**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 мобильных приложений на платформе Android. Часть 11. Разработка приложения менеджера задач»

Студент: Мирганов Э. А.

Институт: ИИТЭ

Направление: 09.03.03

Группа: ПИ-121

Руководитель: Шутов А.В.

Владимир 2024 г.

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc185920136)

[1. Теоретическая часть 4](#_Toc185920137)

[1.1. Понятие и виды мобильных приложений 4](#_Toc185920138)

[1.2. Анализ существующих решений 7](#_Toc185920139)

[1.3. Основные этапы разработки мобильных приложений 8](#_Toc185920140)

[2. Практическая часть 12](#_Toc185920141)

[2.1. Описание поставленной задачи и предметной области 12](#_Toc185920142)

[2.2. Выбор технологий и инструментов разработки 13](#_Toc185920143)

[2.3. Основы архитектуры приложений на платформе Android 14](#_Toc185920144)

[2.4. Разработка клиента (Android-приложение) 17](#_Toc185920145)

[2.5. Разработка сервера (PHP и API) 18](#_Toc185920146)

[2.6. Обзор разработанного приложения 25](#_Toc185920147)

[**ЗАКЛЮЧЕНИЕ** 29](#_Toc185920148)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 30](#_Toc185920149)

## **ВВЕДЕНИЕ**

В современном мире мобильные технологии играют ключевую роль в жизни общества. Одной из наиболее востребованных платформ для разработки мобильных приложений является Android, благодаря её открытости, масштабируемости и широкому спектру возможностей для разработчиков.

Целью данной курсовой работы является создание мобильного приложения для платформы Android, которое обеспечит удобное управление задачами для пользователей. Приложение должно позволять пользователям самостоятельно просматривать, редактировать, добавлять и удалять задачи. Данное приложение должно обеспечить удобный пользовательский интерфейс, простоту навигации и возможность оперативного получения необходимой информации.

Для достижения поставленной цели будут решены следующие задачи:

1. Изучение инструментов и технологий для разработки приложений на платформе Android и выбор оптимальных решений для реализации.
2. Проанализировать существующие доступные решения на рынке мобильных устройств среди менеджеров задач.
3. Разработать архитектуру приложения с учётом требуемого функционала.
4. Разработка пользовательского интерфейса с учётом требований удобства и интуитивной понятности.
5. Провести тестирование приложения, выявить возможные ошибки и гарантировать стабильную работу на различных устройствах.

Таким образом, курсовая работа направлена на приобретение теоретических знаний и практических навыков в области разработки мобильных приложений для Android, что способствует профессиональному развитию в сфере информационных технологий.

## 1. Теоретическая часть

### 1.1. Понятие и виды мобильных приложений

Мобильное приложение представляет собой программу, специально созданную для работы на таких устройствах, как смартфоны и планшеты. Это особый вид прикладного софта, предназначенный для использования на портативных устройствах. Основная цель мобильных приложений — предоставить пользователям возможность выполнять различные задачи в любой момент, без необходимости обращаться к стационарному компьютеру.

Сегодня мобильные приложения занимают одно из ведущих мест в мире разработки, что неудивительно, учитывая повсеместное распространение смартфонов. Их использование стало естественной частью повседневной жизни. Одновременно с ростом популярности мобильных устройств технологии разработки приложений продолжают активно развиваться. Это открывает новые возможности, позволяя создавать приложения для самых разных целей — от работы до развлечений.

Программы можно классифицировать по их предназначению и совместимости с различными операционными системами, такими как Android, iOS, и другие мобильные платформы. Среди них выделяются:

1. Социальные сети. Эти приложения служат для общения и взаимодействия между пользователями. С их помощью можно создавать персональные профили, публиковать контент, обмениваться сообщениями и находить людей с общими интересами.

Основные характеристики социальных сетей:

* 1. Профили пользователей.
  2. Группы, сообщества и каналы.
  3. Взаимодействие между пользователями.
  4. Лента новостей.

1. Игровой контент. Приложения для игр представлены в самых разных жанрах и форматах — от простых мобильных развлечений до масштабных проектов для консолей и ПК. Игры могут быть рассчитаны как на одного игрока, так и на участие в многопользовательских сессиях.

Основные характеристики игрового контента:

* 1. Графика и звук.
  2. Геймплей.
  3. Монетизация.
  4. Интерактивность

1. Промо-приложения. Создаются с целью популяризации брендов, товаров или услуг. Они являются элементом маркетинговой стратегии компании и ориентированы на привлечение и удержание клиентов.

