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 КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЕРЕДАЧИ ТЕКСТОВЫХ СООБЩЕНИЙ ПО СЕТИ ИНТЕРНЕТ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Тыньковского Станислава Евгеньевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванов

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

Саратов 2024

ВВЕДЕНИЕ

В современном мире доступ к информации стал невероятно простым и быстрым, а предпочтительный вид общения посредством технологий изменился. Заполучив персональный компьютер в кармане, люди стали реже звонить друг-другу, предпочтительнее звонкам стало текстовое взаимодействие, а у большинства так и вовсе смартфон не выходит из беззвучного режима.

В данной выпускной работе рассматривается разработка мессенджера, включающего в себя базовые функции текстового обмена сообщениями и дополнительные возможности для повышения удобства и безопасности пользователей. В работе будут подробно описаны архитектура системы, протоколы обмена данными, механизмы аутентификации, а также технологии, использованные при разработке.

Целью выпускной квалификационной работы является разработка и реализация прототипа мессенджера с высокой производительностью и уровнем безопасности, обеспечивающего удобный и надежный обмен сообщениями между пользователями.

В соответствии с целью определены следующие задачи:

- определить требования к функционалу и безопасности разрабатываемого мессенджера;
- разработать архитектуру мессенджера;
- реализовать прототип мессенджера, включающий основные функции текстового обмена сообщениями;
- внедрить механизмы аутентификации для обеспечения безопасности передачи данных;
- выявить возможные направления для дальнейшего улучшения и доработки системы.

Теоретическая значимость бакалаврской работы. В работе проведен обзор теоретических основ разработки клиент-серверных приложений и их архитектуры.

Практическая значимость бакалаврской работы. Разработано клиент-серверное приложение.

Структура и объем работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и 10 приложений. Общий объем работы — 70 страниц, из них 50 страниц — основное содер-

жание, включая введение, основные разделы и заключение. Работа содержит 12 рисунков, цифровой носитель в качестве приложения, список использованных источников из 37 наименования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы разработки клиент-серверных приложений» посвящен теории, которая необходима для разработки программного обеспечения подобного рода.

Программное обеспечение

Программное обеспечение (ПО) представляет собой набор компьютерных программ для выполнения задач. Оно состоит из инструкций и данных, необходимых для работы аппаратных средств. Большинство ПО создается на языках высокого уровня, что упрощает разработку и повышает эффективность программистов. Мобильные приложения разработаны для устройств, таких как смартфоны и планшеты, и предоставляют доступ к различным функциям, от развлечений до бизнес-приложений. Разработка мобильных приложений требует учета особенностей устройств, таких как ограниченные ресурсы и разнообразие размеров экранов.

Архитектурные паттерны

Архитектурные паттерны помогают организовать программное обеспечение для его гибкости и масштабируемости. Чистая архитектура ставит целью изоляцию бизнес-логики от инфраструктуры и внешних зависимостей, что облегчает тестирование и поддержку. Принципы SOLID являются основой этой архитектуры. Гексагональная архитектура (или "Ports and Adapters") стремится к изоляции бизнес-логики и упрощает интеграцию с внешними системами через порты и адаптеры. Feature-Sliced архитектура ориентирована на функциональность приложения, разделяя его на модули для улучшения читаемости, расширяемости и повторного использования кода.

Клиент-серверные приложения

Клиент-серверные приложения обеспечивают взаимодействие между клиентом и сервером для обработки запросов и передачи данных. Основные компоненты таких приложений включают бэкенд (серверная часть, ответственная за обработку данных и логику приложения) и фронтенд (клиентская часть, отвечающая за интерфейс и взаимодействие с пользователем).

Протоколы передачи данных, такие как HTTP и WebSocket, играют ключевую роль в обмене данными между клиентом и сервером. HTTP используется для передачи структурированных данных (например, JSON), а WebSocket обеспечивает непрерывное двустороннее соединение для реального времени.

Для безопасности данных используется хеширование паролей с использованием соли, что позволяет сохранять пароли в зашифрованном виде, устойчивом к атакам.

Авторизация пользователей может осуществляться через использование токенов (Access Token и Refresh Token), что обеспечивает безопасный доступ к ресурсам приложения без передачи учетных данных с каждым запросом.

Таким образом, клиент-серверные приложения представляют собой модульные, масштабируемые системы, обеспечивающие эффективное взаимодействие и обработку данных между компонентами на разных уровнях стека приложения.

Обзор существующих решений

Для определения требований к новому приложению полезно рассмотреть характеристики нескольких популярных мессенджеров:

WhatsApp — предлагает обмен текстовыми сообщениями, аудио и видеозвонки, обмен медиафайлами и функцию статуса.

Telegram — поддерживает отправку сообщений, медиафайлов, создание групп и каналов с большим количеством участников.

