МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»
(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)

УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ДОПУСКАЕТСЯ К ЗАЩИТЕ
Заведующий отделением №3
Университетского колледжа
информационных технологий
_____ И.Г. Дзюба
«_____» _____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (ДИПЛОМНЫЙ ПРОЕКТ)

на тему: Разработка мобильного клиента, работающего под управлением операционной системы Android (на примере сайта www.hramalnevskogo.ru)

студента группы 090207-9о-20/2
специальности 09.02.07 Информационные системы и программирование
Асылбек уулу Бакыта

Студент _____ Б. Асылбек уулу
Руководитель _____ П.В. Миркитанов

Дата защиты «____» _____ 2024 г.
Оценка: _____
Председатель ГЭК _____ А.С. Шипилов

Москва
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных условиях мобильное оборудование становится неотъемлемым элементом нашей жизни, предоставляя доступ к разнообразным информациям и сервисам в удобном виде. Одним из важных направлений в разработке мобильных приложений является создание клиентов для различных веб-сайтов с целью улучшения доступности и комфорта использования онлайн-ресурсов.

В целом, мобильные приложения играют значительную роль в повседневной жизни людей, улучшая их эффективность, доступ к информации и удобство использования технологий.

Разработка мобильного клиента для сайта позволит расширить его аудиторию и обеспечить быстрый и удобный доступ к информации о храме, расписанию богослужений, событиях, истории храма, что повысит удобство использования ресурса на платформе Android.

Мобильное приложение позволит пользователям получить доступ к базовой информации о Храме даже без постоянного подключения к Интернету. Кэширование данных также способствует повышению скорости загрузки, что обеспечит лучшую производительность приложения.

Разработка мобильного клиента под операционную систему Android для сайта Храма Александра Невского будет способствовать повышению узнаваемости и привлечению новых посетителей, что важно для распространения информации о мероприятиях и деятельности храма. Также оно демонстрирует использование современных технологий и подходов к цифровому присутствию, что может улучшить восприятие организации храма в глазах посетителей.

Таким образом, разработка мобильного клиента под управлением операционной системы Android для сайта Храма Александра Невского представляется как целесообразное и перспективное решение для

совершенствования информационного пространства и обслуживания пользователей храма.

Актуальность темы выпускной квалификационной работы имеет целесообразность для обеспечения удобного доступа к информации о храме, расписанию богослужений, новостях и событиях. Такой мобильный клиент может облегчить взаимодействие прихода со своими прихожанами, обеспечивая им удобный и быстрый доступ к необходимой информации о храме и его деятельности. Также это может способствовать цифровизации церковных процессов и облегчению взаимодействия прихода с верующими.

Объектом выпускной квалификационной работы (далее ВКР) является мобильный клиент под управлением операционной системы Android.

Предметом выпускной квалификационной работы являются разработка мобильного приложения для операционной системы Android, которое будет представлять собой клиент на примере сайта www.hramalnevskogo.ru и может быть интегрировано в единую информационную систему прихода Храма Александра Невского.

Цели:

1) Создание удобного и интуитивно понятного мобильного клиента для пользователей операционной системы Android.

2) Предоставление доступа к информации о Храме Александра Невского через мобильное приложение.

Задачи:

1) Разработать дизайн и интерфейс мобильного клиента, опираясь на веб-сайт Храма Александра Невского.

2) Интегрировать функционал для просмотра информации о новостях, расписании служб, контактной информации и других разделов, представленных на сайте храма.

3) Оптимизировать производительность приложения для плавной работы на устройствах под управлением операционной системы Android.

4) Провести тестирование приложения на различных устройствах, чтобы гарантировать стабильную работу и отзывчивость.

Теоретическое значение данного дипломного исследования состоит в изучении современных методов разработки мобильных приложений для Android и основных принципов интеграции приложений с веб-сайтами.

Практическая значимость проекта:

1) Улучшение доступности информации: Мобильный клиент позволит увеличить доступность информации о храме для широкой аудитории пользователей мобильных устройств, что способствует привлечению новых посетителей и повышению осведомленности об объекте.

2) Удобство пользования: Приложение обеспечивает удобный и простой способ получения актуальной информации о храме, его истории, событиях и расписании богослужений, что способствует повышению интереса и участия пользователей.

3) Технологический прогресс: Разработка мобильного приложения демонстрирует использование современных технологий для улучшения взаимодействия организации с пользователем, что актуально в современном мире цифровизации.

Круг рассматриваемых проблем могут включать:

1) Взаимодействие с веб-сервером для получения данных с сайта.

2) Разработка пользовательского интерфейса, учитывающего особенности мобильных устройств.

3) Адаптация контента сайта для удобного отображения на различных размерах экранов мобильных устройств.

4) Тестирование приложения на различных устройствах с разными версиями Android для обеспечения совместимости.

Структура дипломного проекта:

1) Титульный лист.

2) Содержание.

3) Введение.
4) Теоретическая часть.
5) Практическая часть.
6) Заключение.
7) Список используемых источников.
8) Приложения.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исследование предметной области

1.1.1 Описание терминологии предметной области

Мобильный клиент – это мобильное приложение, которое позволяет взаимодействовать с удаленным сервером или службой. Он обеспечивает пользовательский интерфейс для доступа к функциональности, предоставляемой сервером, и обрабатывает данные, получаемые от сервера.

Мобильное приложение – это программа, разработанная для работы и установки на мобильных устройствах. Мобильные приложения обычно предлагают определенный набор функций, которые помогают пользователям выполнять различные задачи, от игр до бизнес-приложений и социальных сетей.

Веб-сервис (или сервер) – это программное обеспечение, которое доступно через интернет и обычно предоставляет определенную функциональность или данные другим программам или пользователям. Веб-сервисы используют стандартные протоколы взаимодействия, такие как HTTP, для передачи данных и выполнения действий. Они широко используются для интеграции различных приложений и систем, обмена информацией между ними и создания распределенных приложений

Эмулятор – это программа или устройство, которое позволяет одной системе (называемой хозяйской) эмулировать работу другой системы (называемой гостевой). Эмуляторы обычно используются для запуска программ или игр, разработанных для одной платформы, на другой платформе, которая может быть совершенно разной по архитектуре или операционной системе.

Паттерны проектирования – это хорошо известные, четко определенные решения распространенных архитектурных проблем при разработке программного обеспечения. Это шаблоны, которые можно использовать для

решения повторяющихся проблем, и они помогают разработчикам создавать более удобный в обслуживании, гибкий и многократно используемый код.

Сайт – это виртуальная площадка в интернете, которая состоит из веб-страниц и других мультимедийных контентов, доступных через специальный адрес (URL). Сайты используются для размещения информации, предоставления услуг, продажи товаров, общения и многого другого. С помощью браузера пользователи могут просматривать сайты, взаимодействовать с контентом и выполнять различные действия онлайн.

Мультиплатформенность (или кроссплатформенность) – способность программного обеспечения, приложений или игр работать на различных платформах или операционных системах без значительных изменений. Это означает, что одно приложение может быть запущено на нескольких различных устройствах или операционных системах, таких как Windows, macOS, iOS, Android и т. д.

1.1.2 Виды мобильных клиентов

Распространенные виды мобильных клиентов включают в себя следующее:

1) Нативные мобильные клиенты – это приложения, разработанные специально для определенной платформы (iOS, Android и т.п.). Нативные приложения обычно имеют более высокую производительность и более гладкий пользовательский интерфейс, чем гибридные или веб-приложения.

2) Гибридные мобильные клиенты – это приложения, которые более универсальны и могут быть запущены на различных платформах, но их производительность обычно ниже, чем у нативных приложений.

3) Веб-мобильные клиенты – это приложения, которые запускаются в веб-браузере устройства и доступны через интернет. Они обычно разработаны с использованием веб-технологий и могут быть доступны на

различных платформах без необходимости установки на устройство. Веб-приложения могут быть менее производительными и иметь ограниченный доступ к некоторым функциям устройства, но они обычно проще в разработке и обновлении.

Нативные приложения имеют такие преимущества как:

1) Более высокая производительность и быстрая работа приложения.

2) Доступ ко всем возможностям устройства, таким как камера, геолокация, датчики и другие.

3) Более интуитивные и привлекательные интерфейсы для пользователей.

Они же имеют следующие недостатки:

1) Необходимость разработки и поддержки двух отдельных версий приложения для разных операционных систем (iOS и Android).

2) Более длительный процесс разработки и обновления.

Веб-приложения для мобильных устройств имеют следующие преимущества:

1) Обеспечивают доступ к важным сервисам и информации в любое время и в любом месте, где есть интернет.

2) Работают на различных операционных системах без необходимости установки приложений.

3) Изменения и обновления в веб-приложении могут быть сразу доступны всем пользователям без необходимости загрузки обновлений из магазина приложений.

4) Меньшие затраты на разработку и поддержку приложения.

Данный вариант имеет такие недостатки как:

1) Мобильные веб-приложения могут иметь худшую производительность по сравнению с нативными приложениями из-за различий в оптимизации и доступу к аппаратным ресурсам устройства.

2) Для работы мобильных веб-клиентов требуется стабильный интернет, что может ограничить доступность в автономном режиме.

3) Некоторые функциональные возможности могут быть ограничены или недоступны в мобильных веб-приложениях.

Гибридные мобильные приложения имеют такие преимущества как:

1) Разработчики могут создать приложение для нескольких платформ, избегая необходимости полной переработки для каждой платформы.

2) Пользователи могут получить доступ к приложению с различных устройств и операционных систем, что повышает доступность и расширяет аудиторию.

Однако у них есть и недостатки:

1) Производительность может быть ниже, чем у нативных приложений.

2) Ограниченный доступ к некоторым функциям устройства.

1.1.3 Преимущества использования мобильных клиентов

Использование мобильных клиентов имеет несколько преимуществ.

Они позволяют получить доступ к сервисам и информации прямо с мобильного устройства в любое время и в любом месте.

Приложения обычно разрабатываются с учетом удобства использования на сенсорных экранах, что делает их более интуитивно понятными для пользователей.

Мобильные клиенты могут отправлять уведомления пользователю, чтобы информировать о важных событиях, новостях или обновлениях в реальном времени.

Некоторые приложения позволяют сохранять данные локально и работать даже без доступа к интернету, что может быть полезно при нестабильном соединении.

Они могут использовать функциональные возможности устройства, такие как камера, геолокация, датчики движения и другие, для более удобного и полезного опыта.

Пользователи могут настраивать их под свои потребности и предпочтения, что делает его использование более персонализированным.

Запуск мобильного приложения обычно занимает всего несколько секунд, что позволяет быстро получить необходимую информацию или выполнить нужное действие.

1.1.4 Уникальность и актуальность разрабатываемого продукта

Диплом на тему «Разработка мобильного клиента, работающего под управлением операционной системы Android (на примере сайта www.hramalnevskogo.ru)» является уникальным в своем роде, так как он разрабатывается специально для храма Александра Невского, что позволит создать уникальный продукт, учитывающий особенности и потребности данной религиозной общины.

Создание мобильного приложения для указанного храма подчеркивает важность использования современных информационных технологий для распространения традиционных духовных ценностей и облегчения доступа к информации для прихожан. К тому же выбор операционной системы Android для реализации данного проекта обусловлен ее широким распространением среди пользователей мобильных устройств, что позволит максимально повысить популяризацию православных ценностей среди верующих и неверующих.

Актуальность разработки подтверждается Указом Президента РФ о сохранении и укреплении традиционных российских духовно-нравственных ценностей.

Таким образом, разработка мобильного клиента для Храма Александра Невского будет не только уникальной, но и актуальной в контексте поддержки и развития православных ценностей в современном цифровом мире.