Основные характеристики промо-приложений:

* 1. Брендирование.
  2. Скидки и акции.
  3. Аналитика.
  4. Обратная связь.
  5. Лояльность.

1. Контентные сервисы. Этот вид приложений предлагают пользователям доступ к разнообразному контенту, включая тексты, видео, аудио или подкасты. Доступ может предоставляться на бесплатной основе или по подписке.

Основные характеристики контентных сервисов:

* 1. Разнообразие контента.
  2. Персонализация.
  3. Монетизация.
  4. Социальные функции

Это – классификация по области использования. Такой подход встречается реже и в большей степени удобен для пользователей. Для разработчиков важнее деление программ на типы, основанные на способах их создания.

С этой точки зрения выделяют следующие виды:

1. Нативные приложения. Наиболее популярный формат. Такие программы разрабатываются индивидуально для каждой платформы (например, Android или iOS) с использованием инструментов и языков программирования, характерных для конкретной системы. Чаще всего под мобильными приложениями подразумеваются именно нативные. Примеры языков: Swift или Objective-C для iOS, а также Java или Kotlin для Android. Основные преимущества нативных приложений включают: высокую производительность, полный доступ к возможностям устройства, улучшенный пользовательский интерфейс и удобство обновления.
2. Кроссплатформенные приложения. Этот тип охватывает приложения, которые визуально и функционально напоминают нативные программы, но создаются с использованием веб-технологий. Они обладают следующими характеристиками: кроссплатформенность, доступ к аппаратным функциям устройства, высокая производительность. Они также улучшают взаимодействие пользователя с программой. Примеры: TikTok, Instagram. Разработка осуществляется с применением таких технологий, как Flutter или React Native.
3. Веб-приложения. Эти программы представлены в виде адаптированных сайтов, доступных через браузер, и не требуют установки. Для удобного доступа их можно добавлять в закладки. Примером могут служить прогрессивные веб-приложения (PWA), такие как Twitter Lite или Spotify Web Player.

### 1.2. Анализ существующих решений

На рынке мобильных приложений для управления задачами существует ряд решений, таких как Todoist, Any.do, Microsoft To Do и другие. Эти приложения предоставляют пользователям различные возможности для организации задач, планирования времени и получения уведомлений. Однако каждое из них имеет свои сильные и слабые стороны, что влияет на выбор пользователя в зависимости от его потребностей.

* T**odoist:** предоставляет удобный интерфейс для создания задач, поддерживает категории и метки для организации дел и позволяет синхронизировать данные между устройствами. Одним из преимуществ является наличие различных уровней приоритетов для задач и возможность интеграции с другими сервисами, такими как Google Calendar. Однако среди недостатков — ограниченные возможности в бесплатной версии.
* A**ny.do:** приложение, которое позволяет не только управлять задачами, но и планировать события, а также синхронизировать данные между мобильными устройствами и компьютерами. Any.do предлагает голосовой ввод задач и возможность делегирования их другим пользователям.
* M**icrosoft To Do:** бесплатное и простое в использовании приложение для управления задачами. Оно интегрируется с другими сервисами Microsoft, такими как Outlook и Teams, что делает его хорошим выбором для пользователей, работающих в экосистеме Microsoft. Приложение позволяет организовывать задачи по категориям и предоставляет базовые функции для планирования. Однако его функционал ограничен, и оно не предлагает таких расширенных опций, как другие более специализированные решения.

Из анализа существующих решений можно выделить несколько ключевых требований для разработки успешного мобильного приложения для управления задачами:

1. И**нтуитивно понятный интерфейс:** приложение должно быть простым в использовании, с возможностью быстро добавлять задачи и отмечать выполненные. Удобство и доступность интерфейса играют ключевую роль для пользователей.
2. Крос**сплатформенность и синхронизация:** приложение должно поддерживать работу на различных устройствах (мобильных, планшетах), чтобы пользователь мог продолжить работу с задачами на любом устройстве.

Таким образом, приложения для управления задачами должны сочетать удобство и функциональность. Это поможет создать инструмент, востребованный как для личного использования, так и для командной работы, обеспечивая эффективность и порядок в выполнении задач.

### 1.3. Основные этапы разработки мобильных приложений

Создание мобильного приложения — это комплексный процесс, включающий анализ, проектирование, программирование, тестирование и поддержку. Каждый этап имеет важное значение для достижения высокого качества, надёжности и соответствия требованиям конечного продукта. Для разработки приложений на платформе Android используются современные технологии, таких как языки программирования Java и PHP, среды разработки Android Studio и Open Server, а также базы данных MySQL.