Signal — обеспечивает шифрование сообщений, аудио и видеозвонки, обмен медиафайлами, не собирает данные для рекламы.

Этот обзор позволяет выявить ключевые функции существующих мессенджеров, необходимые для разработки нового приложения.

Обзор технологий и инструментов, выводы

Для данной выпускной работы как клиентское, так и серверное приложения написаны на одном языке программирования — Kotlin.

Kotlin — кроссплатформенный язык программирования общего назначения с выводом типов, что уменьшает объем кода и повышает его лаконичность.

Преимущества Kotlin:

- Безопасность типов (null safety), избегая ошибок во время выполнения.
- Простой и понятный синтаксис, упрощающий разработку приложений.
- Интероперабельность с Java, облегчающая интеграцию с существующими проектами.
- Поддержка функционального программирования и многопоточности, способствующая созданию чистого и эффективного кода.

Асинхронное программирование позволяет выполнять длительные опера-

ции без блокировки основного потока выполнения.

В Kotlin асинхронное программирование осуществляется с использованием корутин, которые представляют собой приостанавливаемые вычисления, позволяя функциям приостанавливать и возобновлять выполнение в нужный момент.

Среды разработки были выбраны в соответствии с поставленными целями, а также удовлетворяют всем требованиям.

IntelliJ IDEA

— Интегрированная среда разработки для создания приложений на Java и Kotlin.
— Поддерживает полный функционал Kotlin, обеспечивая удобство и эффективность разработки.

Android Studio

— Основная среда разработки мобильных приложений под Android.
— Имеет интуитивно понятный интерфейс и широкий выбор инструментов, таких как редактор кода, отладчик и симуляторы устройств.
— Поддерживает все версии Android и доступна бесплатно для всех разработчиков.

Инструменты разработки мобильной части приложения

Пользовательский интерфейс

Пользовательский интерфейс (UI) является ключевым аспектом взаимодействия пользователей с приложением. Jetpack Compose представляет собой современный инструмент для создания UI на платформе Android, использующий декларативный подход на языке Kotlin. Он предлагает готовые компоненты, упрощает разработку, обеспечивает высокую производительность и поддерживает создание анимаций.

Внедрение зависимостей

Hilt — это библиотека для внедрения зависимостей в Android, которая строится на основе Dagger. Она упрощает управление зависимостями, обеспечивает удобные аннотации и автоматическое управление жизненным циклом компонентов.

HTTP-Клиент

Для взаимодействия с сервером часто используются OkHttp и Retrofit. OkHttp представляет собой мощный HTTP-клиент, а Retrofit упрощает создание

HTTP-запросов и обработку ответов.

СУБД

Room — это библиотека для работы с базами данных SQLite в Android, предоставляющая удобный способ взаимодействия с базой данных, с типобезопасностью и абстракцией над низкоуровневыми операциями.

Хранение простых данных

DataStore предлагает современный способ хранения данных в Android в виде ключ-значение, обеспечивая асинхронное API для записи и чтения данных и защиту от ошибок в рантайме.

Второй раздел «Описание разработки клиент-серверного приложения» посвящен созданию серверного и мобильного приложений.

Разработка серверной части приложения

Следуя принципам гексагональной архитектуры, функциональность серверного приложения разделена на модули, при этом лишь несколько модулей имеют зависимости от других.

Краткий обзор каждого модуля:

- Infra — настройка приложения, точка входа;
- Common — простой код, который может быть полезен для других модулей;
- Core — содержит бизнес-логику, сервисы, которые реализуют сценарии использования, содержит интерфейсы портов для адаптеров, а также в этом модуле находятся бизнес-модели;
- Adapters — обеспечивают специфичные для платформы/фреймворка функции, к примеру, здесь находятся классы репозиториев базы данных и адаптеры;

Модуль Common

С помощью технологии Mapped Diagnostic Context (MDC) можно выдать уникальный id каждому запросу для облегчения логгирования информации.

Модуль Core

Модуль Core является фундаментальной частью системы, определяющей бизнес-логику и основные правила приложения. Он предоставляет интерфейсы для взаимодействия с компонентами, реализованными в других модулях. Этот модуль обеспечивает управление бизнес-правилами, сущностями и сервисами.

Структура модуля Core включает:

- models

— outport
— usecase
— services

Модуль Adapters

Модуль Adapters отвечает за взаимодействие с внешними системами и инфраструктурными компонентами. Включает модули env, persist, primary-web и security.

Регистрация

При регистрации пользователя сохраняется сгенерированная адаптером соль и хешированный пароль в базе данных.

Авторизация и аутентификация

Авторизация на основе JWT токенов с сроком действия 10 минут для Access и полугодовым сроком для Refresh. Используется Token Claims и JWTPayloadHol для хранения информации о пользователе в токене.

Использование механизма аутентификации на основе Principal интегрировано с ktor, что позволяет автоматически идентифицировать пользователя по access key в каждом запросе.

