УДК 004 **Информационные технологии**

Клёмин Никита Алексеевич, *студент факультета информатики и вычислительной техники, Мордовский государственный университет имени Н.П. Огарёва, г. Саранск, Россия*

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ЗАМЕТОК

Аннотация: Данная статья представляет собой обзор процесса разработки мобильного приложения заметок, начиная от концептуального проектирования до финальной реализации. Автор предлагает подробный взгляд на ключевые этапы разработки, методы и инструменты, которые могут быть использованы при создании современного и интуитивно понятного мобильного приложения заметок.

Целью исследования является разработка мобильного приложения для создания заметок, работающего под управлением ОС Android, способного решить проблему забывчивости, с которой сталкиваются многие люди в повседневной жизни.

Ключевые слова: программное обеспечение, мобильное приложение, мобильная разработка, средства реализации, Android, PostgreSQL, заметки, Kotlin, Ktor.

Annotation: This paper provides an overview of the development process of a mobile notes application, from conceptual design to final implementation. The authors offer a detailed view of the key development steps, methods and tools that can be used in creating a modern and intuitive mobile notes application.

The aim of the study is to develop a mobile note-taking application running on Android OS that can solve the problem of forgetfulness that many people face in their daily lives.

Keywords: software, mobile application, mobile development, implementation

tools, Android, PostgreSQL, notes, Kotlin, Ktor

Введение

В современном мире мобильные устройства стали неотъемлемой частью нашей повседневной жизни. Они позволяют нам быть в курсе всего, что происходит в нашем мире. Вследствие чего возросло и наше желание эффективно управлять данной информацией, которая поступает к нам из различных источников.

Одним из наиболее популярных инструментов для организации информации являются приложения заметок. Они позволяют пользователям создавать, редактировать и управлять своими заметками с помощью мобильного устройства, который всегда находится под рукой.

Система заметок, реализованная в мобильном приложении, позволит пользователям более эффективно управлять своими задачами, улучшить свою продуктивность и сосредоточиться на действительно важных аспектах своей жизни и работы. Мобильное приложение предоставляет удобный инструмент для создания, организации и отслеживания заметок.

Целью исследования является разработка мобильного приложения для создания заметок, работающего под управлением ОС Android, способного решить проблему забывчивости, с которой сталкиваются многие люди в повседневной жизни.

Для достижения цели определен следующий список задач:

- изучение и анализ предметной области,
- выбор инструментов для разработки,
- разработка клиентской и серверной части приложения.

Система будет представлять собой полноценный программный продукт. Для обеспечения возможности расширения функционала приложения мы предусмотрим модульную систему, которая позволит добавлять и изменять уже существующие функции с помощью обновлений.

Были выявлены следующие функциональные возможности для продукта:

– создание заметок: пользователь должен иметь возможность создавать заметки в приложении. При создании заметки пользователь может вводить текст и опционально прикреплять фотографии. Пользователь также должен иметь возможность выбрать папку для заметки;

– редактирование заметок: пользователь должен иметь возможность редактировать уже созданные заметки. При редактировании заметки пользователь может изменять текст и фотографии, прикрепленные к заметке;

– удаление заметок: пользователь должен иметь возможность удалять заметки, которые ему больше не нужны;

– просмотр списка заметок: пользователь должен иметь возможность просматривать список своих заметок в какой папке, отсортированных по дате создания. Каждая заметка в списке должна отображать название, статус закреплённости и дату создания;

– просмотр заметки: пользователь должен иметь возможность просмотреть детали заметки, включая текст и фотографии, прикрепленные к заметке;

– поиск заметок: пользователь должен иметь возможность искать заметки по тексту или категории;

– папки для заметок: пользователь должен иметь возможность создавать папки для заметок и прикреплять заметки к этим папкам. Пользователь также должен иметь возможность просмотреть список всех папок и список заметок, относящихся к определенной папке.

– синхронизация заметок: пользователь должен иметь возможность синхронизировать заметки с сервером, чтобы сохранить их на удаленном сервере и синхронизировать с другими устройствами.