Основные этапы разработки включают:

1. Анализ требований — определение целей приложения, изучение потребностей целевой аудитории и постановка задач.

2. Проектирование — создание структуры, прототипов и дизайна интерфейса.

3. Разработка — написание программного кода и создание серверной части.

4. Тестирование — выявление ошибок и проверка соответствия приложения техническому заданию.

5. Развёртывание — публикация приложения в магазинах приложений (например, Google Play).

6. Поддержка и обновление — исправление ошибок, обновление функционала и учёт отзывов пользователей.

Эти этапы обеспечивают полный жизненный цикл приложения и позволяют достичь оптимального результата. Рассмотрим ключевые этапы подробнее.

## 2. Практическая часть

### 2.1. Описание поставленной задачи и предметной области

**Предметная область** разработки мобильного приложения — это организация и управление задачами пользователей, включая их создание, редактирование, удаление, отметку о выполнении, а также отображение прогресса. Основной целью является создание удобного и интуитивного мобильного приложения, которое поможет пользователям эффективно управлять своими задачами, планировать дела и отслеживать их выполнение.

Приложение разработано для людей, которым важно структурировать свои дела, будь то повседневные задачи, рабочие планы или долгосрочные проекты. Основными пользователями приложения являются частные лица, которые стремятся улучшить свою продуктивность и упростить процесс управления задачами.

З**адача проекта** — создать мобильное приложение для платформы Android, которое обеспечивает функциональные возможности для управления задачами.

Перечень функций приложения описано в таблице 1.

|  |
| --- |
| Функциональные возможности |
| Регистрация и авторизация пользователей для индивидуального доступа к данным. |
| Возможность создания, редактирования и удаления задач. |
| Отображение списка задач с функцией отметки о выполнении. |
| Обработку ошибок при отсутствии связи с сервером. |

Таблица 1 – Функциональные возможности приложения

### 2.2. Выбор технологий и инструментов разработки

Для реализации данного проекта было принято решение использовать проверенные и надежные технологии, которые гарантируют стабильную работу мобильного приложения и соответствуют современным стандартам разработки. Эти технологии были выбраны благодаря их популярности, доступности и широким возможностям, что позволяет создать качественное приложение для управления расписанием.

1. Android Studio: это основная 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 для анализа кода и поддержку эмуляторов устройств. Это значительно упрощает процесс разработки и позволяет разработчикам сосредоточиться на функциональности и дизайне.
2. Java: этот язык программирования обеспечивает стабильность и совместимость с Android SDK, что делает её отличным выбором для разработки мобильных приложений. Благодаря большой экосистеме библиотек и фреймворков, Java позволяет ускорить процесс разработки, улучшить производительность и решить сложные задачи, такие как обработка данных и управление ресурсами устройства.
3. PHP: Популярный серверный язык программирования, используемый для создания API и обработки запросов. В этом проекте PHP используется для реализации серверной части приложения, включая взаимодействие с базой данных и обработку данных, поступающих от клиента.
4. Volley: Библиотека для выполнения HTTP-запросов и взаимодействия с сервером. Volley упрощает работу с сетевыми запросами, позволяет эффективно работать с REST API и предоставляет встроенные механизмы кэширования. Эта библиотека помогает быстро отправлять и получать данные с сервера, минимизируя задержки, что особенно важно для приложений с динамически обновляемыми данными.
5. MySQL: Реляционная база данных, используемая для хранения данных о пользователях и их задачах.
6. Postman (тестирование): Инструмент для проверки и отладки API-запросов. С помощью Postman можно протестировать работу серверной части приложения, удостоверившись в правильности обработки данных и минимизации 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.

### 2.3. Основы архитектуры приложений на платформе Android

Архитектура мобильного приложения представляет собой тщательно организованную структуру, включающую все ключевые компоненты, их взаимосвязи, способы управления данными и пользовательским интерфейсом. В основе такой архитектуры лежит клиент-серверная модель, которая обеспечивает гибкость, масштабируемость и надёжное взаимодействие между клиентской и серверной частями.

Этот подход позволяет разделить функциональность на две основные части, что способствует улучшению производительности, упрощению поддержки и развитию приложения.