1.2 Анализ и выбор инструментальных средств

1.2.1 Язык программирования

Разрабатывать приложения для Android можно с использованием таких языков программирования как: C/C++, Python, Java, JavaScript, C#, Dart, Kotlin.

Kotlin – официальный язык разработки для Android, поддерживаемый Google. Этот современный язык стал предпочтительным выбором для создания приложений под эту операционную систему. Он обеспечивает безопасность типов, дополнительные возможности, позволяет писать более компактный код и легко интегрируется с Java. Google рекомендует использовать Kotlin для новых проектов на Android, подчеркивая его актуальность и эффективность.

Относительно других языков:

1) C/C++ можно использовать для создания мобильных приложений, но это требует больше усилий и знаний, что делает его не самым оптимальным выбором для стандартных приложений.

2) Приложения, разработанные на Python, могут быть менее эффективными из-за ограничений на скорость выполнения и не настолько хорошей поддержки инструментария для этой платформы.

3) Java требует большего объема кода для достижения тех же целей, чем, например, Kotlin или Dart.

4) Несмотря на то, что React Native позволяет разрабатывать кроссплатформенные приложения, использование JavaScript может привести к медленной работы и менее эффективному использованию ресурсов устройства Android из-за неоптимального взаимодействия с нативным кодом.

5) C# с .NET MAUI не является подходящим для создания приложений под Android из-за возможных проблем с производительностью и отсутствием оптимальной поддержки некоторых библиотек и функций.

6) Приложения, разработанные на Dart с использованием Flutter, могут не обеспечивать такую же высокую производительность, как нативные приложения, особенно в случае приложений, требующих интенсивной работы с графикой или процессором. Кроме того, они могут иметь больший размер из-за включенных в них библиотек, что может повлиять на время загрузки и использование памяти.

Таким образом, Kotlin представляет собой оптимальный выбор для разработки Android-приложений благодаря своей поддержке со стороны Google, современным возможностям и удобству разработки.

1.2.2 Библиотеки

Для разработки данного приложения можно использовать следующие библиотеки:

1) Для работы с JSON: kotlinx-serialization, Gson, Jackson, Moshi.
2) Для взаимодействия с веб-сервером: ktor-client, Retrofit, Volley.
3) Для выполнения асинхронных операций: kotlinx-coroutines, RxJava.
4) Для работы с базами данных: Room, Realm, SQLDelight.
5) Для инъекции зависимостей: Koin, Dagger 2, Hilt, Kodein.
6) Для реализации навигации между экранами: Navigation Compose, Decompose, Voyager, Appyx.
7) Для работы с изображениями: Coil, Glide, Picasso, Fresco.

Среди вышеперечисленных библиотек были выбраны следующие:

1) Основным критерием выбора kotlinx-serialization является его интеграция с Kotlin, а также он является мультиплатформенным, что позволяет его использовать в различных областях разработки, не ограничиваясь только android приложениями.

2) Ktor-client был выбран из-за своей простоты и легковесности. В отличие от Retrofit, который предоставляет более высокоуровневый API, ktor-client предоставляет минимальный клиент для сетевых запросов, что делает его хорошим выбором для простых приложений. Также он имеет все те же преимущества, что и предыдущая библиотека.

3) Room обеспечивает удобный API для работы с базами данных SQLite на Android. Он автоматически генерирует часть кода и обеспечивает проверку SQL запросов на этапе компиляции, что обеспечивает безопасность и удобство использования.

4) Kotlinx-coroutines обеспечивает легковесную асинхронную обработку в Kotlin приложениях. Он обеспечивает простой и понятный способ управления асинхронными операциями, что делает его предпочтительным выбором. Более того, данная технология обладает теми же преимуществами, что и библиотека, упомянутая в первом пункте.

5) Koin был выбран из-за своей простоты и легковесности. Он предоставляет простой DSL для создания и конфигурирования зависимостей в приложении, что делает его хорошим выбором для небольших проектов. Также вы можете применять его в различных сферах разработки, а не только ограничиваться созданием приложений для Android.

6) Navigation Compose предоставляет декларативный способ управления навигацией в приложении с помощью Jetpack Compose, а также эта библиотека поддерживается Google. Он обеспечивает простой и понятный способ определения навигационного графа и переходов между экранами.

7) Coil был выбран из-за своей простоты и легковесности. Он предоставляет удобный API для загрузки и отображения изображений в приложении, а также обеспечивает поддержку кэширования и обработки изображений.

1.2.3 Среда разработки

1) Android Studio (AS) – Официальная интегрированная среда разработки (IDE) от Google. Она предоставляет широкий набор инструментов и функций для разработки приложений на Kotlin под Android.

2) IntelliJ IDEA – это популярная Java IDE, которая также поддерживает разработку приложений на Kotlin. Для создания приложений под упомянутую ОС можно установить плагин Android Development Tools (ADT).

3) Visual Studio Code – это легковесная и гибкая среда разработки от Microsoft, которая также поддерживает разработку на Kotlin. Для работы с проектами Android в VS Code можно установить необходимые расширения и плагины.

AS был выбран для разработки Android приложения из-за:

1) Официальной поддержки Google, которая разработала его специально для этих целей.

2) Интеграции с Android SDK, а также предлагает богатый набор инструментов, включая эмуляторы и отладчики.

3) Возможности использовать Java, Kotlin и C++. Kotlin особенно хорошо интегрирован.

4) Доступности к множеству библиотек, плагинов и инструментов для улучшения процесса разработки.

1.2.4 Инструменты тестирования

Исходя из уже выбранных библиотек для разработки клиента были отобраны следующие инструменты и библиотеки:

1) Koin-Test позволяет эффективно проводить unit-тесты кода, использующего Koin DI.

2) JUnit и Kotlinx-coroutines-test являются стандартными инструментами для написания unit-тестов в Kotlin, которые также предоставляют расширения для тестирования корутин.

Вышеперечисленные инструменты имеют следующие преимущества:

1) Они предоставляют удобные средства для написания тестов, а также детальную документацию, что делает процесс тестирования более простым и понятным.

2) Поддержка языка Kotlin.

3) Надежность и стабильность помогают обеспечить качество продукта и уверенность в его работоспособности.

1.2.5 Система контроля версий

Git и Subversion – это системы контроля версий (Version Control Systems, VCS) для управления изменениями в коде и других файлов. Они отличаются следующими пунктами:

1) Git является распределенной системой контроля версий, где каждый разработчик имеет полную копию репозитория (включая историю изменений) на своем компьютере. Subversion же представляет собой централизованную систему, где репозиторий располагается на центральном сервере, и разработчики работают с копиями файлов, которые они извлекают и отправляют обратно на сервер.

2) В Git изменения сохраняются локально, и только после явной отправки (push) в удаленный репозиторий они становятся общедоступными. В случае Subversion изменения сразу попадают в центральный репозиторий после фиксации (commit).

3) Git обычно обеспечивает более высокую скорость работы за счет локального хранения всей истории изменений. Subversion, в свою очередь, может иметь некоторые задержки при работе с удаленным сервером.

4) Git очень гибок в ветвлении и объединении изменений, что делает его популярным среди разработчиков для экспериментов и разработки новых функций. В Subversion ветвление также доступно, но оно менее гибкое.

Была выбрана система контроля версий Git из-за его обширного сообщества разработчиков и множества инструментов поддержки. Это обеспечивает доступ к большому количеству примеров, способствующих более эффективному и быстрому обучению.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Спецификация программного изделия

Спецификация программного продукта «Разработка мобильного клиента, работающего под управлением операционной системы Android (на примере сайта www.hramalnevskogo.ru)»

Цель приложения: обеспечить удобный доступ пользователя к информации о храме, расписанию служб, новостях, а также предоставить простой и интуитивно понятный интерфейс для пользователей.

Функциональные требования:

1) Просмотр информации о храме, его истории и деятельности.
2) Просмотр расписания богослужений.
3) Отображение последних новостей и событий, связанных с храмом.
4) Синхронизация с действующим сайтом www.hramalnevskogo.ru для актуализации информации.

Нефункциональные требования:

1) Приложение должно быть разработано под управлением операционной системы Android версии 8.0 и выше.
2) Интерфейс приложения должен быть интуитивно понятным и удобным для использования.
3) Приложение должно быть стабильным, быстрым и безопасным.
4) Должна быть предусмотрена возможность работы в офлайн-режиме для доступа к ранее загруженным данным.
5) Приложение должно поддерживать различные разрешения экранов смартфонов и планшетов.

Техническая реализация:

1) Язык программирования: Kotlin.
2) Использование средств разработки Android Studio.

3) Для получения данных сайта www.hramalnevskogo.ru используется веб-сервис.

4) Дизайн приложения соответствует материальному дизайну Google.

План тестирования:

1) Тестирование функциональности просмотра информации о храме.

2) Тестирование синхронизации с сервером сайта.

3) Тестирование на различных устройствах и версиях ОС.

2.2 Проектирование программного изделия

2.2.1 Описание структуры программы

Программа представляет собой мобильное приложение, разработанное на языке программирования Kotlin с использованием архитектурного подхода Clean Architecture. Приложение разделено на несколько модулей: Domain, Data, и App.(Рисунок 1)

Domain Module содержит бизнес-логику приложения, включая модели данных и репозиторий для работы с статьями. Здесь также определен случай использования GetArticlesUseCase для получения списка статей.

Data Module отвечает за работу с данными приложения. Здесь определены сущности ArticleEntity для работы с базой данных Room, интерфейс IArticleDao для доступа к базе данных, класс ANTDb для работы с базой данных Room, DTO для сериализации данных, методы для преобразования данных между сущностями и моделями, а также репозиторий ArticleRepository для получения списка статей из сети и их сохранения в базе данных.

App Module представляет собой основной модуль приложения. Здесь определен класс App, который настраивает зависимости с помощью фреймворка Koin и инициализирует основной экран MainActivity. Также здесь содержится логика пользовательского интерфейса приложения, включая компоненты Composable для отображения данных и ViewModel ArticleVM для управления данными статей.

Кроме того, в программе определены вспомогательные классы и функции для обработки данных, отображения сообщений, управления навигацией, и отображения пользовательского интерфейса.

Общая структура программы следует принципам чистой архитектуры, где бизнес-логика отделена от деталей реализации, что обеспечивает удобство поддержки, расширения и тестирования приложения.