– безопасность: пользовательские данные должны быть защищены от несанкционированного доступа и утечки [1]. Для этого приложение должно предоставлять безопасную авторизацию и аутентификацию пользователей, а также защищать данные при передаче и хранении на сервере.

Эти функции позволят удовлетворить потребности пользователей в

эффективной и удобной системе заметок.

В ходе исследования были проанализированы пять самых популярных приложений для создания заметок: Google Keep, Evernote, Standard Notes, Typora, OneNote. В результате анализа данных приложений, было принято решение сделать функциональность приложения полностью бесплатной, опираясь на положительные составляющие конкурентов.

При разработке серверной части для мобильного приложения заметок необходимо выбрать наиболее подходящие технологии для реализации.

Основные требования к серверу:

- Быстродействие и масштабируемость;
- Высокая надежность и безопасность;
- Поддержка нескольких клиентов одновременно;
- Возможность хранения и обработки большого количества данных.

Для разработки сервера был выбран язык программирования Kotlin и веб-фреймворк Ktor. Ktor — это легковесный фреймворк для создания серверных приложений на языке Kotlin. Ktor обеспечивает высокую производительность и простоту разработки. Он использует Netty для обработки входящих запросов и позволяет легко создавать RESTful API. Ktor также предоставляет множество дополнительных библиотек, таких как серверная авторизация, обработка ошибок, сериализация и десериализация JSON, и т.д.

Для внедрения зависимостей была использована библиотека Dagger, которая позволяет упростить создание и управление зависимостями в приложении. Он обеспечивает более чистый и модульный код, что делает приложение более поддерживаемым и расширяемым.

Для хранения данных мы будем использовать PostgreSQL, так как это свободная объектно-реляционная система управления базами данных, которая обеспечивает высокую производительность, надежность и масштабируемость [2]. PostgreSQL широко используется в проектах с высокими требованиями к производительности и безопасности, таких как банковские системы и интернет-магазины.

Для работы с PostgreSQL мы будем использовать библиотеку Exposed. Это ORM-библиотека для работы с базами данных на языке Kotlin. Exposed обеспечивает высокую производительность и удобство в использовании [3]. Она позволяет описывать схему базы данных с помощью языка Kotlin, что делает код более читабельным и понятным. Exposed также предоставляет удобный API для работы с базами данных, включая поддержку SQL-запросов, транзакций, миграций и т.д.

Также мы будем использовать библиотеку HikariCP. HikariCP — это быстрый и легковесный пул соединений для баз данных. HikariCP обеспечивает высокую производительность и надежность при работе с базами данных [4]. Он имеет много настроек и опций для оптимизации производительности и управления соединениями

Для обеспечения безопасности мы будем использовать механизм авторизации и аутентификации на основе токенов JSON Web Token (JWT), который позволяет безопасно передавать информацию между клиентом и сервером. Для реализации механизма авторизации и аутентификации мы будем использовать модуль Ktor для работы с JWT.

Для хеширования паролей была выбрана библиотека JBCrypt. JBCrypt обеспечивает высокую степень безопасности при хранении паролей в базе данных. Она использует медленные алгоритмы хеширования, чтобы защитить пароли от атак перебора.

Все выбранные технологии являются совместимыми и обеспечивают высокую производительность, надежность и безопасность серверной части приложения.

Структура сервера представлена на рисунке 1

Data layer

Модуль данных состоит из всех операций, связанных с данными, которые взаимодействуют с уровнем данных приложения, т.е. PostgreSQL. Он включает в себя сущности и таблицы базы данных из фреймворка JetBrains Exposed (ORM) и классы DAO для операций с данными.

Dependency Injection(DI)

Это слой отвечает за управление зависимостями компонент и модулей приложения, что позволяет сделать его более модульным, тестируемым и легко расширяемым.

В данном проекте DI слой состоит из компонентов и модулей. Компоненты определяют группу зависимостей, которые могут быть внедрены в другие компоненты или в классы приложения. Модули содержат определения зависимостей, которые могут быть внедрены в компоненты.

