3}. \quad (6)$$

Пройдя введение можно рассмотреть основные моменты для создания ИИ в играх на основе данной логики.

Основная проблема, которая возникает на первом этапе разработки это: как сделать так, чтобы машина понимала нечеткие понятия и интерпретировала в нужный для нее язык. Чтобы решить данную проблему необходимо простроить следующий механизм, который изображен на рисунке 2 [3].

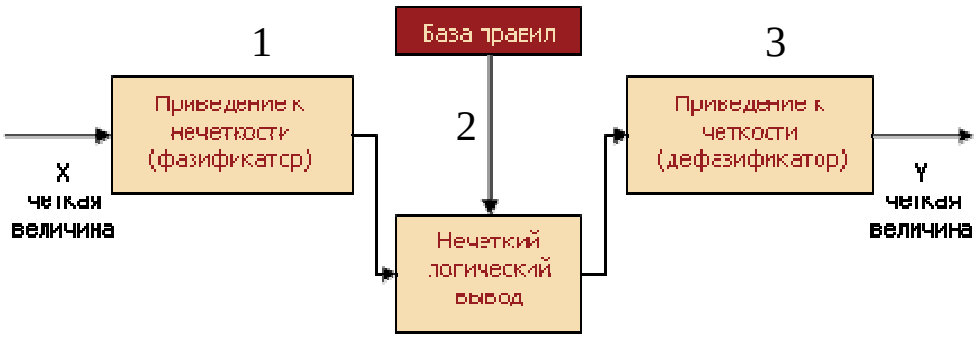

Рис. 2. Механизм для использования нечеткой логики

Суть его такова: разберем тот же пример с дистанцией от машины до препятствия. На вход данного «черного ящика» поступает четкая дистанция, допустим 100 м. Затем элемент ящика (приведение к нечеткости) интерпретирует данное значение в нечеткий терм (из примера «средняя»). Полученный терм переходит на следующий блок (нечеткий логический вывод) где располагаются различные условия, которые оперируют с данной лингвистической переменной (в данном примере «Дистанция»). После выполнения условий полученное действие так же является еще нечетким (к примеру выполнилось условие: если «направление» = «правое» и «дистанция» = «средняя», то «рулевой угол» = «влево») «рулевой угол» = «влево». Данное действие переходит к блоку (приведение к четкости) и получает какой-то четкий угол поворота (к примеру, 15о).

Чтобы воспользоваться данным механизмом в языках программирования необходимо написать свой язык, который будет, поступившее на вход четкое значение преобразовывать в нечеткое, затем с помощью нечетких условий определяет действие и обратно на выход отдавать четкое значение полученного действия.

Пример разработанного формального языка описания поведения агента (юнита) на базе нечетких логических условий и установок:

```
set "R" (100, 1) // Радиус объекта
//Нечеткие множества параметров
set "все слева" (-0.1, 0;1.75, 0.5;3, 1;3.3, 0)
set "более слева" (-0.1, 0;0.8, 1;1.2, 0)
set "чуть слева" (-0.01, 0;0, 1;0.03, 1;1, 0)
//Нечеткие множества действий
set "чуть повернуть направо" (-0.02, 1)
set "резко повернуть направо" (-0.2, 1)
set "чуть повернуть налево" (0.01, 1)
set "резко повернуть налево" (0.2, 1)

//Правила
if "угол на цель" is not "впереди" {
        "скорость корпуса" ~ "стоп"
        if "угол на цель" is "все справа" { "поворот корпуса" ~ "резко повернуть направо" }
        if "угол на цель" is "все слева" { "поворот корпуса" ~ "резко повернуть налево" }
}
```

Данный формальный язык реализует 2 блока (см. рис. 2): приведение к нечеткости и нечеткий логический вывод с базой правил. 3 блок уже реализуется непосредственно с помощью языков программирования получением данных и данного языка (см. рис. 2).

Описание формального языка будет иметь вид (в форме Бэкуса-Наура)[4, 5]:

```
<prog> ::= <def> <block>
<def> ::= set <term> (fp,fp; fp, fp; fp, fp) | <def> set <term> (fp,fp; fp, fp; fp, fp).
<block> ::= <operator> | <prog>
<operator> ::= <ifoperator> | <setoperator>
<ifoperator> ::= if <cond> { <block>} | if <cond> { <block>} else {<block>}
<cond> ::= <term> | <cond> and <cond> | <cond> or <cond> | (<cond>) | not <cond>
<term> ::= name
<setoperator> ::= <lparam> is <term>
<lparam> ::= name
```

Вероятность срабатывания того или иного правила будет высчитываться по тем же формулам, о которых говорилось ранее.

ЛИТЕРАТУРА

1. Шеври, Ф. Нечеткая логика / Ф. Шеври, Ф. Гели. – Вып. 31.
2. ЭР – Нечеткая логика [Электронный ресурс]. – Режим доступа: http://fuzzy-group.narod.ru/files/Fuzzy_Modeling/Lection07.Fuzzy.logic.pdf. – Дата доступа: 20.08.2015.
3. ЭР – Математические основы нечеткой логики [Электронный ресурс]. – Режим доступа: http://bourabai.ru/tpoi/fuzzy.htm. – Дата доступа: 20.08.2015.
4. ЭР – Формальный язык [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA. – Дата доступа: 20.08.2015.
5. ЭР – форма Бэкуса-Наура [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0_%D0%91%D1%8D%D0%BA%D1%83%D1%81%D0%B0_%E2%80%94_%D0%9D%D0%B0%D1%83%D1%80%D0%B0. – Дата доступа: 20.08.2015.

УДК 004

ОСОБЕННОСТИ МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ АГЕНТА ПРИ ОБХОДЕ ПРЕПЯТСТВИЙ С ИСПОЛЬЗОВАНИЕМ НЕЧЕТКОЙ ЛОГИКИ

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

Рассматриваются проблемы при обходе препятствий агентами с использованием нечеткой логики и дальнейшие способы их решения, которые максимально приближают поведение агента к идеальным случаям.

При использовании данной логики при обходе препятствий возникают следующие проблемы:

1. Реализация плавного обхода препятствия/й.
2. Реализация обхода 2-ух впереди стоящих препятствий.
3. Реализация обхода вдоль границы карты в игре.

Непосредственно рассмотрим каждую проблему в отдельности.

На рисунке 1 показано как будет вести себя агент (юнит) пользуясь обычными нечеткими правилами при обходе одного препятствия.

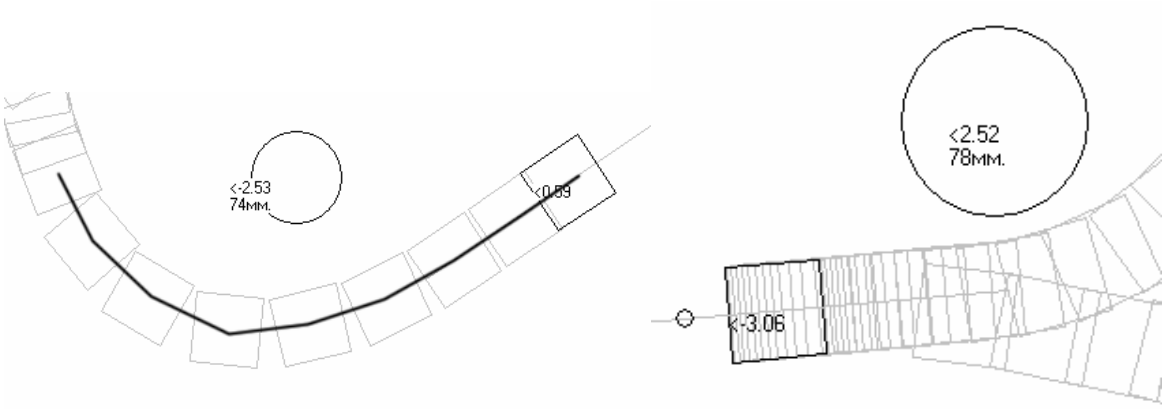

Рис. 1. Плавный обход препятствия агентом

Как говорилось ранее у каждого условия, есть вероятность срабатывания. При любой допустимости срабатывания, даже пусть и при 0.1 сработает какое-либо условие, то в траектории движения будут появляться изгибы. Для того чтобы решить данную проблему и подойти к плавному обходу, необходимо установить порог на вероятность срабатывания, т.е. допустим, установим данный порог на 0.2 из 1, если вероятность срабатывания какого-либо условия меньше данного порога, то оно никогда не сработает, что даст возможность агенту двигаться по более плавной и правильной траектории (см. рис. 1). Такая же проблема возникает, если агенту необходимо пройти вдоль нескольких препятствий, пример приведен на рисунке 2.

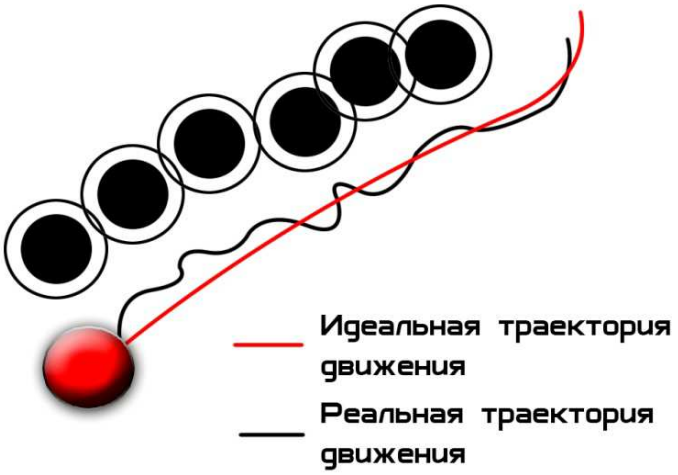

Рис. 2. Обход агентом несколько, подряд стоящих, препятствий

Для решения данной проблемы необходимо анализировать расстояние и угол направления к рядом стоящим препятствиям. Если расстояние достаточное и угол от препятствия довольно большой агент продолжает движение вдоль прямой, пока не покинет предел видимости всех препятствий, но если угол довольно маленький, агент начинает небольшой маневр, до достижения необходимого угла.

Следующая проблема, которая возникает при построении обхода препятствий это обход впереди стоящих 2-ух препятствий. Пример обхода 2-ух впереди стоящих препятствий показан на рисунке 3.