Разработка клиентской части приложения

Модульная архитектура является эффективным подходом к разработке масштабируемых и поддерживаемых мобильных приложений, разделяя их на независимые компоненты.

Краткий обзор каждого модуля:

— app — является основной точкой входа в приложение. Его главная задача — управление навигацией и основной конфигурацией приложения;
— core — содержит основные компоненты и логику, необходимые для работы приложения;
— feature — представляет из себя отдельные функциональные блоки приложения. Они разделяются на логические компоненты для упрощения разработки и поддержки.

Модуль App

Этот модуль зависит от других модулей приложения и включает в себя компоненты Activity и Application.

— Activity — основные компоненты пользовательского интерфейса, обрабатывающие взаимодействие пользователя с приложением.

— Application — подкласс, используемый для инициализации глобальных ресурсов и настроек приложения.

Модуль Core

Модуль Core включает следующие модули:

— common — общие утилиты, константы и вспомогательные классы, используемые в различных частях приложения;
— data — базовая инфраструктура для работы с данными, репозитории, интерфейсы доступа к данным и управление кэшированием;
— datastore — классы для работы с различными источниками данных;
— database — код, связанный с локальным хранением данных, классы для работы с базами данных SQLite и классы DTO;
— designsystem — компоненты пользовательского интерфейса, стили и другие ресурсы для единообразного дизайна приложения;
— domain — бизнес-логика приложения и модели данных, используемые на всех уровнях архитектуры;
— model — классы данных POJO, представляющие данные в приложении;
— network — код, отвечающий за взаимодействие с сетью, клиенты API, обработка запросов и ответов;
— ui — базовые компоненты пользовательского интерфейса, используемые в разных частях приложения.

Модуль Feature

Модуль Feature включает следующие модули, представляющие различные экраны:

— auth — компоненты, связанные с аутентификацией пользователей, экраны входа и регистрации;
— chat — компоненты, связанные с чатом;
— chats — управление списком чатов, отображение списка контактов и управление групповыми чатами;
— contacts — управление списком контактов, поиск и добавление новых контактов;
— settings — экраны и компоненты для настройки приложения и управления учетными записями;
— user — компоненты, связанные с профилями пользователя, настройками, аватаром и т.д.;

— search — компоненты для реализации функциональности поиска в приложении.

Каждый модуль содержит экраны пользовательского интерфейса и вью-модели для управления логикой экрана.

Сильные стороны проекта

Одной из сильных сторон всего приложения является многомодульность как клиентской части, так и серверной, что подразумевает облегчение добавления нового функционала. Также сильной стороной можно назвать тот факт, что мобильное приложение сделано в стиле offline-first, то есть данные после получения устройством кешируются, что позволяет избавиться от длительных экранов загрузки.

Возможные дальнейшие исследования

Учитывая возможности разработанного прототипа, можно расширить функционал приложения, например:

1. ведение нескольких аккаунтов в приложении;
2. создание дополнительного экрана с вводом кода для запуска приложения;
3. восстановление учетной записи пользователя, если он забыл пароль;
4. отправка медиа и голосовых сообщений;
5. создание чата для двух пользователей, используя средства сквозного шифрования.

ЗАКЛЮЧЕНИЕ

В настоящей работе было создано клиент-серверное приложение для отправки и получения сообщений в реальном времени.

В соответствии с полученными результатами были сделаны следующие выводы:

- клиент-серверная архитектура решает список задач, поставленных в выпускной работе;
- мобильные приложения под ОС Android наиболее распространены, поэтому, эта платформа подходит для проектов подобного типа;
- использование языка kotlin является лучшим решением, так как он подходит для написания клиента и сервера;
- для передачи информации в реальном времени подходит протокол WebSocket, а для информации, в которой нет мгновенной потребности, подходит протокол HTTP;
- безопасность является неотъемлемой частью клиент-серверного приложения, поэтому ей необходимо уделить время на начальном этапе проектирования;
- пагинация и кеширование в мобильном приложении позволяют улучшить пользовательский опыт.

Основные источники информации:

1. *Марсикано, К.* Android. Программирование для профессионалов / К. Марсикано, Б. Гарнднер, Б. Филлипс, К. Стюарт. — Санкт-Петербург: ООО Издательство «Питер», 2021;
2. *Мартин, Р.* Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин.— Санкт-Петербург: ООО Издательство «Питер», 2018;
3. Guide to Android app modularization [Электронный ресурс]. — URL: https://develop (Дата обращения 20.05.2024). Загл. с экр. Яз. рус;
4. *Haroon-Sulyman, S.* Client-server model / S. Haroon-Sulyman // IOSR Journal of Computer Engineering 16. — 2014. — Pp. 57–71;
5. *Лоре, А*. Проектирование веб-API / А. Лоре. — Москва: Издательство ДМК, 2020.