МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ВЗАИМОДЕЙСТВИЯ ХУДОЖНИКОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Чернышевского Егора Дмитриевича

Научный руководитель
доцент, к. ф.-м. н. _____ Ю. Н. Кондратова

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

Современное общество невозможно представить без использования мобильных приложений, которые стремительно становятся неотъемлемой частью повседневной жизни. С их помощью люди решают разнообразные задачи: от общения и развлечений до покупок и управления личными финансами. Особое место среди них занимают тематические приложения, являющиеся платформой для взаимодействия людей с определенными интересами. Одним из актуальных направлений в данной области является разработка мобильных приложений для профессионального взаимодействия, в том числе художников.

Мобильный формат, в отличие от веб-платформы, имеет следующие преимущества: доступность в любое время и в любом месте, потенциал использования всех возможностей смартфона, а также высокая скорость взаимодействия. Такое приложение открывает для художников возможности поиска единомышленников, демонстрации профессиональных качеств, привлечение клиентов и заказчиков, возможность учить и учиться.

Целью данной дипломной работы является разработка клиент-серверного приложения с мобильным клиентом, позволяющего художникам взаимодействовать между собой. Приложение должно обеспечивать удобный интерфейс для размещения и чтения статей, а также обеспечить возможность пользователям делиться своими работами и общаться между собой.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести обзор инструментов для реализации приложения.
2. Спроектировать архитектуру приложения.
3. Разработать приложение.

Структура и объем работы. Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 3 основные главы, заключение, список использованных источников из 24 наименований и 3 приложения. Работа изложена на 68 страницах. Первая глава имеет название «Анализ приложений для профессиональных сообществ» и содержит разбор преимуществ и недостатков современных приложений для профессиональных сообществ. Вторая глава имеет название «Инструменты и технологии», в ней находится информация об используемых технологиях разработки и их аналогах. Третья глава имеет название «Разработка приложения», данная глава содержит

подробное описание процесса выполнения работы. Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложения с кодом А-В.

Основное содержание работы

Анализ предметной области. Проведен анализ существующих платформ для художников, таких как ArtStation и DeviantArt, а также профессиональных сообществ в других областях (StackOverflow, SoundBetter). Выявлены их ключевые функции, преимущества и недостатки. Их основными преимуществами выступают возможность демонстрации навыков, социальное взаимодействие и профессиональная направленность. Основными проблемами существующих решений являются ограниченные коммуникационные возможности и отсутствие инструментов для творческого роста.

Инструменты и технологии

Java является одним из наиболее популярных языков программирования, активно используемым для создания серверной части клиент-серверных приложений. Его зрелая экосистема, мультиплатформенность, высокая производительность и обширная поддержка делают его оптимальным выбором для разработки серверной логики современных приложений.

Spring Framework — это мощный инструмент для создания современных серверных приложений на языке Java. Он широко используется благодаря своей гибкости, модульности и поддержке различных паттернов проектирования. Spring позволяет ускорить разработку, упростить поддержку кода и повысить производительность приложений. Для клиент-серверной тематической платформы Spring становится идеальным выбором, обеспечивая создание REST API, управление данными и масштабируемую архитектуру.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), которая славится своей производительностью, масштабируемостью и поддержкой сложных структур данных.

Hibernate — это библиотека, реализующая паттерн ORM, который позволяет разработчикам работать с базой данных через объектную модель, минимизируя необходимость написания SQL-кода.

Spring Data JPA — это надстройка над Hibernate, которая автоматизирует многие аспекты взаимодействия с базой данных. Она предоставляет простой API для работы с данными и автоматически генерирует SQL-запросы на основе имен методов.

Postman — один из широко используемых инструмент для разработки,

тестирования и документирования API. Он предоставляет разработчикам удобную среду для взаимодействия с сервером, что делает его незаменимым в процессе создания веб-приложений и API на базе Spring и PostgreSQL. Использование Postman значительно упрощает работу с сервером и помогает улучшить качество разработки.

Docker — это платформа для контейнеризации, которая позволяет упаковать приложение и его зависимости в единый контейнер. Такой подход обеспечивает стабильную работу приложения в любой среде, вне зависимости от конфигурации системы. Благодаря своей эффективности и универсальности Docker стал неотъемлемой частью разработки и развертывания приложений.

Kotlin — современный язык программирования, разработанный для платформы JVM, который используется для разработки Android-приложений. В данном проекте Kotlin применялся для создания мобильного клиента.

Android Studio — это официальная интегрированная среда разработки (IDE) для создания приложений под платформу Android, разработанная Google и основанная на IntelliJ IDEA от JetBrains. Android Studio предоставляет мощный набор инструментов для создания, тестирования и развертывания Android-приложений, что делает её незаменимым инструментом для разработчиков.