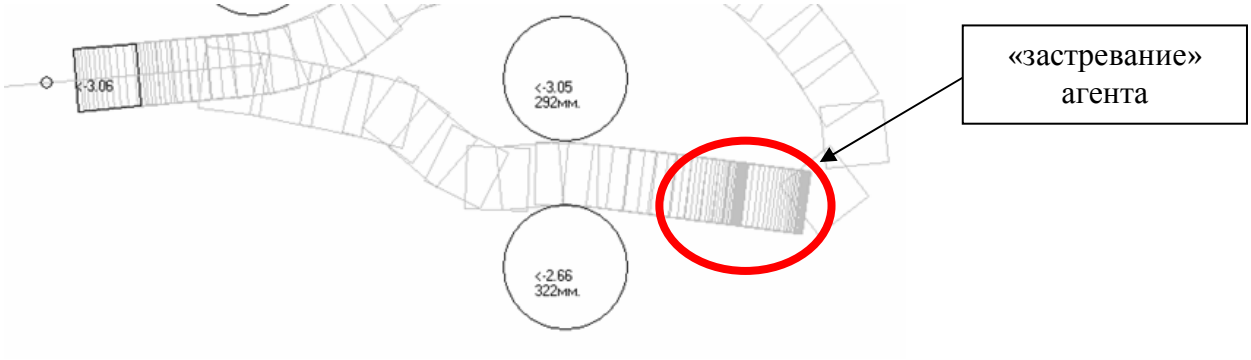

Рис. 3. Обход агентом 2-ух впереди стоящих препятствий

Из рисунка можно увидеть, что агент, пользуясь стандартными условиями, будет по началу пятиться от одного препятствия к другому, уменьшая расстояние до них. В момент, когда уже агент достиг препятствий, возникнет своего рода «застревание» агента между двумя препятствиями, через которые агент в течение длительного времени сможет протиснуться и поехать дальше. Чтобы решить данную проблему, необходимо знать расстояние между препятствиями, если его хватает, тогда агент спокойно проходит по прямой, в противном случае обходит одно из них. Так же чтобы достичь максимально плавный обход, необходимо поставить порог на вероятность срабатывания условий. Еще следует учесть тот момент, какому из препятствий отдавать приоритет, если вдруг расстояния между ними недостаточно, чтобы пройти без проблем, для этого просто в условиях отдаем больший приоритет в ту сторону обхода, которая необходима (лево или право, вверх или вниз).

Итак, 2 из 3 проблем рассмотрено, перейдем к 3-ей. Ее суть заключается в том, как будет вести себя агент по отношению к границе карты. При больших скоростях движения агента может возникнуть такая ситуация, что он выйдет за ее границу и вернется обратно. Пример данной ситуации показан на рисунке 4.

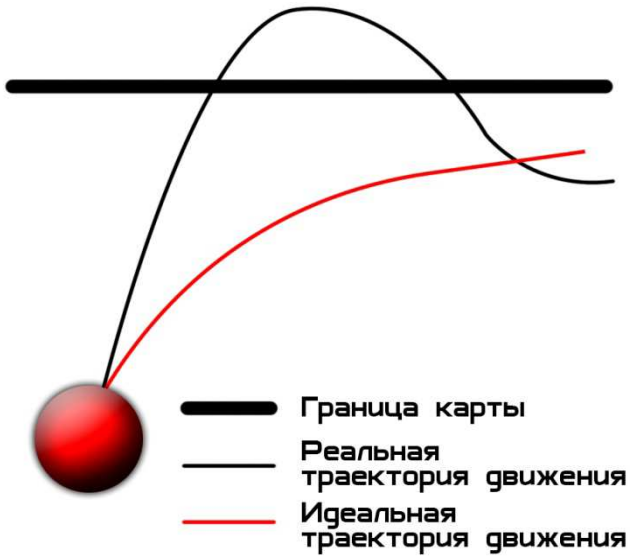

Рис. 4. Траектория движения агента вдоль границы карты

Для решения данной проблемы нам необходимо устанавливать расстояние между агентом и границей. Если агент движется с большой скоростью, а расстояние уже недостаточное для маневра, то он столкнется с границей и потеряет часть жизней. Если расстояния достаточно, то он сбросит скорость и начнет делать маневр и двигаться вдоль границы, при этом каждый раз высчитывается расстояние до границы и угол, на котором от агента она расположена.

На рисунке 5 показана работа программы по обходу препятствий агентами [2].

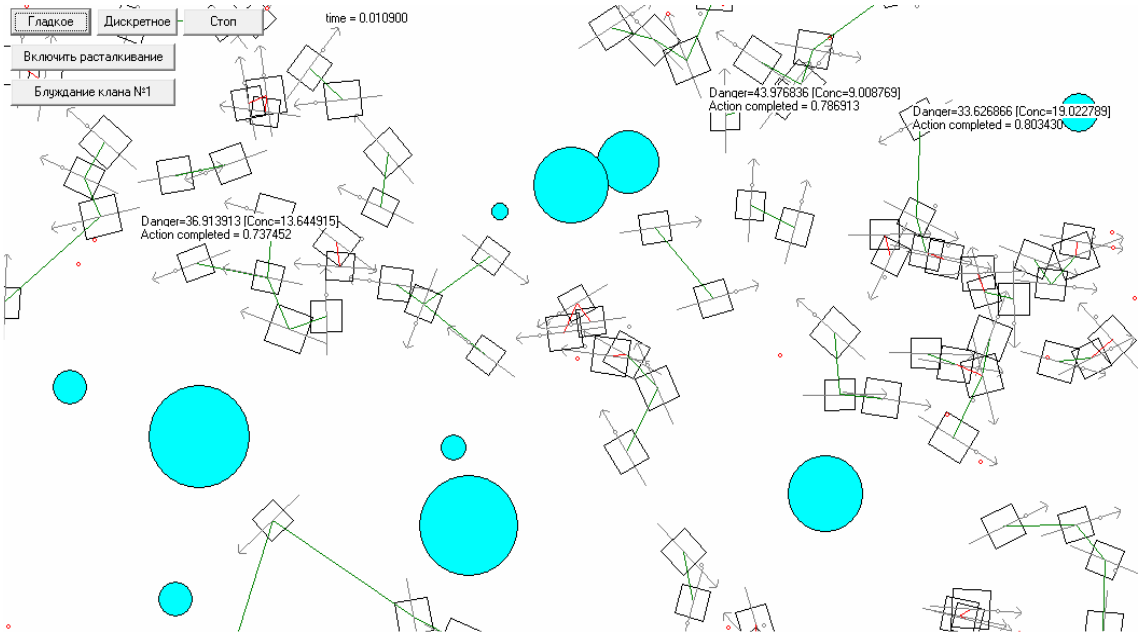

Рис. 5. Обход агентами препятствий

Таким образом, разрабатываемый нами проект будет решать все проблемы, которые были приведены выше, чтобы сделать ИИ в игре максимально простым в реализации и сложным в поведении.

ЛИТЕРАТУРА

1. ЭР – Нечеткая логика на практике [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/125614/. – Дата доступа: 21.08.2015.
2. ЭР – Модель интеллектуальных агентов на нечеткой логике высшего типа [Электронный ресурс]. – Режим доступа: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/84682/02-Yershov.pdf?sequence=1. – Дата доступа: 21.08.2015.
3. Шеври, Ф. Нечеткая логика / Ф. Шеври, Ф. Гели. – Вып. 31.

УДК 004.02

ОТОБРАЖЕНИЕ ОБЪЕКТОВ БАНКА НА КАРТЕ

С.В. РЕЗАНОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассмотрен способ группировки объектов на картах Google Maps SDK. Дано подробное описание проблемы отображения объектов на карте и ее решение.

Перед реализацией научной работы проводилась оценка трудоемкости отдельных модулей приложения. Практически все приложение можно реализовать с помощью готовых средств. Однако самым трудоемким для разработки является группировка объектов на карте. Во время анализа не было найдено ни одного решения данной проблемы. Дело в том, что с появлением первой версии Apple iOS стандартным картографическим сервисом являлся «Google Maps». Многие сторонние разработчики реализовали библиотеки для группировки маркеров. Летом 2012 года с появлением iOS6 Apple отказались от всех сервисов Google, в том числе и от «Google Maps» [1]. Все приложения, которые использовали карты от Google, начали отображать карты от Apple. Таким образом, все ранее написанные библиотеки работали уже с «Apple Maps». В начале 2013 года Google выпустила первую версию «Google Map SDK» для iOS и активно продолжает развивать по сегодняшний день [2]. Однако уже реализованной возможности группировки маркеров нет. Так как карты от Apple не достаточно детализированы для России, а пользователями приложения будут в основном жители Московской области, то «Apple Maps» не подходит для реализации программы. Другие картографические сервисы работают не стабильно на новых версиях ОС. Поэтом альтернатив «Google Maps» нет. Принято решение реализовывать группировку маркеров самому.

Разработка алгоритма. Так как разработанный алгоритм не является простым и коротким в описании, то его сложно изобразить в виде блок-схемы, поэтому будет словесно описана основа алгоритма.

Московский кредитный банк имеет более 4 тысяч объектов в Москве. Если их отобразить на карте, то она потеряет свою информативность. Результат отображения объектов без группировки изображен на рисунке 1.

Проблема неинформативности карты является не самой значимой. Дело в том, что при каждом обновлении положения карты, система перерисовывает все маркеры. Так как маркеров слишком много, то приложение зависает.

Решением этой проблемы могло быть добавление на карту только тех объектов, которые попадают в поле отображения карты. Но если выбрать масштаб карты, при котором будет отображаться вся Москва целиком, то почти все объекты входят в поле отображения карты. Если их добавить на карту, то приложение зависнет.

Для того чтобы карта работала стабильно надо отображать ограниченное число маркеров, а на них указывать количество объектов которые находятся в регионе маркера. Таким образом, на карте прорисовывается до 30 маркеров. Теперь требуется максимально быстро находить объекты, которые нужно группировать. Простой перебор всех объектов занимает слишком много времени, что не допустимо для работы приложения. Выходом стал перебор не всех объектов банка, а маркеров, в которых уже сгруппированы объекты. Для этого был унаследован класс стандартного маркера из Google Maps SDK и помимо методов обработки группировки добавлены три свойства:

– groupZoom – величина масштаба карты, при котором была произведена группировка;

– markers – массив, в котором хранятся сгруппированные маркеры;

– count – количество сгруппированных маркеров.

