МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка алгоритма офлайн-просмотра учебников Росдистанта»

Обучающаяся С.Д. Бутримова _____
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент О.Н.Брега

Тольятти 2023

Аннотация

Тема выпускной квалификационной работы: «разработка алгоритма офлайн-просмотра учебников Росдистанта».

Работа выполнена студенткой Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1902б, Бутримовой Светланой Дмитриевной.

Работа посвящена анализу систем разработки мобильных приложений с возможностью закачки документов на устройство и их просмотра.

Объект исследования – системы разработки мобильных приложений.

Цель работы – разработать мобильное приложение для офлайн-просмотра учебников Росдистанта.

Структура бакалаврской работы представлена введением, тремя разделами, заключением и списком литературы.

Во введении описывается актуальность данной задачи, формулируется цель и ставятся задачи, поставленные в работе.

В первом разделе исследуется предметная область, происходит выбор метода защиты учебников от копирования, а так е способ уменьшения объема скачанных учебников.

Во втором разделе описывается структура и реализация мобильного приложения.

В третьем разделе производится запуск и тестирование разработанного мобильного приложения.

В заключении представлены выводы по проделанной работе.

В работе использована 1 таблица, 34 рисунков, список литературы содержит 30 литературных источников. Общий объем выпускной квалификационной работы составляет 66 страниц.

Abstract

The topic of the graduate qualification work: "Development of the algorithm for offline browsing of textbooks Rosdistance".

The work was done by Butrimova Svetlana Dmitrievna, a student of Togliatti State University, Institute of Mathematics, Physics and Information Technology, group PMIB-1902b.

This work is devoted to the analysis of mobile application development systems with the possibility of uploading documents to the device and viewing them.

The object of the research is mobile application development systems.

The aim of the work is to develop a mobile application for offline browsing textbooks Rosdistant.

The structure of the bachelor's work is represented by an introduction, three sections, the conclusion and a list of references.

The introduction describes the relevance of the task, formulates the goal and sets the objectives of the work.

The first chapter examines the subject area, the choice of method for protecting textbooks from copying, as well as a way to reduce the volume of downloaded textbooks.

The second chapter describes the structure and implementation of the mobile application.

The three chapter tests the mobile application.

The paper concludes with deduction.

The work comprises 1 table, 34 figures, reference list contains 30 literature sources. The total volume of graduate qualification work is 66 pages.

Содержание

Введение

Онлайн-образование становится все более популярным среди пользователей во всем мире. Росдистант – это образовательная платформа, которая предоставляет доступ к онлайн-курсам и учебным материалам для образовательных учреждений и отдельных пользователей. Однако не всегда у пользователей есть возможность постоянно находиться в сети Интернет, что затрудняет их обучение. В таких случаях оффлайн-доступ к материалам платформы Росдистант может быть крайне важным.

В процессе работы были изучены основные аспекты языка программирования Java (байт-код), а также работа с HTTP запросами. В результате чего для операционной системы Android будет разработано мобильное приложение с использованием этих алгоритмов. Рассмотрены различные подходы и методы реализации данного алгоритма, включая использование цифровой подписи для защиты от несанкционированного копирования, а также адаптацию интерфейса для более удобного использования оффлайн-режима.

Цель данного проекта - разработать алгоритм оффлайн просмотра учебников Росдистанта, который позволит пользователям просматривать образовательные материалы, даже если они не имеют доступа к сети Интернет. В соответствии с поставленной целью в работе определены следующие задачи:

- проанализировать предметную область;
- проанализировать исходные данные с сайта вуза и подготовить эскизы и проект продукта;
- рассмотреть алгоритмы защиты данных от копирования;
- рассмотреть алгоритмы сжатия данных;
- реализовать мобильное приложение под управлением выбранной операционной системы;
- применить выбранный алгоритм защиты данных от копирования;

– применить выбранный алгоритм сжатия данных;
– наполнить данными приложение, взятыми с официального сайта;
– протестировать разработанное приложение.

Бакалаврская работа состоит из введения, трех разделов, заключения и списка литературы.

Во введении описывается актуальность данной задачи, формулируется цель и ставятся задачи, поставленные в работе.

В первом разделе исследуется предметная область и происходит выбор инструментов для создания мобильного приложения, осуществляющего офлайн доступ к учебникам Росдистанта.

Во втором разделе описываются структура и архитектура мобильного приложения.

В третьем разделе производится запуск и тестирование разработанного мобильного приложения.

Результаты работы могут быть полезны для пользователей Росдистанта, которые часто сталкиваются с ограничениями доступа к сети Интернет, для которых доступность образовательных материалов в оффлайн-режиме является важным критерием при выборе онлайн-платформы.

1 Анализ существующих технологий

1.1 Математическая модель

Математическая модель офлайн просмотра учебников Росдистанта может быть представлена следующим образом:

Пусть имеется n учебников, каждый учебник i содержит m_i страниц, а каждая страница содержит k символов. Пользователь может загружать учебники на своё устройство и просматривать их офлайн.

Предположим, что при загрузке учебника на устройство, занимаемое им место равно s_i байт.

Тогда общее количество занимаемого места при загрузке всех n учебников будет равно:

$$S = s_1 + s_2 + \ldots + s_n, \quad (1)$$

При чтении каждой страницы учебника, устройство считывает k символов из памяти. Тогда время T, необходимое для загрузки и отображения страницы, может быть описано следующей формулой:

$$T = \left(\frac{s}{k}\right) * t. \quad (2)$$

Таким образом, общее время T_{total}, необходимое для загрузки и отображения учебника, будет равно:

$$T_{total} = T_1 + T_2 + \ldots + T_m. \quad (3)$$

Общее время загрузки всех учебников $T_{general}$ описано формулой:

$$T_{general} = T_{total1} + T_{total2} + \ldots + T_l. \quad (4)$$

Математическая модель офлайн просмотра учебников Росдистанта может быть описана с помощью формул для занимаемого устройством места и времени загрузки.

1.2 Основные требования к приложению

Для получения доступа к оффлайн просмотру учебников Росдистанта было принято решение разработать мобильное приложение.

После авторизации студенту должен быть доступен список его текущих курсов. По каждому курсу доступен список электронных учебников.

По каждому учебнику доступен его онлайн-просмотр в приложении. Доступна возможность сохранить любой учебник на устройство, и просмотр таких сохраненных учебников оффлайн.

Предусмотреть возможность удалить любой отдельный ранее сохраненный на устройство учебник (в т.ч. если соответствующий курс у студента ушел из текущих).

При сохранении учебника на устройство предусмотреть возможность частичной закачки. Т.е. при разрыве соединения во время закачки – при появлении интернета докачиваются только недостающие данные. Также предусмотреть возможность вручную поставить закачку на паузу с возможностью потом возобновить и докапать только оставшиеся данные.

Предусмотреть разделение списка курсов на текущие курсы (т.е. курсы текущего семестра) и ранее пройденные.

Предусмотреть возможность просмотра и сохранения для офлайн-просмотра других учебных материалов курсов, кроме электронных учебников (приложенных файлов, страниц, и др.; тесты и задания не рассматриваем)

Защита от копирования сохраненных учебников.

Учебники не должны занимать много памяти на устройстве.

При обновлении в курсе в LMS материала, который ранее был сохранен на устройство – отображать это в приложении, предлагать обновить сохраненный материал.

Интеграция с LMS Росдистант.

1.3 Обзор и анализ платформ разработки мобильных приложения

В современном мире все чаще люди пользуются мобильными телефонами и все реже персональными компьютерами. Многие компании стараются сделать себе мобильное приложение или мобильную версию сайта, чтобы людям было удобнее пользоваться услугами этих компаний.

Официальное определение гласит: «Мобильное приложение – это программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах» [18]. Основная цель приложений заключается в том, чтобы выполнить определенный набор функций и упростить пользовательскую жизнь. В мобильных приложениях может быть множество функций, но они обязаны быть простыми и понятными в использовании, иначе их не будут скачивать пользователи.

Разработка мобильных приложений представляет собой процесс создания программного обеспечения для смартфонов и других цифровых устройств, в основном для платформ Android и iOS. Программное обеспечение может быть установлено на устройстве заранее, загружено из магазина приложений или доступно через мобильный веб-браузер.

Для создания такого рода программного обеспечения используются различные языки программирования и технологии разметки, такие как Java, Swift, C# и HTML5.

1.3.1 Операционная система Android

При проектировании и разработке программного обеспечения предпочтение отдавалось архитектурным решениям и программным

продуктам, уже доказавшим свою эффективность при решении подобных задач.

Программное обеспечение создавалось на модульной основе с использованием объектно-ориентированного подхода, позволяющего добавлять или изменять функциональные возможности.

В качестве средства разработки программного обеспечения MPM использовался стандартный набор API-функций операционной системы Android 7.0, который совместим с версиями Android 4.0 и выше. На рисунке 1 представлена архитектура системы Android.

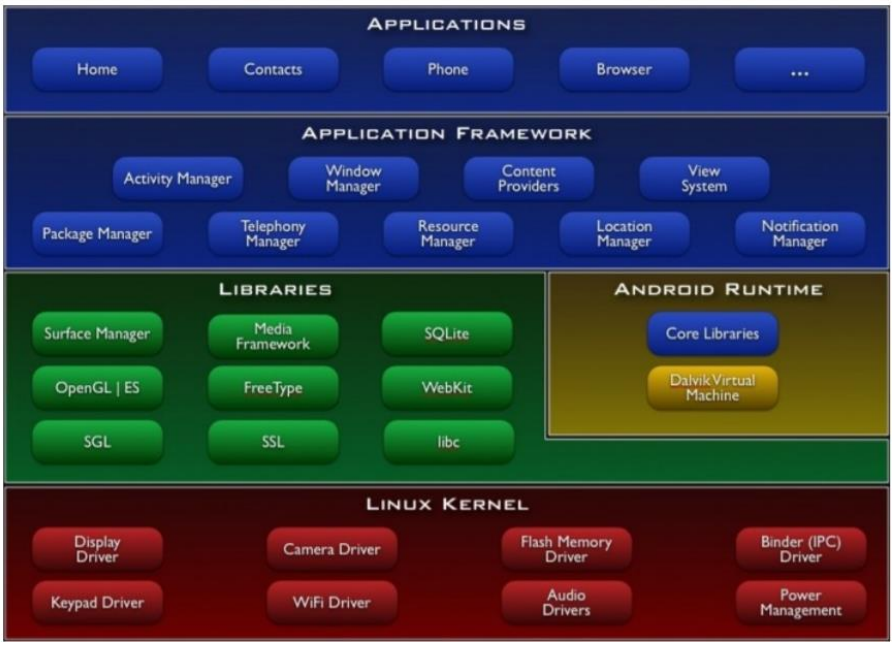


Рисунок 1 – Архитектура системы Android

Для достижения целей, перед нами стоит выбор из многочисленных инструментов. В нашем случае, для разработки мобильного приложения под Android мы выбрали одну из наиболее популярных сред разработки на сегодняшний день - Android Studio. Эта интегрированная среда разработки (IDE) предназначена как для небольших команд разработчиков мобильных

приложений, так и для крупных международных организаций, благодаря своей способности обрабатывать масштабные проекты.

Android Studio является официальной средой разработки под Android. При выборе IDE для Android в настоящее время можно рассмотреть следующие варианты:

- Eclipse, которая ранее была наиболее популярной средой разработки на Java, но в настоящее время не рекомендуется для разработки под Android;
- IntelliJ IDEA, которая также является отличным выбором для разработки под Android;
- Android Studio, которая является наиболее популярной и, возможно, лучшей средой разработки под Android, поскольку постоянно обновляется Google. Кроме того, не нужно заботиться о плагинах, так как эта IDE специально создана для разработки под Android. Она отличается от IDEA некоторыми мелкими деталями, которые значительно облегчают труд разработчика.

Программирование в выбранной нами среде осуществляется на языке программирования Java, который является строго типизированным объектно-ориентированным языком программирования. Программы на Java транслируются в байт-код Java, который выполняется виртуальной машиной Java (JVM) - программой, которая обрабатывает байт-код и передает инструкции оборудованию в качестве интерпретатора.