Modules

Каждый модуль будет определять маршруты, связанные с его функциональностью. Для его настройки будет использоваться расширение функции Route, которое будет получать инжектированный объект контроллера. Этот слой отвечает за реакцию на вызов API, получение параметров и вызов функции контроллера с их помощью.

В данном слое так же присутствуют контроллеры. На каждый модуль будет приходиться один контроллер, и он будет получать путем инъекции те объекты API, которые он будет использовать для обработки запросов модуля. Здесь должна быть определена вся бизнес-логика. Контроллер будет получать/записывать данные через объекты API, а неудачные пути будут обрабатываться с выбросом исключений и ответом в виде страниц состояния.

Plagins

Данный слой отвечает за подключение и настройку сторонних плагинов, которые расширяют функциональность приложения. Он обеспечивает модульность приложения и позволяет легко подключать и настраивать сторонние плагины. Каждый модуль плагина может быть подключен или отключен в зависимости от потребностей приложения, что делает приложение более гибким и расширяемым.

Utils

Здесь будут находится различные вспомогательные функции и утилиты, которые используются в разных частях приложения

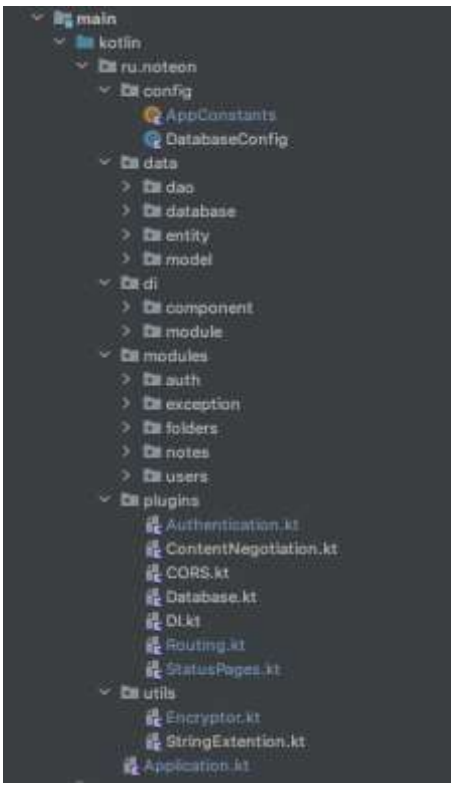

Рисунок 1 – структура веб-сервера

При разработке мобильного приложения для Android, правильный выбор технологий и библиотек может существенно ускорить процесс разработки, улучшить качество приложения и обеспечить более простое сопровождение в дальнейшем.

В данном случае, для разработки мобильного приложения заметок для операционной системы Android, были выбраны следующие технологии и библиотеки:

– Dagger:Hilt является фреймворком для внедрения зависимостей, который позволяет автоматически генерировать код внедрения зависимостей. Dagger:Hilt упрощает процесс внедрения зависимостей в проект и позволяет избежать проблем, связанных с управлением зависимостями вручную.

– Hilt-Navigation-Fragment — это библиотека, которая предоставляет удобный способ внедрения зависимостей в Navigation Graph. Она позволяет инжектировать зависимости во фрагменты, которые находятся в Navigation Graph, и управлять их жизненным циклом.

– Security-Crypto-KTX — это набор Kotlin-расширений, который

упрощает работу с криптографией в Android. Эта библиотека позволяет безопасно хранить конфиденциальную информацию, такую как пароли и ключи шифрования.

– Room — это компонент Android Architecture Components, который позволяет создавать базы данных SQLite на основе объектно-ориентированной модели. Room упрощает работу с базами данных и позволяет выполнять операции с базой данных в фоновом режиме.

– SwipeRefreshLayout — это компонент пользовательского интерфейса Android, который позволяет обновлять содержимое приложения, проводя пальцем вниз по экрану. Он обеспечивает удобный способ обновления содержимого приложения без необходимости перезагрузки приложения.