Таким образом, алгоритм работает по принципу дерева: если масштаб уменьшился, то поиск группируемых объектов проводится среди видимых маркеров. Вместо них создается новый маркер со средними координатами группируемых объектов. Объекты группы записываются во множество markers, в переменную groupZoom записывается масштаб карты, при которой производится группировка, а в count – суммарное значение свойств count группируемых объектов. Далее этот же маркер может быть так же сгруппирован. При уменьшении масштаба элементы дерева рекурсивно перебираются и разгруппировываются до того пока groupZoom объектов меньше текущего масштаба. Результат работы алгоритма изображен на рисунке 2.

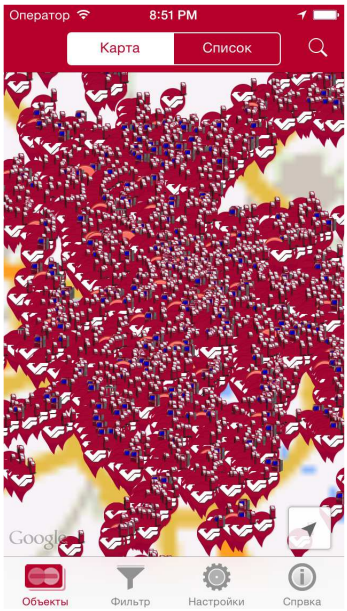

Рис. 1. Отображение объектов на карте без группировки

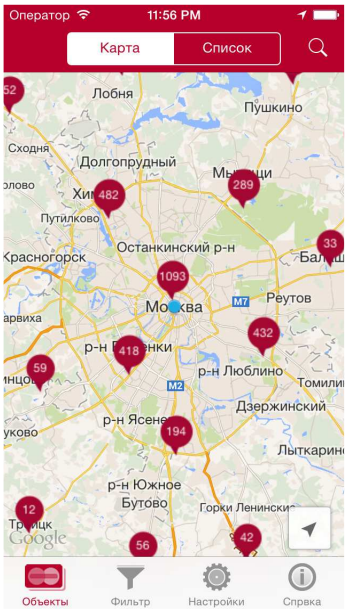

Рис. 2. Результат работы алгоритма

В результате разработки приложения под ОС Apple iOS для отображения объектов Московского кредитного был разработан алгоритм группировки маркеров для карт Google Maps SDK.

ЛИТЕРАТУРА

1. Планета iPhone [Электронный ресурс] // Apple отказалась от Google Maps в iOS 6 из-за голосовой навигации. – Режим доступа: https://www.planetiphone.ru/news/085/apple-otkazalas-ot-google-maps-v-ios-6-iz-za-golosovoj-navigacii.html. – Дата доступа: 16.05.2014.
2. Google Developers [Электронный ресурс] // Google Maps SDK for iOS. – Режим доступа: https://developers.google.com/maps/documentation/ios-sdk/. – Дата доступа: 16.05.2014.

УДК 004.02

ВЗАИМОДЕЙСТВИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ПО ОТОБРАЖЕНИЮ ОБЪЕКТОВ МОСКОВСКОГО КРЕДИТНОГО БАНКА С ПОЛЬЗОВАТЕЛЕМ

С.В. РЕЗАНОВ
(Представлено: Е.Р. СУХАРЕВ)

Рассмотрен интерфейс мобильного приложения под ОС Apple iOS для отображения объектов Московского кредитного банка и взаимодействие пользователя с этим интерфейсом. Отображается реальный интерфейс текущего приложения. Можно проследить весь сценарий действий пользователя.

Пользовательский интерфейс – без сомнения, важная часть проектирования приложения. Пользователь крайне заинтересован в высоком качестве интерфейса приложения и его интуитивного построении, в ином случае таким приложением просто не будут пользоваться. Главной задачей при разработке интерфейса является его проектирование. Относительно не большой экран телефона должен содержать в себе все элементы управления, но не загромождать его. Так же интерфейс приложения должен адаптироваться под все возможные экраны устройств. В этой статье рассмотрен интерфейс игры и особенности взаимодействия игрока с этим интерфейсом.

Проектирование интерфейса приложения.

В приложении «МКБ Мобайл» необходимо реализовать следующие функции:
– отображение объектов на карте;
– отображение объектов в виде списка;
– отображение детальной информации;
– поиск объектов;
– настройка приложения;
– фильтрация объектов;
– отображение справочной информации.

Каждая из вышеперечисленных функций будет реализована на соответствующем экране приложения. Рассмотрим макеты экранов, которые необходимо реализовать в приложении.

Первым экраном является экран отображения объектов на карте (рис. 1). На карте отображаются маркеры, на которых отображены иконки типов объектов или цифра, которая сообщает о количестве сгруппированных объектов. Нажатие на маркер переводит на экран детальной информации об объекте. 4 кнопки таббара переключают экраны представления «Отображение объектов», «Фильтры», «Настройки», «Информация о банке». Кнопка на карте переводит карту на текущее местонахождение пользователя. Сегментконтроль на панели навигации переключает представление объектов в виде списка или карты. Кнопка «Поиск» открывает экран поиска (рис. 2).

Рис. 1. Макет дизайна экрана отображения объектов на карте

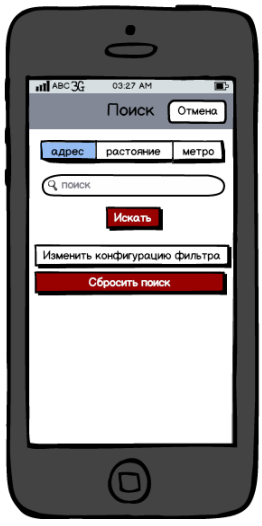

Рис. 2. Макет дизайна экрана «Поиск»

С помощью сегментконтроля можно переключить тип поиска. Кнопка «Искать» запускает поиск и закрывает текущий экран, отображая результат поиска. Так же можно изменить конфигурацию фильтра –

нажав на кнопку «Изменение конфигурации фильтра». Кнопка «Сбросить поиск» отчищает поле ввода и отменяет текущий результат поиска. Кнопка «Отмена» закрывает экран поиска. Если на экране отображения объектов сегментконтроля переключить на «Список» то объекты отобразятся в виде сортированного по удаленности списка (рис. 3). На экране «Список объектов» в панели навигации появляется кнопка «Обновить». Нажатие на нее пересчитывает расстояние до объектов и пересортировывает список. Нажатие на ячейку списка переводит на экран детальной информации об объекте (рис. 4).

Рис. 3. Макет дизайна экрана «Список объектов»

Рис. 4. Макет дизайна экрана «Детальная информация об объекте»

На экране «Детальная информация об объекте» отображается вся известная информация. В заголовке панели навигации отображается тип объекта. Если нажать на вторую кнопку таббара, то можно перейти на экран «Фильтр» (рис. 5). На экране «Фильтр» в виде списка отображаются параметры по которым возможно фильтровать объекты. В конце списка находится кнопка «Сбросить фильтр». Нажатие на эту кнопку переводит фильтрация в режим по умолчанию. Нажатие на третью кнопку таббара отображает экран настроек (рис. 6).

На экране «Настройки» можно переключить тип отображаемой карты, так же включить или выключить автоматическое обновление данных и вручную обновить данные. Последняя кнопка таббара переводит на экран «О Банке» (рис. 7).

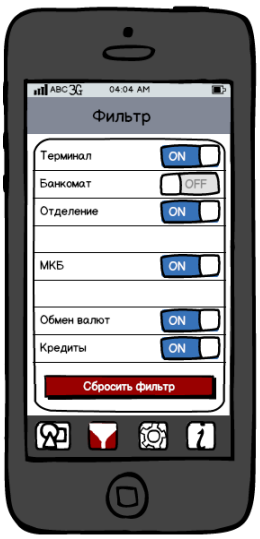

Рис. 5. Макет дизайна экрана «Фильтр»

Рис. 6. Макет дизайна экрана «Настройки»

Рис. 7. Макет дизайна экрана «О Банке»

На экране «О Банке» отображается основная информация о Московском кредитном банке.

Рис. 8. Экран загрузки приложения

Результаты реализации.

Первое, что видит пользователь после установки приложения это иконка. На ней отображен логотип Московского кредитного банка (рис. 8).

После загрузки приложения открывается экран отображения объектов (рис. 9). При изменении масштаба маркеры группируются. На сгруппированных маркерах отображается количество объектов в группе (рис. 10).

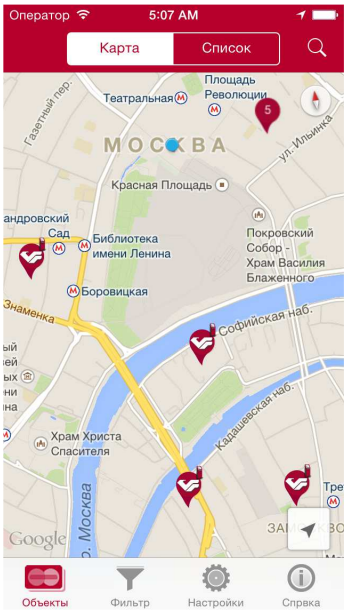

Рис. 9. Экран отображения объектов

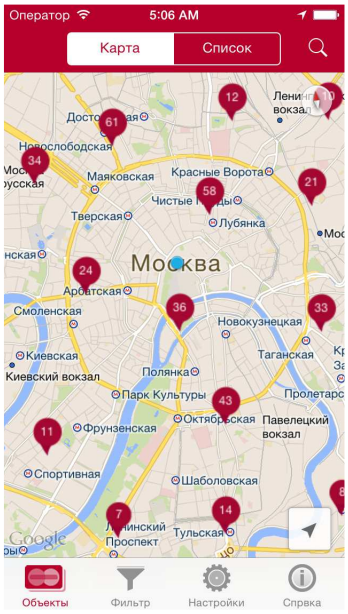

Рис. 10. Группировка объектов на карте

При активном фильтре и поиске отображаются напоминающие информационные сообщения. Это позволит пользователю не забыть о том, что отображаются не все объекты (рис. 11). Отображение объектов в виде сортируется и группируется по расстоянию до объекта (рис. 12).