Jetpack Compose — это современная декларативная библиотека для создания пользовательских интерфейсов (UI) в Android-приложениях. Она была разработана компанией Google как альтернатива традиционному подходу к разработке интерфейсов с использованием XML-разметки. Compose позволяет разработчикам создавать UI с помощью Kotlin-кода, что делает процесс разработки более интуитивным, лаконичным и гибким.

Room — это библиотека для работы с базами данных в Android, входящая в состав Jetpack. Она предоставляет объектно-реляционное отображение (ORM), упрощая работу с локальной базой данных SQLite. Room автоматически генерирует код для выполнения операций с базой данных и позволяет взаимодействовать с ней с помощью аннотаций и Kotlin-кода.

Gradle — это система автоматизации сборки проектов, которая широко применяется в Java- и Kotlin-проектах, как при разработке приложений для сервера, так и для клиента.

Выбранный стек технологий для реализации серверной части (Java, Spring, PostgreSQL) полностью соответствует требованиям проекта, обеспечивая вы-

сокую производительность, масштабируемость и безопасность. Использование этих инструментов позволила создать надежную серверную часть, способную обрабатывать большое количество запросов и эффективно взаимодействовать с клиентом. Использование Docker и Postman дополнительно ускорило процессы развертывания и тестирования.

Применение рассмотренных технологий обеспечивает не только достижение поставленных целей проекта, но и закладывает основу для дальнейшего масштабирования и развития системы.

Выбранный стек технологий для разработки мобильного клиента (Kotlin + Android Studio + Jetpack Compose + Room) полностью соответствует требованиям современной Android-разработки, обеспечивает безопасный и поддерживаемый код, удобный и эффективный процесс разработки, отзывчивый интерфейс, надежное хранение и обработку данных.

Таким образом, применение рассмотренных технологий не только позволило эффективно реализовать текущие задачи проекта, но также заложило необходимый фундамент для его дальнейшего развития и усовершенствования.

Разработка приложения

Постановка задачи. В результате разработки должно получиться клиент-серверное приложение. Клиентом должно выступать мобильное приложение под ОС Android. Приложение должно иметь следующий функционал: регистрацию и авторизацию, взаимодействие с профилем, многопользовательские чаты, еженедельные испытания, возможность создания локальных заметок и заметок сообщества.

Архитектура приложения. Приложение построено по клиент-серверное архитектуре. Клиент-серверная архитектура представляет собой фундаментальную модель организации взаимодействия в распределённых вычислительных системах, где функциональные обязанности чётко разделены между двумя ключевыми компонентами — клиентскими приложениями и серверными ресурсами. Эта парадигма лежит в основе большинства современных сетевых приложений и веб-сервисов.

Архитектура серверного приложения. В реализации серверной части используется трехслойная архитектура (3-layered architecture) — это популярный подход к организации серверной логики, обеспечивающий четкое разделение ответственности между компонентами системы. В данном проекте такая ар-

хитектура была выбрана как оптимальный баланс между простотой реализации и поддержкой ключевых принципов хорошего проектирования: модульности, тестируемости и масштабируемости. Архитектура состоит из трех основных уровней: Controller (Контроллер), Слой Service и Repository (Репозиторий).

Архитектура мобильного приложения. При создании мобильного приложения использовалась чистая архитектура (Clean Architecture) — это современный подход к проектированию программного обеспечения, предложенный Робертом Мартином. Основная ее цель — создание гибкой, поддерживаемой и легко тестируемой системы за счёт строгого разделения ответственности между компонентами и управления зависимостями. В самом проекте мобильного приложения присутствует три слоя — domain, data и presentaion (ядро системы, слой данных и слой представления). Для реализации слоя представления используется паттерн Model-View-Intent (MVI), который является одним из современных подходов, используемых в разработке мобильных приложений. Он помогает организовать поток данных в приложении по принципу одностороннего цикла, что делает поведение системы более предсказуемым и упрощает отладку. MVI был разработан как эволюция паттернов MV* и особенно хорошо подходит для реактивных приложений.

Реализация серверной части. На стороне сервера слой репозиториев отвечает за доступ к данным и их сохранение. Он взаимодействует с базой данных Postgre с помощью фреймворка Hibernate и Spring Data JPA. В базе данных имеются следующие сущности: ChallengeEntity, ChallengeSubmissionEntity, ChatRoom, ChatMessage, CommentEntity, NoteContentEntity, NoteEntity. Также в данном слое хранятся сами репозитории, которые и отвечают за доступ к данным и генерацию SQL-запросов — автоматическое создание запросов на основе названий методов.

Сервисный слой инкапсулирует основную бизнес-логику приложения. Он принимает данные от контроллеров, выполняет над ними требуемые преобразования и бизнес-правила, а также координирует работу с репозиториями для доступа к данным и их сохранения.

