**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

«Владимирский государственный университет имени Александра Григорьевича

и Николая Григорьевича Столетовых» (ВлГУ)

Кафедра ВТиСУ

КУРСОВАЯ РАБОТА

по дисциплине «Разработка мобильных приложений»

на тему «Разработка мобильного приложения расписания занятий»

Студент: Хлопцев Н.Д.

Институт: ИИТЭ

Направление: 09.03.03

Группа: ПИ-121

Руководитель: Шутов А.В.

Владимир 2024 г.

**Оглавление**

[**Введение** 2](#_Toc185509452)

[**1. Теоретическая часть** 5](#_Toc185509453)

[**1.1. Анализ существующих решений** 5](#_Toc185509454)

[**1.2. Выбор технологий и инструментов разработки** 6](#_Toc185509455)

[**1.3. Анализ требований пользователей** 8](#_Toc185509456)

[**1.4. Архитектура приложения** 11](#_Toc185509457)

[**1.5. Структура базы данных** 12](#_Toc185509458)

[**2. Практическая часть** 15](#_Toc185509459)

[**2.1. Создание клиентской части (Android-приложение)** 15](#_Toc185509460)

[**2.2. Реализация серверной части (PHP-скрипты и API)** 16](#_Toc185509461)

[**2.3. Интеграция клиент-серверного взаимодействия** 22](#_Toc185509462)

[**2.4. Обзор пользовательского интерфейса** 24](#_Toc185509463)

[**2.5. Тестирование приложения** 28](#_Toc185509464)

[**2.5.1 Планирование тестирования** 28](#_Toc185509465)

[**2.5.2. Тестирование интерфейса** 30](#_Toc185509466)

[**2.5.3. Тестирование функционала** 31](#_Toc185509467)

[**2.5.4. Обнаружение и исправление ошибок** 31](#_Toc185509468)

[**Заключение** 33](#_Toc185509469)

[**Список литературы** 34](#_Toc185509470)

# **Введение**

В современном обществе мобильные устройства стали неотъемлемой частью повседневной жизни, оказывая значительное влияние на различные сферы деятельности. Мобильные приложения играют важную роль в упрощении и автоматизации множества процессов, связанных с работой, обучением, отдыхом и другими аспектами жизни. Одним из востребованных направлений разработки являются мобильные приложения, предназначенные для упрощения организации учебного процесса, в частности, для управления расписанием занятий.

Мобильные приложения для управления расписанием занятий становятся всё более популярными, поскольку они позволяют студентам и преподавателям эффективно планировать свое время и оперативно отслеживать изменения в расписаниях. В условиях динамично меняющихся учебных графиков такие приложения играют важную роль в оптимизации учебного процесса, обеспечивая своевременное информирование и минимизацию ошибок. Пользователи могут быстро и удобно получать актуальную информацию о предстоящих занятиях, экзаменах, а также других учебных мероприятиях, что особенно важно в условиях быстрого ритма образовательного процесса.

С развитием технологий и увеличением числа мобильных приложений растёт потребность в более эффективных и функциональных решениях для управления расписанием. Важно создать приложение, которое будет не только удобным в использовании, но и достаточно гибким для различных типов учебных заведений. В процессе разработки мобильного приложения для расписания занятий необходимо учитывать такие аспекты, как интуитивно понятный интерфейс, синхронизация с другими образовательными системами и возможность персонализации для разных пользователей.

Целью данной курсовой работы является создание мобильного приложения для платформы Android, которое обеспечит удобное управление расписанием занятий как для студентов, так и для преподавателей. Приложение должно позволять пользователям просматривать, редактировать и синхронизировать расписание, а также получать уведомления о предстоящих занятиях. Для реализации этой цели предстоит решить следующие задачи:

1. Проанализировать существующие решения на рынке мобильных приложений для управления расписанием занятий, выявить их сильные и слабые стороны, а также возможности для улучшений.
2. Определить оптимальные технологии и инструменты разработки для создания мобильного приложения, которые обеспечат высокую производительность и удобство работы.
3. Разработать архитектуру приложения, включая структуру базы данных, с учётом требуемого функционала.
4. Создать клиентскую и серверную части приложения, реализовав функционал для работы с расписанием, системой авторизации и синхронизации данных между клиентом и сервером.
5. Провести тестирование приложения, выявить возможные ошибки и гарантировать стабильную работу на различных устройствах.

Объектом исследования является процесс разработки мобильных приложений для управления расписанием занятий, включая все этапы создания программного обеспечения от концептуального проектирования до тестирования. Предметом исследования является разработка функционала для мобильного приложения на платформе Android с использованием современных технологий и методов программирования, включая Java для клиентской части и MySQL для серверной.

Работа охватывает весь цикл разработки: от проектирования и реализации до тестирования, что позволяет не только развить практические навыки разработки программного обеспечения, но и предложить эффективное решение для автоматизации образовательных процессов.

# **1. Теоретическая часть**

## **1.1. Анализ существующих решений**

На рынке мобильных приложений для управления расписанием занятий существует множество решений, таких как **Timetable**, **MyStudyLife**, **Class Timetable** и другие. Эти приложения предлагают пользователям различные функции, которые помогают эффективно управлять учебным временем, просматривать расписания и получать уведомления. Однако каждое из этих приложений имеет свои особенности и ограничения, которые могут повлиять на выбор пользователя.