Этот метод выполнения программ имеет преимущество в том, что байт-код программы не зависит от операционной системы и оборудования, что позволяет запускать Java-приложения на любом устройстве, где есть соответствующая виртуальная машина. Еще одной важной особенностью технологии Java является гибкая система безопасности, которая позволяет полностью контролировать выполнение программы виртуальной машиной. Любые операции, которые выходят за пределы полномочий программы

(например, попытки несанкционированного доступа к данным или соединения с другим компьютером), приводят к немедленному прерыванию [22].

1.3.2 Среда разработки Android Studio

Android Studio – это интегрированная среда разработки для работы с платформой Android, анонсированная на конференции Google 16 мая 2013 года.

IDE доступна бесплатно с версии 0.1, выпущенной в мае 2013 года, а затем в бета-тестировании с версии 0.8, выпущенной в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, и поддержка плагина Android Development Tools для Eclipse прекратилась.

Android Studio основана на программном обеспечении IntelliJ IDEA компании JetBrains и является официальным инструментом разработки приложений для Android.

Каждая новая версия Android Studio предлагает новые возможности. В настоящее время доступны следующие функции:

- расширенный редактор макетов: WYSIWYG, возможность манипулировать компонентами пользовательского интерфейса с помощью перетаскивания, а также функция предварительного просмотра макета на нескольких конфигурациях экрана;
- создание приложений на основе Gradle;
- генерация различных сборок и нескольких файлов .apk;
- рефакторинг кода;
- статический анализатор кода (Lint), позволяющий находить проблемы с производительностью, несовместимость версий и многое другое;
- встроенные утилиты ProGuard и подписки на приложения;
- шаблоны макетов и компонентов ядра Android;
- поддержка разработки приложений для Android Wear и Android TV;

– встроенная поддержка Google Cloud Platform, включая интеграцию с сервисами Google Cloud Messaging и App Engine;
– Android Studio 2.1 поддерживает Android N Preview SDK, что означает, что разработчики могут начать создавать приложения для новой программной платформы;
– новая версия Android Studio 2.1 работает с обновленным компилятором Jack, а также получила улучшенную поддержку Java 8 и улучшенную функциональность Instant Run;
– Platform-tools 23.1.0 для Linux становится только 64-битной, даже при попытке установить 32-битную версию, и никакого объявления нет. Другими словами, Android Studio больше не работает на 32-битных версиях Linux (выдавая фатальные ошибки) [23].

1.3.3 Инструменты дизайна и проектирования

Существует множество различных инструментов создания мобильных приложений. Они могут разделяться по совместимости с операционной системой компьютера, по совместимости ОС телефона, по назначению. Это самые важные критерии классификации инструментов.

Классификация по совместимости инструмента с операционной системой компьютера:

– MacOS;
– Windows;
– Linux;
– ChromeOS;
– кроссплатформенные.

То есть существуют инструменты, работающие, например, только на операционной системе Windows, или только на MacOS и так далее. Но также существуют инструменты, работающие и на Windows, и на MacOS, и даже на Linux. Они и называются кроссплатформенными инструментами.

Классификация по совместимости инструмента с операционной

системой телефона:

- Android;
- iOS;
- Windows Phone.

Это относится к тому, может ли инструмент создавать макеты приложений для конкретной мобильной операционной системы. Существуют инструменты, позволяющие создавать макеты для одной операционной системы или для нескольких операционных систем.

Инструменты по назначению:

- проверка удобства использования. Доступен ли сайт или приложение для людей с ограниченными возможностями;
- создание анимации. Чтобы сайт выглядел менее "деревянным", нужно добавить анимацию;
- совместная работа. Если вы работаете в команде, вам необходимо синхронизировать свои действия;
- выбор цвета. Это помогает находить приятные цветовые сочетания;
- передача проекта разработчику;
- создание систем проектирования. Они помогают синхронизировать и оптимизировать работу команд и отдельных дизайнеров;
- мониторинг опыта. Изучая поведение пользователей, вы можете выявить недостатки дизайна и улучшить их;
- использование шрифтов. Шрифты, как и цвета, оказывают большое влияние на восприятие продукта людьми, поэтому важно выбрать шрифт, соответствующий теме вашего проекта;
- инструменты информационной архитектуры. Они помогают улучшить удобство использования и информационную архитектуру веб-сайта;
- прототипирование;
- инструменты проектирования интерфейсов;

– создание потоков пользователей.

С помощью таких инструментов можно спланировать путь пользователей через сайт или приложение и улучшить интерфейс [10].

Существует множество других типов инструментов для создания мобильных приложений. В этой статье мы рассмотрим наиболее популярные и универсальные инструменты.

Framer X – это довольно обширная программа, которая, помимо проектирования, позволяет создавать адаптивные макеты и реалистичные прототипы, а затем передавать почти готовый код разработчику [2].

Преимущества:

– позволяет создавать компоненты React;
– позволяет создавать проекты и программировать;
– позволяет создавать и изменять любую анимацию пользовательского интерфейса.

Недостатки.

– очень сложный инструмент, которым будет трудно управлять без обучения;
– это также сложно без понимания кода;
– доступно только для Mac OS.

Adobe XD – это инструмент для проектирования, создания прототипов и масштабных презентаций. Предназначен для того, чтобы сделать рабочий процесс как можно более быстрым [2].

Преимущества:

– подходит для Mac OS и Windows;
– тестирование прототипов непосредственно на мобильных устройствах;
– множество инструментов для ускорения работы.

Недостатки:

– ограничения в использовании компонентов, которые упрощают

процесс редактирования множества похожих элементов;

– анимация слабовата, так как инструмент больше используется для создания прототипов и сборки дизайна из готовых макетов.

InVision Studio – относительно новый инструмент, который хорошо подходит для проектирования интерактивных систем [2].

Преимущества:

– пригодность для работы на MacOS и Windows;
– ускоряет процесс адаптации макетов, так как имеет некоторые нюансы;
– бесплатный инструмент.

Недостаток: большие проекты с большим количеством анимации могут быть очень медленными.

Sketch – это векторный редактор, который позволяет сосредоточиться на создании интерфейсов. Позволяет создавать векторные изображения, иконки, ретушировать фотографии, а также создавать и подготавливать дизайн веб-сайтов.

Преимущества:

– большая база данных бесплатных ресурсов (наборы инструментов пользовательского интерфейса, библиотеки, шаблоны и т.д.);
– простой интерфейс.

Недостатки:

– доступно только для MacOS;
– бесплатно в течение 30 дней;
– нет поддержки совместного редактирования проектов.

Figma – удобный графический редактор, работать с которым можно прямо в браузере или скачать программу на компьютер.

Преимущества:

– подходит для работы на любой платформе: Windows, Mac OS, Chrome OS, Linux;

– вся работа моментально сохраняется в облаке, что значит отсутствие риска не сохранить долгие часы работы;
– есть возможность работать с командой из нескольких человек в одном проекте;
– огромное количество плагинов, ускоряющих работу дизайнера, например, плагины для автоматического заполнения имени и фамилии;
– большая часть функционала бесплатна;
– можно настраивать простую анимацию макета, а также просматривать его на различных моделях телефонов, планшетов и смарт часов.

Недостатки:

– есть платная версия, но она не дает так уж много дополнительного функционала, поэтому смысла покупать ее нет;
– нет возможности работать с полиграфией, так как отсутствует палитра CMYK;
– нет десктопных мокапов, из-за чего нельзя просмотреть макет так, как он будет выглядеть на компьютере.

Когда только начали появляться веб-сайты инструментов для их проектирования было разработано мало. Существовал Photoshop, который мог ретушировать фотографии. В него было заложено довольно много функций, которые подходили для дизайна макетов сайтов, поэтому веб-дизайнеры использовали его. В наши дни Figma и Sketch являются наиболее простыми и удобными редакторами для веб-сайтов, поэтому многие дизайнеры стали переходить туда. Также переход обусловлен тем, что Photoshop при создании двух и более рабочих поверхностей в одном файле начинает очень сильно тормозить или даже зависать, что крайне неудобно. Figma и Sketch выдерживают сразу несколько рабочих поверхностей, практически не теряя при этом в скорости.

Интерфейс Figma, в сравнение с Photoshop предельно простой. Никаких излишеств:

- редактирование цвета;
- редактирование текста;
- редактирование формы объекта;
- добавление теней, бликов, бордеров;
- компоненты;
- Auto layout;
- плагины.

Здесь приведены самые широкие функции Figma, но есть и другие, более узкой направленности. Так как Sketch предназначен только для компьютеров на Mac OS, а Photoshop не предназначен для веб-разработки вообще, в данной работе будет использоваться Figma. [2]

1.4 Защита учебников от копирования

1.4.1 Цифровая подпись

Защита учебников от копирования может быть обеспечена с помощью различных технологий, таких как DRM (Digital Rights Management), водяные знаки или шифрование. Одной из возможных математических моделей защиты учебников от копирования является модель, основанная на использовании цифровой подписи.

Предположим, что учебник представлен в виде электронного документа, содержащего информацию об авторских правах и цифровую подпись. Цифровая подпись гарантирует, что документ не был изменен после подписания и что он был подписан авторизованным лицом. Для проверки подписи используется открытый ключ, который может быть распространен отдельно от документа.

При попытке скопировать документ, копия будет содержать другую цифровую подпись, что позволит легко определить факт копирования. Кроме того, можно использовать дополнительные меры защиты, например, ограничение на количество устройств, на которых можно открыть документ с одной цифровой подписью, или использование временной блокировки после определенного количества открытий.

Математическая модель защиты учебников от копирования с использованием цифровой подписи может быть описана следующим образом:

Пусть учебник представлен в виде электронного документа D, содержащего информацию об авторских правах и цифровую подпись S. Цифровая подпись S создается авторизованным лицом с использованием закрытого ключа K и алгоритма подписания Sign. При этом подпись S является уникальной для каждого документа и гарантирует целостность и подлинность документа.

При попытке скопировать документ M, копия M' будет содержать другую цифровую подпись S', которая не будет совпадать с подписью S. Таким образом, можно легко определить факт копирования.

Математически, алгоритм подписания Sign может быть описан следующей формулой:

$$S = Sign(M, D), \tag{5}$$

где M – электронный документ,

D – закрытый ключ.

Для проверки подписи S используется открытый ключ E', который может быть распространен отдельно от документа. Проверка подписи осуществляется с помощью алгоритма проверки подписи Verify:

$$Verify(D, S, E') = true, \tag{6}$$

если подпись S является действительной для документа M и открытого ключа E'.

Для дополнительной защиты от копирования можно использовать ограничение на количество устройств, на которых можно открыть документ с одной цифровой подписью. Пусть N – максимальное количество устройств, на которых можно открыть документ с одной цифровой подписью. Тогда при каждом открытии документа на устройстве i, счетчик открытий на устройстве i увеличивается на 1. Если счетчик открытий на устройстве i достигает значения N, то документ блокируется на этом устройстве. Это может быть описано следующим образом:

Пусть C_i будет счетчиком открытий документов на устройстве i. Если $\mathrm{C}_i \geq \mathrm{N}$, то заблокировать документ на устройстве i.

Кроме того, можно использовать временную блокировку после определенного количества открытий. Пусть P – максимальное количество открытий документа. Тогда при каждом открытии документа счетчик открытий увеличивается на 1. Если счетчик достигает значения P, то документ блокируется на определенный период времени T. Это может быть описано следующим образом:

Пусть O – счетчик открытий документов на устройстве i. Если $O \geq P$, то заблокировать документ на время T.

Таким образом, математическая модель защиты учебников от копирования может быть описана с помощью формул для создания цифровой подписи и проверки её действительности, а также использования ограничений на количество устройств и количество открытий документа. Эти формулы могут быть использованы для разработки системы защиты учебников от копирования с помощью цифровых подписей.