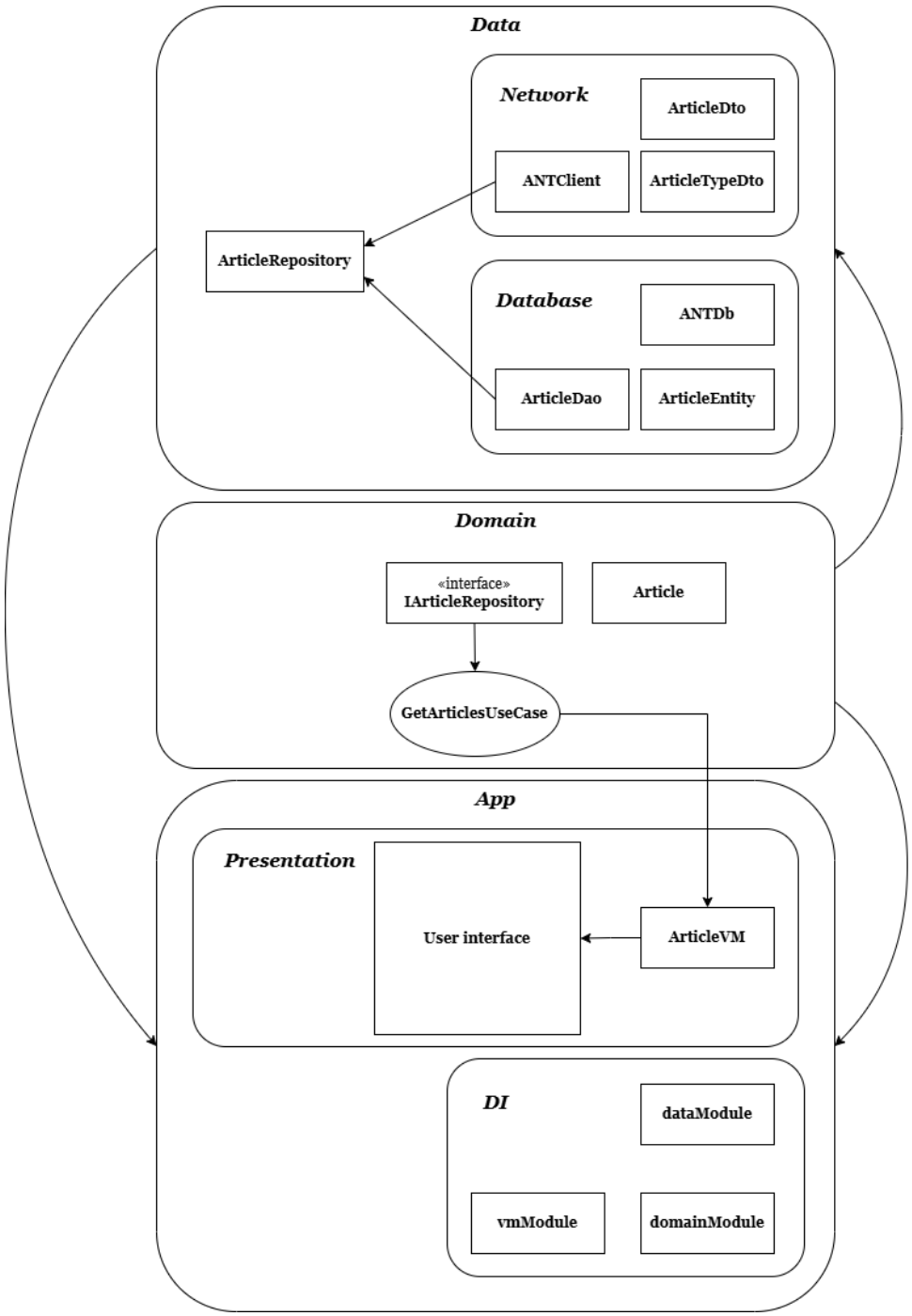

Рисунок 1 - диаграмма архитектуры программы

2.2.2 Описание и анализ библиотек

Библиотеки, использованные для разработки мобильного клиента, имеют следующие цели и особенности:

Kotlinx-serialization – это библиотека для сериализации данных в Kotlin. Она позволяет преобразовать данные из JSON в объекты Kotlin и обратно. В данном приложении она используется для работы с данными, получаемыми с сервера.

Ktor-client – это библиотека для создания HTTP-клиентов в Kotlin. Она позволяет выполнять сетевые запросы для получения данных с сервера. В данном случае ктор-клиент используется для получения информации о храме из веб-сервиса.

Kotlinx-coroutines - это библиотека для асинхронного программирования в Kotlin. Она позволяет управлять параллельными задачами и обрабатывать асинхронные операции без блокировки потоков. Это полезно для обработки сетевых запросов и длительных операций в мобильном приложении.

Room – это библиотека для работы с базой данных SQLite в приложениях на Android. Она предоставляет удобный интерфейс для доступа к данным, хранящимся локально на устройстве. Room используется в данном приложении для кэширования информации о храме.

Koin – это фреймворк для управления зависимостями в Android-приложениях. Он облегчает внедрение зависимостей и управление жизненным циклом зависимостей в приложении. Koin используется для внедрения зависимостей в различные компоненты приложения.

Navigation Compose – это библиотека для навигации между экранами в приложении, разработанная специально для Jetpack Compose, нового инструмента для создания пользовательских интерфейсов на Android. Она позволяет удобно управлять переходами между различными экранами и фрагментами приложения.

Coil – это библиотека для загрузки и отображения изображений в приложениях на Android. Она предоставляет простой и эффективный способ загрузки изображений из сети или локального хранилища и их отображения в

пользовательском интерфейсе. Coil используется для загрузки изображений, связанных с храмом в приложении.

2.2.3 Описание модели базы данных

В локальной базе данных мобильного приложения имеется таблица под названием «Статьи», которая хранит следующие данные:

1) Заголовок.
2) Описание.
3) Дата публикации.
4) Категория.
5) Содержание статьи, представленное в виде списка ссылок на изображения или контактную информацию, преобразованное в строку.
6) Уникальный идентификатор представляет собой первичный ключ с автоматическим увеличением значения значения при добавлении и имеющий числовой тип данных.

Таким образом, все атрибуты, за исключением последнего, имеют строковый тип данных. Каждая запись в этой сущности содержит статьи с указанными выше характеристиками.

2.2.4 Обеспечение информационной безопасности

Цель приложения заключается в обеспечении удобного доступа пользователя к информации о храме, и в данном случае информационная безопасность менее приоритетна, поскольку данные общедоступны и не содержат конфиденциальную информацию пользователей.

Однако были приняты следующие меры для обеспечения безопасности мобильного приложения:

1) Использование безопасного протокола HTTPS гарантирует защиту передачи данных между мобильным приложением и сервером.

2) Реализация механизмов обработки ошибок помогает предотвращать возможные уязвимости и сбои приложения, повышая его стабильность и защищенность.

3) Синхронизация данных через защищенные каналы и проверенные методы синхронизации обеспечивает безопасность передачи и хранения информации между мобильным приложением и сайтом.

2.3 Разработка программного изделия

2.3.1 Описание разработки программного продукта

Программный продукт разрабатывается с применением языка программирования Kotlin, который структурирован на три основных модуля.

Модуль Domain представляет собой бизнес-логику приложения. В нем реализованы классы Article и GetArticlesUseCase для работы с информацией о статьях и получения списка статей со статусами подключения. Также определен интерфейс IArticleRepository и перечисление ConnectionStatus для обработки статусов подключения.

Модуль Data отвечает за работу с данными. Здесь определены класс ArticleEntity для работы с данными статей, интерфейс IArticleDao для доступа к базе данных Room и классы ArticleRepository и ANTClient для работы с сетью и хранения данных статей.

Модуль App содержит логику отображения пользовательского интерфейса. Здесь определены класс ArticleVM для связи бизнес-логики и пользовательского интерфейса, а также различные composable функции для отображения экранов и элементов интерфейса.

Помимо основных модулей, также определены вспомогательные классы и функции для работы с данными, отображения сообщений и работы с навигацией.

Полный жизненный цикл приложения начинается с инициализации модулей с помощью фреймворка Koin в классе App, который также отвечает за создание базы данных и клиента для работы с сетью. Далее приложение запускается в MainActivity, где на экране отображается главный экран с возможностью навигации по различным разделам приложения.

Все компоненты приложения написаны с учетом современных стандартов и лучших практик разработки, обеспечивая быструю и эффективную работу приложения для пользователей.

2.3.2 Описание методов разработки и паттернов проектирования

Методология разработки программного продукта в данном проекте была основана на Agile подходе, который предполагал гибкость и итеративность разработки.

Начальный этап проекта включал определение основных целей, задач и требований. Здесь также проводилась разработка общего плана работы.

На этапе разработки происходило создание программного кода, тестирование функционала и внедрение новых компонентов.

Каждая итерация разработки включало в себя тестирование для выявления ошибок и дальнейшей их коррекции.

Архитектура продукта была рассчитана на масштабирование и добавление нового функционала без больших изменений в коде.

Таким образом, Agile методология помогала эффективно управлять процессом разработки программного продукта, быстро адаптироваться, обеспечивать качество и актуальность продукта.

Паттерны проектирования:

1) Repository Pattern: Использование репозитория IArticleRepository и ArticleRepository для абстрагирования работы с данными от источника данных. Этот паттерн позволяет легко заменять источник данных без изменения бизнес-логики.

2) Data Access Object (DAO): Порождающий паттерн, позволяющий абстрагировать доступ к данным из их хранения. В коде присутствует интерфейс IArticleDao, который определяет методы для взаимодействия с базой данных для работы с объектами ArticleEntity.

3) MVVM (Model-View-ViewModel) Pattern: Использование ArticleVM для управления данными и их отображения в пользовательском интерфейсе. ViewModel отвечает за обработку данных из Domain Module и их представление в Presentation Module.

4) Dependency Injection: Использование фреймворка Koin для внедрения зависимостей и управления жизненным циклом объектов. Это позволяет легко создавать и связывать объекты без явного создания зависимостей вручную.

5) Single Responsibility Principle: Классы и функции однозначно выделены по своей ответственности.

6) Clean Architecture: Разделение кода на слои (Domain, Data, Presentation) помогает избежать зависимостей между слоями и сохранить чистоту кода. Каждый слой отвечает за конкретную область приложения и обеспечивает легкость его тестирования и поддержки.

2.3.3 Этапы разработки программного кода

Модуль бизнес-логики:

1) Разработан класса Article для представления информации о статье.

2) Определение интерфейс IArticleRepository для работы с хранилищем статей.

3) Создание перечисления ConnectionStatus для определения статуса подключения.

4) Определены классы ActionStatus для представления различных состояний выполнения действия.

5) Разработан класс GetArticlesUseCase для получения списка статей.

Модуль данных:

1) Создана сущность ArticleEntity для хранения информации о статье в базе данных.

2) Определен интерфейс IArticleDao для взаимодействия с базой данных.

3) Разработан класс базы данных Room.

4) Реализованы DTO классы для сериализации данных.

5) Определены фунции преобразования между DTO, сущностью и доменной моделью.

6) Создан клиент для выполнения HTTP запросов.

7) Реализван ArticleRepository для получения списка статей из сети и сохранения их в базу данных.

8) Описаны зависимости базы данных, HTTP-клиента и хранилища данных в dataModule.

Модуль приложения:

1) Разработан domainModule для предоставления зависимостей для работы с бизнес-логикой.

2) Создан vmModule для предоставления зависимостей для ViewModels.

3) Разработан класс App для инициализации фреймворка Koin и настройки зависимостей.

4) Создан класс ArticleVM для работы с данными статей в пользовательском интерфейсе.

5) Определение функций для отображения основных элементов пользовательского интерфейса, таких как кнопки и другие.

6) Реализованы экраны для отображения различных категорий статей, контактной информации и другой информации.

Тестирование и отладка:

1) Проведено тестирование функциональности приложения на эмуляторе и устройстве.

2) Проведена отладка для выявления и исправления ошибок.

Оптимизация и доработка:

1) Проведена оптимизация работы приложения для повышения производительности.

2) Внесены доработки и улучшения пользовательского интерфейса и функциональности.

2.3.4 Возникшие трудности и их решения

В ходе разработки приложения были выявлены следующие проблемы:

1) Отсутствие оповещения пользователя об ошибке.

Решение: Для отображения сообщения об ошибке был создан метод displayMessage, который позволяет отобразить сообщение пользователю с указанным текстом ошибки.

2) Отсутствие связи с сервером.

Решение: Для обработки ситуации отсутствия связи с сервером был реализован механизм отображения сообщения об ошибке «Нет интернета». Это сообщение выводится пользователю в случае возникновения ошибки при загрузке данных с сервера.

3) Отображение контактов.

Решение: Для отображения контактной информации был создан экран «Контакты», где пользователь может найти информацию о телефоне, электронной почте, Telegram и VK. Также был добавлен функционал для непосредственного запуска соответствующих приложений при выборе соответствующего контакта.

4) Отображение списка статей.

Решение: Для отображения списка статей был создан компонент ContentList, который отображает карточки с информацией о статьях. При нажатии на карточку пользователь может увидеть детальную информацию о статье.