– Work-Runtime — это библиотека, которая предоставляет удобный способ запуска фоновых задач в Android. Эта библиотека позволяет выполнять задачи в фоновом режиме и управлять их выполнением.

– Navigation-UI — это компонент Android Architecture Components, который предоставляет удобный способ управления навигацией в приложении. Он упрощает процесс навигации между фрагментами и активностями в приложении.

– Lifecycle-ViewModel — это компонент, предоставляющий удобный способ управления жизненным циклом приложения. Он позволяет создавать модели представления, которые сохраняют свои данные при изменении конфигурации устройства, таких как поворот экрана.

– Lifecycle-Runtime — это компонент Android Architecture Components, который предоставляет удобный способ управления жизненным циклом приложения в рантайме. Он позволяет регистрировать и отменять регистрацию компонентов в соответствии с жизненным циклом приложения.

– Retrofit2 — это библиотека, которая предоставляет удобный способ работы с REST API в Android. Он упрощает процесс отправки запросов на сервер и получения ответов.

– Datastore-Preferences — это библиотека, которая предоставляет

удобный способ работы с хранилищем настроек в Android. Она предоставляет более безопасный способ работы с данными, чем SharedPreferences.

– Moshi-Kotlin — это библиотека, которая предоставляет удобный способ работы с JSON-данными в Android. Он позволяет сериализовать и десериализовать JSON-данные в Kotlin-объекты.

Выбор технологий для мобильного приложения заметок для Android зависит от конкретных требований и задач проекта. Однако, использование перечисленных технологий может значительно упростить процесс разработки и обеспечить успешную реализацию проекта.

Для разработки мобильного приложения была выбрана архитектура MVVM (Model-View-ViewModel) в сочетании с Clean Architecture.

Архитектура MVVM была выбрана из-за ее простоты, расширяемости и возможности использования современных фреймворков и библиотек. Она разделяет приложение на три основных компонента: Model (модель), View (представление) и ViewModel (модель представления). Каждый компонент имеет свои задачи и ответственности. Данная архитектура показана на рисунке 2.

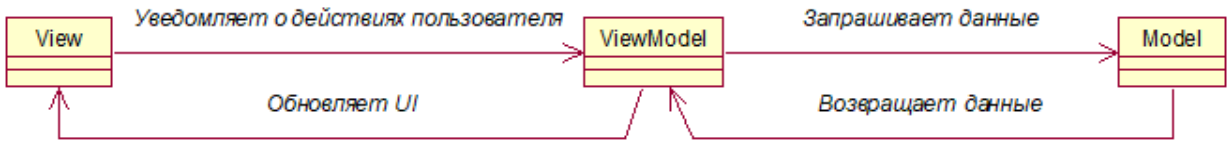

Рисунок 2 – Схема архитектуры MVVM

Model отвечает за хранение данных и бизнес-логику приложения. Он представляет собой набор классов и структур данных, которые отображают состояние приложения.

View отображает данные, полученные из модели. Она представляет собой набор пользовательских интерфейсов, которые взаимодействуют с пользователем.

ViewModel является посредником между Model и View. Он получает данные из Model и обрабатывает их, чтобы они были готовы к отображению в

View. ViewModel также отвечает за управление состоянием View.

Clean Architecture, с другой стороны, помогает разделить приложение на слои, которые обеспечивают независимость компонентов и упрощают тестирование [5]. В Clean Architecture приложение делится на 3 слоя: Presentation, Domain, Data.

Presentation отвечает за пользовательский интерфейс и взаимодействие с пользователем. Он связан с ViewModel и представляет собой набор экранов и компонентов пользовательского интерфейса.

Domain слой содержит бизнес-логику приложения. Он представляет собой набор классов и структур, которые описывают предметную область приложения и определяют правила его работы.

Data слой отвечает за доступ к данным. Он связан с Model и представляет собой набор классов и структур, которые обеспечивают получение и сохранение данных.