* **Timetable**: приложение, позволяющее пользователям создавать расписания занятий с возможностью добавления различных типов событий, включая экзамены и дополнительные мероприятия. Одним из его преимуществ является синхронизация с Google Calendar, что делает управление временем ещё удобнее. Однако среди минусов — ограниченные возможности настройки интерфейса под индивидуальные предпочтения, что может не устраивать пользователей с особыми требованиями.
* **MyStudyLife**: это приложение ориентировано на студентов и предоставляет функции не только для создания расписаний, но и для планирования домашних заданий и экзаменов. Оно поддерживает синхронизацию между различными устройствами и доступно на нескольких платформах. Однако MyStudyLife имеет ограничения по количеству настроек уведомлений и частоте обновлений, что может вызывать неудобства у пользователей с плотным графиком.
* **Class Timetable**: простое в использовании приложение, предоставляющее возможность добавления уведомлений о предстоящих занятиях и экзаменах. Оно также позволяет синхронизировать расписание между устройствами. Однако функционал ограничивается только работой с расписанием, а отсутствие дополнительных инструментов для управления заданиями или интеграции с другими образовательными платформами может стать его слабым местом.

Из проведённого анализа можно выделить несколько ключевых требований, которые необходимо учитывать при разработке успешного мобильного приложения для управления расписанием:

1. **Простой и удобный интерфейс**: приложение должно быть интуитивно понятным, с чётким и доступным дизайном, который позволит пользователю легко добавлять и изменять занятия, а также просматривать расписание.
2. **Совместимость и надёжность**: приложение должно быть доступным для использования на разных устройствах, поддерживать различные версии операционной системы и обеспечивать стабильную работу без потери данных.

Таким образом, приложения для управления расписанием должны сочетать в себе простоту использования, богатый функционал и гибкость настройки. Эти требования помогут создать инструмент, востребованный как среди студентов, так и среди преподавателей, обеспечивающий удобство и функциональность при планировании учебного времени.

## **1.2. Выбор технологий и инструментов разработки**

Для реализации данного проекта было принято решение использовать проверенные и эффективные технологии, которые обеспечивают стабильную работу мобильного приложения и соответствуют современным стандартам разработки. Выбор этих технологий обусловлен их популярностью, доступностью и широким функционалом, что позволяет создать качественное приложение для управления расписанием занятий.

1. **Android Studio**: это интегрированная среда разработки (IDE) для создания приложений на платформе Android. Android Studio предоставляет все необходимые инструменты для разработки, включая мощные средства визуального проектирования интерфейса, встроенные инструменты анализа кода, а также поддержку эмуляторов устройств. Это значительно упрощает процесс создания приложений и позволяет разработчикам сосредоточиться на функциональности и дизайне.
2. **Java**: один из самых популярных языков программирования для разработки мобильных приложений на платформе Android. Java обеспечивает высокую стабильность и совместимость с Android SDK, что делает её идеальным выбором для создания мобильных приложений. Её широкая экосистема библиотек и фреймворков позволяет ускорить разработку, улучшить производительность приложения и упростить решение сложных задач, таких как обработка данных или управление ресурсами устройства.
3. **Volley**: библиотека для выполнения HTTP-запросов и взаимодействия с сервером. Volley значительно упрощает обработку сетевых запросов, обеспечивает эффективную работу с REST API и предоставляет встроенные механизмы кэширования. Эта библиотека позволяет быстро отправлять и получать данные с сервера, минимизируя задержки, что критично для приложений с динамически обновляемыми данными, таких как расписания занятий.
4. **PHP**: популярный серверный язык программирования, используемый для создания API и обработки запросов. PHP отличается гибкостью, простотой в использовании и возможностью интеграции с различными базами данных. В данном проекте PHP применён для реализации серверной части приложения, включая функционал взаимодействия с базой данных и обработку данных, поступающих от клиента. Он позволяет эффективно обрабатывать запросы и обеспечивать доступность данных в реальном времени.
5. **MySQL**: реляционная база данных, подходящая для хранения информации о пользователях и расписаниях. MySQL предоставляет высокую производительность, масштабируемость и надёжность при работе с большими объёмами данных. Использование индексов и связей между таблицами помогает оптимизировать запросы и минимизировать вероятность ошибок, что особенно важно для приложений с большим количеством пользователей и постоянно обновляемой информацией.
6. **Postman** (дополнительно для тестирования): инструмент для проверки и отладки API-запросов. С его помощью можно проводить тестирование работы серверной части приложения, обеспечивая корректную обработку данных и минимизацию ошибок в сетевых взаимодействиях.

Выбор этих технологий объясняется их совместимостью и возможностью интеграции между собой. В совокупности они обеспечивают надёжную и масштабируемую архитектуру, позволяющую создать функциональное, удобное и устойчивое приложение для управления расписанием занятий. Это решение отвечает всем современным требованиям, предъявляемым к образовательным мобильным приложениям, и способствует повышению эффективности учебного процесса за счёт автоматизации управления данными.

## **1.3. Анализ требований пользователей**

Для успешной разработки мобильного приложения для управления расписанием занятий необходимо учитывать потребности и ожидания конечных пользователей — студентов, преподавателей и административных работников учебных заведений. Требования к приложению зависят от специфики образовательных учреждений и задач, которые оно должно решать. В этом разделе выделяются ключевые требования пользователей к функциональности и интерфейсу приложения.

1. **Студенты**

Основная цель студентов — это удобное и быстрое управление своим расписанием, а также получение актуальной информации о предстоящих занятиях. Ключевые требования:

* **Просмотр расписания**: возможность быстрого и простого доступа к личному расписанию, с отображением информации о днях недели, времени начала и окончания занятий, аудитории.
* **Уведомления и напоминания**: приложение должно напоминать о предстоящих занятиях за определённое время до начала (например, за 15 минут), а также уведомлять о любых изменениях в расписании.
* **Пользовательская настройка**: возможность настроить отображение расписания по своим предпочтениям, например, выбор цвета для отдельных предметов или преподавателей.
* **Интеграция с другими сервисами**: возможность синхронизации расписания с календарём смартфона для более удобного отображения.
* **Доступ оффлайн**: предоставление возможности просматривать ранее загруженное расписание без подключения к интернету.