После реализации указанных решений и устранения проблем, приложение стало функциональным и готовым к использованию для пользователей.

2.4 Тестирование программного изделия

Получение списка статей.

Цель: Получить список статей для отображения на главном экране приложения.

Условия, подлежащие проверке:

1) Успешное получение данных из репозитория и API.
2) Отсутствие интернет-соединения.
3) Ошибка соединения с сервером.
4) Отсутствие данных.
5) Ошибка сериализации данных.
6) Неизвестная ошибка.

Шаги:

1) Открыть приложение.
2) Если данные успешно получены, отобразить список статей.
3) Если отсутствует интернет-соединение, отобразить сообщение «Нет интернета».
4) Если произошла ошибка соединения с сервером, отобразить сообщение «Ошибка соединения с сервером».
5) Если данные не были получены, отобразить сообщение «Нет данных».
6) Если произошла ошибка сериализации данных, отобразить сообщение «Сервер отправил некорректные данные».
7) В случае неизвестной ошибки, отобразить сообщение «Неизвестная ошибка».

Навигация по разделам приложения.

Цель: Переходить между различными разделами приложения.

Условия, подлежащие проверке:

1) Успешная навигация по разделам.

Шаги:

1) Открыть приложение.

2) Выбрать раздел для просмотра.

3) Перейти на страницу выбранного раздела.

4) Если навигация успешна, отобразить содержимое выбранного раздела.

Взаимодействие с контактной информацией.

Цель: Взаимодействие с контактными данными (телефон, электронная почта, мессенджеры) из приложения.

Условия, подлежащие проверке:

1) Успешное взаимодействие с контактной информацией.

Шаги:

1) Открыть приложение.

2) Перейти в раздел «Контакты».

3) Нажать на контактные данные (телефон, электронная почта, мессенджеры).

4) Если взаимодействие прошло успешно, выполнить соответствующее действие (позвонить, отправить сообщение и т.д.).

Просмотр подробной информации о событии.

Цель: Просмотреть подробную информацию конкретного события.

Условия, подлежащие проверке:

1) Успешный просмотр информации о событии.

Шаги:

1) Открыть приложение.

2) Перейти в раздел «Истории»

3) Выбрать событие для просмотра подробной информации.

4) Отобразить заголовок, описание, дату и фотографии, если они имеются.

5) Если данные успешно отображены, позволить пользователю ознакомиться с данными.

Просмотр расписания богослужений.

Цель: Просмотреть расписание богослужений и событий в приложении.

Условия, подлежащие проверке:

1) Успешный просмотр расписания.

Шаги:

1) Открыть приложение.

2) Выбрать раздел с расписанием богослужений.

3) Отобразить расписание богослужений.

4) Если информация успешно отображена, предоставить возможность ознакомиться с расписанием.

Добровольческая служба.

Цель: Получить информацию о добровольческой службе в приходе.

Условия, подлежащие проверке:

1) Успешное отображение информации о добровольческой службе.

Шаги:

1) Открыть приложение.

2) Перейти в раздел с информацией о добровольческой службе.

3) Отобразить информацию о добровольческой службе.

4) Если информация успешно отображена, предоставить возможность ознакомиться с необходимыми деталями.

После всего предыдущего было также проведено модульное тестирование с использованием Dependency Injection и ViewModel.

FakeRepository имитирует получение данных, а ExampleUnitTest проверяют корректную работу Koin DI и ArticleVM, который работает с FakeRepository для получения списка статей.

В методе setUp инициализируем тестовое окружение перед запуском каждого теста, устанавливаем UnconfinedTestDispatcher для Kotlin Coroutines и запускаем Koin с заданными модулями.

В тесте isLoading мы проверяем, что при запуске приложения статус загрузки устанавливается на LOADING и список статей пуст.

В тесте isCorrectData мы проверяем, что после задержки в 1 секунду данные успешно загружаются и список статей заполняется правильными данными.

Метод verifyKoinApp проверяет правильность настроенных модулей DI.

В методе tearDown происходит освобождение ресурсов после завершения каждого теста.

Таким образом, тесты покрывают основные сценарии работы с ViewModel и проверяют правильность работы Dependency Injection в приложении.

ЗАКЛЮЧЕНИЕ

Дипломная работа по теме «Разработка мобильного клиента, работающего под управлением операционной системы Android (на примере сайта www.hramalnevskogo.ru)» были успешно выполнены поставленные цели.

Результатами данного проекта, по разработке приложения, являются:

1) Удобный и интуитивно понятный мобильный клиент для пользователей Android.

2) Интерфейс, частично напоминающий дизайн сайта, включает удобные функции навигации и взаимодействия с контентом.

3) Производительность приложения оптимизирована для плавной работы на устройствах с операционной системой Android.

4) Мобильный клиент предоставляет пользователю удобный и быстрый доступ к информации о храме, благодаря успешной интеграции с функционалом сайта.

5) Проведено тестирование на различных устройствах для обеспечения стабильной работы.

Завершая настоящий дипломный проект, важно отметить следующие выводы:

1) Разработка мобильного приложения позволила значительно увеличить удобство использования и доступ к информации о Храме для пользователей.

2) Оптимизация производительности обеспечивает плавную работу приложения на различных устройствах под Android, что гарантирует удобство использования и повышает общую эффективность приложения.

3) Тестирование на различных устройствах подтвердило стабильную работу приложения.

Предложения по дальнейшему развитию проекта:

1) Реализовать дизайн, соответствующий стилю сайта Храма Александра Невского.

2) Разработать возможность получения уведомлений о предстоящих событиях, мероприятиях или новостях.

3) Внедрить функции онлайн-пожертвований для поддержки храма.

4) Регулярно обновлять приложение, исправлять ошибки и обеспечивать его совместимость с новыми версиями Android.

5) Убедиться, что приложение соответствует требованиям платформы Google Play для успешной публикации.

Таким образом, разработка мобильного клиента под управлением Android для сайта храма Александра Невского имеет существенную практическую значимость, обеспечивая удобный доступ к информации о храме, улучшая взаимодействие с посетителями и способствуя повышению интереса к объекту культурного наследия.

СПИСОК ИСПОЛЬЗ ИСТОЧНИКОВ

1) Анализ технологий разработки мобильных приложений и информационных систем на базе операционной системы Android: cyberleninka [Электронный ресурс]. – 2022. URL: https://cyberleninka.ru/article/n/analiz-tehnologiy-razrabotki-mobilnyh-prilozheniy-i-informatsionnyh-sistem-na-baze-operatsionnoy-sistemy-android (дата обращения 05.03.2024).

2) Введение в клиент-серверные мобильные разработки Бузыкова Ю.С., Зуфарова А.С. Москва 2023.

3) Вы за это заплатите! Цена Чистой Архитектуры. Часть 1: Habr [Электронный ресурс]. – 2024. URL: https://habr.com/ru/companies/vk/articles/801393/ (дата обращения 15.03.2024).

4) Вы за это заплатите! Цена Чистой Архитектуры. Часть 2: Habr [Электронный ресурс]. – 2024. URL: https://habr.com/ru/companies/vk/articles/801849/ (дата обращения 15.03.2024).

5) ГОСТ 2.105-95 (последняя редакция 2019 года) Единая система конструкторской документации. ОБЩИЕ ТРЕБОВАНИЯ К ТЕКСТОВЫМ ДОКУМЕНТАМ.

6) ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. ОТЧЁТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ.

7) К вопросу выбора оптимального языка программирования для разработки мобильного программирования: cyberleninka.ru [Электронный ресурс]. – 2022. URL: https://cyberleninka.ru/article/n/k-voprosu-vybora-optimalnogo-yazyka-programmirovaniya-dlya-razrabotki-mobilnogo-prilozheniya (дата обращения 02.03.2024)

8) Koin: Простой и легковесный фреймворк для внедрения зависимостей: Habr [Электронный ресурс]. – 2023. URL: https://habr.com/ru/companies/otus/articles/756124/ (дата обращения 20.03.2024)

9) Kotlin как язык программирования будущего: cyberleninka.ru [Электронный ресурс]. – 2022. URL: https://cyberleninka.ru/article/n/kotlin-kak-yazyk-programmirovaniya-buduschego (дата обращения 03.03.2024).

10) Мобильное приложение: romi.center [Электронный ресурс]. – 2024. URL: https://romi.center/ru/learning/glossary/mobile-app/ (дата обращения 29.05.2024)

11) Особенности языков программирования для Android приложений: cyberleninka.ru [Электронный ресурс]. – 2024. URL: https://cyberleninka.ru/article/n/osobennosti-yazykov-programmirovaniya-dlya-android-prilozheniy (дата обращения 02.03.2024).

12) Оценка и выбор наиболее подходящих инструментов разработки мобильных приложений: cyberleninka.ru [Электронный ресурс]. – 2023. URL: https://cyberleninka.ru/article/n/otsenka-i-vybor-naibolee-podhodyaschih-instrumentov-razrabotki-mobilnyh-prilozheniy (дата обращения 03.03.2024).

13) Паттерн проектирования MVVM, как один из способов написания «Чистого» кода в Android-приложении на Jetpack Compose: cyberleninka.ru [Электронный ресурс]. – 2023. URL: https://cyberleninka.ru/article/n/pattern-proektirovaniya-mvvm-kak-odin-iz-sposobov-napisaniya-chistogo-koda-v-android-prilozhenii-na-jetpack-compose (дата обращения 11.05.2024).

14) Перспективы использования языка Kotlin в программировании: cyberleninka.ru [Электронный ресурс]. – 2023. URL: https://cyberleninka.ru/article/n/perspektivy-ispolzovaniya-yazyka-kotlin-v-programmirovanii (дата обращения 04.03.2024)

15) Практическое применение языка программирования Kotlin: cyberleninka.ru [Электронный ресурс]. – 2023. URL: https://cyberleninka.ru/article/n/prakticheskoe-primenenie-yazyka-programmirovaniya-kotlin (дата обращения 05.03.2024).

16) Причины выбора Kotlin как основного языка мобильной разработки на Android: cyberleninka.ru [Электронный ресурс]. – 2023. URL:

https://cyberleninka.ru/article/n/prichiny-vybora-kotlin-kak-osnovnogo-yazyka-mobilnoy-razrabotki-na-android (дата обращения 05.03.2024).

17) Разница между GIT и SVN: GeeksforGeeks [Электронный ресурс] – 2024. URL: https://www.geeksforgeeks.org/difference-between-git-and-svn/ (дата обращения 28.05.2024).

18) Разработка базы данных для мобильного приложения на операционной системе Android: cyberleninka.ru [Электронный ресурс]. – 2022. URL: https://cyberleninka.ru/article/n/razrabotka-bazy-dannyh-dlya-mobilnogo-prilozheniya-na-operatsionnoy-sisteme-android (дата обращения 16.05.2024).

19) Чем новее, тем лучше: как мы с GSON на Kotlinx.Serialization переезжали: Habr [Электронный ресурс]. – 2023. URL: https://habr.com/ru/companies/tbank/articles/728928/ (дата обращения 15.05.2024).

20) Язык программирования Kotlin: cyberleninka.ru [Электронный ресурс]. – 2020. URL: https://cyberleninka.ru/article/n/yazyk-programmirovaniya-kotlin-1 (дата обращения 04.03.2024)

21) Guide to app architecture: Android Developers [Электронный ресурс]. – 2023. URL: https://developer.android.com/topic/architecture (дата обращения 16.03.2024).

22) Kotlin in Action, Second Edition/ Sebastian Aigner, Roman Elizarov, Svetlana Isakova, and Dmitry Jemerov, 2024. - 560 с.