Основные компоненты архитектуры:

1. Клиентская часть

Клиентская часть реализована в виде Android-приложения на языке Java и представляет собой интерфейс для взаимодействия пользователя с системой. Она включает следующие элементы:

Activity — основной компонент, представляющий экран или окно пользовательского интерфейса. Каждое приложение может состоять из нескольких Activity для реализации различных функций, между которыми можно переключаться.

Fragment — модуль пользовательского интерфейса, который может быть повторно использован в разных Activity. Это делает интерфейс приложения гибким и адаптивным для различных устройств.

Service — компонент для выполнения фоновых операций, таких как воспроизведение музыки или синхронизация данных.

Broadcast Receiver — обрабатывает системные и пользовательские уведомления, такие как завершение зарядки устройства или получение сообщений.

Content Provider — обеспечивает доступ к данным приложения и их обмен с другими приложениями (например, работа с базами данных или файлами).

Клиентская часть отправляет HTTP-запросы к серверу для получения, изменения или удаления данных, таких как расписания или пользовательская информация. Для сетевого взаимодействия используется библиотека **Volley**, которая ускоряет обработку запросов и упрощает работу с REST AP.

2. Серверная часть

Серверная сторона написана на PHP и работает с реляционной базой данных MySQL. Она выполняет функции хранения, обработки и предоставления данных клиенту.

3. API (Application Programming Interface)

API является связующим звеном между клиентом и сервером. Приложение использует REST API для передачи данных через стандартные HTTP-запросы (GET, POST, PUT, DELETE).

4. Особенности Android-архитектуры

Android-приложения должны быть устойчивыми к ограниченным ресурсам системы, что требует разработки автономных компонентов. Каждый компонент приложения может запускаться или завершаться независимо, в том числе при дефиците ресурсов устройства. Данные и состояния рекомендуется сохранять в устойчивых хранилищах, таких как базы данных или файлы, чтобы избежать потерь информации.

Кроме того, важным элементом архитектуры является файл манифеста, который описывает основные компоненты приложения и их связи. Операционная система Android использует манифест для определения работы приложения на устройстве.

Для лучшего понимания как данные взаимодействуют друг с другом, перед процессом разработки необходимо изобразить архитектуру приложения.

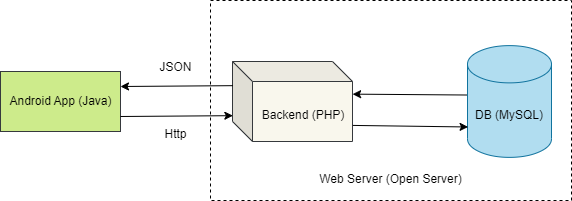


Рисунок 1 – Общая архитектура приложения

### 2.4. Разработка клиента (Android-приложение)

Клиентская часть была реализована с использованием языка программирования Java и среды разработки Android Studio.

Основные этапы разработки:

1. Инициализация проекта:
   * Создание нового проекта в Android Studio.
   * Настройка необходимых библиотек и зависимостей, включая Volley для обработки HTTP-запросов.
   * Установка базовых параметров, таких как название приложения, иконки, а также версии SDK, для обеспечения совместимости с различными устройствами.
2. Создание интерфейса приложения:
   * Экран авторизации и регистрации: предоставляет пользователям возможность авторизоваться в приложении, введя свои данные, или зарегистрировать новую учётную запись. Это обеспечивает безопасность данных и персонализацию работы с приложением.
   * Главный экран: отображает список задач. На данном экране пользователь может просматривать список дел, видеть статус выполнения задач и получать доступ к функциям управления.
   * Экран добавления/редактирования задач: реализует возможность создания новых записей или изменения существующих данных. Пользователь может вводить название задачи.
3. Интеграция клиент-серверного взаимодействия:
   * Настроены взаимодействия через API для передачи данных, таких как аутентификация пользователя, загрузка списка задач и отправка новых записей на сервер. Это позволяет обеспечить актуальность данных в приложении.
   * Реализация механизма обработки ошибок сети: приложение проверяет наличие интернет-соединения и при его отсутствии уведомляет пользователя. Последние данные сохраняются локально, что позволяет использовать приложение в оффлайн-режиме.

Такой подход к разработке клиентской части позволил добиться стабильной и надёжной работы приложения, обеспечив плавное взаимодействие с сервером и удобство использования для пользователей.

### 2.5. Разработка сервера (PHP и API)

Серверная был реализован с использованием языка PHP. Он выполняет важную роль в обработке запросов, отправленных клиентом приложения, а также обеспечивает взаимодействие с базой данных.

