УДК 004.62

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС ДЛЯ МОНИТОРИНГА ПОСЕЩАЕМОСТИ СТУДЕНТОВ: СЕРВЕРНАЯ ЧАСТЬ

А. В. ТОНКИХ
(Представлено: канд. техн. наук, доц. В. М. ЧЕРТКОВ, И. С. РУСЕЦКИЙ)

Работа посвящена разработке системы автоматизации мониторинга посещаемости студентов с использованием программно-аппаратного комплекса на базе микроконтроллерной платы ESP32 и использованием HTTP-запросов. Целью работы является повышение эффективности учета посещаемости и упрощение взаимодействия между студентами и преподавателями.

Введение. Современные технологии автоматизации и цифровизации активно внедряются в различные сферы, включая образование. Одной из актуальных задач является мониторинг посещаемости занятий, что позволяет повысить эффективность учебного процесса и обеспечить прозрачность учета студентов. Традиционные методы, такие как бумажные журналы или устные переклички, не всегда удобны и требуют больших временных затрат. В этой связи возникает необходимость в разработке систем, которые автоматизируют процесс регистрации и контроля посещаемости.

Теоретическая часть. HTTP (Hypertext Transfer Protocol) – это основной протокол прикладного уровня, обеспечивающий взаимодействие клиентских и серверных компонентов в распределенных системах.

REST API (Representational State Transfer Application Programming Interface) — это стиль проектирования программных интерфейсов, основанный на использовании стандартных HTTP-запросов для взаимодействия между клиентом и сервером.

Основными HTTP-методами, используемыми в REST API, являются:

– GET, запрос на получение данных с сервера (например, запрос списка студентов или журналов посещаемости);
– POST, отправка данных на сервер (например, регистрация нового посещения);
– PUT, обновление данных на сервере;
– DELETE, удаление данных.

PHP (Hypertext Preprocessor) является мощным и гибким инструментом для реализации серверной логики в веб-приложениях и взаимодействия с базами данных.

Основными функциями PHP-скриптов в системе являются:

– обработка POST-запросов;
– обработка GET-запросов;
– интеграция с базой данных;
– безопасность и валидация данных.

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

В классификацию по модели данных обычно включают:

– иерархические;
– объектные или объектно-ориентированные;
– объектно-реляционные;
– реляционные;
– сетевые;
– функциональные.

SQL (Structured Query Language) – язык структурированных запросов, декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Декларативный язык означает, что пользователи формулируют, что они хотят получить (например, извлечение данных), а не как именно это следует реализовать.

Операторы SQL делятся на:

– операторы определения данных (Data Definition Language, DDL): CREATE, ALTER, DROP;
– операторы манипуляции данными (Data Manipulation Language, DML): SELECT, INSERT, UPDATE, DELETE;
– операторы определения доступа к данным (Data Control Language, DCL): GRANT, REVOKE, DENY;
– операторы управления транзакциями (Transaction Control Language, TCL): COMMIT, ROLLBACK, SAVEPOINT.

Flutter – это фреймворк с открытым исходным кодом для создания мобильных, веб- и настольных приложений с использованием одного кода на языке программирования Dart. Он был разработан Google и позволяет разработчикам создавать высококачественные приложения для iOS, Android, Windows, macOS и Linux [1].

Firebase Cloud Storage – это масштабируемое облачное решение для хранения файлов, предоставляемое Google разработчикам приложений [2].

Практическая часть. В данной практической части рассматривается архитектура и реализация серверной части программно-аппаратного комплекса, предназначенного для мониторинга посещаемости занятий. Основной задачей серверной компоненты является организация надежного и эффективного взаимодействия между клиентским приложением, физическим устройством на базе микроконтроллерной платы ESP32 DEVKIT V1, Firebase Cloud Storage и реляционной базой данных, что обеспечивает целостность и доступность данных.

В данном разделе будет детально рассмотрена последовательность выполнения операций и взаимодействие ключевых компонентов системы. При активации программно-аппаратного комплекса осуществляется инициализация всех аппаратных модулей, составляющих систему. Наиболее критическим этапом является проверка наличия интернет-соединения, так как результат данной проверки определяет последующие действия системы и сценарии её функционирования. Интернет-соединение необходимо для взаимодействия с сервером, передачи данных и синхронизации информации, обеспечивая выполнение основной задачи комплекса — мониторинга посещаемости.

При наличии интернет-соединения программно-аппаратный комплекс переходит к активному режиму взаимодействия с сервером и базой данных. На этом этапе происходит следующее:

сбор и отправка данных на сервер: устройство, используя RFID-модуль, считывает уникальные идентификаторы меток. Считанные данные передаются на микроконтроллерную плату ESP32, которая формирует HTTP-запросы (POST-запросы) для отправки на сервер. Каждый запрос содержит ключевую информацию, такую как UID карты, который формируется в виде строки. UID включается в строку запроса и передается через HTTP POST-запрос на сервер. URL указывается отдельно в какой-либо переменной, в данной работе использовалась переменная с именем serverName (sn). Затем устройство отправляет UID через POST-запрос и получает ответ от сервера. Пример формирования URL и отправки POST-запроса представлен на рисунке 1.

```
String url = sn + "?uid=" + uid;

HTTPClient http;
http.begin(url);
int httpResponseCode = http.POST(uid);
```

Рисунок 1. – Формирование URL и отправка POST-запроса

Пример обработки ответа сервера представлен на рисунке 2.

```
if (httpResponseCode > 0) {
  String response = http.getString();
  Serial.print("HTTP Response code: ");
  Serial.println(httpResponseCode);
} else {



  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
```

Рисунок 2. – Формирование URL и отправка POST-запроса

Таким образом, описанная последовательность операций демонстрирует принцип функционирования микроконтроллерной платы при наличии активного интернет-соединения.