1. **Преподаватели**

Для преподавателей приложение должно обеспечивать удобство управления своими занятиями и взаимодействия со студентами. Важные требования:

* **Просмотр расписания**: преподаватель должен иметь доступ к расписанию всех своих занятий, а также информации о группе студентов, месте проведения лекций и семинаров.
* **Редактирование расписания**: возможность вносить изменения в расписание, включая перенос или отмену занятий.
* **Отчёты и статистика**: наличие функций для просмотра статистики посещаемости или изменений в расписании на протяжении учебного периода.
* **Интерфейс для связи со студентами**: возможность добавления заметок или уведомлений для студентов.
* **Поддержка мультиязычности**: для преподавателей, работающих с иностранными студентами, приложение должно поддерживать несколько языков.

1. **Административные работники учебного заведения**

Администрация образовательного учреждения будет использовать приложение для более эффективного планирования и контроля учебного процесса. Основные требования:

* **Управление расписанием**: возможность создавать и редактировать расписания для различных групп студентов и преподавателей.
* **Интерфейс для массовых изменений**: возможность массового редактирования расписаний для быстрого внесения изменений.
* **Интерфейс для мониторинга**: контроль за актуальностью расписания и возможность редактировать информацию о занятиях в случае необходимости.
* **Безопасность данных**: приложение должно обеспечивать высокий уровень защиты данных, чтобы информация о расписаниях, пользователях и учебных материалах была защищена от несанкционированного доступа.
* **Поддержка отчётов**: генерация отчётов об использовании расписания, что может быть полезно для анализа и улучшения учебного процесса.

1. **Общие требования**

Для всех категорий пользователей приложению необходимо удовлетворять следующие общие требования:

* **Интуитивно понятный интерфейс**: приложение должно быть простым в использовании, с логично организованным интерфейсом, чтобы пользователи могли быстро ориентироваться.
* **Доступность на различных устройствах**: поддержка различных разрешений экранов и мобильных устройств, включая смартфоны и планшеты.
* **Производительность и скорость**: приложение должно быть быстрым, обеспечивать мгновенный доступ к расписанию и реагировать на действия пользователя без задержек.
* **Гибкость настроек**: возможность настройки интерфейса и функций приложения под потребности разных категорий пользователей.
* **Обновления и поддержка**: приложение должно быть легко обновляемым, чтобы можно было быстро добавлять новые функции и исправлять ошибки.

Анализ требований пользователей позволяет точно определить, какие функциональные возможности должны быть реализованы в приложении, а также какие особенности интерфейса и взаимодействия с сервером необходимы для достижения наилучшего пользовательского опыта. Эти требования являются основой для проектирования и реализации приложения, способного удовлетворить запросы всех категорий пользователей.

## **1.4. Архитектура приложения**

Архитектура приложения строится на основе клиент-серверной модели, которая предоставляет гибкость и масштабируемость, обеспечивая эффективное взаимодействие между мобильным приложением и сервером. Данная модель позволяет разделить логику приложения на две основные части: клиентскую и серверную, что способствует улучшению производительности и удобству обслуживания.

Основные компоненты архитектуры:

1. **Клиентская часть**: это Android-приложение, разработанное с использованием языка программирования Java. Оно служит интерфейсом для пользователя, позволяя ему взаимодействовать с системой. Клиентская часть отправляет запросы на сервер для получения или изменения данных (например, расписания занятий), а затем обрабатывает ответы от сервера. Android-приложение включает в себя различные экраны и элементы управления, которые делают использование приложения интуитивно понятным и удобным для пользователей. Для повышения удобства взаимодействия реализованы функции уведомлений, оффлайн-доступа к расписанию и персонализации интерфейса.
2. **Серверная часть**: реализована с использованием PHP, что позволяет эффективно обрабатывать запросы от клиента и управлять данными. Серверная часть обеспечивает хранение, извлечение и обработку информации, такой как расписания, данные пользователей и прочие элементы, необходимые для корректной работы приложения. Для управления данными используется реляционная база данных MySQL, которая гарантирует надёжность и быстродействие при работе с большим объёмом данных. На серверной стороне также реализована аутентификация и авторизация пользователей с использованием токенов безопасности.
3. **API (Application Programming Interface)**: это важный компонент, который организует взаимодействие между клиентом и сервером. Используемое REST API позволяет обмениваться данными через HTTP-запросы. Для упрощения процесса отправки запросов и обработки ответов применяется библиотека Volley. Она оптимизирует работу с сетевыми запросами, обеспечивая быструю и безопасную передачу данных. REST API поддерживает стандартные методы (GET, POST, PUT, DELETE), что упрощает расширение функциональности приложения. Взаимодействие через REST API делает систему гибкой, так как позволяет вносить изменения в серверную часть, не затрагивая клиентскую сторону.
4. **Дополнительные механизмы безопасности**: для защиты данных пользователей реализовано шифрование паролей с использованием алгоритма bcrypt. Все передаваемые данные между клиентом и сервером защищены с помощью протокола HTTPS, что предотвращает их перехват злоумышленниками.

Такое разделение на клиентскую и серверную части значительно упрощает управление приложением, его поддержку и развитие. Благодаря стандартизации через API взаимодействие между компонентами становится независимым, что упрощает масштабирование приложения в будущем. Кроме того, гибкость архитектуры позволяет легко добавлять новые функции или интеграции с внешними системами.