Проект содердит следующие сервисы:

- ChallengeService — описывает логику работы с испытаниями.
- ChatService — описывает логику работы с чатами.
- NoteService — описывает логику работы с заметками.

— UserService — описывает логику работы с аккаунтами пользователей.

Контроллерный слой принимает HTTP-запросы от клиента, делегирует их обработку сервисному слою и формирует HTTP-ответы на основе полученных результатов. Он служит промежуточным звеном между внешними запросами и внутренней бизнес-логикой.

В проекте представлены следующие контроллеры:

— ChallengeController — для принятия запросов по работе с еженедельными испытаниями.
— ChatController — для предоставления API эндпоинтов по работе с чатами.
— NoteController — позволяет клиентам взаимодействовать с заметками сообщества.
— UserController — для принятия запросов по работе с профилями пользователей.

Реализация клиентской части. Доменный слой приложения отвечает за построение бизнес-логики и правил. Этот слой представлен моделями данных данного слоя, репозиториями и вариантами использования, которые описывают возможное поведение приложения.

Репозитории представляют собой набор интерфейсов, которые используются для реализации паттерна репозиторий, который позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Имеется следующий список интерфейсов:

1. AuthRepository — для работы с авторизацией.
2. ChatRepository — для работы с чатами.
3. ChallengeRepository — для работы с еженедельными испытаниями.
4. FavoriteRepository — для работы с избранным.
5. LocNoteRepository — для работы с локальными заметками.
6. NoteRepository — для работы с заметками сообщества.
7. TokenRepository — для получения токена доступа.
8. UserProfileRepository — для работы с профилями пользователей.

Варианты использования представляют собой набор всех возможных вариантов взаимодействия пользователя с программой. Они представлены следующими категориями:

1. auth — описывает варианты взаимодействия пользователя при авторизации.
2. chat — описывает варианты взаимодействия пользователя с чатами.
3. comnote — описывает варианты взаимодействия пользователя с заметками сообщества.
4. challenge — описывает варианты взаимодействия пользователя с еженедельными испытаниями.
5. locnote — описывает варианты взаимодействия пользователя с локальными заметками.
6. profile — описывает варианты взаимодействия пользователя с профилем.
7. favorite — описывает варианты взаимодействия пользователя с избранным.

Слой данных предоставляет доступ к внешним источникам данных – к API серверной части и к локальной базе данных Room, в которой хранятся локальные заметки пользователя и информация об избранном.

Данный слой в проекте содержит в себе следующее: реализацию репозиториев слоя домена, модели данных этого слоя, преобразователи для данных, логику работы с базой данных, интерфейсы для взаимодействия с api.

Раздел с репозиториями содержит реализации интерфейсов репозиториев слоя домена, в этих реализациях описана конкретная логика получения и обработки данных, которые потом будут переданы на слой представления.

Раздел для работы с базой данных содержит в себе сущности базы данных, DAO, описывающие методы для работы с базой данных, конвертеры для преобразование объектов котлина в сущности базы данных Room, а также миграции. Сама база данных используется для хранения и работы с локальными заметками и избранным.

Слой представления отвечает за отображение данных на следующих экранах приложения: экране авторизации, главном экране, экране чата, создания заметок, экранах конкретной заметки сообщества или локальной заметки, еженедельного испытания, экране избранного и профиля.

При запуске приложения на экране пользователя отображается страница авторизации, на которой предлагается ввести логин и пароль или зарегистрироваться. После успешной авторизации пользователь попадает на главную страницу. Отображение главного экрана можно увидеть на рисунке 1.