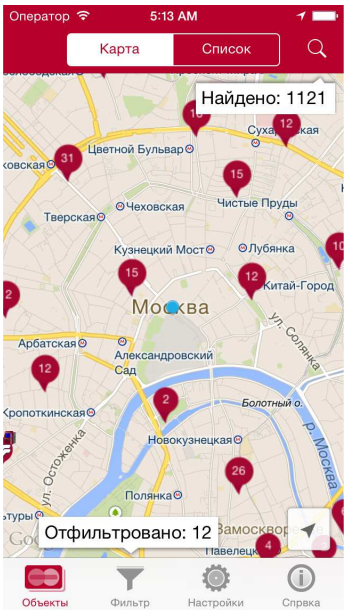

Рис. 11. Информационные сообщения о фильтрации и поиске

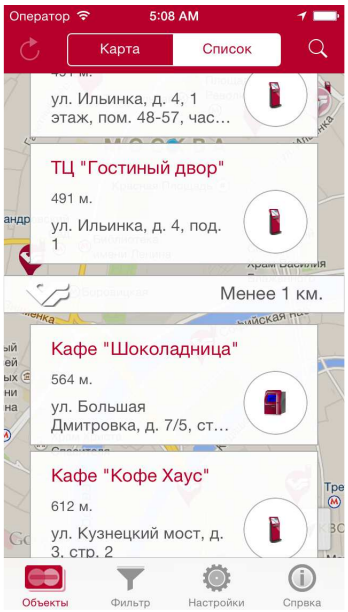

Рис. 12. Отображение объектов в виде списка

Приложения поддерживает все устройства под управление iOS7 в различных ориентациях экрана. На экранах iPad можно просматривать одновременно больше информации (рис. 13).

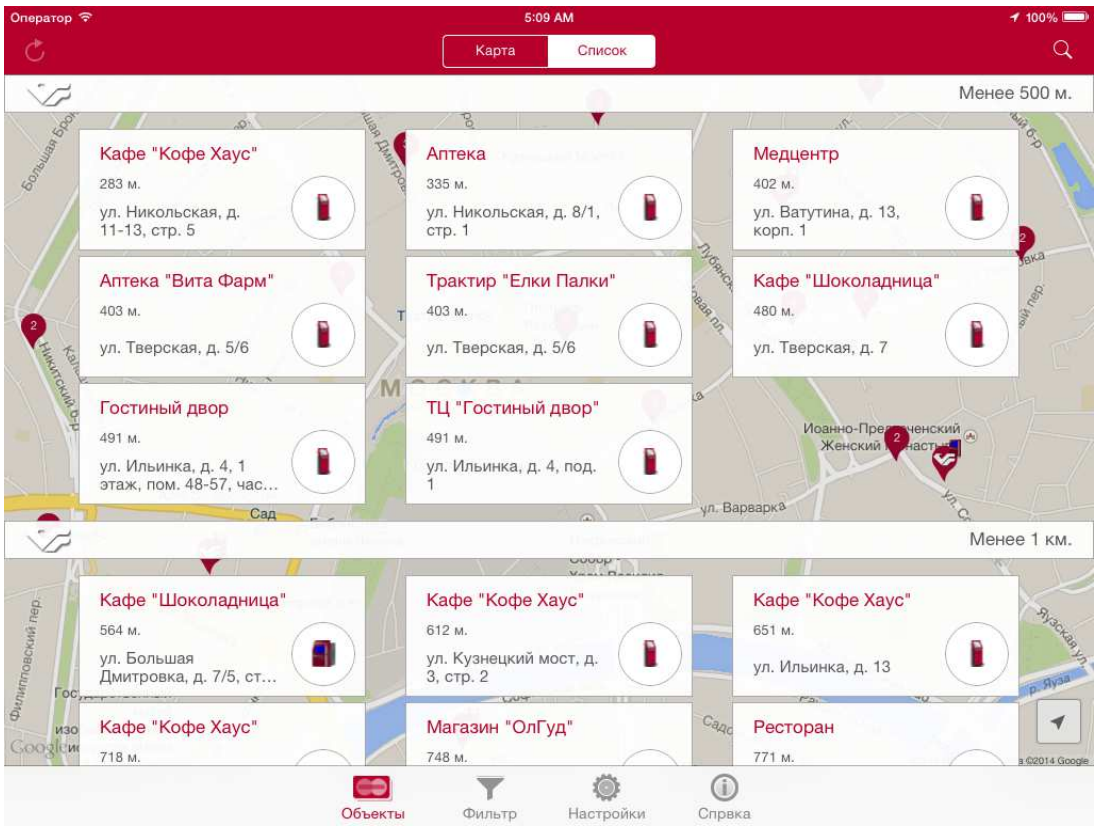

Рис. 13. Список объектов на iPad в горизонтальной ориентации

Для большей информативности объекты отличаются по типу графически. Перечень маркеров и иконок для типов объектов отображен в таблице.

Таблица

Перечень маркеров и иконок типов объектов

Тип	На карте	В списке
Терминал		
Банкомат		
Отделение		
Операционная касса		
Остальные		

Все сообщения об ошибках корректно отображаются в диалоговых окнах (рис. 14).

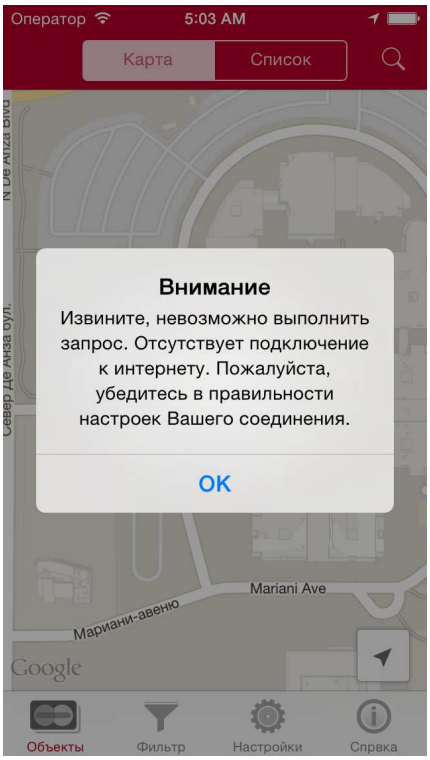

Рис. 14. Отображение сообщений

В статье подробно рассмотрен пользовательский интерфейс мобильного приложения под ОС Apple iOS для отображения объектов Московского кредитного банка, а так же сценарий действий пользователя. Приведены подробные иллюстрации игрового интерфейса.

УДК 004.42

ВЗАИМОДЕЙСТВИЕ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА И МОБИЛЬНОГО УСТРОЙСТВА НА БАЗЕ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID ПОСРЕДСТВОМ ТЕХНОЛОГИИ NFC

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

Произведены исследования организации взаимодействия персонального компьютера и мобильного устройства на базе операционной системы Android с современной технологии близкого контакта NFC. Исследованы протоколы взаимодействия и формат передаваемых данных, а также организация пиринговой сети между ПК и мобильным устройством через NFC.

NFC (Near Field Communication) – это технология, позволяющая смартфонам и другим устройствам обмениваться данными по беспроводной высокочастотной связи друг с другом при близком контакте, обычно около десяти сантиметров [1]. Технология NFC используется во многих областях:

1. Коммерция.
2. Устройства NFC могут применяться в бесконтактных платежных системах, в электронных билетных смарткартах, а также для осуществления платежей с мобильных устройств.
3. Обмен информацией с другими людьми.
4. NFC может быть использована для обмена контактами, фотографиями, видео или другими файлами, а также для входа в мультиплеерные игры.
5. Подтверждение личности и коды доступа.
6. NFC-устройства используются как электронные удостоверения личности.
7. Другие возможности [1].

Смартфоны с NFC могут работать вместе с NFC-тегами, которые могут быть запрограммированы NFC-приложениями для автоматизации некоторых задач. Эти приложения могут позволить изменить настройки телефона, создать и отправить текст, запустить другое приложение или выполнить любое другое действие, которое позволяет смартфон.

Также смартфоны с NFC можно использовать для работы с персональными компьютерами. Для этого необходимо иметь специальное NFC устройство для персонального компьютера, позволяющее считывать и записывать информацию на NFC-теги, например, ACR 122 USB NFC Reader.

Рассмотрим, как можно обмениваться данными между персональным компьютером не только с NFC-тегами, но и мобильными устройствами и организовать пиринговую сеть посредством технологии NFC.

В качестве формата обмена данных между NFC устройствами используется NDEF. NDEF (NFC Data Exchange Format) – это стандартизированный формат данных, который можно использовать для обмена информацией между любым NFC совместимым устройством и другим NFC устройством или NFC-тегом.

NDEF формат используется для хранения и обмена информацией, например ссылок, контактов, текста и т.д., используя общий понимаемый формат. NFC-теги, такие как карты Mifare Classic, могут быть настроены как NDEF-теги и данные, записанные на них одним NFC устройством, могут быть прочитаны другим. Также NDEF сообщения могут использоваться для обмена данными между двумя активными NFC устройствами в пиринговом режиме. Придерживаясь NDEF формата обмена данными, NFC устройства, не зная информации друг о друге, способны обмениваться данными в организованной и взаимно понятной форме.

Необходимо исследовать, как происходит взаимодействие с NFC на базе операционной системы Android. С помощью операционной системы Android 4.4 и выше можно использовать режим host-based card emulation (HCE), который позволяет любым Android приложениям эмулировать NFC карту и общаться им напрямую с NFC считывателем (рис. 1).

Рис. 1. Схема работы Host Card Emulation

HCE использует Application Protocol Data Units (APDU). Этот протокол используется для взаимодействия Android приложений и NFC считывателя. APDU – это командные и ответные пакеты для обеспечения связи между смарт-картами. Команда APDU содержит код инструкции и ассоциированные с ним данные. APDU ответ содержит данные и код состояния ответа на запрос предыдущей команды APDU. Архитектура Host-based Card Emulation в Android основана на сервисных компонентах операционной системы, называемые HCE сервисы. Одно из ключевых преимуществ сервисов заключается в том, что они могут быть запущены в фоновом режиме без пользовательского интерфейса. Это является естественным для многих HCE приложений, в которых пользователь не должен запускать само приложение, чтобы использовать функции NFC. Работает всю следующим образом: когда пользователь подносит мобильное устройство к NFC считывателю, системе Android необходимо знать, какой HCE сервис необходим для NFC считывателя. Здесь приходит на помощь спецификация ISO/IEC 7816-4, которая описывает способ выбора приложений на основе AID (Application ID), который содержит не больше 16 байт информации. Также существует особенность работы Android с NFC контроллером, когда экран мобильного

устройства отключен. В это время операционное система выключает NFC контроллер и HCE сервисы не работают. Однако это можно исправить, прописав необходимые свойства в приложение. После этого при поднесении устройства к NFC считывателю, Android предложит пользователю разблокировать устройства. После разблокировки, появится диалоговое окно для завершения транзакции. Эта мера защиты необходима, т.к. пользователь может подносить мобильное устройство к NFC считывателю случайно, не подозревая об этом, т.к. NFC является бесконтактной технологией [2].