## **1.5. Структура базы данных**

Для хранения данных в мобильном приложении используется реляционная база данных MySQL. Структура базы данных организована таким образом, чтобы обеспечивать целостность, удобство обработки и быстродействие при работе с данными пользователей и расписаниями занятий. Основные таблицы базы данных:

1. **Users**: таблица, хранящая информацию о зарегистрированных пользователях приложения. Она включает следующие поля:
   * ID: уникальный идентификатор пользователя.
   * username: имя пользователя для входа в систему.
   * password: зашифрованный пароль пользователя для безопасного хранения.
   * role: роль пользователя (например, студент, преподаватель, администратор).

Эта таблица отвечает за управление пользователями и их доступом к функционалу приложения.

1. **Schedule**: таблица, хранящая информацию о расписаниях занятий. Она включает следующие поля:
   * id: уникальный идентификатор записи расписания.
   * day\_of\_week: день недели, в который проводится занятие (например, понедельник, вторник и т.д.).
   * start\_time: время начала занятия.
   * end\_time: время окончания занятия.
   * subject: название учебной дисциплины.
   * room: аудитория, в которой проводится занятие.

Эта таблица хранит все данные о занятиях, позволяя студентам и преподавателям просматривать актуальное расписание.

**Связи между таблицами:**

* В таблице Users поле ID связано с другими таблицами через внешние ключи, что обеспечивает целостность данных и предотвращает потерю информации при удалении или изменении записей.
* В таблице Schedule информация о расписаниях может быть связана с пользователями (например, преподавателем), что поможет учесть роль каждого пользователя в контексте занятий.

**Возможности расширения модели:**

В будущем можно добавить дополнительные таблицы, такие как:

* **Attendance**: для учёта посещаемости студентов.
* **Subjects**: для хранения информации о предметах.
* **Groups**: для связи расписаний с конкретными группами студентов.

Такое проектирование базы данных помогает эффективно управлять данными, минимизировать вероятность ошибок и улучшить взаимодействие между клиентской и серверной частями приложения. Благодаря использованию индексов для часто запрашиваемых полей достигается оптимизация производительности базы данных.

# **2. Практическая часть**

## **2.1. Создание клиентской части (Android-приложение)**

Клиентская часть была реализована с использованием языка программирования Java и среды разработки Android Studio. Основные этапы разработки включали:

1. **Настройка проекта**:
   * Создание нового проекта в Android Studio.
   * Настройка зависимостей, включая библиотеку Volley для работы с HTTP-запросами.
   * Добавление базовых настроек проекта, таких как иконки, название приложения и версия SDK, для обеспечения корректной работы приложения на различных устройствах.
2. **Реализация экранов приложения**:
   * **Экран авторизации и регистрации**: предоставляет пользователям возможность войти в свою учётную запись, введя логин и пароль, а также зарегистрироваться, если они ещё не имеют аккаунта. Это критично для обеспечения безопасности и персонализации использования приложения.
   * **Главный экран**: отображает расписание занятий в виде списка, сгруппированного по дням недели. На этом экране пользователи могут легко просматривать предстоящие занятия, включая время начала и окончания, название предмета и аудиторию.
   * **Экран добавления/редактирования расписания**: предоставляет возможность пользователям с правами администратора добавлять, изменять или удалять записи о занятиях. Поля ввода включают название предмета, день недели, время начала и окончания занятия, а также аудиторию. Для удобства реализованы элементы календаря и выпадающих списков.
3. **Интеграция клиент-серверного взаимодействия**:
   * Настроены API-запросы для получения данных о расписании, авторизации пользователя и добавления новых записей. Это позволяет приложению взаимодействовать с сервером, обеспечивая актуальность и доступность информации в реальном времени.
   * Реализована обработка ошибок сети, включая проверку наличия активного интернет-соединения. При отсутствии подключения пользователю выводится соответствующее уведомление, а приложение сохраняет последний загруженный вариант расписания для просмотра оффлайн.

Этот подход обеспечил стабильную работу клиентской части приложения и плавное взаимодействие с сервером.

## **2.2. Реализация серверной части (PHP-скрипты и API)**

Серверная часть была реализована с использованием языка PHP. Она выполняет важную роль в обработке запросов, отправленных клиентской частью приложения, а также обеспечивает взаимодействие с базой данных.

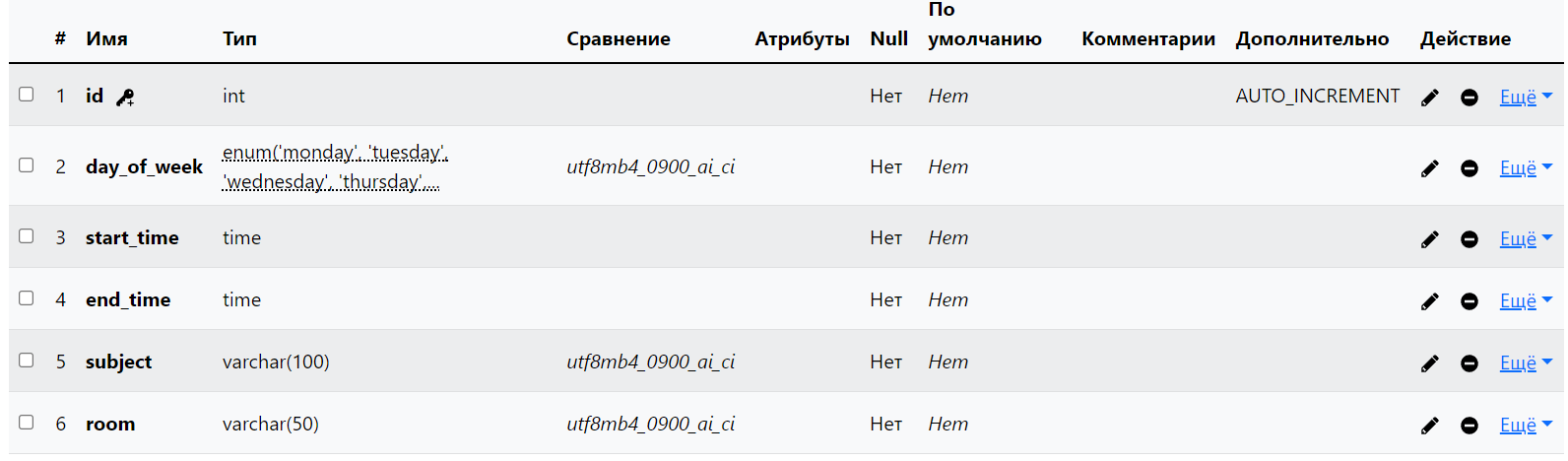


Рис. 6 – Структура таблицы Schedule (расписание)