ПРИЛОЖЕНИЕ А

Код проекта

```
//Domain Module
data class Article(
    val title: String,
    val description: String,
    val date: String,
    val articleType: String,
    val content: List<String>,
    val id: Int? = null
)
interface IArticleRepository {
    fun getList() : Flow<ActionStatus<Article>>
}
enum class ConnectionStatus {
    LOADING, SUCCESS, CONNECTION_ERROR, NO_INTERNET, NO_DATA,
SERIALIZATION_ERROR, UNKNOWN
}
sealed class ActionStatus<M>(val list: List<M>, val status: ConnectionStatus)
{
    class Loading<M>(list: List<M>) : ActionStatus<M>(list,
ConnectionStatus.LOADING)
    class Success<M>(list: List<M>) : ActionStatus<M>(list,
ConnectionStatus.SUCCESS)
    class Error<M>(list: List<M>, status : ConnectionStatus) :
ActionStatus<M>(list, status)
}
class GetArticlesUseCase(private val repository: IArticleRepository) {
    operator fun invoke(): Flow<ActionStatus<Article>> = repository.getList()
}
//Data Module
@Entity(tableName = "Articles")
data class ArticleEntity(
    val title: String,
    val description: String,
    val dateOrBanner: String,
    val catalog: String,
    val content: String,
    @PrimaryKey(autoGenerate = true)
    val id: Long? = null,
)
@Dao
interface IArticleDao {
    @Upsert
    suspend fun insert(article : ArticleEntity)
    @Query("SELECT * FROM Articles")
    suspend fun getAll() : List<ArticleEntity>
}
@Database(entities = [ArticleEntity::class], version = 4, exportSchema =
false)
abstract class ANTDb : RoomDatabase() {
    abstract fun articleDao() : IArticleDao
    companion object {
        const val DB_NAME = "ant.db"
    }
}
@Serializable
data class ArticleTypeDto(val id: Int, val name: String)
@Serializable
data class ArticleDto(
    val id: Long,
```

```
    val catalog: ArticleTypeDto,
    val title: String,
    val description: String,
    val dateOrBanner: String,
    val content: List<String>
)
fun ArticleDto.toEntity() = ArticleEntity(
    title, description, dateOrBanner, catalog.name,
Json.encodeToString(content), id
)
fun ArticleDto.toModel() = Article(title, description, dateOrBanner,
catalog.name, content)
fun ArticleEntity.toModel() = Article(
    title, description, dateOrBanner, catalog,
    Json.decodeFromString<List<String>>(content)
)
object ANTClient {
    val client = HttpClient(OkHttp) {
        expectSuccess = true
        install(Logging) {
            level = LogLevel.ALL
        }
        install(ContentNegotiation) {
            json(
                Json {
                    prettyPrint = true
                    isLenient = true
                }
            )
        }
        defaultRequest {
            url(DEFAULT_URL)
        }
    }
    private const val DEFAULT_URL = "https://ip:port/api/"
}
class ArticleRepository(
    private val articleDao: IArticleDao,
    private val client: HttpClient
) : IArticleRepository {
    override fun getList() : Flow<ActionStatus<Article>> = flow {
        val articleList = articleDao.getAll().map { it.toModel() }
        emit(Loading(articleList))
val articleDtoList = client.get("/api/v1/chapter").body<List<ArticleDto>?>()
        if(!articleDtoList.isNullOrEmpty()) {
            articleDtoList.forEach { articleDao.insert(it.toEntity()) }
            emit(Success(articleDtoList.map { it.toModel() }))
        } else emit(Error(articleList, NO_DATA))
    }.catch { ex ->
        val articleList = articleDao.getAll().map { it.toModel() }
        when(ex) {
is ServerResponseException -> emit(Error(articleList, CONNECTION_ERROR))
is RedirectResponseException -> emit(Error(articleList, CONNECTION_ERROR))
is HttpRequestTimeoutException -> emit(Error(articleList, CONNECTION_ERROR))
is ClientRequestException -> emit(Error(articleList, NO_INTERNET))
is JsonConvertException -> emit(Error(articleList, SERIALIZATION_ERROR))
else -> emit(Error(articleList, UNKNOWN))
        }
    }
}
val dataModule = module {
    single(createdAtStart = true) {
        Room.databaseBuilder(get(), ANTDb::class.java, ANTDb.DB_NAME).build()
    }
```

```
    single(createdAtStart = true) { ANTClient.client }
    single<IArticleRepository>() {
        ArticleRepository(get<ANTDb>().articleDao(), get())
    }
}
//Presentation Module
class ArticleVM(private val getArticlesUseCase: GetArticlesUseCase) :
ViewModel() {
    private val allArticles =
MutableStateFlow<ActionStatus<Article>>(Loading(emptyList()))
    val articlesSF = allArticles.asStateFlow()
    init {
        viewModelScope.launch {
            getArticlesUseCase().flowOn(Dispatchers.IO).collectLatest {
                allArticles.emit(it)
            }
        }
    }
}
val domainModule = module {
    factory() { GetArticlesUseCase(get()) }
}
val vmModule = module {
    viewModel() { ArticleVM(get()) }
}
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        startKoin {
            androidContext(this@App)
            androidLogger()
            modules(dataModule, domainModule, vmModule)
        }
    }
}
object Info {
    const val MAIN_TITLE = "Храм Александра Невского"
    const val MENU = "Меню"
    // error messages
    const val CONNECTION_ERROR = "Ошибка соединения с сервером"
    const val NO_INTERNET = "Нет интернета"
    const val NO_DATA = "Нет данных"
    const val SERIALIZATION_ERROR = "Сервер отправил некорректные данные"
    const val UNKNOWN = "Неизвестная ошибка"
    // screen routes
    const val MAIN = "Главная страница"
    const val PARISH_LIFE = "Приходская жизнь"
    const val SCHEDULE = "Расписание Богослужений"
    const val SPIRITUAL_TALKS = "Духовные беседы"
    const val YOUTH_CLUB = "Молодежный клуб"
    const val PRIESTHOOD = "Священство"
    const val ADVICES = "Советы священника"
    const val HISTORY = "История"
    const val SACRAMENTS = "Требы"
    const val CONTACTS = "Контакты"
    const val INFORMATION = "Информация"
    const val VOLUNTEERISM = "Приходская добровольческая служба"
    // URLs
const val SPIRITUAL_TALKS_URL = "https://hramalnevskogo.ru/page40967215.html"
const val INFORMATION_URL = "https://hramalnevskogo.ru/page42533272.html"
    // other
    const val PHONE = "Телефон"
    const val EMAIL = "Электронная почта"
    const val TELEGRAM = "Телеграм"
```

```
    const val VK = "VK"
}
fun Context.displayMessage(
    message : String, duration : Int = Toast.LENGTH_SHORT
) = Toast.makeText(this, message, duration).show()
fun ConnectionStatus.getMessage() = when(this) {
    CONNECTION_ERROR -> Info.CONNECTION_ERROR
    NO_INTERNET -> Info.NO_INTERNET
    NO_DATA -> Info.NO_DATA
    SERIALIZATION_ERROR -> Info.SERIALIZATION_ERROR
    UNKNOWN -> Info.UNKNOWN
    else -> ""
}
fun ConnectionStatus.isError() = (this != SUCCESS) && (this != LOADING)
fun List<String>.getNotNull(index: Int): String =
this.getOrNull(index).toString()
// Aliases for convenience and simplification
typealias SAction = (String) -> Unit
typealias Action = () -> Unit
@Composable
fun FredText(text: String, modifier: Modifier = Modifier, textAlign:
TextAlign = TextAlign.Justify) {
Text(text, modifier, fontFamily = FontFamily.Serif, textAlign = textAlign)
}
@Composable
fun FredTitle(text: String) {
    Text(
        text,
        Modifier.fillMaxWidth(),
        fontWeight = FontWeight.SemiBold,
        fontFamily = FontFamily.Serif,
        textAlign = TextAlign.Center
    )
}
@Composable
fun FredTButton(onClick: Action, text: String, modifier: Modifier = Modifier)
{
    TextButton(onClick, modifier) {
        FredText(text, Modifier.wrapContentSize())
    }
}
@Composable
fun FredIconButton(onClick: Action, icon: ImageVector, description: String,
modifier: Modifier = Modifier) {
    IconButton(onClick, modifier) {
        Icon(icon, description)
    }
}
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun FredTopAppBar(openDrawer: Action) {
    TopAppBar(
title = { FredText(Info.MAIN_TITLE) },
Modifier.fillMaxWidth().border(1.dp, MaterialTheme.colorScheme.primary),
navigationIcon = { FredIconButton(openDrawer, Icons.Outlined.Menu, Info.MENU)
}
    )
}
@Composable
fun FredNavigationDrawerItem(text: String, selected: Boolean, onClick:
Action) {
    NavigationDrawerItem(
        label = { FredText(text) },
        selected,
```

```
        onClick,
        Modifier.fillMaxWidth(),
        shape = MaterialTheme.shapes.small,
        colors = NavigationDrawerItemDefaults.colors()
    )
}
@Composable
fun FredFloatingActionButton(onClick: Action, icon: ImageVector) {
    FloatingActionButton(
        onClick = onClick,
        modifier = Modifier.border(1.dp, MaterialTheme.colorScheme.primary,
MaterialTheme.shapes.medium),
        shape = MaterialTheme.shapes.medium
    ) {
        Icon(icon, Info.SCHEDULE)
    }
}
@Composable
fun FredCard(onClick: Action, uri: String?, title: String, date: String,
modifier: Modifier = Modifier) {
    Card(
        onClick,
        modifier.border(2.dp, MaterialTheme.colorScheme.primary,
MaterialTheme.shapes.medium).padding(8.dp),
        shape = MaterialTheme.shapes.small,
        colors = CardDefaults.outlinedCardColors()
    ) {
        if(!uri.isNullOrBlank()) AsyncImage(model = uri.toUri(),
contentDescription = title, modifier =
Modifier.fillMaxWidth().height(250.dp))
        Spacer(Modifier.height(4.dp))
        Text(title,
Modifier.align(Alignment.CenterHorizontally).padding(horizontal = 4.dp),
fontFamily = FontFamily.Serif, textAlign = TextAlign.Center, overflow =
TextOverflow.Ellipsis, maxLines = 1)
        Spacer(Modifier.height(4.dp))
        FredText(date, modifier = Modifier.align(Alignment.End))
    }
}
@Composable
fun ImageSlider(article: Article, modifier: Modifier = Modifier) {
    LazyRow(modifier, Arrangement.SpaceEvenly, Alignment.CenterVertically) {
        items(article.content) { photo ->
            AsyncImage(model = photo.toUri(), contentDescription =
article.title, Modifier.height(300.dp).border(2.dp,
MaterialTheme.colorScheme.background))
        }
    }
}
@Composable
fun ListItemDetails(article: Article, modifier: Modifier = Modifier) {
    Column(modifier, Arrangement.Center, Alignment.CenterHorizontally) {
        FredTitle(article.articleType)
        Spacer(Modifier.height(8.dp))
        FredTitle(article.title)
        Spacer(Modifier.height(4.dp))
        if(article.date.isNotBlank()) FredText(article.date, modifier =
Modifier.align(Alignment.End))
        Spacer(Modifier.height(4.dp))
        ImageSlider(article = article, Modifier.fillMaxWidth())
        Spacer(Modifier.height(8.dp))
        FredText(article.description)
    }
}
```

```
@Composable
fun ListItem(isShowDialog: (Boolean) -> Unit, article: Article) {
LazyColumn(Modifier.fillMaxSize().background(MaterialTheme.colorScheme.backgr
ound)) {
item { ListItemDetails(article, Modifier.wrapContentSize().padding(8.dp)) }
        item {
            Box(Modifier.fillMaxWidth().wrapContentHeight()) {
                FredIconButton(
                    onClick = { isShowDialog(false) }, icon =
Icons.Default.Close, description = article.title,
                    modifier =
Modifier.align(Alignment.BottomCenter).padding(8.dp).border(2.dp,
MaterialTheme.colorScheme.error, MaterialTheme.shapes.medium)
                )
            }
        }
    }
}
@Composable
fun ContentList(state: ActionStatus<Article>, condition: (Article) ->
Boolean) {
    var isShowDialog by rememberSaveable { mutableStateOf(false) }
    var articleIndex by rememberSaveable { mutableIntStateOf(0) }
    LazyColumn(Modifier.fillMaxSize().padding(8.dp)) {
        itemsIndexed(state.list) { index, it ->
            if(!condition(it)) return@itemsIndexed
            FredCard(
                onClick = { articleIndex = index; isShowDialog = true },
                uri = it.content.getOrNull(0),
                title = it.title,
                date = it.date
            )
            Spacer(Modifier.height(4.dp))
        }
    }
    if(isShowDialog) {
        ListItem(isShowDialog = { isShowDialog = it }, article =
state.list[articleIndex])
    }
}
@Composable
fun ParishLife(vm: ArticleVM) {
    val state = vm.articlesSF.collectAsState().value
    ContentList(state) { it.articleType == Info.PARISH_LIFE }
}
@Composable
fun YouthClub(vm: ArticleVM) {
    val state = vm.articlesSF.collectAsState().value
    ContentList(state) { it.articleType == Info.YOUTH_CLUB }
}
@Composable
fun Advices(vm: ArticleVM) {
    val state = vm.articlesSF.collectAsState().value
    ContentList(state) { it.articleType == Info.ADVICES }
}
@Composable
fun History(vm: ArticleVM) {
    val state = vm.articlesSF.collectAsState().value
    ContentList(state) { it.articleType == Info.HISTORY }
}
@Composable
fun MainScreen(vm: ArticleVM) {
Column(Modifier.fillMaxSize().verticalScroll(rememberScrollState()).padding(8
.dp)) {
```

```
        vm.articlesSF.collectAsState().value.list.forEach {
            if(it.articleType != Info.MAIN) return@forEach
            ListItemDetails(it, Modifier.fillMaxWidth())
        }
    }
}
@Composable
fun Priesthood(vm: ArticleVM) {
Column(Modifier.fillMaxSize().verticalScroll(rememberScrollState()).padding(8
.dp)) {
        vm.articlesSF.collectAsState().value.list.forEach {
            if(it.articleType != Info.PRIESTHOOD) return@forEach
            ListItemDetails(it, Modifier.fillMaxWidth())
        }
    }
}
@Composable
fun Schedule(vm: ArticleVM) {
    Column(Modifier.fillMaxSize().verticalScroll(rememberScrollState())) {
        vm.articlesSF.collectAsState().value.list.forEach {
            if(it.articleType != Info.SCHEDULE) return@forEach
            AsyncImage(model = it.title, contentDescription = it.title,
modifier = Modifier.fillMaxWidth())
            AsyncImage(model = it.description, contentDescription =
it.description, modifier = Modifier.fillMaxWidth())
        }
    }
}
@Composable
fun Sacraments(vm: ArticleVM) {
Column(Modifier.fillMaxSize().verticalScroll(rememberScrollState()).padding(8
.dp), Arrangement.Center) {
        vm.articlesSF.collectAsState().value.list.forEach {
            if(it.articleType != Info.SACRAMENTS) return@forEach
            FredTitle(text = it.title)
            Spacer(modifier = Modifier.height(8.dp))
            FredText(text = it.description, modifier =
Modifier.fillMaxWidth(), TextAlign.Center)
        }
    }
}
@Composable
fun Contacts(vm: ArticleVM, openSomeApp: SAction) {
Column(Modifier.fillMaxSize().verticalScroll(rememberScrollState()).padding(8
.dp), horizontalAlignment = Alignment.CenterHorizontally) {
        val state = vm.articlesSF.collectAsState().value
        state.list.forEach {
            if(it.articleType != Info.CONTACTS) return@forEach
            AndroidView(factory = { context ->
if(state.status.isError()) context.displayMessage(state.status.getMessage())
                TextView(context).apply {
                    text = it.title
                    textSize = 24f
                    setTextColor(resources.getColor(R.color.black, null))
                    textAlignment = TextView.TEXT_ALIGNMENT_CENTER
                }
            }, Modifier.fillMaxWidth())
            Spacer(modifier = Modifier.height(8.dp))
            ContactsCard(contentList = it.content, openSomeApp)
        }
    }
}
@Composable
private fun ContactsCard(contentList: List<String>, openSomeApp: SAction) {
```

```
    Row(Modifier.fillMaxWidth().wrapContentHeight(), Arrangement.SpaceEvenly,
Alignment.CenterVertically) {
        FredIconButton({ openSomeApp(contentList.getNotNull(2)) },
Icons.Default.Phone, Info.PHONE)
        FredTButton({ openSomeApp(contentList.getNotNull(0)) },
Info.TELEGRAM)
        FredTButton({ openSomeApp(contentList.getNotNull(1)) }, Info.VK)
        FredIconButton({ openSomeApp(contentList.getNotNull(3)) },
Icons.Default.MailOutline, Info.EMAIL)
    }
}
@Composable
fun Volunteerism(vm: ArticleVM) {
    Column(modifier =
Modifier.fillMaxSize().verticalScroll(rememberScrollState()).padding(8.dp)) {
        vm.articlesSF.collectAsState().value.list.forEach {
            if(it.articleType != Info.VOLUNTEERISM) return@forEach
            ListItemDetails(article = it, Modifier.fillMaxWidth())
        }
    }
}
@Composable
fun MainNavHost(
    controller: NavHostController,
    navItems: List<String>,
    openSomeApp: SAction,
    modifier: Modifier = Modifier,
    articleVM: ArticleVM = koinViewModel()
) {
    Column(modifier) {
        NavHost(controller, startDestination = navItems[0]) {
            composable(navItems[0]) { MainScreen(articleVM) }
            composable(navItems[1]) { ParishLife(articleVM) }
            composable(navItems[2]) { Schedule(articleVM) }
            composable(navItems[3]) { Box(modifier) {
CircularProgressIndicator(Modifier.align(Alignment.Center)) } }
            composable(navItems[4]) { YouthClub(articleVM) }
            composable(navItems[5]) { Priesthood(articleVM) }
            composable(navItems[6]) { Advices(articleVM) }
            composable(navItems[7]) { History(articleVM) }
            composable(navItems[8]) { Sacraments(articleVM) }
            composable(navItems[9]) { Contacts(articleVM, openSomeApp) }
            composable(navItems[10]) { Box(modifier) {
LinearProgressIndicator(Modifier.align(Alignment.Center)) } }
            composable(navItems[11]) { Volunteerism(articleVM) }
        }
    }
}
@Composable
fun MainEntryPoint(navItems: List<String>, openSomeApp: SAction) {
    val controller = rememberNavController()
    val drawerState = rememberDrawerState(Closed)
    val scope = rememberCoroutineScope()
    var selectedItemIndex by rememberSaveable { mutableIntStateOf(0) }
    val navigateTo: (Int, String) -> Unit = { index, route ->
        controller.navigate(route)
        selectedItemIndex = index
        scope.launch { drawerState.close() }
    }
    ModalNavigationDrawer(
        drawerContent = {
            ModalDrawerSheet {
                Column(Modifier.fillMaxSize(), Arrangement.Center) {
                    navItems.forEachIndexed { index, route ->
```

```
                        FredNavigationDrawerItem(
                            text = route,
                            selected = index == selectedItemIndex,
                            onClick = {
                                when(index) {
                                    3 -> openSomeApp(Info.SPIRITUAL_TALKS_URL)
                                    10 -> openSomeApp(Info.INFORMATION_URL)
                                    else -> navigateTo(index, route)
                                }
                            }
                        )
                        Spacer(Modifier.height(2.dp))
                    }
                }
            }
        }, drawerState = drawerState) {
        val currentRoute =
controller.currentBackStackEntryAsState().value?.destination?.route
        Scaffold(
            Modifier.fillMaxSize(),
          topBar = { FredTopAppBar { scope.launch { drawerState.open() } } },
          floatingActionButton = { if(currentRoute != navItems[2])
FredFloatingActionButton({ navigateTo(2, navItems[2]) },
Icons.Outlined.CalendarMonth) },
        ) { padding ->
            MainNavHost(controller, navItems, openSomeApp,
Modifier.fillMaxSize().padding(padding))
        }
    }
}
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val navItems = listOf(Info.MAIN, Info.PARISH_LIFE, Info.SCHEDULE,
Info.SPIRITUAL_TALKS, Info.YOUTH_CLUB, Info.PRIESTHOOD, Info.ADVICES,
Info.HISTORY, Info.SACRAMENTS, Info.CONTACTS, Info.INFORMATION,
Info.VOLUNTEERISM)
        setContent {
            AlexanderNevskyTempleTheme {
Surface(Modifier.fillMaxSize(),color = MaterialTheme.colorScheme.background){
                    MainEntryPoint(navItems) { data ->
startActivity(Intent(Intent.ACTION_VIEW, Uri.parse(data))) }
                }
            }
        }
    }
}
```