Работа с NFC на персональном компьютере отличается от таковой на операционной системе Android. Если Android эмулирует NFC карты, то программному обеспечению на ПК необходимо производить различные запросы к приложению на мобильном устройстве. Эти запросы должны соответствовать APDU, а данные можно передавать и получать с помощью формата NDEF. Также для организации работы с NFC на компьютере необходимы специальный драйвер и библиотека. Самой популярной библиотекой для работы с NFC на языке Java является nfctools, которая упрощает организацию пиринговой сети с мобильным устройством через NFC.

Технология Near Field Communication (NFC) в настоящее время много где используется и продолжает набирать популярность. Организация взаимодействия мобильного устройства на базе операционной системы Android и персонального компьютера посредством NFC открывает некоторые возможности для упрощения работы устройств между собой, хранения и доступа к данным. Однако существуют некоторые ограничения по организации пиринговой сети с мобильным устройством, например, устройство должно находиться в разблокированном состоянии. Данное ограничение обусловлено в непредумышленном контакте посредством NFC и защите пользовательских данных. В ходе работы были проведены исследования, а также разработано ПО по обеспечению пиринговой сети с помощью технологии NFC между персональным компьютером и мобильным устройством.

ЛИТЕРАТУРА

1. Near Field Communication [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Near_field_communication. – Дата доступа: 30.03.2015.
2. Host-based Card Emulation [Электронный ресурс] – Режим доступа: https://developer.android.com/intl/ru/guide/topics/connectivity/nfc/hce.html. – Дата доступа: 28.03.2015.

УДК 004.49

МЕНЕДЖЕР ПАРОЛЕЙ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ NFC

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

Для безопасного использования веб-сайтов, где требуется аутентификация, необходимо использовать сложные и длинные пароли, которые трудно запомнить. Исследуются проблемы хранения паролей существующими способами, а также предлагается несколько иной способ хранения и доступа к паролям.

Сейчас глобальная сеть Интернет используется повсеместно. Огромное количество людей посещают миллионы сайтов каждый день. Некоторые сайты для работы с ними требуют пройти аутентификацию пользователей с помощью учетных записи, обычно имя пользователя и его пароля. Однако существует несколько проблем связанных с этим процессом.

В среднем каждый пользователь имеет десятки учетных записей [4]. Запомнить даже несколько различных сложных паролей практически невозможно. Помимо этого, большинство людей используют очень легкие пароли. Также одна из самых серьезных проблем является многократное использование одного пароля, т.к. происходят утечки паролей каждый год на огромном количестве сайтов. Чтобы предотвратить утечку паролей, пользователь должен использовать сложный уникальный пароль на каждом веб-сайте. Однако запомнить все пароли практически невозможно. Для этих целей используются так называемые менеджеры паролей.

Менеджер паролей – программное обеспечение, которое помогает пользователю хранить и организовывать пароли. Менеджеры паролей обычно хранят пароли в зашифрованном виде и требуют у пользователя создать мастер пароль. Мастер-пароль – единый, довольно сложный пароль, который предоставляет пользователю доступ к базе его паролей. Некоторые менеджеры паролей хранят пароли на устройстве пользователя, другие – на своих серверах. Также большинство менеджеров предоставляют дополнительные возможности, например, автозаполнение форм и создание сложных, случайных паролей [3].

Основные ошибки использования и хранения паролей пользователями:

Простые пароли – короткие, которые используют слова из словарей, не используют различные типы символов (цифры, пунктуацию, специальные символы, верхний и нижний регистры) или попросту говоря, являются легко отгадываемыми.

Пароли можно обнаружить – на стикерах на мониторах, в блокноте компьютера, в документе на компьютера, хранящиеся на устройствах в открытом виде и так далее.

Использование одного пароля – использование одного пароля на множестве веб сайтах, никогда не меняя его.

Использование общего пароля – передача пароля другим лицам, пересылка незашифрованных писем с парольной информацией, использование наемными рабочими одного пароля для всех учетных записей.

Обычно пользователи совершают хотя бы одну из этих ошибок. Это предоставляет легкий путь для злоумышленников получить доступ к индивидуальным учетным записям или даже к ресурсам компании. Поэтому использование менеджеров паролей крайне важно.

Менеджеры паролей также могут быть использованы для защиты против фишинга. Фишинг (англ. phishing, от fishing – рыбная ловля, выуживание) – вид интернет-мошенничества, целью которого является получение доступа к конфиденциальным данным пользователей – логинам и паролям [5]. В отличие от человека, менеджер паролей содержит автоматизированный скрипт входа, который сравнивает Uniform Resource Locator (URL) текущего сайта с URL сайта, хранимого в базе менеджера. Если они не совпадают, то менеджер паролей автоматически не заполняет веб-форму. Это используется в качестве защиты от визуальных подражаний и похожих друг на друга веб-сайтов [2].

Использование менеджера паролей на переносимом устройстве, в данном случае на смартфоне с технологией Near Field Communication (NFC), дает дополнительные преимущества, такие как: пароли хранятся в зашифрованном виде только в одном месте, к ним не имеют доступа посторонние лица, использовать менеджер паролей можно также для аутентификации не только на веб сайтах, но и для доступа к физическим устройствам (при написании соответствующего программного обеспечения (ПО) для них).

NFC – это технология, позволяющая смартфонам и другим устройствам обмениваться данными по беспроводной высокочастотной связи друг с другом при близком контакте, обычно около десяти сантиметров [1]. Технология NFC используется во многих областях.

Рассмотрим некоторые аналоги. Существует несколько популярных менеджеров паролей, например, LastPass, Dashlane, KeePass, 1Password и RoboForm. 4 из 5 из этих менеджеров являются платными и поставляются с закрытым исходным кодом. Также большинство из них используют свои сервера для хранения паролей, что можно считать существенным минусом. На мой взгляд, лучшим из представленных выше менеджеров паролей является KeePass. Преимуществом разработанного менеджера пароля является то, что его можно использовать не только с браузером и вообще с компьютером, но и с другими устройствами, поддерживающими технологию NFC для аутентификации на них.

В ходе реализации работы были выбраны алгоритмы для защиты данных. Для шифрования данных в менеджере паролей используется симметричный алгоритм блочного шифрования AES (Advanced Encryption Standard). Алгоритм обладает очень высокой защищенностью. Единственный работающий способ взлома шифра AES – это атаки по побочным каналам. Такие атаки не связаны с математическими особенностями AES, а используют определенные особенности реализации систем, использующих шифр, с целью раскрыть частично или полностью секретные данные, в том числе ключ. Также алгоритму присуще очень высокая скорость шифрования. Программная реализация на машине с частотой 2 ГГц позволяет шифровать данные со скоростью 700 Мбит/с.

Для обмена ключами для AES между мобильным устройством и расширением для браузера, используется ассиметричный алгоритм шифрования RSA. RSA – один из наиболее успешных асимметричных алгоритмов шифрования на сегодняшний день. Безопасность RSA основана на математической проблеме факторизации целых чисел. Шифруемое сообщение рассматривается как одно большое число. Во время шифрования оно возводится в степень ключа и делится с остатком на произведение первых двух. Повторяя процесс с другим ключом, можно получить исходный текст.

Для хранения мастер-пароля на мобильном приложении используется алгоритм SHA-2 (Secure Hash Algorithm 2). Хэш-алгоритмы SHA-2 называются безопасными, потому что по заданному алгоритму невозможно вычислить следующее: 1) восстановить сообщение по конкретному дайджесту сообщения, или 2) найти два различных сообщения, у которых один и тот же дайджест сообщения (найти коллизию). Любые изменения в сообщении, с очень высокой вероятностью, приводят к различным хэш-значениям. Это свойство полезно при создании и проверке цифровых подписей, при аутентификации сообщений, при создании случайных чисел.

В ходе работы была исследована проблема надежного хранения паролей от веб-ресурсов. Запомнить все пароли невозможно, а использовать одинаковые небезопасно, наилучшее решение этой проблемы – использование менеджеров паролей. Были проанализированы существующие менеджеры паролей, среди которых выявлены недостатки, например, хранение пользовательских данных на серверах компаний. Разработанный менеджер паролей хранит данные на устройстве пользователя, которое поддерживает технологию NFC, а по запросу, предоставляет эти данные. Аналогов среди менеджеров паролей, использующих данный принцип работы, не существует.

В ходе выполнения работы было проделано следующее:

– исследованы проблемы хранения паролей.

– ознакомление с разработкой для операционной системы Android.

– разработан менеджер паролей на базе операционной системы Android.

– разработан протокол обмена учетными данными между менеджером паролей и другими устройствами.

– разработано программное обеспечение для ПК, поддерживающее протокол обмена данных.

– ознакомление с разработкой расширений для браузера Google Chrome и разработаны расширения, для взаимодействия с приложением на базе ОС Android.

– разработано расширение для браузера Google Chrome, с поддержкой автоматического заполнения веб-форм, а также автоматического извлечения из них учетных данных для последующей передачи их в менеджер паролей.

Таким образом, разработанное ПО можно расширять для использования в других расширениях, а также в других приложениях для ПК. Также менеджер паролей можно использовать с другими устройствами, поддерживающие технологию NFC и протокол обмена данных менеджера паролей. Главным недостатком разработанных приложений является обязательная поддержка на мобильном устройстве и ПК технологии NFC.

ЛИТЕРАТУРА

1. Near Field Communication [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Near_field_communication. – Дата доступа: 30.03.2015.
2. Password manager [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Password_manager. – Дата доступа: 28.03.2015.
3. Should You Use a Password Manager? [Электронный ресурс]. – Режим доступа: http://www.tomsguide.com/us/password-manager-pros-cons,news-19018.html. – Дата доступа: 28.03.2015.
4. Three quarters of Britons risking online safety [Электронный ресурс]. – Режим доступа: https://www.cyberstreetwise.com/blog/three-quarters-britons-risking-online-safety
5. Фишинг [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/фишинг. – Дата доступа: 03.04.2015.