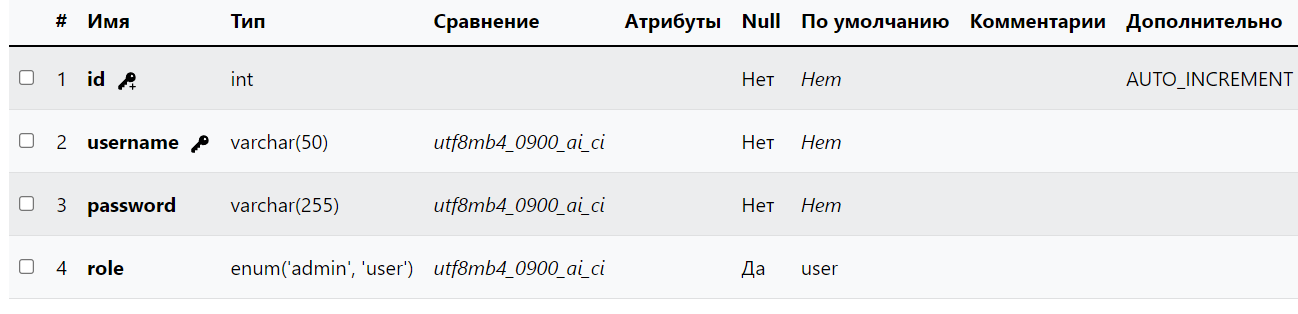


Рис. 7 – Структура таблицы Users (пользователи)

Основные реализованные функции:

1. **API для работы с пользователями**:
   * **Регистрация нового пользователя**: серверная часть обрабатывает запросы на создание новой учётной записи, включая валидацию данных и сохранение информации о пользователе в базе данных.

<?php

$response = array();

if (isset($\_POST['username']) && isset($\_POST['password']))

{

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_BCRYPT);

if (isset($\_POST['role'])) $role = $\_POST['role'];

else $role = 'user';

$result = $con->query("INSERT INTO users (username, password, role) VALUES ('$username', '$password', '$role')");

if ($result)

{

$response["success"] = 1;

$response["message"] = "User successfully registered.";

}

else

{

$response["success"] = 0;

$response["message"] = "Registration failed.";

}

}

else

{

$response["success"] = 0;

$response["message"] = "Required field(s) missing.";

}

echo json\_encode($response);

?>

* + **Авторизация пользователя**: при авторизации проверяется правильность введённых данных (логин и пароль). При успешной проверке сервер генерирует и возвращает токен для дальнейших запросов.

<?php

$response = array();

if (isset($\_POST['username']) && isset($\_POST['password'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

$username = $\_POST['username'];

$password = $\_POST['password'];

$result = $con->query("SELECT \* FROM users WHERE username = '$username'");

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

if (password\_verify($password, $row['password'])) {

$response["success"] = 1;

$response["message"] = "Login successful.";

$response["user"] = array(

"id" => $row["id"],

"username" => $row["username"],

"role" => $row["role"]

);

} else {

$response["success"] = 0;

$response["message"] = "Invalid password.";

}

} else {

$response["success"] = 0;

$response["message"] = "User not found.";

}

} else {

$response["success"] = 0;

$response["message"] = "Required fields are missing.";

}

echo json\_encode($response);

?>

1. **API для работы с расписанием**:
   * **Получение списка всех расписаний**: сервер обрабатывает GET-запросы на получение актуальной информации о расписании занятий для всех пользователей.

<?php

$response = array();

if (isset($\_GET['day\_of\_week'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

$con->set\_charset('utf8');

$day\_of\_week = $con->real\_escape\_string($\_GET['day\_of\_week']);

$result = $con->query("SELECT \* FROM schedule WHERE day\_of\_week = '$day\_of\_week' ORDER BY start\_time ASC");

$response["schedule"] = array();

while ($row = $result->fetch\_assoc()) {

$entry = array();

$entry["id"] = $row["id"];

$entry["start\_time"] = $row["start\_time"];

$entry["end\_time"] = $row["end\_time"];

$entry["subject"] = $row["subject"];

$entry["room"] = $row["room"];

array\_push($response["schedule"], $entry);

}

$response["success"] = 1;

} else {

$response["success"] = 0;

$response["message"] = "Required field(s) missing.";

}

echo json\_encode($response);

?>

* + **Добавление нового занятия**: доступно только пользователям с правами администратора. Сервер принимает POST-запросы с данными для создания новой записи в расписании (предмет, день недели, время и аудитория) и сохраняет их в базе данных.