1.4.2 Алгоритм подписания Sign и Алгоритм проверки подписи (Verify)

Алгоритм подписания (Sign) обычно используется для создания цифровой подписи сообщения, чтобы гарантировать его подлинность и

целостность. Цифровая подпись создается с использованием закрытого ключа отправителя сообщения, который необходим для расшифровки подписи.

В целом, алгоритм подписания состоит из трех основных шагов: вычисления хеш-значения сообщения, создания цифровой подписи с использованием закрытого ключа отправителя и добавления подписи к сообщению.

Алгоритм подписания (Sign) может быть описан с помощью математической модели, которая включает в себя следующие шаги:

а) выбор простых чисел p и q, которые используются для генерации открытого и закрытого ключей RSA;

б) вычисление модуля n, который является произведением простых чисел:

$$n = p * q; \quad (7)$$

в) вычисление функции Эйлера от n, которая равна:

$$\varphi(n) = (p - 1) * (q - 1),; \quad (8)$$

г) выбор открытого ключа e, который является целым числом, взаимно простым с $\varphi(n)$;

д) вычисление закрытого ключа d, который является обратным элементом для e по модулю φ (n), то есть:

$$d * e = 1 \left(mod\varphi(n)\right); \quad (9)$$

е) выбор хеш-функции H, которая используется для вычисления хеш-значения сообщения, хеш-функция обычно выбирается из надежных криптографических алгоритмов, таких как SHA-256 или SHA-512;

ж) вычисление хеш-значения сообщения m с использованием выбранной хеш-функции:

$$h(m) = H(m); \quad (10)$$

з) создание цифровой подписи s с использованием закрытого ключа d и хеш-значения сообщения $h(m)$:

$$s = h(m)^d mod\ n; \quad (11)$$

и) добавление цифровой подписи s к сообщению.

Когда получатель получает подписанное сообщение, он может использовать открытый ключ e и модуль n для расшифровки подписи и проверки подлинности сообщения.

Главное преимущество алгоритма подписания RSA заключается в том, что он обеспечивает высокий уровень безопасности при передаче данных, так как для подписания сообщения используется закрытый ключ, который является известным только отправителю.

Алгоритм проверки подписи (Verify) обычно используется для проверки подлинности сообщения, которое было подписано с использованием алгоритма создания подписи (Sign). Этот алгоритм обычно используется в криптографии для обеспечения конфиденциальности и целостности данных.

Для проверки подписи сообщения необходимо выполнить следующие шаги:

а) извлечь цифровую подпись s из сообщения;

б) вычислить хеш-значение сообщения m с использованием выбранной хеш-функции:

$$h(m) = H(m); \quad (12)$$

в) использовать открытый ключ e и модуль n для расшифровки подписи:

$$s' = s^e mod\ n; \quad (13)$$

г) сравнить расшифрованную подпись s' с вычисленным хеш-значением сообщения $h(m)$. Если они совпадают, то подпись действительна и сообщение не было изменено после создания подписи;

д) сравнивается расшифрованная подпись s' с вычисленным хеш-значением сообщения $h(m)$. Если они совпадают, то подпись действительна и сообщение не было изменено после создания подписи.

В целом, алгоритм проверки подписи Verify может быть реализован с использованием различных криптографических примитивов и формул, но основная идея заключается в том, чтобы использовать открытый ключ для проверки подписи, которая была создана с использованием секретного ключа. Криптографические хеш-функции используются для генерации фиксированного размера хеш-значения, которое служит для проверки целостности сообщения. При этом, подпись должна быть создана таким образом, чтобы невозможно было подделать подпись без знания секретного ключа.

На рисунке 2 показана общая схема цифровой подписи RSA:

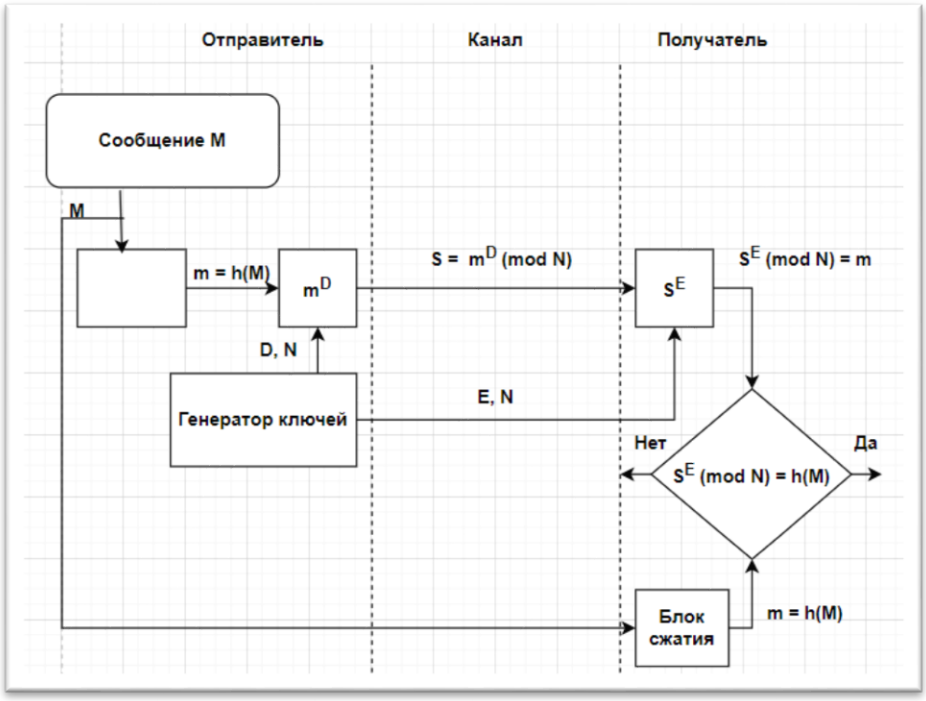


Рисунок 2 – Общая схема цифровой подписи RSA

1.5 Уменьшение занимаемого места

1.5.1 Выбор способа уменьшения занимаемого места

Для уменьшения занимаемого места при загрузке учебника можно использовать формулу $S = s\_1 + s\_2 + \ldots + s\_n$, где s_i – размер учебника i в байтах.

Существует несколько способов уменьшения размера учебника:

– использование сжатия данных: можно использовать сжатие данных для уменьшения размера файлов учебников. Это может быть выполнено с использованием алгоритмов сжатия данных, таких как gzip или zlib. Однако, необходимо учитывать, что сжатие может увеличить время, необходимое для загрузки и отображения учебника, так как приложению придется распаковывать данные перед отображением;

– использование менее затратных форматов файлов: можно использовать более компактные форматы файлов, такие как EPUB или MOBI, вместо традиционных форматов, таких как PDF. Эти форматы обеспечивают более эффективное использование места за счет использования сжатия данных и других технологий, таких как динамическое изменение размера изображений и сжатие текста;
– удаление ненужной информации: можно удалить ненужную информацию из учебников, такую как метаданные, скрытые слои и неиспользуемые ресурсы. Это может помочь уменьшить размер учебника и сократить время загрузки;
– разделение учебника на отдельные части: можно разделить учебник на отдельные части, например, на главы или разделы, и загружать только необходимые части при открытии учебника. Это может помочь уменьшить объем данных, необходимых для загрузки, и сократить время загрузки.

Все эти методы могут быть использованы в комбинации для достижения максимального уменьшения занимаемого места при загрузке учебника. Однако, при выборе методов необходимо учитывать потенциальные негативные последствия, такие как потеря качества изображений и текста, увеличение времени загрузки или потеря некоторых функций и возможностей. В данном случае воспользуемся сжатием данных.

1.5.2 Алгоритм сжатия GZIP

Алгоритм сжатия GZIP является одним из наиболее распространенных алгоритмов сжатия данных и используется в различных приложениях и протоколах, включая HTTP и SMTP.

Алгоритм GZIP основан на комбинации двух алгоритмов: Lempel-Ziv (LZ77) и Хаффмана. LZ77 используется для поиска повторяющихся блоков данных, которые могут быть заменены ссылками на их первое вхождение, а Хаффман используется для построения оптимального кода для замены блоков данных.

Алгоритм gzip работает следующим образом:

а) оригинальный набор данных разделяется на блоки;

б) каждый блок проходит через алгоритм LZ77, который ищет повторяющиеся блоки данных;

в) каждый повторяющийся блок заменяется ссылкой на его первое вхождение в блоке данных;

г) затем, для каждого блока данных, сформированный набор ссылок и символьные данные проходят через алгоритм Хаффмана, который строит оптимальный код для замены блоков данных;

д) наконец, полученные блоки данных и ссылки объединяются в один сжатый поток данных.

При распаковке данных алгоритм GZIP использует обратный процесс. Сначала выделяются блоки данных и ссылки, затем декодируются ссылки и символьные данные, и наконец объединяются в оригинальный набор данных.

Алгоритм gzip обычно достигает хорошего уровня сжатия для текстовых данных, таких как HTML, CSS, JavaScript и XML. Однако, для бинарных данных, таких как изображения и видео, хорошие результаты могут быть достигнуты только с помощью специализированных алгоритмов сжатия, таких как JPEG и MPEG.

Алгоритм сжатия GZIP можно описать математически следующим образом:

Пусть $X = (x_1, x_2, \dots, x_n)$ – исходный набор данных размером n байтов Данные X разбиваются на блоки длиной L байтов, где L - параметр алгоритма сжатия. $Z = (z_1, z_2, \dots, z_m)$ - итоговый сжатый поток данных.

Для каждого блока данных $X[i: i + L - 1]$ $(0 \leq i < n - L + 1)$ выполняется следующий алгоритм сжатия:

а) выполнить алгоритм LZ77, который ищет в блоке $X[i: i + L - 1]$ повторяющиеся подстроки и заменяет их на ссылки на первое вхождение. И получить модифицированный блок данных Y;

б) полученный модифицированный блок данных Y подвергается обработке алгоритмом Хаффмана, который строит оптимальный код для замены блоков данных на более короткие битовые последовательности. Получаем битовую последовательность H;

в) полученный сжатый блок данных H добавляется в сжатый поток данных Y. После сжатия каждого блока данных получается сжатый поток данных Y. Объединив все сжатые блоки данных, получаем итоговый сжатый поток данных Z.

Таким образом, алгоритм сжатия GZIP представляет собой комбинацию двух алгоритмов: LZ77 и Хаффмана, которые позволяют достичь высокой степени сжатия данных при сохранении качества данных.

Выводы по первому разделу

В процессе написания данного раздела были изучены основы разработки мобильных приложений, их виды и типы. Также были изложены основные требования к приложению.

После этого был проведен обзор и анализ платформ для разработки мобильных приложений, а также инструментов дизайна и проектирования.

В заключении был изучен такой метод осуществления защиты учебников от копирования как использование цифровой подписи. Так же был рассмотрен алгоритм сжатия данных GZIP, с помощью которого мы уменьшаем занимаемое место на устройстве.

На основании всего вышеописанного можно сделать вывод что осуществление офлайн просмотра учебников Росдистанта достаточно трудоемкий процесс, но с помощью некоторых инструментов можно облегчить этот процесс. На основании изложенной информации можно перейти непосредственно к разработке мобильного приложения в среде Android Studio.

2 Проектирование мобильного приложения

2.1 Структура проекта

Для написания мобильного приложения, опишем классы и их содержимое:

1. AdapterCourse – адаптер списка курсов:
   - Inflater – объект класса LayoutInflater, который используется для создания объектов View из XML-файлов;
   - data – список объектов класса dataCourse, которые содержат информацию о курсах;
   - activity –объект класса MainActivity;
   - fragment – объект класса IndexFragment;
   - all – количество документов на курсе;
   - now – количество загруженных документов на курсе;
   - header – объект класса TextView для отображения заголовка курса;
   - course_status – объект класса TextView для отображения статуса загрузки курса;
   - course_count – объект класса TextView для отображения количества документов на курсе;
   - course_read – объект класса Button для начала чтения курса;
   - course_delete – объект класса Button для удаления загруженных документов курса;
   - courseWeb – объект класса WebView для загрузки страницы курса.