УДК 512.643

О ПРЕДСТАВЛЕНИИ (2 × 2)-МАТРИЦЫ С ПОЛОЖИТЕЛЬНЫМ ОПРЕДЕЛИТЕЛЕМ В ВИДЕ ПРОИЗВЕДЕНИЯ ТРЕУГОЛЬНЫХ МАТРИЦ С ПОЛОЖИТЕЛЬНЫМИ ДИАГОНАЛЬНЫМИ ЭЛЕМЕНТАМИ

В.А. ЗАЙЦЕВ, Д.А. ГОЛУБЕВ
(Представлено: канд. физ.-мат. наук, доц. А.А. КОЗЛОВ)

Одним из основных вопросов теории матриц является задача о факторизации матриц, т.е. о разложении матрицы из некоторого класса матриц в произведение матриц, принадлежащих некоторым (возможно, иным) подмножествам матриц. Рассматривается задача представления квадратной (2 × 2)-матрицы с положительным определителем в виде произведения семи треугольных матриц. Отличительной особенностью найденного разложения от иных и, прежде всего, от LU –разложения, является то, что все матрицы-сомножители в нем имеют положительную диагональ.

Имеет место
Лемма *Для матриц*

$$\alpha := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \beta := \alpha^{\mathrm{T}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \gamma := \alpha^2 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \tag{1}$$

выполняются равенства

$$\alpha^{-1}=\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix},\quad \beta^{-1}=\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix},$$

$$\alpha^{-1}\cdot\beta\cdot\alpha^{-1}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}:-\mathrm{J},\quad \alpha^{-1}\cdot\beta\cdot\alpha^{-1}=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}=\mathrm{J}^{-1}. \qquad (2)$$

$$\alpha\cdot\beta^{-1}\cdot\gamma\cdot\beta^{-1}\cdot\alpha=\begin{pmatrix}-1 & 0\\ 0 & -1\end{pmatrix}=-\mathrm{E}.$$

Доказательство леммы приводится непосредственным перемножением матриц из (1) и обратных к матрицам α и β, которые существуют, ввиду очевидной невырожденности α и β:

$$\alpha^{-1}\cdot\beta\cdot\alpha^{-1}=\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & -1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}:=\mathrm{J}$$

$$\alpha\cdot\beta^{-1}\cdot\alpha=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & 1\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}=\mathrm{J}^{-1}$$

$$\alpha\cdot\beta^{-1}\cdot\gamma\cdot\beta^{-1}\cdot\alpha=\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 2\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}0 & 1\\ -1 & -1\end{pmatrix}\cdot\begin{pmatrix}1 & 2\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=$$

$$=\begin{pmatrix}0 & 1\\ -1 & -1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -1 & 1\end{pmatrix}\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}-1 & 1\\ 0 & -1\end{pmatrix}\cdot\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}=\begin{pmatrix}-1 & 0\\ 0 & -1\end{pmatrix}=-\mathrm{E}$$

Лемма доказана.

На основании этой леммы удалось получить следующую теорему:

Теорема Любую вещественную (2 × 2)-матрицу с положительным определителем можно представить в виде произведения семи вещественных треугольных (2 × 2)-матриц с положительными диагональными элементами.

Доказательство. Возьмем произвольную матрицу $H=\left\{h_{ij}\right\}_{ij}^{2}\in \mathrm{M}_2$ с положительным определителем. Тогда возможны два случая: $h_{11}\neq 0$ и $h_{11}=0$.

Пусть $h_{11}\neq 0$. Поскольку $\det H>0$, то главные угловые миноры [1, с. 30] матрицы H ненулевые, и, следовательно, для этой матрицы можно применить теоремы о LU –разложении [1, с. 194]. Тогда $H=L\cdot U$, где $L=\left\{l_{ij}\right\}_{ij=1}^{2}$ $U=\left\{U_{ij}\right\}_{ij=1}^{2}\in M_2$ – соответственно нижне- и верхнетреугольная матрицы, причем на диагонали матрицы U стоят единицы, т.е. $U_{ij=1}$, $i=1,2$. Так как $0<\det H=\det(L\cdot U)=\det L\cdot\det U=i_{11}\cdot l_{22}\cdot u_{11}\cdot u_{22}$, то диагональные элементы матрицы L одного знака. Пусть $l_{ij>0}$, $i=1,2$, тогда L – нижнетреугольная матрица с положительными диагональными элементами. Отсюда, ввиду, что единичная матрица имеет положительную диагональ и является треугольной, имеем искомое представление

$$H=\underbrace{E\cdot\ldots\cdot E}_{5\text{ сомножителей}}\cdot L\cdot U, \qquad (4)$$

Пусть теперь $l_{ii<0}$, $i=1,2$, т.е. на диагонали матрицы L стоят только отрицательные числа, тогда диагональные элементы матрицы $-L$ – положительны. Используя формулу (3) леммы, получим равенства

$$H=L\cdot U=-E\cdot(-L)\cdot U=\alpha\cdot\beta^{-1}\cdot\gamma\cdot\beta^{-1}\cdot\alpha\cdot(-L)\cdot U, \qquad (5)$$

которые, с учетом того, что матрицы $\alpha^{\pm 1}$, $\beta^{\pm 1}$, γ имеют положительные диагональные элементы и являются треугольными, дают требуемое в теореме представление.

Прежде, чем перейти ко второму случаю, введем в рассмотрение матрицу

$$J:\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}.$$

Очевидно, что эта матрица обратима, причем выполняется легко проверяемое соотношение

$$J^{-1}=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}.$$

Рассмотрим теперь случай $h_{11} = 0$, т.е. когда матрица H имеет вид

$$\mathrm{H}=\begin{pmatrix}0 & h_{12}\\ h_{21} & h_{22}\end{pmatrix},$$

и, поскольку среди главных угловых миноров матрицы H есть нулевые ($h_{11} = 0$), когда для этой матрицы нельзя применить теорему о LU–разложении [1, с. 194]. Так как по условию $\det \mathrm{H} > 0$, то для рассматриваемого случая выполняется неравенство $h_{12}h_{22} < 0$, т.е. элементы побочной диагонали матрицы H имеют разные знаки.
Пусть $h_{12} < 0$, тогда у верхнетреугольной матрицы

$$G_1 \coloneqq \begin{pmatrix}h_{21} & h_{22}\\ 0 & -h_{12}\end{pmatrix}$$

на диагонали стоят положительные элементы. Так как выполняются равенства

$$J\cdot G_1=\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\begin{pmatrix}h_{21} & h_{22}\\ 0 & -h_{12}\end{pmatrix}=\begin{pmatrix}0 & h_{12}\\ h_{21} & h_{22}\end{pmatrix}=H,$$

то, отсюда, ввиду второй формулы из (2) и определения J, вытекает представление

$$H=\alpha^{-1}\cdot\beta\cdot\alpha^{-1}\cdot\underbrace{E\cdot\ldots\cdot E}_{\text{3 сомножителя}}\cdot G_1. \qquad (6)$$

Если же выполняется неравенство $h_{21} < 0$, тогда матрица

$$G_2 \coloneqq \begin{pmatrix}-h_{21} & -h_{22}\\ 0 & h_{12}\end{pmatrix}$$

имеет только положительные диагональные элементы. Поскольку справедливы равенства

$$J^{-1}\cdot G_2=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}\begin{pmatrix}-h_{21} & -h_{22}\\ 0 & h_{12}\end{pmatrix}=\begin{pmatrix}0 & h_{12}\\ h_{21} & a_{22}\end{pmatrix}=H,$$

то, отсюда, ввиду второй формулы из (2) и определения J^{1}, вытекает представление

$$H=\alpha^{-1}\cdot\beta\cdot\alpha^{-1}\cdot\underbrace{E\cdot\ldots\cdot E}_{\text{3 сомножителя}}\cdot G_2. \qquad (7)$$

В каждом из найденных представлений (4)-(7) матрицы H матрицы-сомножители, стоящие в правой части, суть треугольные и имеют положительные диагональные элементы.
Теорема доказана.

ЛИТЕРАТУРА

1. Хорн, Р. Матричный анализ / Р. Хорн, Ч. Джонсон. – М. : Мир. – 1989. – 655 с.

УДК 681.3.06

ИСПОЛЬЗОВАНИЕ ПЛИС С АРХИТЕКТУРОЙ FPGA ДЛЯ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

А.В. АНДРОЩУК
(Представлено: В.М. ЧЕРТКОВ)

Рассмотрена возможность повышения производительности цифровой обработки сигналов (ЦОС) при помощи методов параллельных вычислений. Представлены особенности использования ПЛИС. Определены преимущества использования ПЛИС с архитектурой FPGA по сравнению с DSP процессорами.

Термин «цифровая обработка сигналов» (DSP, Digital Signal Processing) охватывает довольно широкую область, границы которой трудно очертить однозначно. Часто отличительной чертой устройств DSP (сигнальных процессоров) считают наличие в них аппаратной поддержки операции «умножение с накоплением» (MAC, Multiply and Accumulate). Само по себе умножение с накоплением сводится к вычислению суммы произведений вида:

$$y = \sum_{i=0}^{n} k_i \cdot x_i. \quad (1)$$

Эта сумма приближенно может считаться равной интегралу

$$\int_t k(t) \cdot x(t) dt, \quad (2)$$

к которому, в свою очередь, сводится большой набор математических методов анализа сигналов [1].

Одно из основных требований к системам цифровой обработки информации – высокая производительность. Достигнуть высокой скорости вычислений можно с помощью методов параллельных расчетов, которые в большинстве случаев реализуется при помощи специализированных интегральных схем (ИС), таких как ПЛИС с архитектурой Field Programmable Gate Array (FPGA).

ИС такого типа представляют собой программируемую логическую матрицу (ПЛМ), между элементами которой проложены электрически коммутируемые соединения. Это позволяет конфигурировать отдельные компоненты и создавать связи между ними путем загрузки в ПЛИС потока данных, включающего требуемые цепи и узлы коммутации. В результате из имеющихся в составе ПЛМ ресурсов создается требуемая цифровая схема, которая при необходимости может быть легко модифицирована. Современные ПЛИС имеют достаточно большой объем ресурсов, достигающий миллионов эквивалентных логических вентилей, составляющих сотни тысяч логических ячеек, что позволяет проектировать цифровые устройства практически любой сложности [2].

Особенности использования ПЛИС. Практически с момента своего появления FPGA позиционировались как устройства, превосходящие сигнальные процессоры по отношению производительность/цена. Однако надо иметь в виду, что по сравнению с относительно дешевыми микроконтроллерами и сигнальными процессорами ПЛИС не оправдывают свое применение в случае повторения широко распространенных процессорных архитектур или однопоточных вычислений. Преимущества ПЛИС в системах ЦОС проявляются только в случае реализации массово-параллельных вычислительных архитектур. В них максимально полно используется высокая суммарная пропускная способность накристальной памяти FPGA, блоков цифровой обработки сигналов и, при организации обмена данными с внешними устройствами, скоростных последовательных приемопередатчиков. Соответственно, наиболее эффективны для реализации в ПЛИС методы и алгоритмы, использующие параллельную обработку нескольких потоков данных.