<?php

$response = array();

if (isset($\_POST['day\_of\_week']) && isset($\_POST['start\_time']) && isset($\_POST['end\_time']) && isset($\_POST['subject']) && isset($\_POST['room'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

$con->set\_charset('utf8');

$day\_of\_week = $\_POST['day\_of\_week'];

$start\_time = $\_POST['start\_time'];

$end\_time = $\_POST['end\_time'];

$subject = $\_POST['subject'];

$room = $\_POST['room'];

$result = $con->query("INSERT INTO schedule (day\_of\_week, start\_time, end\_time, subject, room) VALUES ('$day\_of\_week', '$start\_time', '$end\_time', '$subject', '$room')");

if ($result) {

$response["success"] = 1;

$response["message"] = "Schedule entry successfully added.";

} else {

$response["success"] = 0;

$response["message"] = "Failed to add schedule.";

}

} else {

$response["success"] = 0;

$response["message"] = "Required field(s) missing.";

}

echo json\_encode($response);

?>

* + **Удаление расписания**: доступно только пользователям с правами администратора. Сервер обрабатывает запросы на удаление записей расписания, гарантируя корректность изменений.

<?php

$response = array();

if (isset($\_POST['schedule\_id'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

$con->set\_charset('utf8');

$schedule\_id = $\_POST['schedule\_id'];

$result = $con->query("DELETE FROM schedule WHERE id = '$schedule\_id'");

if ($result) {

if ($con->affected\_rows > 0) {

$response["success"] = 1;

$response["message"] = "Schedule entry successfully deleted.";

} else {

$response["success"] = 0;

$response["message"] = "No entry found with the provided ID.";

}

} else {

$response["success"] = 0;

$response["message"] = "Failed to delete schedule.";

}

} else {

$response["success"] = 0;

$response["message"] = "Required field(s) missing.";

}

echo json\_encode($response);

?>

1. **Обработка ошибок**:
   * Реализована проверка корректности данных, отправляемых клиентом. В случае ошибки сервер возвращает соответствующий HTTP-код состояния (например, 400 — неправильные данные, 401 — неавторизован).
   * Сообщения об ошибках включают описание проблемы для упрощения их обработки на клиентской стороне.
2. **Настройка базы данных**:
   * Созданы таблицы Users и Schedule для хранения информации о пользователях и расписаниях.
   * Настроены индексы и связи между таблицами для повышения производительности запросов и обеспечения целостности данных.

С помощью этого подхода серверная часть предоставляет надёжные и гибкие интерфейсы для работы с расписанием, а также гарантирует безопасность и корректность данных.

## **2.3. Интеграция клиент-серверного взаимодействия**

Интеграция клиентской и серверной части приложения осуществлялась через настройку HTTP-запросов с использованием библиотеки Volley, что обеспечивало эффективное взаимодействие между Android-клиентом и сервером. Основные этапы интеграции включают:

1. **Настройка авторизации**:
   * Клиентская часть отправляет на сервер данные пользователя (логин и пароль) с помощью POST-запроса.
   * Сервер проверяет эти данные и возвращает токен. Токен используется для аутентификации всех последующих запросов, что обеспечивает безопасность взаимодействия между клиентом и сервером.
2. **Получение списка расписаний**:
   * Для получения актуальных данных о расписании отправляется GET-запрос на сервер.
   * Сервер возвращает список записей, который отображается на экране пользователя с использованием компонента LinearLayout для удобного представления данных.
3. **Добавление расписания**:
   * Пользователь с правами администратора может отправлять POST-запросы с данными нового расписания (например, день недели, время начала и окончания, название предмета и аудитория).
   * Сервер обрабатывает запрос и добавляет запись в базу данных, после чего клиент обновляет интерфейс, отображая актуальные данные.
4. **Обработка ошибок и тестирование запросов**:
   * Для проверки корректности работы API использовался инструмент **Postman**. Тестировались ключевые операции, включая GET-запросы на получение расписания и POST-запросы для добавления новых записей.
   * Реализована обработка ошибок на клиентской стороне. Например, при отсутствии соединения или получении некорректного ответа сервер возвращает сообщение об ошибке, которое отображается пользователю.

Эта схема интеграции гарантирует стабильное и безопасное взаимодействие между клиентской и серверной частями, а также упрощает дальнейшую настройку, тестирование и расширение функциональности системы.

## **2.4. Обзор пользовательского интерфейса**

Пользовательский интерфейс приложения был разработан с акцентом на простоту и удобство использования. Основные экраны приложения включают:

1. **Экран авторизации и регистрации**: на этих экранах рис. 1-3 пользователи могут ввести свой логин и пароль для входа в приложение. Также предусмотрена кнопка регистрации для создания нового аккаунта. Это позволяет пользователю легко войти в приложение или создать новый профиль, если он ещё не зарегистрирован.