Сочетание архитектуры MVVM и Clean Architecture отображено на рисунке 3.

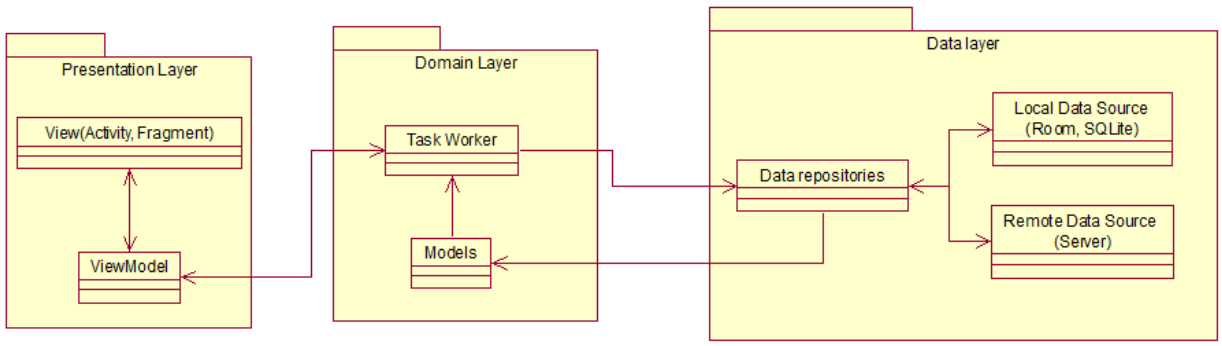

Рисунок 3 – Схема сочетания архитектуры MVVM и Clean Architecture

Такое сочетание позволяет создавать гибкие и расширяемые мобильные приложения, которые легко поддерживать и модифицировать.

Выводы

Одной из ключевых задач при создании приложения заметок было обеспечение максимальной удобности и простоты использования. Для этого

была спроектирована удобная и интуитивно понятная пользовательская интерфейсная часть приложения, которая позволяет быстро создавать и редактировать заметки. Также была разработана функциональность для сохранения заметок и синхронизации их между устройствами пользователя.

В процессе разработки приложения были использованы современные технологии обработки данных и хранения информации. В частности, для хранения заметок была использована база данных PostgreSQL, которая обеспечивает быстрое и надежное хранение данных.

В результате выполнения задач было создано качественное и удобное в использовании приложение заметок для операционной системы Android, которое обладает высокой степенью функциональности и надежности. Разработанный программный продукт может быть успешно использован пользователями для ведения заметок и организации своей рабочей деятельности.

Библиографический список:

1. Рындюк В. А. Современные задачи прикладной криптографии / В. А. Рындюк, А. А. Джамолуев, И. М. Султыгов. – Текст: непосредственный // Университетские чтения – 2020: Материалы научно-методических чтений ПГУ, Пятигорск, 09–10 января 2020 года. – Пятигорск: Пятигорский государственный университет, 2020. – С. 61-67. – Текст: непосредственный.

2. What Is PostgreSQL? // postgresql.org – postgresql: [сайт]. – URL: https://www.postgresql.org/docs/current/intro-whatis.html (дата обращения: 04.05.2022). – Режим доступа: сеть Интернет. – Текст: электронный.

3. Руководство по работе с фреймворком Kotlin Exposed // habr.com – habr: [сайт]. – URL: https://habr.com/ru/companies/otus/articles/555134/ (дата обращения: 04.05.2022). – Режим доступа: сеть Интернет. – Текст: электронный.

4. Introduction to HikariCP // baeldung.com – baeldung: [сайт]. – URL: https://www.baeldung.com/hikaricp (дата обращения: 04.05.2022). – Режим доступа: сеть Интернет. – Текст: электронный.

5. Волков А. А. Методы автоматизированного формирования

электронных документов в информационных системах / А. А. Волков, Н. А. Вехтева, М. А. Иванов // The World of Science Without Borders, 11 февраля 2022 года, 2022. – Р. 360-362. – Текст: непосредственный.