Основные реализованные функции:

1. API для работы с пользователями:
   * Регистрация нового пользователя: сервер 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);

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

if ($result) {

$response["success"] = 1;

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

} else {

$response["success"] = 0;

$response["message"] = "Error during registration.";

}

} else {

$response["success"] = 0;

$response["message"] = "Required fields are 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"]

);

} 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 списком задач:
   * Получение списка всех задач: сервер обрабатывает GET-запросы на получение актуальной информации о доступных задачах для всех пользователей.

<?php

$response = array();

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

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

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

$result = $con->query("SELECT \* FROM notes WHERE user\_id = '$user\_id'");

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

$response["success"] = 1;

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

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

$note = array();

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

$note["note"] = $row["note"];

$note["enabled"] = $row["enabled"];

array\_push($response["notes"], $note);

}

} else {

$response["success"] = 0;

$response["message"] = "No notes found.";

}

} else {

$response["success"] = 0;

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

}

echo json\_encode($response);

?>

* + Добавление новой задачи: сервер принимает POST-запросы с данными для создания новой записи в списке задач и сохраняет ее в базе данных.

<?php

$response = array();

if (isset($\_POST['user\_id']) && isset($\_POST['note'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

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

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

$result = $con->query("INSERT INTO notes (user\_id, note, enabled) VALUES ('$user\_id', '$note', true)");

if ($result) {

$response["success"] = 1;

$response["message"] = "Note added successfully.";

} else {

$response["success"] = 0;

$response["message"] = "Error adding note.";

}

} else {

$response["success"] = 0;

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

}

echo json\_encode($response);

?>

* + Удаление задачи: сервер обрабатывает DELETE-запросы на удаление записей задач.

<?php

$response = array();

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

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

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

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

if ($result) {

$response["success"] = 1;

$response["message"] = "Note deleted successfully.";

} else {

$response["success"] = 0;

$response["message"] = "Error deleting note.";

}

} else {

$response["success"] = 0;

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

}

echo json\_encode($response);

?>

* + Редактирование задачи: сервер обрабатывает UPDATE-запросы на изменение содержания записи задачи.

<?php

$response = array();

if (isset($\_POST['note\_id']) && isset($\_POST['note\_text'])) {

require 'db\_connect.php';

$db = new DB\_CONNECT();

$con = $db->con;

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

$note\_text = $\_POST['note\_text'];

$stmt = $con->prepare("UPDATE notes SET note = '$note\_text' WHERE id = '$note\_id'");

if ($stmt->execute()) {

$response["success"] = 1;

$response["message"] = "Note updated successfully.";

} else {

$response["success"] = 0;

$response["message"] = "Error updating note.";

}

$stmt->close();

} else {

$response["success"] = 0;

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

}

echo json\_encode($response);

?>

1. Настройка базы данных:
   * Созданы таблицы Users и Notes для хранения информации о пользователях и их задачах (рис).
   * Настроены связи между таблицами для обеспечения целостности данных.

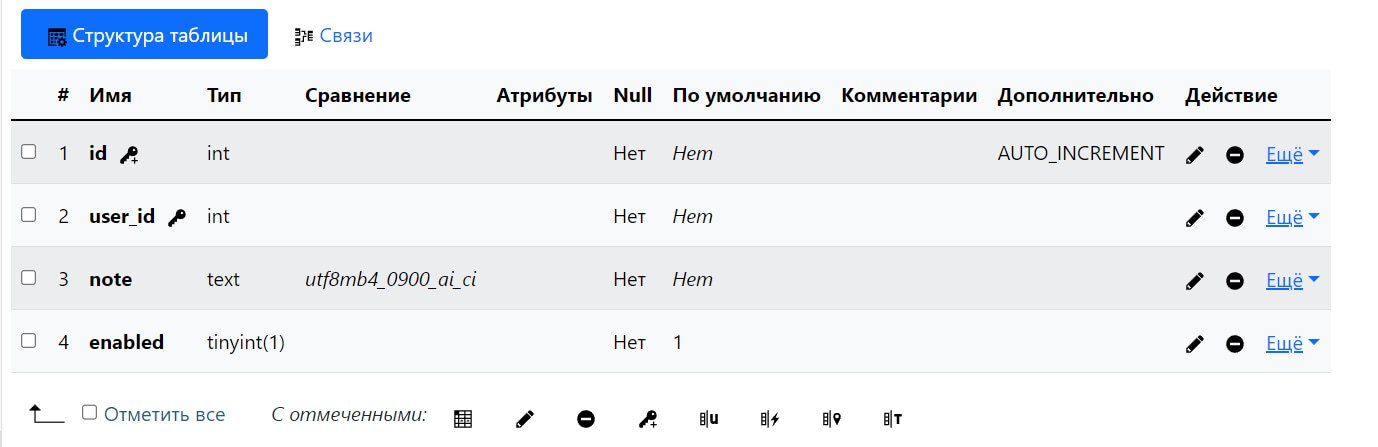


Рис. 2 – Структура таблицы Notes (задачи)