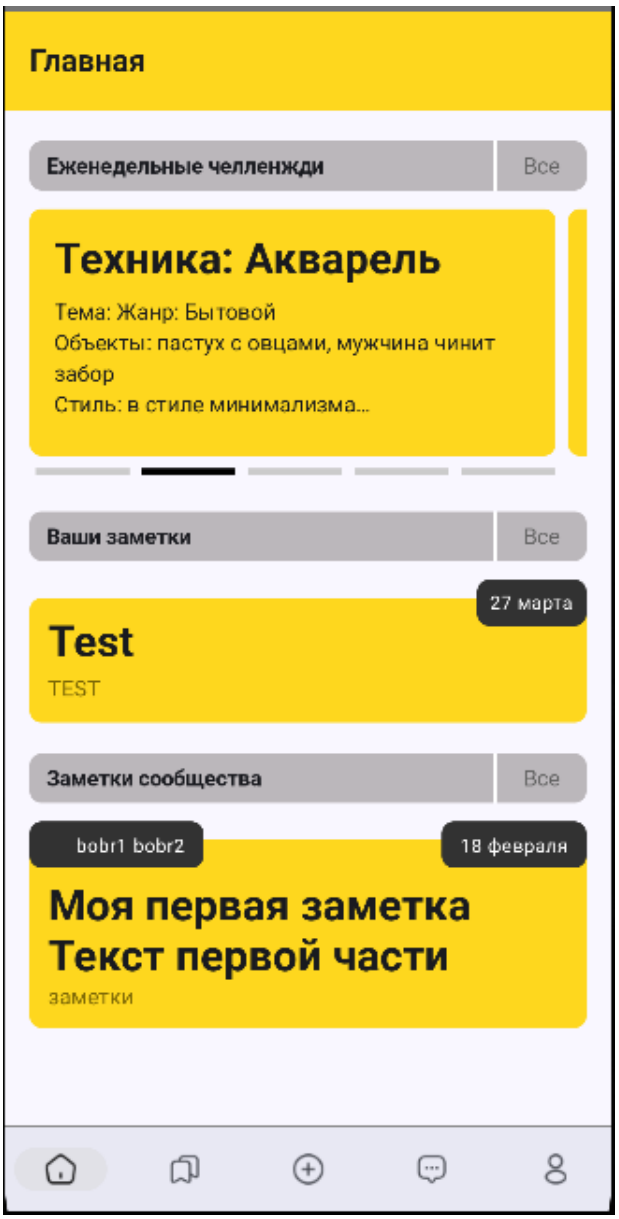

Рисунок 1 – Главный экран

На нем находится слайдер с еженедельными испытаниями, при нажатии на элемент которого пользователю покажет детальное описание конкретного испытания и прикрепленные к нему работы других пользователей.

Также на главном экране можно увидеть превью заметки сообщества и локальной заметки и при необходимости открыть список со всеми заметками или детальное отображение конкретной заметки, где можно будет оставить комментарий к ней.

Внизу экрана располагается панель навигации с 5-ю иконками: главная, избранное, создание заметки, чат, профиль. При нажатии на каждую из иконок происходит навигация на соответствующий экран. В избранном отображаются заметки сообщества и локальные заметки, которые пользователь добавил

в избранное. На экране создания заметки пользователю предлагается создать локальную заметку или заметку сообщества.

На экране чата пользователю предлагается выбрать чат для общения. Для каждой техники рисования существует отдельный чат. В чате можно отправлять текстовые сообщения и видеть сообщения от других пользователей.

Таким образом, разработка клиент-серверного приложения для художников реализована с использованием современных подходов и технологий, обеспечивающих высокую масштабируемость, устойчивость и удобство поддержки кода.

Применение трехслойной архитектуры для серверной части и чистой архитектуры для клиентской позволило четко разделить ответственность между слоями приложения, обеспечив слабую связанность компонентов и независимость бизнес-логики от деталей реализации. Это позволит в будущем легко расширять функциональность, тестировать компоненты и адаптировать приложение к изменяющимся требованиям.

Использование баз данных Postgres и Room обеспечивает централизованное хранение и легкий доступ к данным. В базах данных успешно сохраняется вся необходимая информация о пользователях, заметках, чатах, что позволяет гибко и быстро ее получать для отображения в приложении.

Выбор паттерна Model-View-Intent (MVI) для слоя представления мобильного клиента оправдан ясностью структуры взаимодействия, удобством работы с потоками данных и упрощенной тестируемостью. MVI обеспечивает единое управление состоянием экрана, что снижает вероятность ошибок и облегчает обработку пользовательских событий.

ЗАКЛЮЧЕНИЕ

В настоящей работе была выполнена разработка клиент-серверного приложения с мобильным клиентом под операционную систему Andoid, позволяющего художникам взаимодействовать между собой. В ходе работы были решены следующие задачи: проведен анализ инструментов для решения, продумана архитектура приложения, разработано приложение.

Для реализации приложения использовались следующие инструменты и технологии:

- Java и Spring Framework — для разработки серверной части.
- Docker и Postman — для контейнеризации и тестирования.
- Kotlin — для разработки мобильного клиента под операционную систему Android.
- Jetpack Compose — для создания пользовательского интерфейса.
- Android Studio — как основная среда разработки мобильного приложения.
- PostgreSQL и Room — для хранения данных.
- Gradle — в качестве системы сборки проектов.

Основной функционал разработанного приложения включает в себя:

- Регистрация и авторизация.
- Поддержка чатов.
- Генерация и отображение еженедельных испытаний.
- Создание и просмотр локальных заметок и заметок сообщества.

В дальнейшем данное приложение можно доработать и расширить его функциональные возможности. Например, добавить поддержку push уведомлений о новых сообщениях и расширение аналитики с помощью сбора статистики по выполнению испытаний.