Ссылки на проекты

FredNekrasov/ANT-Web-API: Веб-сервис для синхронизации данных мобильного приложения и сайта. (github.com)

FredNekrasov/ANT-Parser: Разработка парсера-клиента, который производит скраппинг данных с сайта и отправляет их на сервер (github.com)

FredNekrasov/ANT-Android-App: Разработка мобильного клиента для православного храма под управлением операционной системы Android. (github.com)

ПРИЛОЖЕНИЕ Б

Unit-тесты

```
class FakeRepository : IArticleRepository {
    override fun getList(): Flow<ActionStatus<Article>> = flow {
        emit(ActionStatus.Loading(emptyList()))
        val articleList = listOf(
Article("title1", "description1", "date1", "1", listOf("a", "1")),
Article("title2", "description2", "date2", "2", listOf("b", "2")),
Article("title3", "description3", "date3", "3", listOf("c", "3")),
Article("title4", "description4", "date4", "4", listOf("d", "4")),
Article("title5", "description5", "date5", "5", listOf("e", "5"))
        )
        emit(ActionStatus.Success(articleList))
    }
}

@OptIn(ExperimentalCoroutinesApi::class)
class ExampleUnitTest : KoinTest {
    private val fakeDataModule = module(createdAtStart = true) {
        singleOf(::FakeRepository) withOptions {
            bind<IArticleRepository>()
            createdAtStart()
        }
    }
    @Before
    fun setUp() {
        Dispatchers.setMain(UnconfinedTestDispatcher())
        startKoin {
            modules(fakeDataModule, domainModule, vmModule)
        }
    }
    private val articleVM: ArticleVM by inject<ArticleVM>()
    @Test
    fun isLoading() {
        val res = articleVM.articlesSF.value
        assertNotNull(res.status)
        assertEquals(ConnectionStatus.LOADING, res.status)
        assertNotNull(res.list)
        assertEquals(emptyList<Article>(), res.list)
    }
    @Test
    fun isCorrectData(): Unit = runTest {
        val res = articleVM.articlesSF.value
        delay(1000L)
        assertEquals(ConnectionStatus.SUCCESS, res.status)
        assertNotNull(res.list.firstOrNull())
        assertEquals("title1", res.list.firstOrNull()?.title)
        assertEquals(
        Article("title1", "description1", "date1", "1", listOf("a", "1")),
        res.list.firstOrNull()
        )
    }
    @Test
    fun verifyKoinApp() = getKoin().checkModules()
    @After
    fun tearDown() {
        Dispatchers.resetMain()
        stopKoin()
    }
}
```