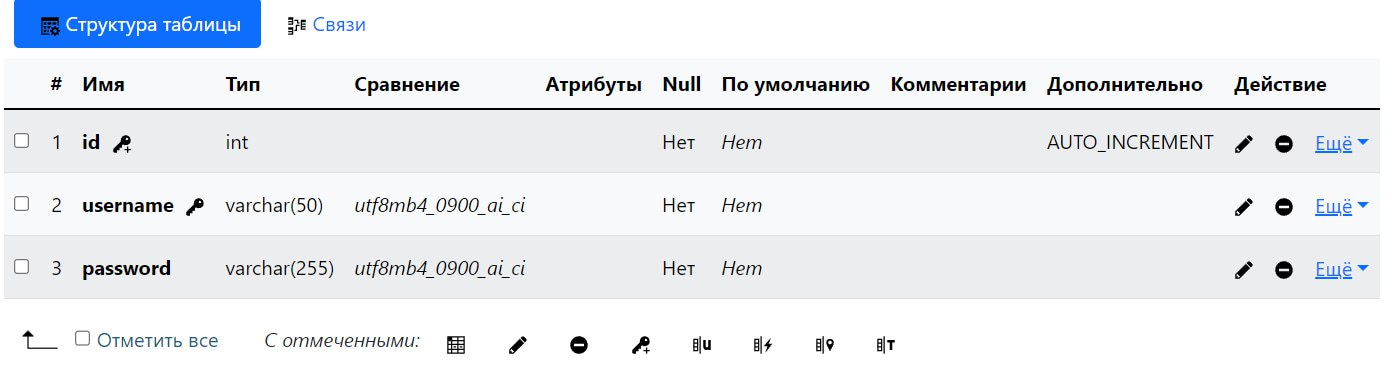


Рис. 3 – Структура таблицы Users (пользователи)

С помощью этого подхода серверная часть предоставляет надёжные интерфейсы для работы с задачами, а также гарантирует безопасность данных.

## 2.6. Обзор разработанного приложения

Интерфейс приложения был разработан с упором на простоту и удобство.

Основные экраны приложения включают:

1. Экран авторизации и регистрации: при открытии приложения любой пользователь в первую очередь проходит авторизацию, либо регистрацию в случае отсутствия действующего аккаунта. Для идентификации пользователь использует логин и пароль, которые хранятся в объекте «Users» (рис. 3). Это позволяет пользователю легко войти в приложение или создать новый профиль, если он ещё не зарегистрирован.