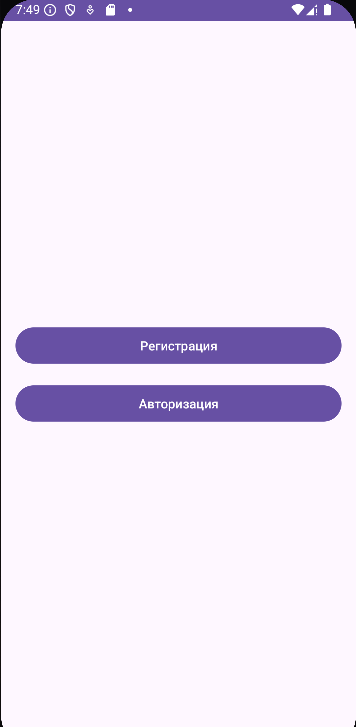


Рис. 1 – Окно входа

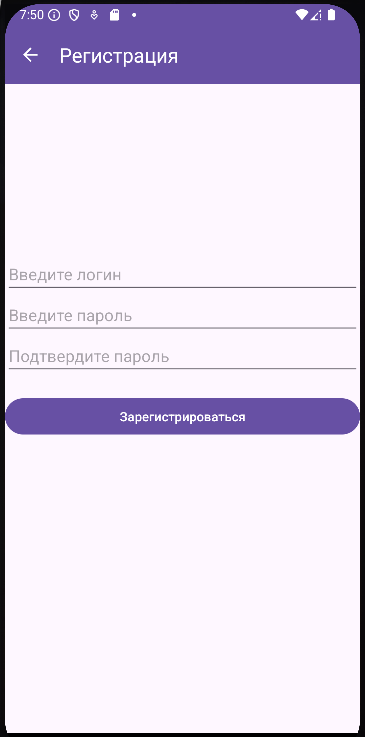


Рис. 2 – Окно регистрации

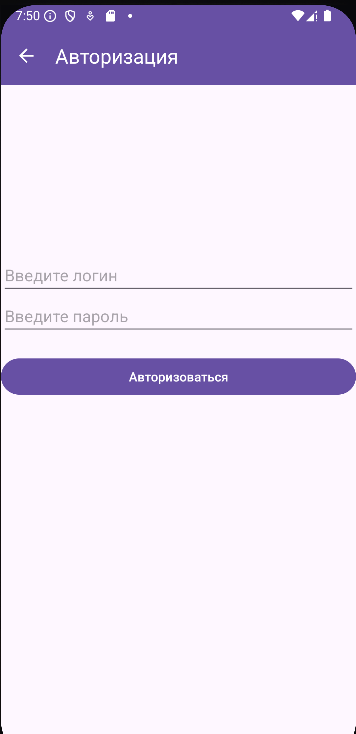


Рис. 3 – Окно авторизации

1. **Главный экран**: данный экран на рис. 4 отображает расписание занятий в виде списка дней недели. Для каждого дня отображаются занятия с указанием времени начала и окончания, а также предмета и аудитории. Пользователь с ролью «admin» имеет возможность редактировать расписание, добавлять новые записи, изменять или удалять существующие. Это позволяет администраторам и преподавателям эффективно управлять расписанием, предоставляя пользователю актуальную информацию.

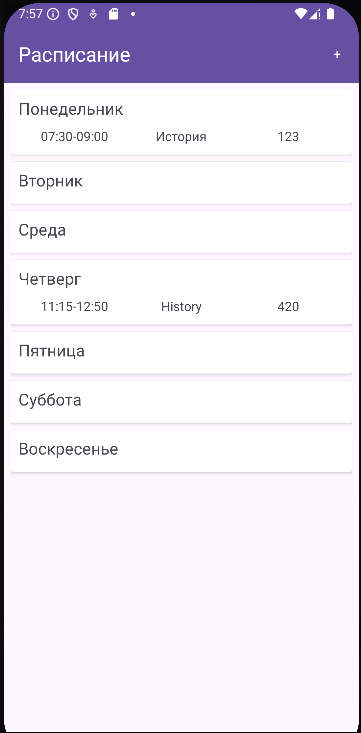


Рис. 4 – Главное окно с расписанием

1. **Экран добавления/редактирования расписания**: данный экран на рис. 5 позволяет пользователям с соответствующими правами вносить изменения в расписание. Включает удобные поля для ввода времени, названия предмета и выбора аудитории. Реализован встроенный календарь для выбора даты и выпадающие списки для минимизации ошибок ввода.

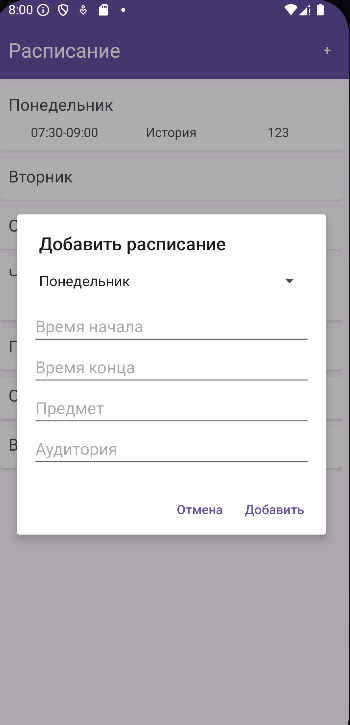


Рис. 5 – Диалоговое окно добавления расписания

## **2.5. Тестирование приложения**

Тестирование является важной частью разработки мобильного приложения, так как оно помогает убедиться в корректности работы всех функций и интерфейса, а также позволяет выявить и устранить ошибки на ранних стадиях. В ходе выполнения курсовой работы тестирование было выполнено в нескольких направлениях.

### **2.5.1 Планирование тестирования**

Тестирование мобильного приложения — это процесс проверки его работы для выявления ошибок и подтверждения того, что все функции выполняются корректно. В рамках курсовой работы планируется провести несколько типов тестирования для проверки функциональности и стабильности приложения.

**Цели тестирования**

Основными целями тестирования являются:

* Проверка корректности работы всех функций (например, добавление и редактирование расписания).
* Убедиться, что приложение работает на разных устройствах и версиях Android.
* Проверить, что приложение безопасно и не передаёт личные данные пользователя без разрешения.

**Типы тестирования**

1. **Модульное тестирование**:
   * Проверка отдельных функций приложения, например, добавление нового занятия в расписание или изменение времени урока.
2. **Интеграционное тестирование**:
   * Проверка взаимодействия между клиентом и сервером, например, отправка данных о расписании на сервер и получение ответа.
3. **Тестирование пользовательского интерфейса (UI)**:
   * Проверка того, насколько удобно и интуитивно понятно работать с интерфейсом приложения. Например, протестировать, легко ли добавить расписание или найти нужный день.
4. **Тестирование совместимости**:
   * Проверка работы приложения на разных устройствах и версиях Android. Приложение должно корректно отображаться и работать на различных экранах и устройствах.
5. **Тестирование производительности**:
   * Проверка того, насколько быстро приложение работает, например, при загрузке расписания или изменении данных.