Поля inflater, data, activity и fragment в конструкторе adapterCourse() и методе onCreateViewHolder() для создания объекта MyHolder, который содержит отображаемые элементы списка курсов.

Поля header, course_status, course_count, course_read, course_delete и courseWeb используются для отображения информации о курсе в элементе списка. Поля all и now используются для отображения статуса загрузки курса. Метод isNetworkAvailable() используется для проверки доступности сети. Методы startDownload() и inArray() используются для загрузки документов курса и сохранения их в локальном хранилище. Методы onBindViewHolder() и getItemCount() используются для заполнения элементов списка и получения количества элементов в списке соответственно.

2. AdapterDocs – адаптер списка документов:
   - inflater: объект LayoutInflater, используется для создания макетов элементов списка, то есть для создания объектов View, которые будут использоваться для отображения каждого элемента списка;
   - data: список объектов dataCourse, которые будут отображаться в списке;
   - activity: объект MainActivity, к которому привязан адаптер. Используется для запуска активности, которая будет отображать документы в приложении;
   - header: объект TextView, используется для отображения заголовка документа;
   - mainView: объект View, используется для хранения ссылки на основной макет элемента списка документов;
   - course_read: объект Button, используется для отображения кнопки, которая позволяет открыть документ в приложении.
3. DataCourse – дата-класс для работы со списками курсов и документов, основные методы и поля класса DataCourse:
   - URL: строка, содержащая URL-адрес документа;
   - Title: строка, содержащая заголовок документа;
   - ID: целое число, содержащее идентификатор документа;

- dataCourse(String, String, int): конструктор класса, который инициализирует поля URL, Title и ID.

4. Класс DownloadFileFromUrl является подклассом AsyncTask и используется для загрузки файла по URL-адресу. Он содержит следующие поля и методы:
   - url: строка, содержащая URL-адрес файла;
   - filename: строка, содержащая имя файла;
   - callBack: объект CallBackDownload, используется для вызова метода onComplete(), когда загрузка завершена;
   - DownloadFileFromUrl(String, String, CallBackDownload): конструктор класса, который инициализирует поля url, filename и callBack;
   - onPreExecute(): метод, который вызывается перед выполнением задачи. В данном случае он не используется;
   - doInBackground(String...): метод, который выполняет фоновые операции. Он загружает файл по URL-адресу и сохраняет его на устройство;
   - onProgressUpdate(String...): метод, который вызывается при обновлении прогресса загрузки. В данном случае он не используется;
   - onPostExecute(String): метод, который вызывается после выполнения задачи. Он вызывает метод onComplete() объекта callBack, чтобы сообщить об окончании загрузки;
   - интерфейс CallBackDownload содержит метод onComplete(), который вызывается при завершении загрузки файла.
5. IndexFragment – главный фрагмент – загружается список всех курсов, основные методы и поля класса IndexFragment:
   - mView: объект View, который содержит макет фрагмента;
   - activity: объект MainActivity, к которому привязан фрагмент;

- IndexFragment(MainActivity): конструктор класса, который инициализирует поле activity;
- onCreate(Bundle): метод, который вызывается при создании фрагмента;
- onCreateView(LayoutInflater, ViewGroup, Bundle): View: метод, который создает и возвращает макет фрагмента и загружает список курсов;
- loadCourses(): метод, который загружает список курсов из SharedPreferences и отображает его в списке.

6. MainActivity – главная активность, в которой происходит работа всех фрагментов, основные методы и поля класса MainActivity:
   - mSectionsPagerAdapter: объект SectionsPagerAdapter, который управляет вкладками;
   - mViewPager: объект ViewPager, который отображает содержимое вкладок;
   - toolbar: объект Toolbar, который содержит панель инструментов;
   - drawerLayout: объект DrawerLayout, который содержит боковую панель навигации;
   - navigationView: объект NavigationView, который содержит элементы боковой панели навигации;
   - navHeader: объект View, который содержит заголовок боковой панели навигации;
   - navHeaderTitle: объект TextView, который содержит заголовок боковой панели навигации;
   - navHeaderSubtitle: объект TextView, который содержит подзаголовок боковой панели навигации;

- onCreate(Bundle): метод, который вызывается при создании активности. Он инициализирует макет активности, загружает фрагмент SplashFragment и отображает его;
- setupToolbar(): метод, который настраивает панель инструментов;
- setupViewPager(ViewPager): метод, который настраивает ViewPager;
- setupDrawerLayout(DrawerLayout): метод, который настраивает боковую панель навигации;
- openFragment(Fragment): метод, который открывает переданный фрагмент;
- loadFragment(Fragment, int): метод, который заменяет текущий фрагмент переданным фрагментом.

7. LoginFragment – фрагмент авторизации, основные методы и поля класса LoginFragment:
   - activity: объект MainActivity, к которому привязан фрагмент;
   - mView: объект View, который содержит макет фрагмента;
   - LoginFragment(MainActivity): конструктор класса, который инициализирует поле activity;
   - onCreate(Bundle): метод, который вызывается при создании фрагмента. В данном случае он не используется;
   - onCreateView(LayoutInflater, ViewGroup, Bundle): View: метод, который создает и возвращает макет фрагмента и добавляет обработчик нажатия на кнопку "Войти".
8. ReadFragment – список документов соответствующего курса, основные методы и поля класса ReadFragment:
   - activity: объект MainActivity, к которому привязан фрагмент;
   - ID: идентификатор курса;
   - mView: объект View, который содержит макет фрагмента;

- ReadFragment(MainActivity, int): конструктор класса, который инициализирует поля activity и ID;
- onCreate(Bundle): метод, который вызывается при создании фрагмента. В данном случае он не используется;
- onCreateView(LayoutInflater, ViewGroup, Bundle): View: метод, который создает и возвращает макет фрагмента и вызывает метод loadDocs();
- loadDocs(): метод, который загружает список документов для данного курса и отображает его RecyclerView.

9. SplashFragment – экран загрузки данных, основные методы и поля класса SplashFragment:
   - activity: объект MainActivity, к которому привязан фрагмент;
   - mView: объект View, который содержит макет фрагмента;
   - SplashFragment(MainActivity): конструктор класса, который инициализирует поле activity;
   - onCreate(Bundle): метод, который вызывается при создании фрагмента. В данном случае он не используется;
   - onCreateView(LayoutInflater, ViewGroup, Bundle): View: метод, который создает и возвращает макет фрагмента и загружает страницу веб-сайта в WebView;
   - isNetworkAvailable(Context): boolean: статический метод, который проверяет наличие подключения к интернету.

На рисунке 3 приведена диаграмма классов, содержащая в себе основные компоненты программы.

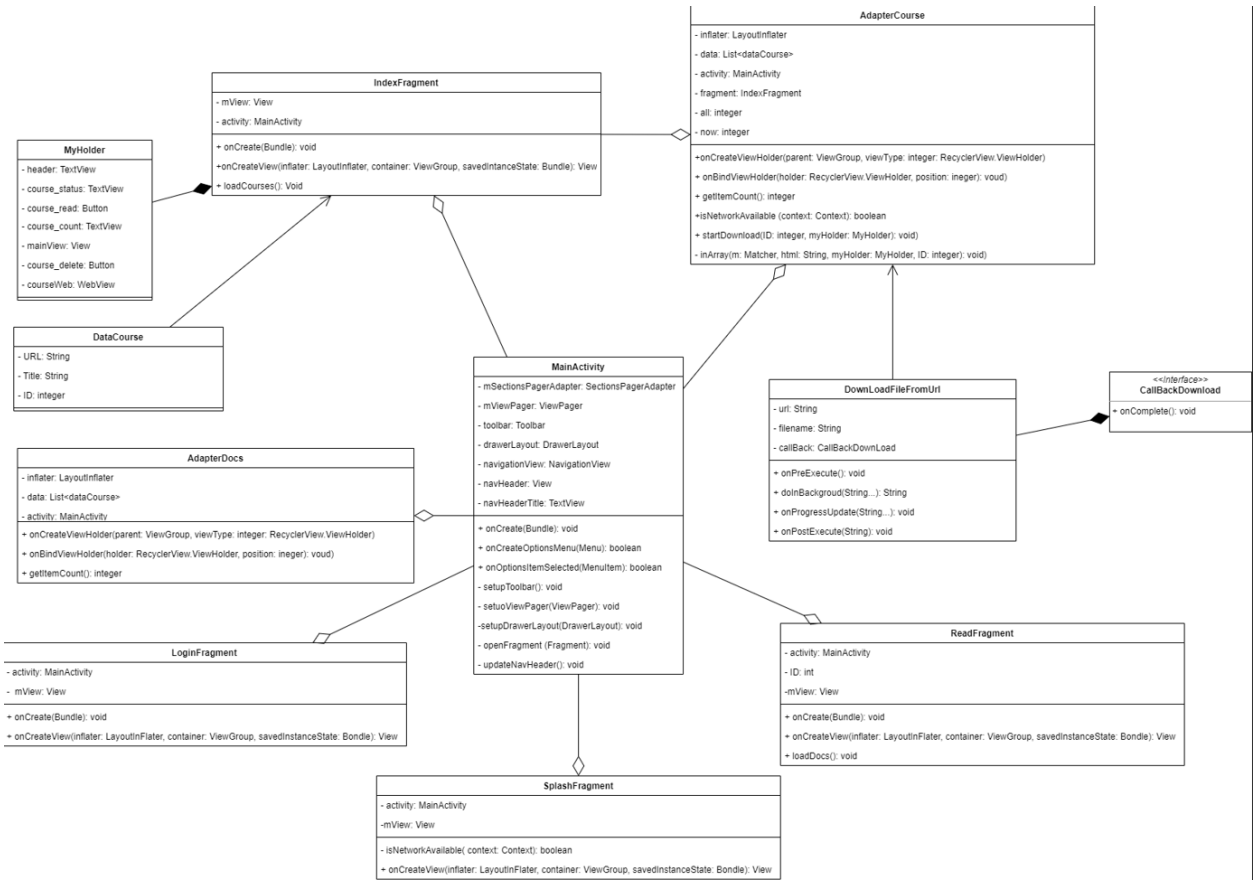


Рисунок 3 – Диаграмма классов основных компонентов

Таким образом, мы рассмотрели структуру проекта и разработали диаграмму классов.

2.2 Архитектура проекта

Архитектура приложения представлена на рисунке 4.

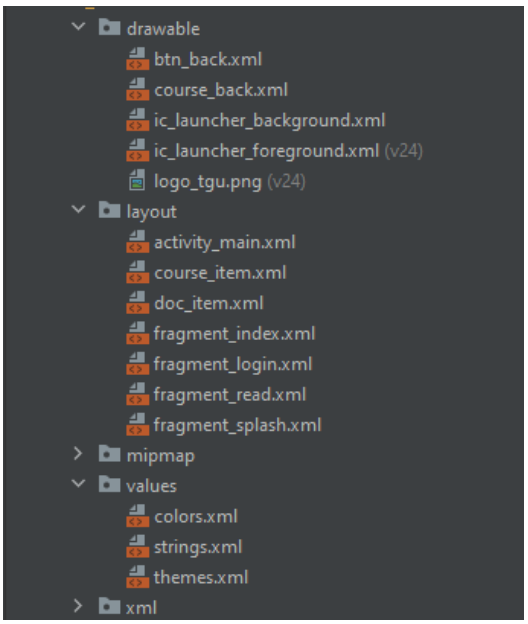


Рисунок 4 – Архитектура приложения

Папка drawable содержит все картинки и XML-фигуры:

– btn_back – «бэкграунд» (задний фон) кнопок;
– course_back – бэкграунд списка курсов;
– ic_launcher_background и ic_launcher_foreground – иконка приложения;
– logo_tgu – логотип «РосДистант».

В папке layout хранятся макеты экранов. В основном, это макеты фрагментов, указанных в начале. Course_item и doc_item – макеты элементов списков Курсы и Документы соответственно.

В папке mipmap хранятся виды иконок приложения для каждой версии Android.