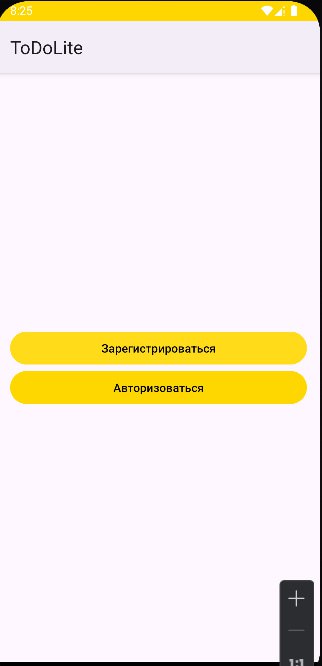


Рис. 4 – Окно входа

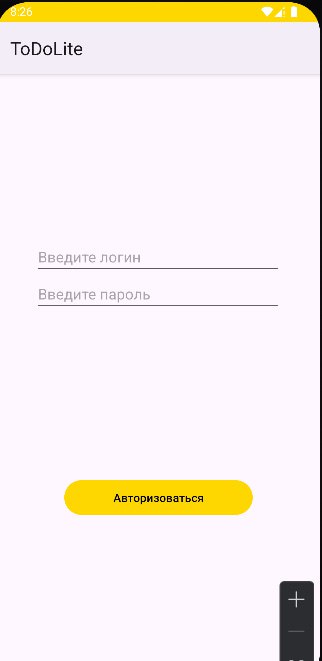
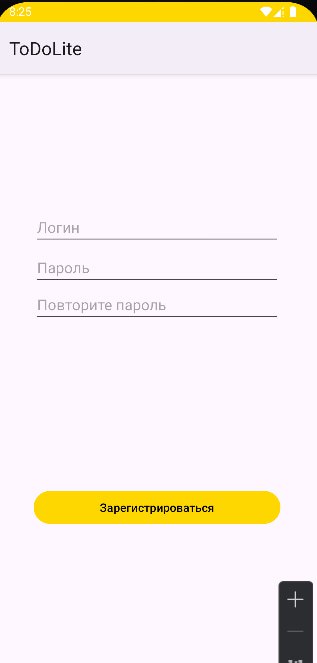
 

Рис. 5 и 6 – Экран авторизации пользователя (рис. 5) и

экран регистрации пользователя (рис. 6)

1. Главный экран: данный экран на рис. 7 отображает задачи в виде списка. Для каждой задачи отображается ее название и статус выполнения. Пользователь имеет возможность редактировать или удалять существующие задачи, а также добавлять новые. Также, чтобы пометить задачу как выполненой или наоборот отметить ее актуальной, каждая задача имеет встроенную кнопку checkbox, при нажатии на которую происходит соответствующее изменение. Это позволяет пользователю эффективно управлять своими задачами и планами, предоставляя простой и интуитивный опыт взаимодействия.

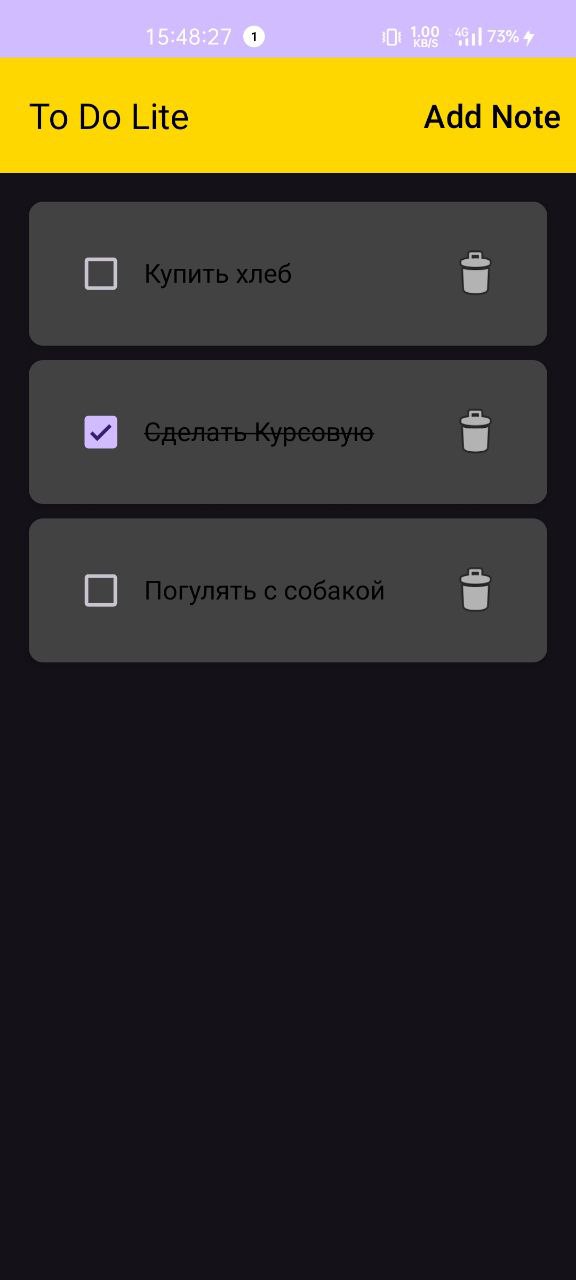


Рис. 7 – Главный экран с списком задач

1. Экран добавления/редактирования задачи: данный экран на рис. 8 позволяет пользователям вносить изменения в список задач. Имеет удобное поле для ввода названия задачи.