Сигнальный процессор, созданный по сопоставимой технологии, в среднем имеет более высокую тактовую частоту, однако единственный поток исполнения команд снижает общую производительность. Несмотря на то, что некоторые сигнальные процессоры допускают выполнение двух или четырех MAC операций одновременно, при расчете фильтров высокого порядка общая скорость вычислений существенно снижается. В то же время FPGA с большим числом блоков DSP вполне может обеспечить однотактное исполнение всех операций, используя параллельный расчет (рис.1). Для эффективного использования данного преимущества следует ориентироваться на алгоритмы и методы, подразумевающие распараллеливание операций – фильтры высоких порядков, быстрое преобразование Фурье, вейвлет-анализ, статистическая обработка данных и т.п.

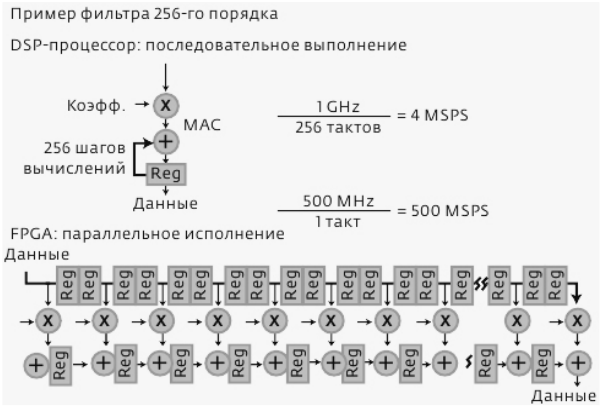

Рис. 1. Выполнение цифровой фильтрации в сигнальном процессоре и FPGA

Еще одно важное преимущество FPGA – способность обеспечивать обработку данных непрерывно и со стабильной скоростью. Дело в том, что понятие «пиковая производительность» имеет разный смысл для сигнальных процессоров и FPGA. В случае с сигнальным процессором тактовая частота условно соответствует количеству операций в секунду непосредственно с фильтром. Однако в программе процессора могут быть предусмотрены и другие действия, например, обработка прерываний. Из-за необходимости выполнять дополнительные операции пиковая скорость может упасть. Таким образом, понятие "пиковая производительность" имеет статистический характер, а реальное значение производительности может меняться не только в зависимости от выбранного алгоритма, но и в процессе работы программы при возникновении соответствующих условий.

В то же время для FPGA термин «максимальная тактовая частота» относится к наиболее выгодным условиям трассировки кристалла – все соединения выполнены с использованием коротких цепей, связанные программируемые ячейки расположены рядом, максимальная длина цепей ускоренного переноса ограничена (разрядность счетчиков, как правило, невелика). Неудачная трассировка снижает допустимую тактовую частоту, однако весьма важно то, что после завершения проектирования она остается постоянной. Некоторые проблемы с дополнительными тактами ожидания может внести использование внешней памяти. Но наличие скоростных синхронных ресурсов и достаточного количества внутренней памяти существенно облегчает построение стандартных узлов ЦОС. При необходимости в проектах на ПЛИС тоже можно реализовать процессоры (например, процессор на логических ячейках типа MicroBlaze), однако через этот процессор совершенно необязательно пропускать весь поток обрабатываемых данных. Более того, рекомендуется реализовывать высокопроизводительную цифровую обработку с использованием независимых от процессора ресурсов DSP. Процессор может выполнять организацию интерфейса, загрузку коэффициентов и прочие операции, которые сложно реализовать аппаратно. При этом единственное процессорное ядро может обеспечивать управление несколькими сотнями DSP-блоков ПЛИС, которые непрерывно выполняют обработку входного потока [1].

Средства разработки. Программирование DSP под определенную задачу, как правило, выполняется с использованием языка высокого уровня, например C, и с использованием библиотек ориентированных на определенную задачу, например библиотека для использования в приложениях беспроводной связи. Это значительно сокращает время проектирования устройств на базе DSP.

До недавнего времени разработка устройств на базе ПЛИС являлась трудной задачей, требующей больших временных затрат. Однако ситуация изменилась с появлением новых методов проектирования устройств на базе ПЛИС. Одним из новых методов являться написание алгоритма работы устройства на языке высокого уровня с последующей трансляцией программы на уровень регистровых передач. Другим вариантом для сокращения времени проектирования является использование встраиваемых процессоров в ПЛИС. При этом алгоритм пишется на языке высокого уровня, и программа выполняется во встроенном процессоре [3].

Использование ПЛИС с архитектурой FPGA в цифровой обработке на сегодняшний день является наиболее эффективным решением для повышения производительности устройств ЦОС. Данное решение позволяет реализовать методы и алгоритмы, использующие параллельную обработку нескольких потоков данных, тем самым повысив общую скорость вычислений. Так же благодаря новым методам проектирования устройств на базе ПЛИС время разработки устройств на их основе сравнима с временем разработки устройств на базе DSP. Это позволяет говорить, что использование ПЛИС для реализации сложных алгоритмов цифровой обработки сигналов выглядит предпочтительнее, чем разработка на базе DSP.

ЛИТЕРАТУРА

1. Тарасов, И. ПЛИС Xilinx и цифровая обработка сигналов особенности, преимущества, перспективы / И. Тарасов // Электроника: наука, технология, бизнес. – 2011. – №3(00109). – С. 116.
2. Тарасов, И.Е. Программируемые логические схемы и их применение в схемотехнических решениях : учеб. пособие / И.Е. Тарасов, Е.Ф. Певцов. – М., 2012. – 184 с.
3. Шидловский, Д.Ю. Сравнение характеристик ПЛИС и ЦСП для определения целесообразности разработки устройств на их основе в области цифровой обработки сигнала / Д.Ю. Шидловский, М.В. Руфицкий // Труды Международного симпозиума «Надежность и качество». – 2007. – С. 2.

УДК 004.9+004.056

ИСПОЛЬЗОВАНИЕ ПРОТОКОЛА АУТЕНТИФИКАЦИИ OAUTH 2 ПРИ РАЗРАБОТКЕ RESTFUL API

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

Рассмотрены основные принципы протокола аутентификации OAuth 2. Обращено внимание на основные цели в области обеспечения безопасности предоставления данных сторонним приложениям, достигаемые использованием этого протокола.

Развитие веб-сайтов в настоящее время получило широчайшее распространение; они используются с целью предоставления информации, программных услуг, в качестве средства программного доступа к распределенным приложениям. Развитие веб-сайтов построено на базе протокола HTTP.

Протокол гипертекстовой передачи данных (англ. Hypertext Transfer Protocol – HTTP) является протоколом прикладного уровня, обеспечивающим легкость и скорость, требуемую для распределенных взаимодействующих информационных систем [1]. HTTP, наряду с традиционным использованием для передачи гипертекстовых документов и другими случаями использования, применяется как транспортный протокол для построенных по принципу REST веб-сервисов. REST (Representational State Transfer – «передача репрезентативного состояния») представляет собой архитектурный стиль для построения распределенных систем [2], который широко распространен для разработки программных интерфейсов (англ. Application Programming Interface – API) для интернет-приложений.

Такие программные интерфейсы должны предоставлять способы для сторонних приложений действовать от имени пользователя посредством процесса аутентификации. Метод аутентификации должен обеспечить наивысший уровень безопасности аутентификационных данных и наименьший возможный требуемый уровень взаимного доверия среди сторон процесса аутентификации (иными словами, метод аутентификации должен быть безопасен даже для случаев, когда предоставляющая программный интерфейс система и ее пользователи не доверяют в полной степени клиенту этого интерфейса).

Фактическим стандартом в области аутентификации для RESTful API является OAuth2 [3] – вторая версия открытого протокола авторизации. Этот протокол используют для предоставления доступа посредством программного интерфейса к защищенным пользовательским данным такие технологические компании, как Google, Facebook, GitHub, Amazon и прочие. OAuth 2 предоставляет стороннему приложению возможность получить ограниченный доступ к веб-сервису от имени конечного пользователя (иными словами, владельца защищенных данных). При этом отсутствует необходимость в предоставлении пользовательских данных аутентификации (обычно логина и пароля) стороннему приложению. Это достигается посредством разделения ролей клиента сервиса (стороннего приложения) и владельца защищенного ресурса: вместо использования данных аутентификации владельца ресурса, клиент получает т.н. токен доступа – строку, выражающую определенные границы доступа, временной интервал доступа и прочие атрибуты; токен доступа затем используется для получения защищенных ресурсов.

Обобщенная схема потока аутентификации согласно протоколу OAuth 2 приведена на рисунке 1.

Таким образом, аутентификации состоит из ряда шагов.

Запрос аутентификации. Клиентское веб-приложение выполняет запрос аутентификации к предоставляющему программный интерфейс веб-сервису. Запрос осуществляется опосредовано посредством перенаправления веб-браузера пользователя на специальный URL веб-сервиса. Такой URL является составной частью программного интерфейса веб-сервиса.

Возвращение кода аутентификации. Веб-сервис выполняет авторизацию пользователя стандартным для себя способом (обычно запросом у пользователя через веб-браузер логина и пароля, так же, как и в случае прямого обращение пользователя к веб-сервису; в качестве дополнительной защиты может использоваться двухфакторная аутентификация), явным образом запрашивает у пользователя разрешение на предоставление доступа к его данным указанному клиентскому приложению. В случае согласия пользователя перенаправляет его веб-браузер на специальный URL клиентского веб-приложения, передав параметром этого URL так называемый код авторизации. Такой URL передается клиентским приложением в виде параметра запроса аутентификации. В случае если пользователь не смог выполнить авторизацию, либо отказал в предоставлении клиентскому приложению разрешения на доступ к данным, веб-сервис перенаправляет браузер на тот же специальный URL клиентского приложения, но в этом случае в качестве параметра передает подробную информацию о причинах отказа аутентификации.

Запрос токена доступа. Клиентское веб-приложение выполняет прямой межсерверный HTTP-запрос к специальному URL веб-сервиса с запросом токена доступа на основании ранее полученного кода аутентификации. Этот специальный URL также является частью программного интерфейса веб-сервиса. Ранее полученный код аутентификации передается в виде параметра запроса (либо в теле запроса).