Однако для полного функционала программно-аппаратного комплекса необходимо предусмотреть возможность удалённого управления устройством. Далее мы рассмотрим разработку программы внешнего управления устройством, позволяющей пользователю не только контролировать его работу, но и вносить изменения в его параметры и настройки через интерфейс удалённого доступа.

При запуске приложения отображается стартовый экран с приветствием и возможностью входа или регистрации. В случае выбора регистрации пользователю предлагается выбрать роль (студент или преподаватель), после чего открывается форма ввода данных для регистрации. Здесь создается Map на языке программирования Dart [3], содержащий все необходимые данные, которые будут отправлены на сервер, а затем он преобразуется в JSON-формат. Создание данных для отправки запроса представлено на рисунке 3.

```
final data = {
  'firstName': _firstNameController.text,
  'lastName': _lastNameController.text,
  'middleName': _middleNameController.text,
  'faculty': _selectedFaculty,
  'group': _groupController.text,
  'subg': _selectedsubg,
  'cardUid': _cardUidController.text,
  'email': _emailController.text,
  'password': _confirmPasswordController.text,
  'userType': _selectedUserType ?? '',
};
final jsonData = json.encode(data);
print('Отправляемые данные: $jsonData');
```

Рисунок 3. – Создание данных для отправки запроса

По нажатию на кнопку отправляется HTTP-запрос на сервер для сохранения введённых данных в базе данных. Пример отправки запроса на сервер представлен на рисунке 4.

```
try {
  final response = await http.post(
    Uri.parse('http://vh140.by2080.ihb.by/login.php'),
    headers: <String, String>{'Content-Type': 'application/json'},
    body: jsonData,
  );
  print('Ответ от сервера: ${response.body}');
```

Рисунок 4. – Создание данных для отправки запроса

Этот механизм обеспечивает надежное взаимодействие между клиентом и сервером, позволяя пользователям легко управлять своими данными. Последующие запросы к серверу будут однотипными и могут включать получение информации, например, пользователи смогут просматривать данные о посещаемости, успеваемости, а также просматривать другую информацию, связанную с их аккаунтом. Эти запросы будут структурированы и стандартизированы, что облегчает обработку на сервере.

В приложении предусмотрено четкое разделение функционала в зависимости от роли пользователя — студента или преподавателя. Это разделение обеспечивает удобство и безопасность работы с данными, а также оптимизирует взаимодействие пользователей с интерфейсом.

Студенты имеют доступ к информации о своей успеваемости, представленной в виде графика, основанного на ежедневном посещении учебного заведения. Это позволяет им отслеживать свои достижения и посещаемость. Также в приложении присутствуют базовые наборы функций, такие как редактирование профиля и настроек внутри приложения. Студенты могут отправлять запросы на получение дополнительных данных, таких как расписание занятий и информации о дополнительных мероприятиях для каждой группы на любом факультете.

Преподаватели могут управлять посещаемостью с помощью приложения и устройства. После того как студент использует устройство метки для регистрации посещаемости, соответствующие данные записываются в базу данных. При этом метка, связанная с конкретным пользователем, записывается

вместе с персональными данными. Это связывание осуществляется на этапе регистрации, когда студент указывает свою метку.

Программно-аппаратный комплекс служит для фиксации посещаемости студентов в реальном времени. Когда студент фиксирует свою метку, информация о его посещаемости автоматически записывается в базу данных. При этом каждая метка связана с конкретным пользователем, что позволяет создать уникальную связь между данными о посещаемости и учетной записью студента. При регистрации в приложении студент вводит свои персональные данные и указывает свою метку. Эта информация сохраняется в базе данных, а затем используется для сопоставления меток с пользователями. Когда пользователь осуществляет вход в приложение, данные о посещаемости, полученные с устройства метки, становятся доступными ему в виде отчетов о посещаемости.

Заключение. Следует отметить, что использование данного метода контроля позволяет проводить учет студентов без необходимости вручную прокручивать списки или учитывать подписи. Отметка о посещении фиксируется автоматически при проходе через определенные точки, оснащенные RFID-считывателями. Полученные данные записываются на SD-карту, что позволяет хранить информацию о посещаемости на долгосрочной основе и проводить анализ студенческой активности.

ЛИТЕРАТУРА

1. Документация Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev. – Дата доступа: 29.04.2024.
2. Документация Firebase [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs?hl=ru. – Дата доступа: 15.05.2024.
3. Документация Dart SDK [Электронный ресурс]. – Режим доступа: https://dart.dev/get-dart. – Дата доступа: 20.05.2024.