Папка values включает в себя XML-файлы содержания цветов (colors.xml), текстовых строк (string.xml) и тем приложения (themes.xml).

В папке xml содержатся различные настройки в виде XML-разметки.

Для корректной работы приложения запрашиваются следующие разрешения (рисунок 5):

– проверка интернет соединения;

– доступ в интернет;
– запись, работа и чтение файлов во внутренней памяти.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
    tools:ignore="ScopedStorage" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Рисунок 5 – Разрешения в проекте

Проект Android Studio работает с JDK (Java Development Kit) 11.0.12 b Gradle 7.2.1. Минимальный API – 24, компиляционный – 33 (самый новый).

Чтобы не засорять проект библиотеками, были использованы только самые необходимые (рисунок 6):

– appcompat – View-элементы;
– material – работа с материалами Google (Android);
– constraintlayout – ConstraintLayout (корневой элемент экранов);
– junit, espresso-core – работа с тестированием проекта.

```
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.material:material:1.8.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
testImplementation 'junit:junit:4.13.2'
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
```

Рисунок 6 – Подключение библиотек

2.3 Описание программной части

Приложение построено на принципе Одна активность – множество фрагментов. Данный принцип позволяет менять содержимое экрана без задержки для пользователя.

При запуске приложения используется фрагмент LoginFragment, который имеет код, представленный на рисунках 7 и 8.

```
mView.findViewById(R.id.gologin).setOnClickListener(new View.OnClickListener() {
    @SuppressLint("SetJavaScriptEnabled")
    @Override
    public void onClick(View view) {
        EditText edt1 = mView.findViewById(R.id.user_login), edt2 = mView.findViewById(R.id.user_pass);
        String login = edt1.getText().toString();
        String pass = edt2.getText().toString();
        WebView web = mView.findViewById(R.id.courseWeb);
        web.loadUrl("https://edu.rosdistant.ru/login/index.php");
        CookieManager.getInstance().setAcceptThirdPartyCookies(web, b: true);
        WebSettings webSettings = web.getSettings();
        webSettings.setJavaScriptEnabled(true);
        final int[] started = {0};
        web.getSettings().setJavaScriptEnabled(true);
        web.setWebViewClient(new WebViewClient() {
            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                return false;
            }

            @Override
            public void onPageStarted(WebView view, String url, Bitmap favicon) {
                super.onPageStarted(view, url, favicon);

                if(Objects.equals(url, b: "https://edu.rosdistant.ru/my/")) {
                    SharedPreferences sp = activity.getSharedPreferences( name: "UserData", Context.MODE_PRIVATE);
                    SharedPreferences.Editor editor = sp.edit();
                    editor.putString( s: "Login", login);
                    editor.putString( s: "Pass", pass);
                    editor.putInt( s: "isLogin",  i: 1);
                    editor.apply();
                    activity.loadFragment(new SplashFragment(activity),  type: 1);
                }
            }

            @Override
            public void onPageFinished(WebView view, String url) {
                super.onPageFinished(view, url);
```

Рисунок 7 – Фрагмент LoginFragment часть 1

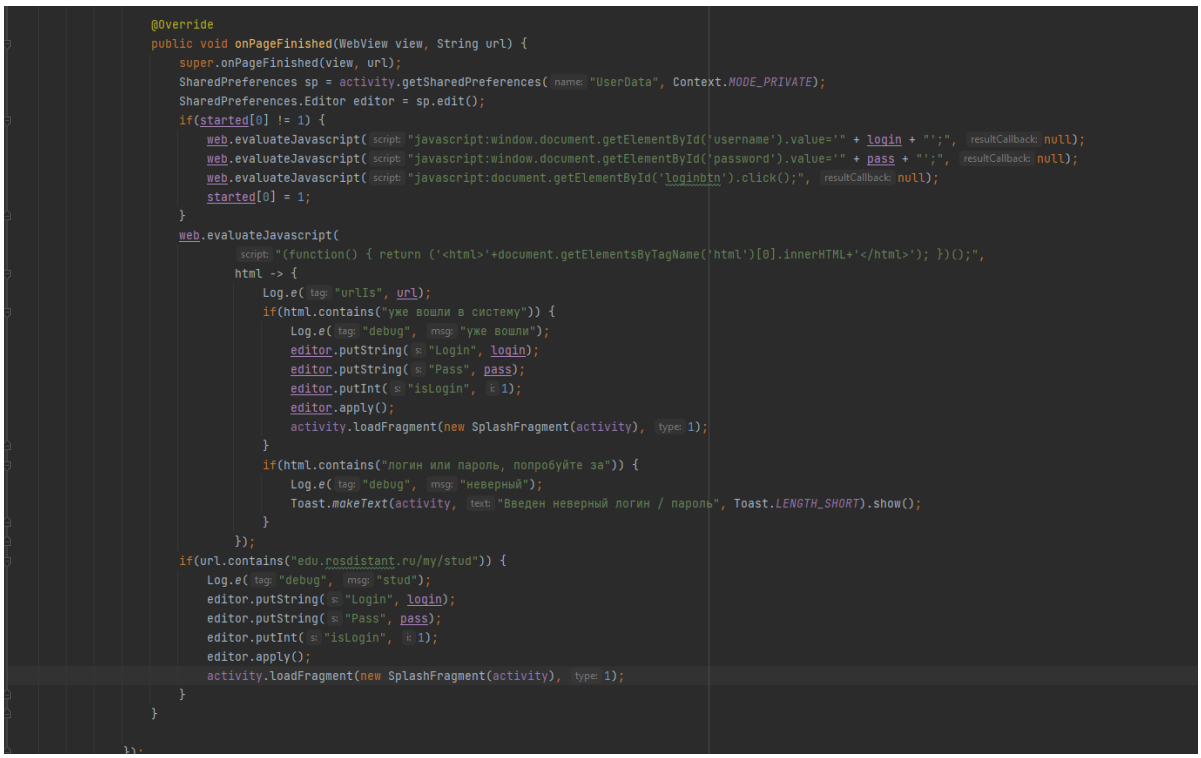


Рисунок 8 – Фрагмент LoginFragment часть 2

При нажатии на кнопку «Войти» мы сначала считываем введенные логин и пароль, а затем с помощью экземпляра браузера открываем страницу авторизации. Данный экземпляр позволяет нам внедрять собственный JavaScript-код. Например, при авторизации мы используем код, который вводит в форму в браузере логин и пароль, а затем нажимаем кнопку «Войти».

Далее при обновлении страниц в этом браузере, мы проверяем адрес страницы. Если авторизация прошла успешно, в приложении открывается фрагмент SplashFragment, если неудачная попытка входа – выводим соответствующий Toast-текст для пользователя.

Также, если мы успешно вошли в аккаунт, сохраняем данные для того, чтобы при дальнейших запусках приложения не вводить их заново.

Переходим к фрагменту SplashFragment, представленному на рисунке 9.

Рисунок 9 – Фрагмент SplashFragment

Здесь мы сначала проверяем, есть ли подключение к интернету на устройстве пользователя с помощью функции isNetworkAvailable (ее функционал отображен на рисунке 10).

```
public static boolean isNetworkAvailable(Context context) {
    ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = cm.getActiveNetworkInfo();
    return networkInfo != null && networkInfo.isConnected();
}
```

Рисунок 10 – Функция проверки подключения

Если подключения к интернету нет – мы загружаем фрагмент IndexFragment. Если есть – загружаем список курсов с личного кабинета в «РосДистант».

Система загрузки списка курсов выглядит следующим образом: для начала мы проверяем, авторизован ли пользователь, если нет – перенаправляем его на предыдущий фрагмент LoginFragment (рисунок 11).

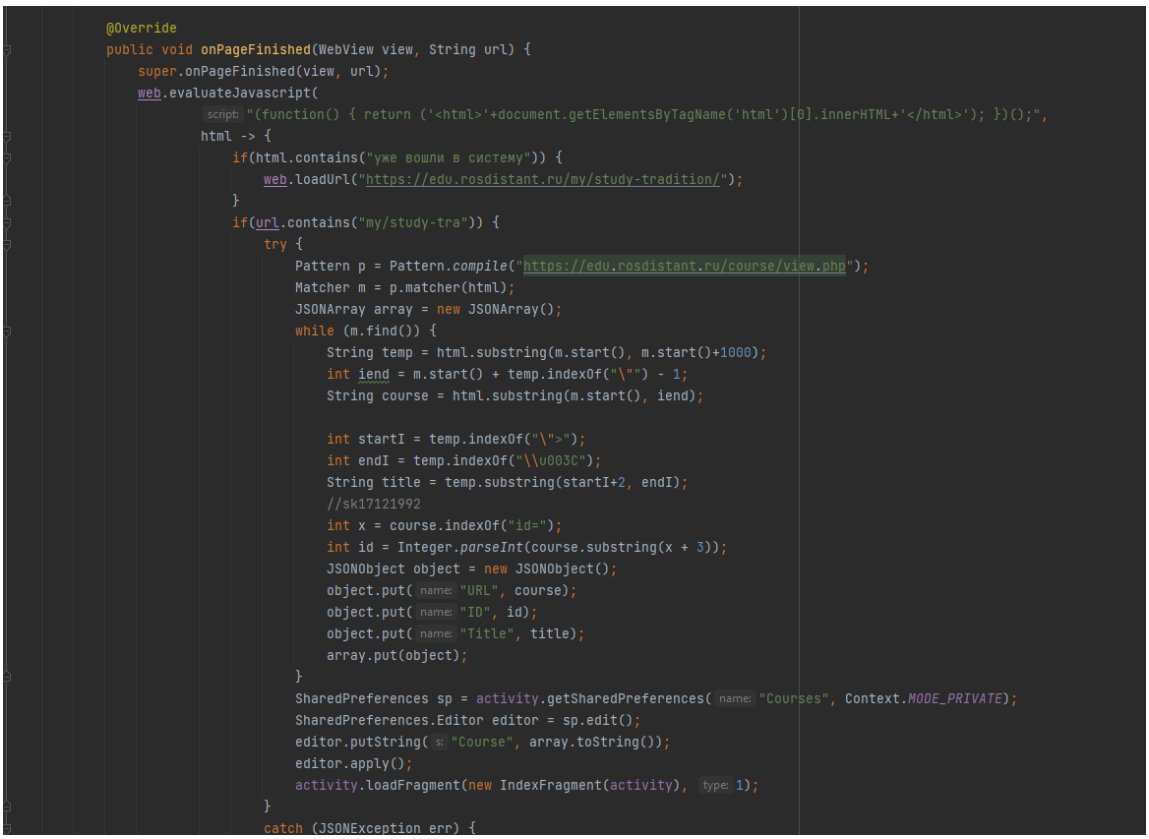


Рисунок 11 – Функция загрузки списка курсов

Затем загружаем содержимое страницы «Мои курсы» и начинаем ее «парсить». Мы берем оттуда все ссылки вида https://edu.rosdistant.ru/course/view.php так как именно они ведут на страницы соответствующих курсов. С помощью парсинга мы получаем: ссылку на курс, его ID и название.

Все полученные данные сохраняются с помощью SharedPreferences в ячейку Courses в виде JSON-массива.

После загрузки списка курсов пользователю открывается IndexFragment – список всех курсов, документы которых можно просмотреть.

Данный фрагмент содержит лишь код загрузки списка в RecyclerView, вся остальная работа со списком производится в adapterCourse, о котором будет рассказано чуть позже.