Возвращение токена доступа. Веб-сервис отвечает на выполненный HTTP-запрос, передав в теле ответа токен доступа (авторизационный токен) в случае, если код аутентификации корректен и не просрочен. В противном случае веб-сервис возвратит ответ с соответствующим кодом и сообщением об ошибке.

Обращение к защищенному ресурсу. Клиентское веб-приложение выполняет обращение (посредством прямого межсерверного HTTP-запроса) к ресурсу веб-сервиса, защищенному от прямого доступа. Обращение выполняется к специальному URL, соответствующему требуемому ресурсу в соответствии с программным интерфейсом веб-сервиса. При запросе передается ранее полученный токен доступа, подтверждающий, что этому приложению можно возвратить запрашиваемый защищенный ресурс.

Предоставление защищенных данных. Веб-сервис, проверив переданный токен доступа, возвращает в виде ответа на HTTP-запрос затребованный защищенный ресурс в случае, если переданный токен доступа корректен и не просрочен. В противном случае веб-сервис возвратит ответ с соответствующим кодом и сообщением об ошибке.

Рис. 1. Схема потока аутентификации OAuth 2

Таким образом, протокол аутентификации OAuth 2 предоставляет относительно высокий уровень безопасности. Он не требует от пользователя веб-сервиса доверять свои данные авторизации стороннему приложению, не требует хранения пароля в обратимой форме на сервере веб-сервиса. Этот протокол предоставляет возможность осуществлять тонкое разделение разрешений доступа посредством границ доступа (пользователь может предоставлять сторонним приложениям не полный доступ к своим данным в веб-приложении, а доступ лишь к требуемой части данных). Протокол обеспечивает явное согласие пользователя на предоставления доступа к своим данным сторонним приложениям. Также OAuth 2 не требует передачи в открытом виде пароля пользователя в ходе аутентификации своих запросов, т.к. для этих целей используется токен доступа, привязанный к конкретному стороннему приложению.

ЛИТЕРАТУРА

1. Berners-Lee, T. RFC 1945 – Hypertext Transfer Protocol – HTTP/1.0 [Electronic Resource] / Tim Berners-Lee, Roy T. Fielding, Henrik Frystyk Nielsen. – Mode of access: http://tools.ietf.org/pdf/rfc1945.pdf. – Date of access: 30.09.2015.
2. Fielding, R. Architectural Styles and the Design of Network-based Software Architectures / Roy Thomas Fielding. – University of California – Irvine, 2000. – 162 p.
3. Hardt, D. RFC 6749 – The OAuth 2.0 Authorization Framework [Electronic Resource] / Dick Hardt. – Mode of access: http://tools.ietf.org/pdf/rfc6749.pdf. – Date of access: 30.09.2015.

УДК 004.9

ИСПОЛЬЗОВАНИЕ RESTFUL API ПРИ РАЗРАБОТКЕ ВЕБ-СЕРВИСОВ

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

Рассмотрены основные принципы архитектурного стиля REST. Обращено внимание на основные преимущества, достигаемые при проектировании распределенных систем с учетом формальных ограничений REST.

Развитие веб-сайтов в настоящее время получило широчайшее распространение; они используются с целью предоставления информации, программных услуг, в качестве средства программного доступа к распределенным приложениям. Развитие веб-сайтов построено на базе протокола HTTP.

Протокол гипертекстовой передачи данных (англ. Hypertext Transfer Protocol – HTTP) является протоколом прикладного уровня, обеспечивающим легкость и скорость, требуемую для распределенных взаимодействующих информационных систем [1]. При разработке веб-сайтов посредством этого протокола к веб-браузеру пользователя поставляются документы различных форматов, формирующие пользовательский интерфейс сайта: это обычно документы разметки в формате HTTP, документы описания стилей в формате CSS, документы исполняемых веб-браузером динамических сценариев в формате JS, а также изображения и шрифты.

Помимо предоставления пользовательского интерфейса от веб-сайтов зачастую требуется предоставление программного интерфейса (API) для сторонних приложений. Такой программный интерфейс преимущественно предназначен для приложений, разработанных для мобильных устройств, но может использоваться и сторонними веб-сайтами. Функциональные возможности программного интерфейса в целом должны повторять таковые пользовательского интерфейса. Распространённым и эффективным стилем реализации программного интерфейса, предоставляемого веб-ресурсами, является RESTful API [2] – стиль построения протокола поверх стандартного HTTP с передачей данных в общеизвестном структурированном формате, обычно JSON или XML.

REST определяет стиль построения архитектур программных продуктов для всемирной сети. Он предоставляет набор ограничений, которым должна соответствовать структура компонентов распределенной системы для того, чтобы достигать наиболее производительной и поддерживаемой архитектурой. Системы, соответствующие принципам REST, называются RESTful-системами. Они обычно взаимодействуют поверх протокола HTTP с помощью стандартных команд протокола (GET, POST, PUT, DELETE и пр.). Соответствующие программные интерфейсы обычно оперируют коллекциями ресурсов, которые идентифицируются посредством URL.

Стиль REST определяет ряд формальных ограничений.

Клиент-серверная архитектура. Подразумевается четкое разделение клиента и сервера, взаимодействующих посредством унифицированного интерфейса. Таким способом достигается разделение обязанностей: клиент не ответственен за хранение данных и управление ими, а сервер не ответственен за пользовательский интерфейс и его состояние. Таким образом, клиент и сервер могут в широкой степени разрабатываться раздельно, они также могут быть легко заменены на аналогичные при условии сохранения интерфейса взаимодействия.

Отсутствие состояния. Взаимодействие клиента и сервера ограничивается отсутствием сохранения на сервере какого-либо клиентского контекста между запросами. Каждый запрос от клиента к серверу должен содержать всю необходимую информацию для выполнения этого запроса. Состояние сессии при этом хранится на стороне клиента.

Возможность кэширования. Клиенты должны иметь возможность кэшировать ответы. По этой причине ответы должны быть явным или неявным образом определены как разрешённые для кэширования, либо не разрешённые, для предотвращения использования клиентом устаревших или некорректных данных. Корректно реализованное кэширование способно частично или полностью исключить часть клиент-серверного взаимодействия, тем самым улучшая масштабируемость и производительность.

Многослойность системы. Клиент не должен иметь возможность определить, взаимодействует ли он напрямую с конечным сервером, либо взаимодействие происходит через промежуточные серверы. Промежуточные серверы могут улучшать масштабируемость системы, выполняя балансировку нагрузки либо выполняя кэширование разделяемых данных. Они также могут обеспечивать выполнение дополнительных политик безопасности.

Динамический код по требованию (необязательное ограничение). Сервер должен иметь возможность временно расширить функциональность клиента путем передачи в ходе взаимодействия исполняемого кода. Примером такого кода служат сценарии JavaScript либо скомпилированные Java-апплеты. Это ограничение является единственным необязательным в REST-архитектуре.

Унифицированный интерфейс взаимодействия. Это ограничение является фундаментальным для построения любой REST-системы. Унифицированный интерфейс упрощает и разделяет архитектуру, что позволяет развивать каждую ее часть независимо. Существует четыре ограничения при построении унифицированного интерфейса.

а) *Идентификация ресурса*. Отдельные ресурсы должны быть идентифицированы в запросах, например посредством URI в случае основанных на всемирной сети системах. Сами по себе ресурсы должны быть отделены от представления, возвращаемого клиенту, например, данные могут пересылаться

в виде HTML, XML, JSON, причем ни одно из этих представлений может не соответствовать внутреннему представлению данных на сервере.

б) *Управление ресурсами через их представление*. Наличия у клиента представления ресурса, в том числе метаданных, должно быть достаточно для возможности клиента изменить либо удалить ресурс.

в) *Самоописываемые сообщения*. Каждое сообщения должно содержать достаточно сведений для описания способа обработки этого сообщения.

В случае, когда стиль REST применяется к проектированию программных интерфейсов веб-ресурсов, указанные ограничения кратко описываются следующими аспектами:

- наличием базового URI, описывающим глобальную точку доступа к программному интерфейсу;
- наличием определённого типа данных в качестве представления передаваемой в сообщениях информации; наиболее распространёнными являются форматы JSON и XML;
- использование стандартных методов протокола HTTP;
- использование гиперссылок для указания на состояния;
- использование гиперссылок для взаимной связи ресурсов.

Наиболее распространенными структурированными форматами передачи данных при построении RESTful-систем являются JSON [3] и XML [4].

JSON – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 [5]. JSON – текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными. В качестве значений в JSON используются структуры:

- Объект – это неупорядоченное множество пар ключ-значение, заключённое в фигурные скобки. Ключ описывается строкой, между ним и значением стоит символ двоеточия. Пары ключ-значение отделяются друг от друга запятыми.
- Массив – это упорядоченное множество значений. Массив заключается в квадратные скобки. Значения разделяются запятыми.
- Значение может быть строкой в двойных кавычках, числом, объектом, массивом, одним из литералов: true, false или null. Таким образом структуры могут быть вложены друг в друга.
- Строка – это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки.

XML – расширяемый язык разметки. Разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программам и одновременно удобный для чтения и создания документов человеком, с подчеркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

Таким образом, стиль REST позволяет задать широкий спектр требований к архитектуре распределенных систем, позволяющих в дальнейшем улучшить масштабируемость и возможность поддержки и развития этих систем.

ЛИТЕРАТУРА

1. RFC 1945 – Hypertext Transfer Protocol – HTTP/1.0 [Electronic Resource] / Tim Berners-Lee, Roy T. Fielding, Henrik Frystyk Nielsen. – Mode of access: http://tools.ietf.org/pdf/rfc1945.pdf. – Date of access: 30.09.2015.
2. Fielding, Roy. Architectural Styles and the Design of Network-based Software Architectures / Roy Thomas Fielding; University of California. – Irvine, 2000. – 162 p.
3. RFC 7159 – The JavaScript Object Notation (JSON) Data Interchange Format [Electronic Resource] / Tim Bray. Mode of access: https://tools.ietf.org/pdf/rfc7159.pdf. Date of access: 30.09.2015.
4. Extensible Markup Language (XML) 1.0 [Electronic Resource] / World Wide Web Consortium. – Mode of access: http://www.w3.org/TR/REC-xml/. – Date of access: 30.09.2015.
5. ECMA-262. ECMAScript 2015 Language Specification [Electronic Resource] / Ecma International. – Mode of access: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf. – Date of access: 30.09.2015.