ПРИЛОЖЕНИЕ В

Вывод команды git log

```
commit ddfe2d4cec57bbe314eb429a5dcfdfcabbb8d02e (HEAD -> master, origin/master)
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 26 23:26:24 2024 +0300
    creating unit tests for koin, use cases, and ViewModels

commit 49613cf834aca5f8bbbfbec157f93c94e5310373
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 26 23:24:08 2024 +0300
    creating a FakeRepository for unit testing

commit 6be214835df5776b91c8a8b6a251ec56b1598d77
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 26 23:23:24 2024 +0300
    adding dependencies for unit testing

commit 76e3107b73ecd4a45fd960c722dbb388db3026f8
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 25 22:40:51 2024 +0300
    adding comments in the libs.versions.toml

commit 94f3d062f4a8a4b1e708d830110e749de20bb184
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 25 22:35:53 2024 +0300
    adding comments in the app module

commit b0578998d52b9bcf03928d54cae0a299e3149634
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 25 22:35:19 2024 +0300
    adding comments in the data module

commit 1f916f383adbf773e71a54df6d0513fc499394aa
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 25 22:33:17 2024 +0300
    adding comments in the domain module

commit d60e94443028c63875d8867d6bd37da7bdb136db
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu May 23 13:21:04 2024 +0300
    refactoring the ArticleRepository

commit 0c0217f98319417cf63daca11ce2ce990ff430a2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu May 23 13:19:32 2024 +0300
    refactoring the IArticleDao

commit 59a7ac397dbc362c90c36d1ac33c5c3909677938
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu May 23 13:17:53 2024 +0300
    changing the version of the compose compiler

commit 52bb2e49a2bc868bf13303022b3478d0b2806d34
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu May 23 13:16:55 2024 +0300
    updating the kotlin and ksp versions

commit b2704023d3a66de337d5e4e3bc08e5e75f4bf4bb
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 15:02:21 2024 +0300
    refactoring the OnePageScreens.kt
```

```
commit 896c9164d93ae8dfb341462271d16956f592a953
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 12:24:02 2024 +0300
    refactoring the MainActivity and the MainEntryPoint

commit 56738590a659c157da0fea110ffdbbd4f840d4e8
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 10:51:58 2024 +0300
    renaming the Strings object to Info and refactoring it

commit df3228d5fd430b754bd48f8bea4d0273ca5e2e15
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 10:21:22 2024 +0300
    refactoring the MainNavHost

commit b9090efc933cc761d906512a7fe53b2370315e56
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 10:19:53 2024 +0300
    refactoring the MainEntryPoint

commit 055a169e24b09b0a4264a65a525c73bc6612dde5
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 10:19:23 2024 +0300
    refactoring the MainActivity

commit ec01a8286cc9266f0d29f808d31be83686907f79
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed May 22 10:15:03 2024 +0300
    removing an unnecessary extension method

commit 27c7eec2250030dae96d1a3e1ed4c3df8184e22b
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:27:00 2024 +0300
    refactoring the OnePageScreens.kt file

commit 89eef748e469fdfbf84f4f4ac56e8309dd23dbb5
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:26:15 2024 +0300
    refactoring the CustomViews.kt

commit f1972a3ac44593de3849c729c2242f283293e159
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:25:53 2024 +0300
    refactoring the MainNavHost

commit f768d12bb8370626b9fb0847b45783029a123910
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:25:16 2024 +0300
    refactoring the MainEntryPoint and deleting the MainScaffold.kt file

commit fc1f095a60bd214bde388897920e21529344c256
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:23:05 2024 +0300
    refactoring the MainActivity

commit 353881c776e6ea27e1d6e8e6f3e296b1e7be97f1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:22:22 2024 +0300
    adding new data to the Strings object

commit 69d8ebc096870d17bc7b699c1b8d499f49a83e4d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Tue May 21 23:21:09 2024 +0300
```

```
    adding some extension functions and custom data types

commit 162179aa1744e52854b9050200fa70c620a93c8a
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Mon May 20 19:34:07 2024 +0300
    refactoring the UI again. ImageSlider.kt and ListItem.kt

commit db6d8c3188d0adaf95e388125b7935a024c8a8c1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 22:42:09 2024 +0300
    refactoring the UI again. CustomViews.kt, ListItem.kt, OnePageScreens.kt

commit 1f06c591516bd42e7e2057673a440ad8f6a6c716
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:57:55 2024 +0300
    refactoring the ContentList.kt file

commit 3ab6c9ff8d0e2c2192b18003ed2fc44eec823cb6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:57:04 2024 +0300
    refactoring the ImageSlider.kt file

commit 33b0780a2b99110a2e42aea704a6191eb8844353
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:56:48 2024 +0300
    refactoring the OnePageScreens.kt file

commit bd62262e13023f3df6a7b3d69ed0a806132baf4d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:56:13 2024 +0300
    refactoring the ListItemDialog.kt file and renaming it to ListItem.kt

commit 7552de987ab123f24f08cf76622cbfe034189dc3
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:55:19 2024 +0300
    refactoring the ListItem.kt file and renaming it to ListItemDetails.kt

commit 17e6290673b69945e3115b48247f923f30ea019f
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 21:52:07 2024 +0300
    refactoring the CustomViews.kt file

commit e3d426936c699c486d4ac4b74bdda2f57d3a4014
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:07:36 2024 +0300
    refactoring the CustomViews.kt file

commit c567064e7ee0558726e4b53c7730398b3671824b
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:06:36 2024 +0300
    refactoring the MainNavHost

commit b0c65a084bf258199e36c2edf49f93fafc82e792
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:06:16 2024 +0300
    refactoring the MainScaffold

commit ddcff48056e933f001cc521a6d941e69d68cddc8
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:04:59 2024 +0300
    refactoring the MainEntryPoint

commit 1aafe511723b2338f8a64558c19b3b6455cca3d7
Author: FredNekrasov <frednekrasov49@gmail.com>
```

```
Date:   Sun May 19 16:04:16 2024 +0300
    refactoring the MainActivity

commit 2bf120d81c67e1a914bb0fe835d0d158b0ed3327
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:03:12 2024 +0300
    creating screens to display information about articles, divided by their
type

commit 86a2c0d45447273a2aba3188c31bd71741467040
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:01:05 2024 +0300
    creating a ContentList function to display a list of articles

commit bce485a6fc382ee805b83bfe85274934dc26d3d0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 16:00:43 2024 +0300
    creating a ListItemDialog function to display detailed information about
articles on the dialog screen

commit 4de69df41d9d48e1edbde37bd9fcac35dde13144
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 15:59:25 2024 +0300
    creating a ListItem function to display detailed information about articles

commit a0390877e6c3e5f3131071f445266a732e29d0a6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 15:58:16 2024 +0300
    creating an ImageSlider function to display a list of photo

commit a9fa8b5eddb84dd1ee0407e6623802d650ad04e1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 15:52:32 2024 +0300
    creating a Extensions.kt file with extension methods

commit 0100a0271eb2d6c8ce0dc3f2e733e12c046b2257
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 15:50:11 2024 +0300
    creating a Strings object with string data

commit d5a3955842dc06e9f53d935f30a6c8efadc48e71
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 14:35:54 2024 +0300
    renaming the VMModule.kt to the PresentationModule.kt

commit 2c91c96e15dd435931b72d2c5f6e7403ce4178d0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 14:35:14 2024 +0300
    refactoring the ArticleVM

commit fe7bcce727c455e736d57b72fcd2a3d2f11fb7f2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 14:29:20 2024 +0300
    refactoring the GetArticlesUseCase

commit 3931e5ecc9b539bcd7c84d628e5e9391e5142726
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 14:28:57 2024 +0300
    refactoring the ArticleRepository

commit 0afeffa310dbed035e93b93cc834db6943334894
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 19 14:28:26 2024 +0300
    refactoring the ANTDb class
```

```
commit 0f9e96944911565f82c8fb77069184b4812bf43d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:41:06 2024 +0300
    refactoring the ANTDb class

commit 2b410216aea0f9e5a28f91cb25682536bd55b614
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:40:28 2024 +0300
    refactoring the DataModule.kt

commit 828b2619d47abe9b82b5b662097ffd54ee23f8f2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:39:46 2024 +0300
    refactoring the GetArticlesUseCase

commit 0be3b9e3fecc47aaff963a5795b5c1df93c174a8
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:39:08 2024 +0300
    refactoring repositories

commit 62dd9cae6a1c05e1b47d5208e55c063afbcdad79
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:37:07 2024 +0300
    refactoring the ArticleMapper.kt

commit a00f8743e78a8fc11460bc20fcaaa8e7e0cc32cc
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:36:45 2024 +0300
    refactoring the Article model

commit 9bb52be72f70573dc5393dcfdb3c5ba468792141
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:35:23 2024 +0300
    refactoring the UseCaseModule.kt file and renaming it to DomainModule.kt

commit 618221dfc1e067e2a91d3fe27ff2743d8cd0f256
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:32:39 2024 +0300
    deleting an unused variable in the DIStrings object

commit 9798503d2bde740b8496003cd20f2d8fda9a03e6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 18 11:31:06 2024 +0300
    deleting unnecessary classes

commit 9eeb51b5a8e841553809aa846bffd427c6a77a07
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 19:22:28 2024 +0300
    updating the compose compiler from version 1.5.8 to version 1.5.13

commit 6d1147849accbb266ba7ec7bf8cb5eddbaade8c5
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 19:21:59 2024 +0300
    refactoring the App class

commit e4018d76207af57f4118ec5a26f78586440dd341
Merge: a47607c d822e25
Author: fred nekrasov <152185797+FredNekrasov@users.noreply.github.com>
Date:   Fri May 17 19:00:45 2024 +0300
    Merge pull request #3 from FredNekrasov/Data
    refactoring the data layer

commit d822e25ef2f58b9f95ec384bce8e323259760d1c (origin/Data)
```

```
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:10:29 2024 +0300
    creating the Data DI Module

commit 250734f0aec15a75c0ce50e5386780b1454d0a17
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:08:36 2024 +0300
    moving DIStrings from the app module to the data module and refactoring
vmModule

commit d7ed885f69741efdcd8c33e7b2e3f37d94a135c7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:05:22 2024 +0300
    refactoring repositories

commit 921c811da775a63a97c91f8249ddd9fc87f6711a
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:03:35 2024 +0300
    creating a HttpClient

commit ad52bb9d33ccc21fdfc83502779242cd0fb297b6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:01:55 2024 +0300
    creating the database class

commit 8d37ba04936033ee6def69f2a3deec22d72c8731
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:01:13 2024 +0300
    creating DAOs

commit d900c060c917a551f4758b2b38f68149b524f4a3
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 16:00:35 2024 +0300
    refactoring mappers

commit 38c77e6fece86a984ab57a4cca80739f8651ced4
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 15:59:56 2024 +0300
    creating entities

commit 6294357bf35c626e3222fcb5ce56084a46213ce5
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 15:58:48 2024 +0300
    refactoring DTOs

commit b4d2252150bcc68dbad4d64593bb3c7be4133d9d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 15:57:43 2024 +0300
    removing unnecessary classes/files

commit 5568ed369fd27d369bdf2d55dde55f825ccc985a
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri May 17 15:53:46 2024 +0300
    adding the ksp and kotlinx-serialization plugins. replacing retrofit and
converter-gson with ktor and kotlinx-serialization. adding room for data caching

commit a47607c71fc52e282f8e43e4476a9cdf1111e5d3
Merge: d209ef7 998e688
Author: fred nekrasov <152185797+FredNekrasov@users.noreply.github.com>
Date:   Sun May 12 15:18:27 2024 +0300
    Merge pull request #2 from FredNekrasov/appLogic
    completion of the model