При загрузке списка (рисунок 12) мы получаем JSON-массив, сохраненный на фрагменте SplashFragment, а затем проходимся по нему с помощью цикла.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    mView = inflater.inflate(R.layout.fragment_index, container,  attachToRoot: false);

    loadCourses();

    return mView;
}
public void loadCourses() {
    RecyclerView rcl = mView.findViewById(R.id.coursesList);

    SharedPreferences sp = activity.getSharedPreferences( name: "Courses", Context.MODE_PRIVATE);
    String response = sp.getString( s: "Course",  s1: "[]");

    ArrayList<dataCourse> data=new ArrayList<>();
    try {
        JSONArray jArray = new JSONArray(response);
        for(int i=0;i<jArray.length();i++){
            JSONObject json_data = jArray.getJSONObject(i);
            dataCourse fishData = new dataCourse();
            fishData.ID = json_data.getInt( name: "ID");
            fishData.Title = json_data.getString( name: "Title");
            fishData.URL = json_data.getString( name: "URL");
            data.add(fishData);
        }
        adapterCourse mAdapter = new adapterCourse(activity, activity.getApplicationContext(), data,  fragment: this);
        rcl.setAdapter(mAdapter);
        rcl.setLayoutManager(new LinearLayoutManager(activity.getApplicationContext()));
        rcl.setOverScrollMode(RecyclerView.OVER_SCROLL_NEVER);
    } catch (JSONException e) {
        Log.e( tag: "CourseException", e.toString());
    }
}
```

Рисунок 12 – Код загрузки

Тело цикла содержит: создание экземпляра JSONObject’a, экземпляра дата-класса, который хранит информацию о курсе (представлен на рисунке 13) и публикация дата-класса в общий массив.

```
public class dataCourse {
    public String URL, Title;
    public int ID;
}
```

Рисунок 13 – Объект JsonObject

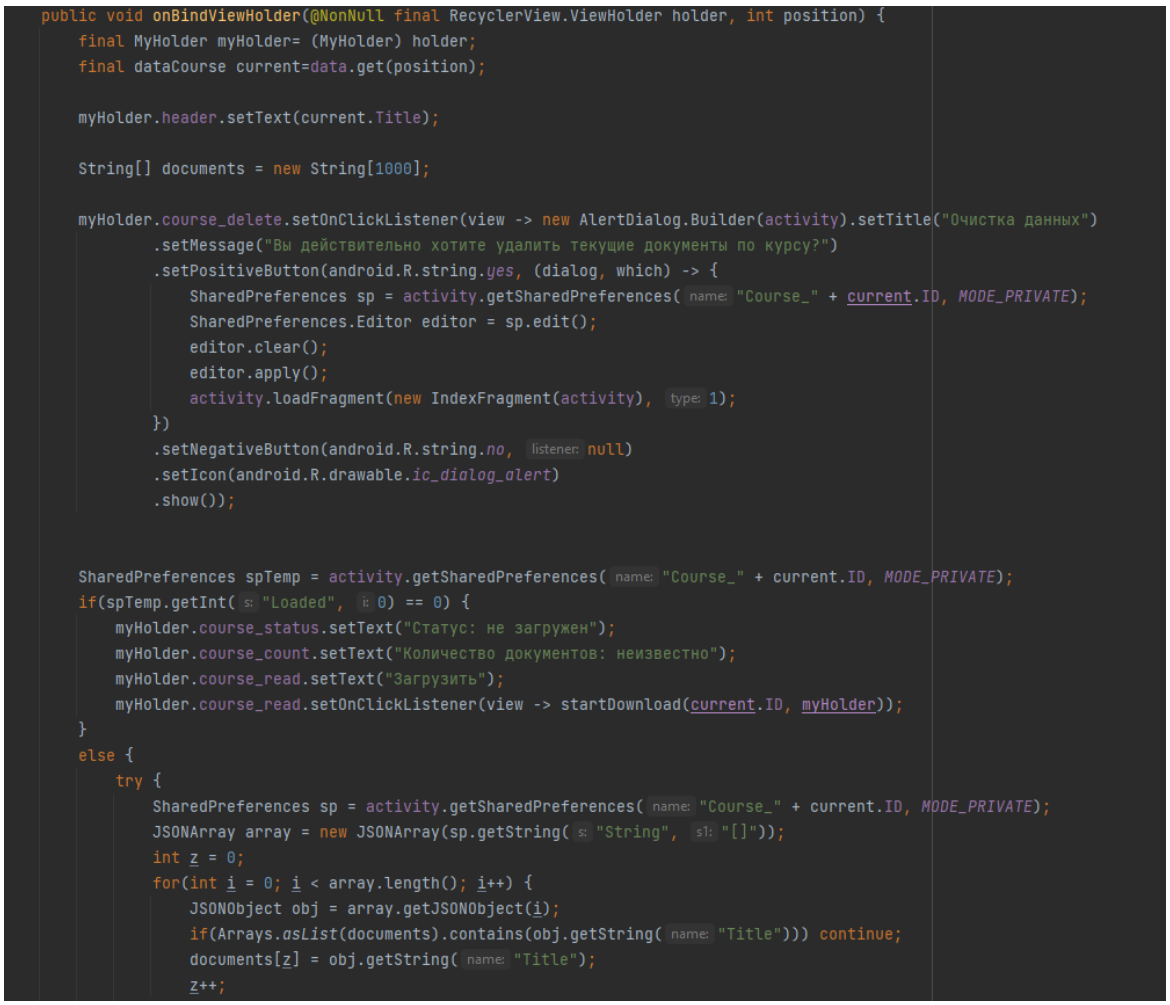
Как можно заметить, дата-класс dataCourse содержит информацию о ссылке, названии и уникальном номере курса.

Далее, после цикла идет создание экземпляра адаптера списка, настройка RecyclerView (установка адаптера, вида расположения элементов, свойства прокрутки) (представлен на рисунке 14).

```
private final LayoutInflater inflater;
List<dataCourse> data;
MainActivity activity;
public adapterDocs(MainActivity activity, Context context, List<dataCourse> data) {
    inflater= LayoutInflater.from(context);
    this.data=data;
    this.activity = activity;
}
@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    Context ctx = inflater.getContext();
    SharedPreferences sp = activity.getSharedPreferences( name: "UserInfo", MODE_PRIVATE);
    ctx.setTheme(sp.getInt( s: "Theme", R.style.Theme_RosDistantOffline));
    View view=inflater.inflate(R.layout.doc_item, parent, attachToRoot: false);
    return new MyHolder(view);
}
```

Рисунок 14 – Настройка RecyclerView

В классе адаптера мы сначала получаем данные для адаптера (экземпляр активности, контекста и список данных). Далее идет метод onCreateViewHolder – он отвечает за установку XML-макета каждому элементу списка (рисунок 15).



```
public void onBindViewHolder(@NonNull final RecyclerView.ViewHolder holder, int position) {
    final MyHolder myHolder= (MyHolder) holder;
    final dataCourse current=data.get(position);

    myHolder.header.setText(current.Title);

    String[] documents = new String[1000];

    myHolder.course_delete.setOnClickListener(view -> new AlertDialog.Builder(activity).setTitle("Очистка данных")
            .setMessage("Вы действительно хотите удалить текущие документы по курсу?")
            .setPositiveButton(android.R.string.yes, (dialog, which) -> {
                SharedPreferences sp = activity.getSharedPreferences( name: "Course_" + current.ID, MODE_PRIVATE);
                SharedPreferences.Editor editor = sp.edit();
                editor.clear();
                editor.apply();
                activity.loadFragment(new IndexFragment(activity),  type: 1);
            })
            .setNegativeButton(android.R.string.no,  listener: null)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show());


    SharedPreferences spTemp = activity.getSharedPreferences( name: "Course_" + current.ID, MODE_PRIVATE);
    if(spTemp.getInt( s: "Loaded",  i: 0) == 0) {
        myHolder.course_status.setText("Статус: не загружен");
        myHolder.course_count.setText("Количество документов: неизвестно");
        myHolder.course_read.setText("Загрузить");
        myHolder.course_read.setOnClickListener(view -> startDownload(current.ID, myHolder));
    }
    else {
        try {
            SharedPreferences sp = activity.getSharedPreferences( name: "Course_" + current.ID, MODE_PRIVATE);
            JSONArray array = new JSONArray(sp.getString( s: "String",  s1: "[]"));
            int z = 0;
            for(int i = 0; i < array.length(); i++) {
                JSONObject obj = array.getJSONObject(i);
                if(Arrays.asList(documents).contains(obj.getString( name: "Title"))) continue;
                documents[z] = obj.getString( name: "Title");
                z++;
```

Рисунок 15 – Метод OnCreateViewHolder

В методе onBindViewHolder мы устанавливаем значения View-элементов каждого элемента списка, например, если курс не загружен, выведем об этом информацию и предложим пользователю его загрузить. При нажатии на кнопку «Очистить данные по курсу» выполним соответствующие действия, но перед этим обязательно спросим пользователя, действительно ли нужно это сделать.

Если курс не загружен – с помощью парсинга, как это было на фрагменте SplashFragment, загрузим нужные документы по курсу с сервера https://edu.rosdistant.ru (рисунок16).

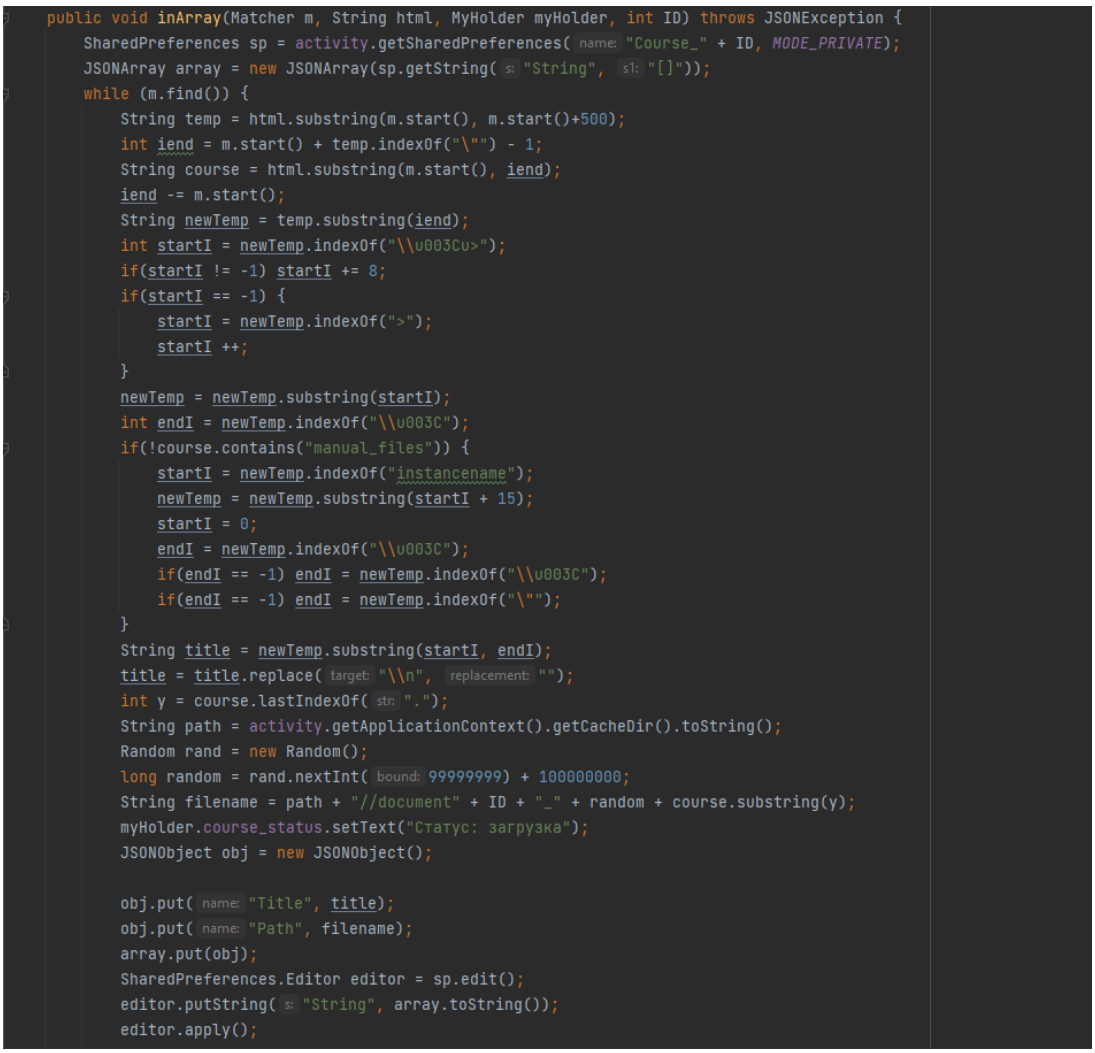
```
try {
    Pattern ptemp = Pattern.compile("https://edu.rosdistant.ru/manual_files");
    Matcher mtemp = ptemp.matcher(html);
    while(mtemp.find()) all++;

    ptemp = Pattern.compile("https://edu.rosdistant.ru/mod/");
    Matcher m2temp = ptemp.matcher(html);
    while(m2temp.find()) all++;