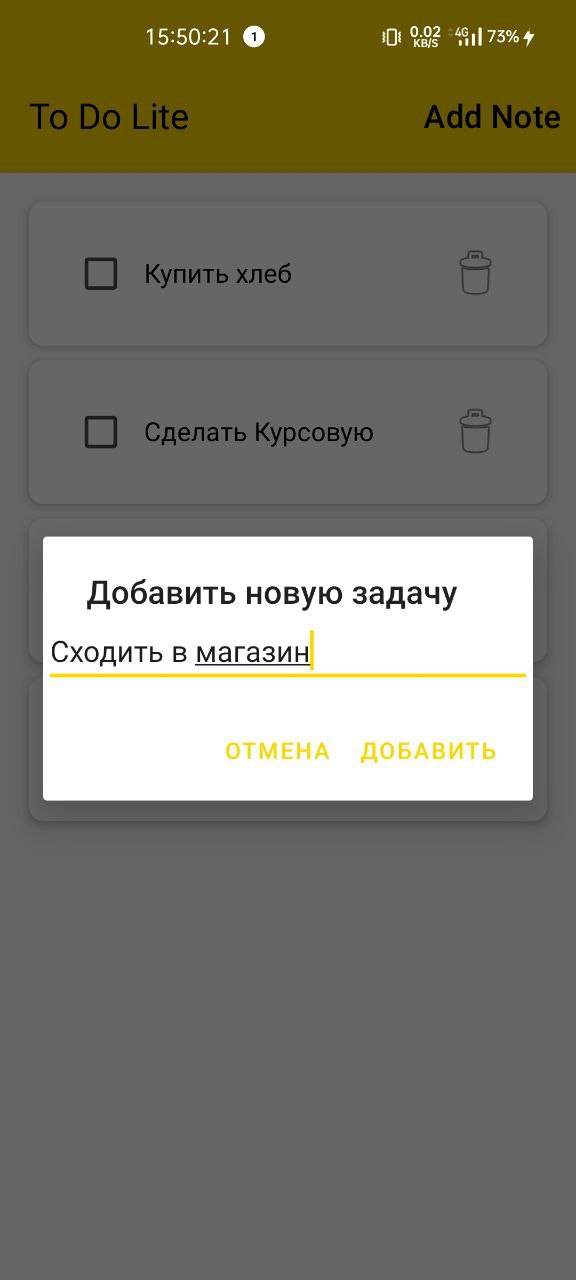
 

Рис. 8 и 9 – Экран добавления задачи (рис. 8) и

экран редактирования задачи (рис. 9)

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была достигнута основная цель — разработка мобильного приложения менеджера задач на платформе Android. Проведённый анализ предметной области и существующих решений позволил определить ключевые требования к функциональности приложения, что стало основой для его проектирования и реализации.

На этапе проектирования были созданы архитектура приложения и структура базы данных. Реализованная клиент-серверная модель обеспечила надёжное взаимодействие между Android-приложением и серверной частью, разработанной с использованием PHP, а также базой данных MySQL. Для клиентской части были выбраны современные технологии, включая Java и среду разработки Android Studio. Взаимодействие между клиентом и сервером организовано через REST API, что позволило обеспечить быстрое и безопасное выполнение всех запросов.

Разработанное приложение поддерживает основные функции, необходимые для управления задачами:

1. Регистрация и авторизация пользователей.

2. Отображение списка задач с возможностью добавления, редактирования, удаления и отметки их выполнения.

3. Обработка ошибок, таких как отсутствие соединения с сервером, с отображением уведомлений для пользователей.

В процессе тестирования приложение показало стабильную работу и соответствие заданным требованиям.

Таким образом, задачи курсовой работы были успешно выполнены, а разработанное приложение сочетает в себе необходимую функциональность, надёжность и удобство.

## **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Гарсиа-Молина Г., Уидом Д., Ульман Д.Д. Системы баз данных. Полный курс. — М.: Юрайт, 2017. — 1248 с.
2. Соловьев А.М. Архитектура мобильных приложений. – М.: Вильямс, 2015. – 288 с.
3. OpenServer. Документация по использованию программной среды [Электронный ресурс]. — Режим доступа: https://ospanel.io/
4. Вайнберг Т. Разработка мобильных приложений под Android. — М.: ДМК Пресс, 2019. — 352 с.
5. Хортон Дж. Android Studio. Разработка приложений на Kotlin и Java. — М.: Питер, 2020. — 688 с.
6. Романенко А.В. Разработка мобильных приложений для Android. – М.: БХВ-Петербург, 2017. – 448 с.
7. Хэллманн Б. PHP и MySQL. Разработка веб-приложений. — СПб.: Питер, 2018. — 608 с.