**Этапы тестирования**

1. **Подготовка тестов**:
   * Написание тестовых случаев, которые описывают, что нужно протестировать (например, добавление нового занятия в расписание).
2. **Основное тестирование**:
   * Проведение тестов, проверка корректности работы всех функций, взаимодействия с сервером и удобства интерфейса.
3. **Исправление ошибок**:
   * Исправление найденных проблем и повторное тестирование.
4. **Заключительное тестирование**:
   * Проверка исправленных ошибок и подтверждение, что приложение работает как надо.

**Инструменты для тестирования**

Для тестирования планируется использовать:

* **Postman** — для тестирования взаимодействия с сервером.

**Критерии завершённости тестирования**

Тестирование будет считаться завершённым, если:

* Все основные функции работают корректно.
* Приложение работает на разных устройствах.
* Ошибки, выявленные в ходе тестирования, исправлены.

Тестирование поможет убедиться, что приложение соответствует требованиям и будет удобно в использовании.

### **2.5.2. Тестирование интерфейса**

Тестирование интерфейса направлено на проверку удобства взаимодействия пользователя с приложением. Важно, чтобы все элементы были на своих местах и корректно реагировали на действия пользователя. На данном этапе были проверены следующие моменты:

* **Отображение элементов интерфейса**: были проверены все экраны приложения (экран авторизации, главный экран, экран расписания и другие), чтобы убедиться, что все элементы отображаются корректно на различных устройствах с разными разрешениями экранов.
* **Интерактивность**: проверялось, что кнопки, текстовые поля, переключатели и другие элементы взаимодействуют с пользователем должным образом (например, кнопки работают при нажатии, текстовые поля принимают ввод и т.д.).
* **Пользовательский опыт**: проводились тесты с пользователями, чтобы убедиться, что интерфейс интуитивно понятен и не вызывает затруднений в использовании.

### **2.5.3. Тестирование функционала**

Тестирование функционала включает проверку всех рабочих процессов приложения, чтобы убедиться, что каждая функция выполняет свою задачу. Для приложения, которое управляет расписанием, были протестированы следующие аспекты:

* **Авторизация и регистрация**: проверялась корректность работы функций регистрации нового пользователя и авторизации, а также правильность обработки неверных данных (например, неправильный логин или пароль).
* **Отображение расписания**: проверялось правильное отображение списка занятий, а также возможность фильтрации расписания по дням недели.
* **Добавление и редактирование расписания**: тестировалась возможность добавления новых предметов в расписание и редактирования существующих данных. Проверялось, что изменения отображаются правильно.
* **Удаление записей**: проверялась возможность удаления записи из расписания и корректная обработка таких действий.

### **2.5.4. Обнаружение и исправление ошибок**

В ходе тестирования были выявлены несколько ошибок, которые требовали исправления:

* **Проблемы с отображением на устройствах с низким разрешением**: некоторые элементы интерфейса некорректно отображались на старых моделях смартфонов. Были сделаны улучшения для оптимизации отображения на устройствах с разными размерами экранов.
* **Ошибки в логике фильтрации расписания**: в некоторых случаях приложение неправильно сортировало расписания по дням недели. Ошибка была исправлена путём уточнения логики фильтрации данных.
* **Проблемы с сохранением изменений**: после редактирования расписания некоторые изменения не сохранялись в базе данных. Это было связано с неправильной обработкой данных на сервере и было исправлено путём доработки обработки запросов на сервере.
* **Ошибки при работе с пользовательскими данными**: при регистрации новых пользователей возникали проблемы с проверкой уникальности логинов. Проблема была решена путём добавления соответствующей валидации на серверной стороне.

После устранения выявленных ошибок приложение было протестировано повторно, и все проблемы были успешно решены.

# **Заключение**

В ходе выполнения курсовой работы была достигнута основная цель — разработка мобильного приложения для управления расписанием занятий на платформе Android. Проведенный анализ существующих решений позволил определить ключевые требования к функциональности приложения, что стало основой для проектирования и реализации.

На этапе проектирования были разработаны архитектура приложения и структура базы данных. Реализованная архитектура на основе клиент-серверной модели обеспечила стабильное взаимодействие между клиентской частью (Android-приложением) и сервером, созданным на языке PHP с использованием базы данных MySQL. Также удалось интегрировать основные функции приложения, такие как регистрация и авторизация пользователей, просмотр и редактирование расписания, а также управление данными через API.

В рамках реализации проекта удалось решить все поставленные задачи:

1. Проведен анализ существующих приложений для управления расписанием, что позволило учесть их преимущества и недостатки.
2. Выбраны технологии и инструменты, соответствующие требованиям современного рынка разработки мобильных приложений.
3. Создано Android-приложение с интуитивно понятным интерфейсом, поддерживающее основные функции для работы с расписанием.
4. Реализовано взаимодействие клиента и сервера с использованием REST API.
5. Обеспечена стабильная работа приложения в процессе тестирования.

Таким образом, цели курсовой работы были успешно достигнуты. Разработанное приложение соответствует поставленным требованиям и обладает потенциалом для дальнейшего развития.

# **Список литературы**

Вот обновленный список литературы, без повторяющихся источников из вашего примера:

1. W3Schools. Руководство по использованию библиотеки Volley для работы с HTTP-запросами [Электронный ресурс]. – Режим доступа: <https://www.w3schools.com/>.
2. Романенко А.В. Разработка мобильных приложений для Android. – М.: БХВ-Петербург, 2017. – 448 с.
3. Соловьев А.М. Архитектура мобильных приложений. – М.: Вильямс, 2015. – 288 с.