    Pattern p = Pattern.compile("https://edu.rosdistant.ru/manual_files");
    Matcher m = p.matcher(html);
    inArray(m, html, myHolder, ID);

    p = Pattern.compile("https://edu.rosdistant.ru/mod/");
    Matcher m2 = p.matcher(html);
    inArray(m2, html, myHolder, ID);
}
catch (Exception err) {
    Log.e( tag: "MyErr",  msg: err + " | data: " + ID);
}
```

Рисунок 16 – Парсинг

Так как ссылки на документы имеют уже два вида, вместо одного в предыдущем фрагменте, будем использовать их раздельно.

Метод inArray проходится по массиву найденных вхождений подстроки, получает ссылку и заголовок документа.

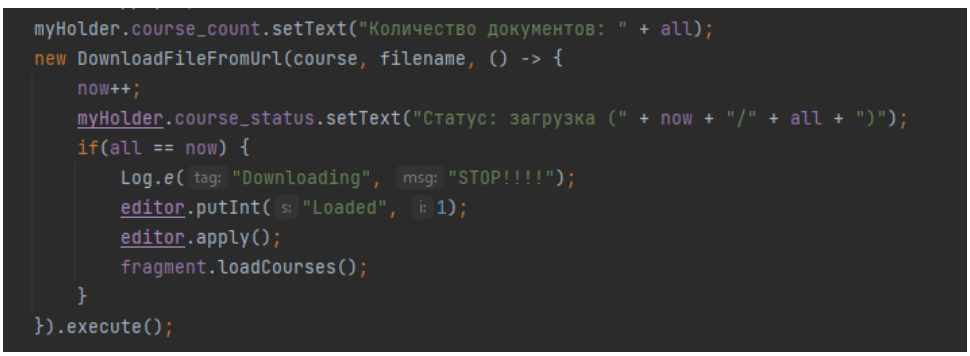
После загрузки файла данные сохраняются в JSON-массив соответствующего курса (рисунок 17).



```
public void inArray(Matcher m, String html, MyHolder myHolder, int ID) throws JSONException {
    SharedPreferences sp = activity.getSharedPreferences( name: "Course_" + ID, MODE_PRIVATE);
    JSONArray array = new JSONArray(sp.getString( s: "String", s1: "[]"));
    while (m.find()) {
        String temp = html.substring(m.start(), m.start()+500);
        int iend = m.start() + temp.indexOf("\"") - 1;
        String course = html.substring(m.start(), iend);
        iend -= m.start();
        String newTemp = temp.substring(iend);
        int startI = newTemp.indexOf("\\u003Cu>");
        if(startI != -1) startI += 8;
        if(startI == -1) {
            startI = newTemp.indexOf(">");
            startI ++;
        }
        newTemp = newTemp.substring(startI);
        int endI = newTemp.indexOf("\\u003C");
        if(!course.contains("manual_files")) {
            startI = newTemp.indexOf("instancename");
            newTemp = newTemp.substring(startI + 15);
            startI = 0;
            endI = newTemp.indexOf("\\u003C");
            if(endI == -1) endI = newTemp.indexOf("\\u003C");
            if(endI == -1) endI = newTemp.indexOf("\"");
        }
        String title = newTemp.substring(startI, endI);
        title = title.replace( target: "\\n", replacement: "");
        int y = course.lastIndexOf( str: ".");
        String path = activity.getApplicationContext().getCacheDir().toString();
        Random rand = new Random();
        long random = rand.nextInt( bound: 99999999) + 100000000;
        String filename = path + "//document" + ID + "_" + random + course.substring(y);
        myHolder.course_status.setText("Статус: загрузка");
        JSONObject obj = new JSONObject();

        obj.put( name: "Title", title);
        obj.put( name: "Path", filename);
        array.put(obj);
        SharedPreferences.Editor editor = sp.edit();
        editor.putString( s: "String", array.toString());
        editor.apply();
```

Рисунок 17 – Использование ссылок

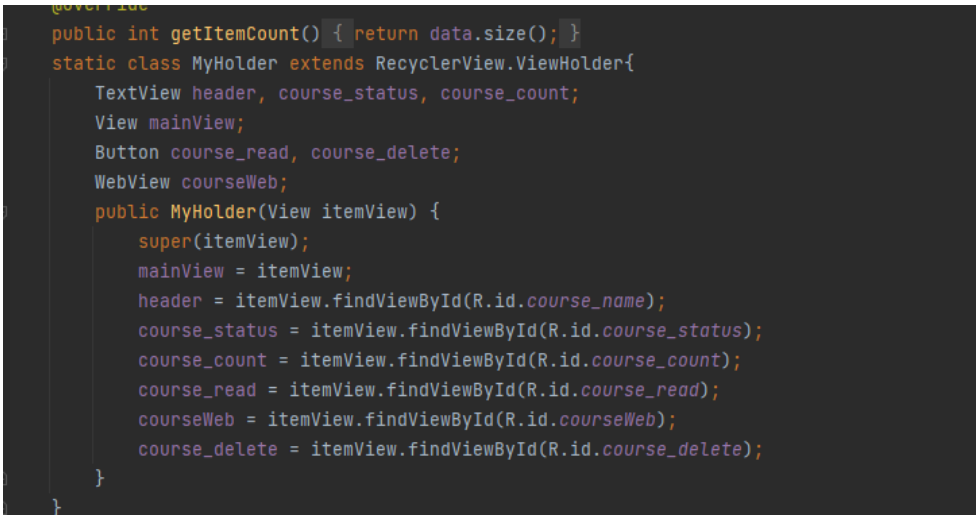
Процесс загрузки файлов виден для пользователя и имеет следующий код (рисунок 18).



```
myHolder.course_count.setText("Количество документов: " + all);
new DownloadFileFromUrl(course, filename, () -> {
    now++;
    myHolder.course_status.setText("Статус: загрузка (" + now + "/" + all + ")");
    if(all == now) {
        Log.e( tag: "Downloading", msg: "STOP!!!!");
        editor.putInt( s: "Loaded", i: 1);
        editor.apply();
        fragment.loadCourses();
    }
}).execute();
```

Рисунок 18 – Процесс загрузки

Также адаптер имеет класс создания нового XML-экземпляра нашей разметки (представлен на рисунке 19).



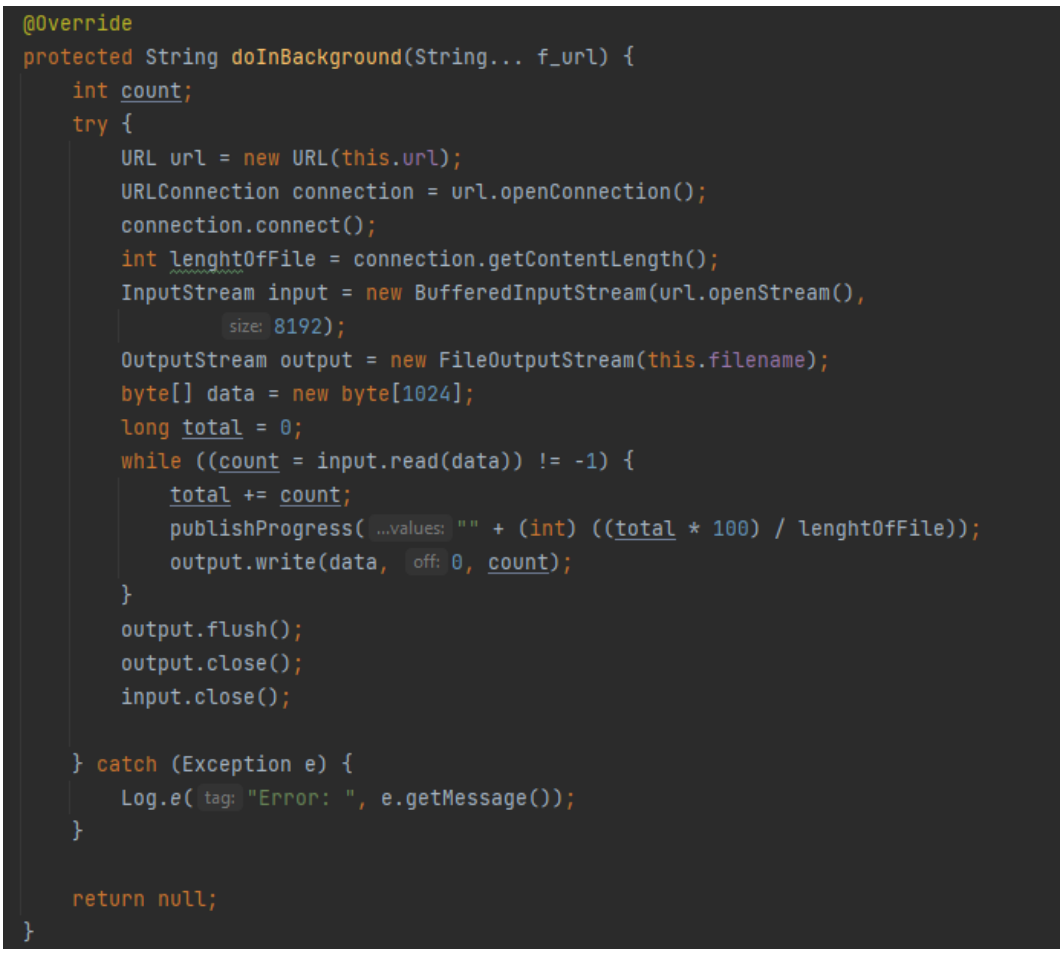
```
public int getItemCount() { return data.size(); }
static class MyHolder extends RecyclerView.ViewHolder{
    TextView header, course_status, course_count;
    View mainView;
    Button course_read, course_delete;
    WebView courseWeb;
    public MyHolder(View itemView) {
        super(itemView);
        mainView = itemView;
        header = itemView.findViewById(R.id.course_name);
        course_status = itemView.findViewById(R.id.course_status);
        course_count = itemView.findViewById(R.id.course_count);
        course_read = itemView.findViewById(R.id.course_read);
        courseWeb = itemView.findViewById(R.id.courseWeb);
        course_delete = itemView.findViewById(R.id.course_delete);
    }
}
```

Рисунок 19 – Создание экземпляра разметки

С его помощью мы можем создать экземпляры текстовых элементов, кнопок и веб-браузера, который как раз-таки и загружает наши файлы.

Стоит отметить, что при загрузке документов из интернета, в панели уведомлений устройства не отображается информация о файлах – это также способствует защите наших документов.

Также имеется дополнительный класс загрузки данных из интернета (рассмотрим рисунок 20).