commit 998e688a23426193f788bbd192a7d1a3597a8765
```

```
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 15:14:11 2024 +0300
    refactoring an App class

commit 2dddb8e2a1640952b1bac4edb0e1a82bfb2a5207
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 15:13:27 2024 +0300
    refactoring the CustomViews.kt

commit 26aaf0da714a6010cd0e09dd86cf0317288fab91
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 15:13:06 2024 +0300
    refactoring the MainEntryPoint

commit 6addd2ec06af225353120548602a136ee6a919b7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 15:12:02 2024 +0300
    refactoring the MainScaffold

commit 51afc8059a217f94372adee96c5ec1657ad6e882
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 15:10:07 2024 +0300
    creating the MainNavHost

commit 5e73ff7fcdf3e7443a9d628c21ab9c26cc7c1631
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 14:32:44 2024 +0300
    refactoring the MainActivity

commit 33986cdd5603146adca3b17ee52725a4b7fe56c0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 14:32:13 2024 +0300
    refactoring the MainEntryPoint

commit 5e83bb8ffa56a5b31fa09e0005772fa7632aa24e
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 14:30:17 2024 +0300
    creating the VM DI module

commit fb63e00cfc02469348ea44687156854c39af24ad
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 14:06:07 2024 +0300
    creating viewModels

commit 75183a86bef827ebfb3e30bb449a2352f66a1bc7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 13:57:13 2024 +0300
    refactoring repositories and use cases

commit d0efca3fd6c291b80adcb8d48d0c3651a218e29a
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 11:14:10 2024 +0300
    refactoring services/APIs

commit de61de317efe9ab89516e9e3a146d02051d69a42
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 11:12:00 2024 +0300
    refactoring repositories

commit a11c62a213360ba0bfe518098675ac3834a8d770
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun May 12 11:11:03 2024 +0300
    editing the ArticleDto
```

```
commit 11cbb765b88ea5d09b03c31665c8d38b3cefcc73
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:32:04 2024 +0300
    creating a ContentDto class

commit d7c891239cd270b60f56aec68f8e9f2607f7d42e
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:30:49 2024 +0300
    refactoring mappers

commit 1066359d013c6bc883562403e0a4676a744798d7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:29:53 2024 +0300
    refactoring DTOs

commit d209ef7f8d29d2dda3418396825da92a55287e2a
Merge: 5ce138a 1e0c880
Author: fred nekrasov <152185797+FredNekrasov@users.noreply.github.com>
Date:   Sat May 11 19:19:52 2024 +0300
    Merge pull request #1 from FredNekrasov/refactor
    Refactor

commit 1e0c880440b7d17b12bff8f8282482207b4089e8
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:07:37 2024 +0300
    removing unused imports

commit bfbb2044967bdb2f3b6ccb0a8b2c444fcde1554b
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:05:58 2024 +0300
    refactoring repositories

commit f0d18dcdaf826e6393062434f82a6487640e9bd2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:05:39 2024 +0300
    refactoring an IRepository

commit 64e35f0db2c426a7cd03dcd3a1913e00aff34dc4
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:04:46 2024 +0300
    deleting an unused variable

commit 9b6492cc6c58fd2d7d90661b8b8a5cfe2252ab27
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 19:02:46 2024 +0300
    refactoring DI modules

commit 8b1e8b8de4faf4e8879e926374561def36633fc5
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:52:50 2024 +0300
    refactoring and moving use cases from the impl folder to the "useCases"
folder

commit d6c4cdf491ed003701750b1aec417c3a87d44b0e
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:51:07 2024 +0300
    removing an unnecessary IGetListUseCase.kt interface

commit c0280985a9059133d391093536c4d2c806d6a5c6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:49:39 2024 +0300
    creating a ApiProvider file to get instances of IArticleService and
IArticleTypeService
```

```
commit c41dbcc76e2b031a88ec0ccb36e850e21e8dc8b7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:47:18 2024 +0300
    removing unnecessary functions from the IArticleTypeService and
IArticleService interfaces

commit 1c59ca8612636f41237a507d46af72d7c62f4193
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:43:37 2024 +0300
    refactoring the ConnectionStatus class

commit 2ebb04a0cab0cad70ebad970a2a3af88f29c56e1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:43:01 2024 +0300
    creating an ActionStatus to get information about the remote data

commit f35a9c48f20e3bd9c3692d7dc7f79a1f5c2ab187
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat May 11 17:37:18 2024 +0300
    updating libraries/plugins version

commit 5ce138afa80dfa109b96252b3e4b3687de5010dc
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 24 11:28:20 2024 +0300
    editing the Theme.kt file

commit 9d0083235c8d5812406deb3ab2f58a46a3a19114
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 24 11:27:24 2024 +0300
    adding a permission to the manifest and setting values to parameters

commit 3730ac29a6335dcf7a326394f5f4c9bf24dc8e84
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 16:52:13 2024 +0300
    deleting unnecessary classes

commit 342dc7c9aae4ca03ab7f65a89f7cf1bb3c9b2ea7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 12:37:18 2024 +0300
    refactoring the ServiceModule.kt file

commit 5a1d1b3bf973df8b75f3a63e4b001ac22ae1d7b3
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 12:34:22 2024 +0300
    deleting the base url and refactoring a function in the IArticleTypeService
interface

commit f2cd754a4ae7dca56da2c97285c4967eab748bd9
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 12:25:22 2024 +0300
    creating a function that creates a Retrofit instance that implements an
IArticleService interface for network requests

commit 7f026c09bbeec6155b3dea10f08d742edbca56fd
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 12:14:42 2024 +0300
    creating a file network_configuration.xml to work with the API

commit 6cb72a22954a161cb43099464c04a0849f00a104
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 11:33:48 2024 +0300
    creating an app class

commit 9728f48a364f0d0e86e84b54c04bae112962f8fd
```

```
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 11:32:35 2024 +0300
    creating a DI for the domain module

commit ee44a1b3c00b49a15f4e1ec0ce8405f47fb95a6b
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 11:31:51 2024 +0300
    creating a DI for the data module

commit cffc6653c112adeeab88d26073907776cd2e87f1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:16:54 2024 +0300
    creating repositories for checking data from the rest api

commit 49c961d08c07d8e3109cad17bdc22959f47258d0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:13:01 2024 +0300
    creating interfaces for getting data from the rest api

commit 07a760ac650de1b34b0a64d48d08ba6f2c89443c
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:11:22 2024 +0300
    refactoring the ArticleMapper.kt file

commit fe636318587e02a19cc7e71303ead93ea7191107
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:10:20 2024 +0300
    creating mappers from dto to model

commit 2d08dbba498817fd9762d0a7ea54d03115b953bc
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:08:34 2024 +0300
    renaming the variable name in the Article.kt class

commit 314e641eec9d6874955f582c91eabc7352533b03
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:06:42 2024 +0300
    creating the dto classes

commit 6e56a5910f7f56bcf27eaf79b6a7ce890432eca7
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 09:01:28 2024 +0300
    creating the GetArticleTypesUseCase.kt and GetArticlesUseCase.kt classes

commit 7fd6168664530ae8f0a28184b58df2dc6e3cca87
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 08:57:38 2024 +0300
    creating an interface for GetListUseCases in the domain layer

commit de6b09b2a13eb6d6bc7a4d2f749ed7366d01ce74
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 08:56:42 2024 +0300
    adding a coroutine dependency and a domain layer connection to the gradle
dependencies of the data module


commit 02e15096184fedf72783e64107aced09b69b5f69
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 08:46:55 2024 +0300
    refactoring an interface for repositories in the domain layer

commit 44b925651ac20799f393b65aee96cb3107301ac3
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 23 08:46:32 2024 +0300
```

```
    refactoring the ConnectionStatus.kt class in the domain layer. adding
parameters to the sealed class constructor.

commit 3ad88a75c60e390fd99f764b2e0c3e4c4789efd1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu Mar 21 10:47:57 2024 +0300
    adding dependencies to domain and application modules.

commit 0ef8e3c044d71beebcd6d547149cb8fe7b39b47e
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu Mar 21 01:34:28 2024 +0300
    creating an interface for repositories in the domain layer

commit 4677aa4b31fed301ec009fab5dd1399aa9103cb2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu Mar 21 01:33:13 2024 +0300
    creating a ConnectionStatus class to send an error message and a list of
data

commit 0d1992135d54059dc062b5fff7040de8875ab603
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Thu Mar 21 01:17:06 2024 +0300
    creating models in the domain layer

commit 66232b9ec90dd60782b629152904bb1af8ee7c45
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Wed Mar 20 20:33:09 2024 +0300
    Добавление зависимостей в модули данных(retrofit и gson) и приложения(koin)

commit b0a7b7b727d3682086066c536e22ca1dd43e8a49
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:59:14 2024 +0300
    Добавление модулей domain и данных в приложение

commit e9d747c83861e3f1d29cf1af19d385980d941473
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:55:17 2024 +0300
    создание domain слоя

commit 05a62e305f87ecdbfc2fa7e3fd98a8d303007c31
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:46:09 2024 +0300
    создание новых личных представлений

commit dec1855db7f9311fc4906b04ca78db7841a564f6
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:45:30 2024 +0300
    переименование данных в ScreenRoute.kt и TitleStrings.kt

commit dac5a687f72d483decb17ff984631ef4aaba0fc0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:41:45 2024 +0300
    удаление ненужных зависимостей

commit e435b89f5c02ab58c0106ab2bbc1ce35d9ee0ad4
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sun Mar 17 16:19:29 2024 +0300
    создание модуля данных

commit f596238b0dc6fadb7d92c560391f1b8baa17ffe5
Author: fred nekrasov <152185797+FredNekrasov@users.noreply.github.com>
Date:   Sun Mar 17 16:05:54 2024 +0300
    Create LICENSE
```

```
commit a31ba5a04be95cc95d8dbbfbd13f05fc568d9ab1
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Sat Mar 9 14:05:51 2024 +0300
    Переход с папки ui в основную папку

commit d19e7f061d3e38b7e6f240d9813def8472c8509d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 22:28:49 2024 +0300
    Создание каркаса приложения

commit 6e4bf79ba9dda03413e5b654c007a516ab6a0f2c
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 22:27:34 2024 +0300
    Создание выдвижной панели

commit bfcbb2796bd7f893d74edf2aa248bae495a9cad2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 22:25:45 2024 +0300
    рефакторинг представлений и создание новых пользовательских представлений

commit 2af303ef9251f08c41c756b9aa1d8e763d3b607a
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 21:29:25 2024 +0300
    Создание своих представлений: кнопка, 2 вариации обычного текста, кнопка с
иконкой, верхняя панель приложения

commit ccb6ca0e109afb74a7b9cf54b39a7a43793ec224
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 21:24:54 2024 +0300
    Создание класса для работы с navigation drawer

commit 355b746eda8036c341f13f3771fb9641406442d2
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 21:16:22 2024 +0300
    Создание путей для основных экранов

commit 103676295bf45206761a9008b6abe2fefaa85439
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 21:15:53 2024 +0300
    Создание констант с наименованием заголовка

commit eb872966f8507d69660b6e0fe8c5c398b43457b3
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 17:05:41 2024 +0300
    добавление зависимостей в проект

commit 15993a30f60023e043433e8f73f60aece4d0a06d
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 16:43:29 2024 +0300
    изменение настроек приложения

commit 838a1befd4bba49cc953654cab815c550615f0a0
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 16:36:15 2024 +0300
    очистка класса MainActivity.kt

commit 66a28cae04afae3e2461f4ed28721ab592df2591
Author: FredNekrasov <frednekrasov49@gmail.com>
Date:   Fri Mar 8 16:34:32 2024 +0300
    создание проекта
```