```
@Override
protected String doInBackground(String... f_url) {
    int count;
    try {
        URL url = new URL(this.url);
        URLConnection connection = url.openConnection();
        connection.connect();
        int lenghtOfFile = connection.getContentLength();
        InputStream input = new BufferedInputStream(url.openStream(),
                size: 8192);
        OutputStream output = new FileOutputStream(this.filename);
        byte[] data = new byte[1024];
        long total = 0;
        while ((count = input.read(data)) != -1) {
            total += count;
            publishProgress( ...values: "" + (int) ((total * 100) / lenghtOfFile));
            output.write(data,  off: 0, count);
        }
        output.flush();
        output.close();
        input.close();

    } catch (Exception e) {
        Log.e( tag: "Error: ", e.getMessage());
    }

    return null;
}
```

Рисунок 20 – Создание ссылки

Здесь мы создаем URL-экземпляр нашей ссылки, получаем соединение и загружаем файл считывая его по 8192 символа.

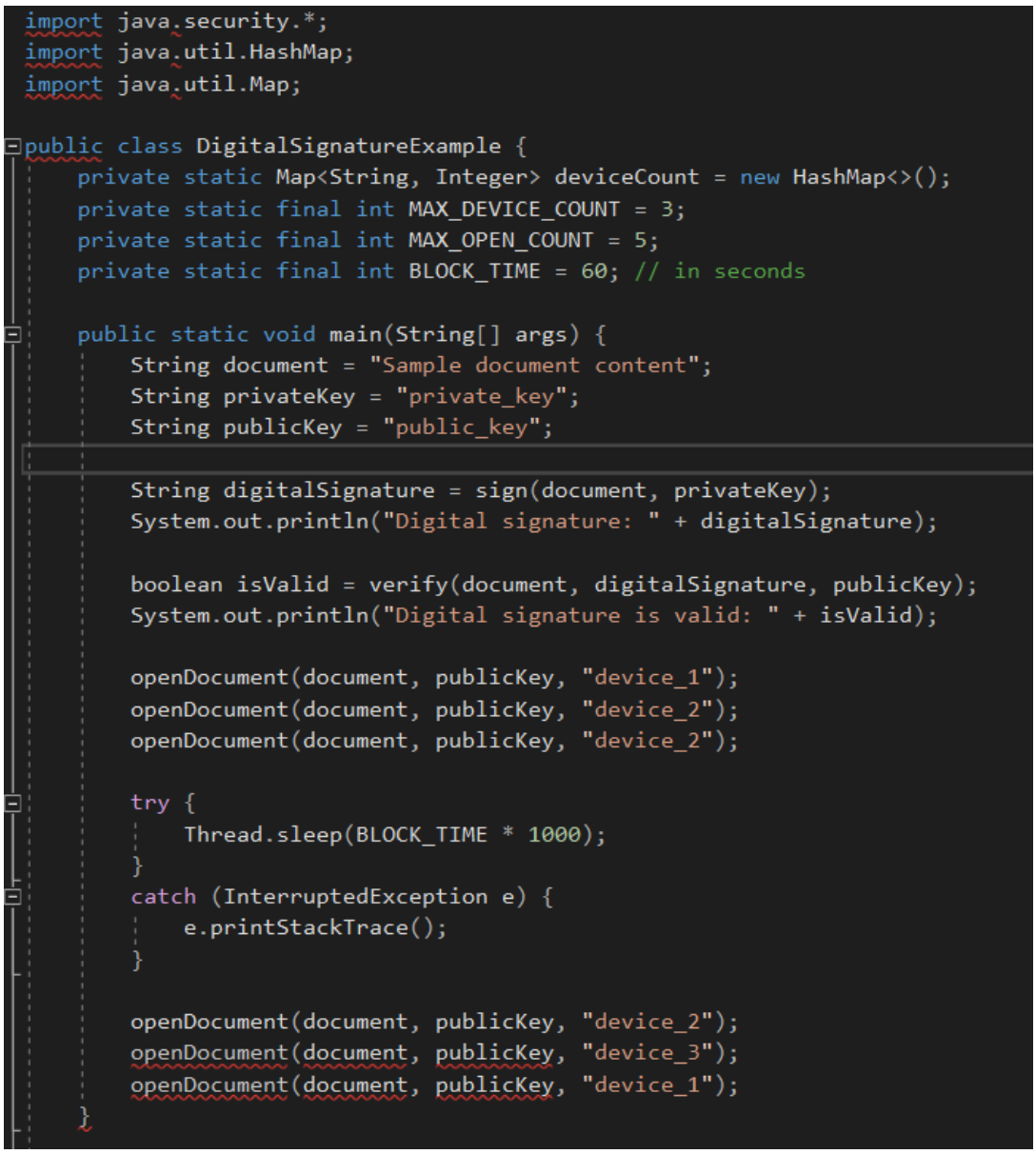
Вывод списка документов соответствующего курса аналогичен выводу списка самих курсов. Единственное различие заключается в функционале загрузки данных каждого элемента списка, в нашем случае будет выполняться следующий код (рисунок 21).

Рисунок 21 – Вывод информации

Здесь мы сначала выводим информацию об элементе, а затем, при нажатии на кнопку создаем Uri-экземпляр файла и открываем новую активность с помощью Intent.ACTION_VIEW (действие просмотра) соответствующего документа. В большинстве случаев, система будет запрашивать выбор приложения для открытия файла.

2.4 Применение цифровой подписи

Реализация математической модели защиты учебников от копирования с использованием цифровой подписи показана на рисунках 22, 23 и 24.



```java
import java.security.*;
import java.util.HashMap;
import java.util.Map;

public class DigitalSignatureExample {
    private static Map<String, Integer> deviceCount = new HashMap<>();
    private static final int MAX_DEVICE_COUNT = 3;
    private static final int MAX_OPEN_COUNT = 5;
    private static final int BLOCK_TIME = 60; // in seconds

    public static void main(String[] args) {
        String document = "Sample document content";
        String privateKey = "private_key";
        String publicKey = "public_key";

        String digitalSignature = sign(document, privateKey);
        System.out.println("Digital signature: " + digitalSignature);

        boolean isValid = verify(document, digitalSignature, publicKey);
        System.out.println("Digital signature is valid: " + isValid);

        openDocument(document, publicKey, "device_1");
        openDocument(document, publicKey, "device_2");
        openDocument(document, publicKey, "device_2");

        try {
            Thread.sleep(BLOCK_TIME * 1000);
        }
        catch (InterruptedException e) {
            e.printStackTrace();
        }

        openDocument(document, publicKey, "device_2");
        openDocument(document, publicKey, "device_3");
        openDocument(document, publicKey, "device_1");
    }
```

Рисунок 22 – Применение цифровой подписи (часть 1)

```
private static String sign(String document, String privateKey) {
    try {
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initSign(getPrivateKey(privateKey));
        signature.update(document.getBytes());
        byte[] digitalSignature = signature.sign();
        return bytesToHex(digitalSignature);
    }
    catch (Exception e) {
        e.printStackTrace();
        return "";
    }
}

private static boolean verify(String document, String digitalSignature, String publicKey) {
    try {
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initVerify(getPublicKey(publicKey));
        signature.update(document.getBytes());
        return signature.verify(hexToBytes(digitalSignature));
    }
    catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

private static void openDocument(String document, String publicKey, String device) {
    int openCount = deviceCount.getOrDefault(device, 0);
    if (openCount >= MAX_OPEN_COUNT) {
        System.out.println("Document is blocked on device " + device);
        return;
    }
    deviceCount.put(device, openCount + 1);
```

Рисунок 23 – Применение цифровой подписи (часть 2)

```
        }
        else {
            System.out.println("Invalid digital signature on device " + device);
        }
    }

    private static PrivateKey getPrivateKey(String privateKey) throws Exception {
        byte[] privateKeyBytes = hexToBytes(privateKey);
        PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(privateKeyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePrivate(keySpec);
    }

    private static PublicKey getPublicKey(String publicKey) throws Exception {
        byte[] publicKeyBytes = hexToBytes(publicKey);
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(publicKeyBytes);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        return keyFactory.generatePublic(keySpec);
    }

    private static String bytesToHex(byte[] bytes) {
        StringBuilder sb = new StringBuilder();
        for (byte b : bytes) {
            sb.append(String.format("%02X", b));
        }
        return sb.toString();
    }

    private static byte[] hexToBytes(String hex) {
        int len = hex.length();
        byte[] bytes = new byte[len / 2];
        for (int i = 0; i < len; i += 2) {
            bytes[i / 2] = (byte)((Character.digit(hex.charAt(i), 16) << 4)
                + Character.digit(hex.charAt(i + 1), 16));
        }
        return bytes;
    }
```

Рисунок 24 – Применение цифровой подписи (часть 3)

В данном фрагменте используется класс Signature для создания и проверки цифровой подписи, а также PrivateKey и PublicKey для работы с закрытым и открытым ключами соответственно. Также в коде использованы методы bytesToHex и hexToBytes для преобразования байтов в шестнадцатеричную строку и обратно.

Метод sign создает цифровую подпись для указанного документа с использованием указанного закрытого ключа и алгоритма "SHA256withRSA". Метод verify проверяет действительность цифровой подписи для указанного документа и открытого ключа.

Метод openDocument открывает документ на указанном устройстве с помощью проверки цифровой подписи. Если количество открытий документа на указанном устройстве достигло максимального значения или цифровая подпись недействительна, то документ не открывается.

Дополнительные ограничения на количество устройств и количество открытий документа реализованы с помощью использования deviceCount, хранящей количество открытий документа на каждом устройстве. Если количество открытий на устройстве достигло максимального значения, то документ блокируется на этом устройстве. Если количество открытий документа достигло максимального значения, то документ блокируется на определенный период времени BLOCK_TIME.

2.5 Реализация сжатия учебников алгоритмом GZIP

Для сжатия и разжатия данных учебников Росдистанта можно использовать алгоритм GZIP, который предоставляет стандартный пакет java.util.zip.

Для сжатия данных учебника достаточно использовать объект GZIPOutputStream, который позволяет записывать данные в сжатом виде, например, в файл или в буфер. Для этого можно использовать следующий код (рисунок 25).

```
import java.util.zip.GZIPOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.OutputStream;

// ...

public byte[] compressData(byte[] data) {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutp
utStream();
    try {
        OutputStream gzipOutputStream = new GZIPOutputStream(byteAr
rayOutputStream);
        gzipOutputStream.write(data);
        gzipOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return byteArrayOutputStream.toByteArray();
}
```

Рисунок 25 – Сжатие данных алгоритмом GZIP

Этот метод принимает массив байтов data, который представляет собой содержимое учебника, и возвращает сжатый массив байтов.

Для разжатия данных учебника можно использовать класс GZIPInputStream. Этот класс позволяет читать сжатые данные из файла или буфера и возвращать их в исходном виде. Фрагмент кода для разжатия данных показан на рисунке 26.

```
import java.util.zip.GZIPInputStream;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ByteArrayOutputStream;

// ...

public byte[] decompressData(byte[] data) {
    ByteArrayInputStream byteArrayInputStream = new ByteArrayInputS
tream(data);
    InputStream gzipInputStream = null;
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutp
utStream();
    byte[] buffer = new byte[1024];
    int len;
    try {
        gzipInputStream = new GZIPInputStream(byteArrayInputStrea
m);
        while ((len = gzipInputStream.read(buffer)) != -1) {
            byteArrayOutputStream.write(buffer, 0, len);
        }
        gzipInputStream.close();
        byteArrayOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return byteArrayOutputStream.toByteArray();
}
```

Рисунок 26 – Разжатие данных

Важно отметить, что при сжатии данных алгоритмом GZIP размер данных может значительно уменьшиться, что позволяет сократить объем передаваемых данных по сети.

Выводы по второму разделу

В данном разделе были описаны структура и архитектура мобильного приложения. Так же показана реализация цифровой подписи для защиты учебников от копирования и реализация сжатия учебников алгоритмом GZIP. Описана и реализована программная часть. Разработка выполнялась на платформе Android Studio. Программный код написан на языке программирования JAVA.

3 Тестирование мобильного приложения

3.1 Руководство пользователя

При первом запуске приложения нам необходимо авторизоваться, вводим наши данные (рисунок 27).

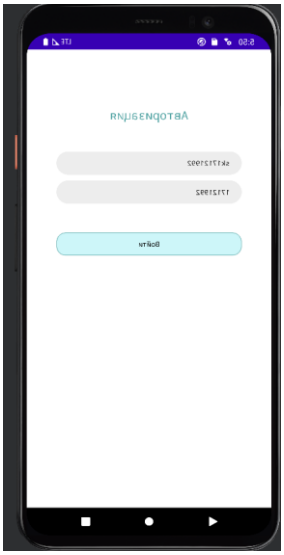


Рисунок 27 – Авторизация

Далее подождем, пока система подгрузит необходимые данные.

После того, как данные будут загружены, мы получим список курсов (рисунок 28).

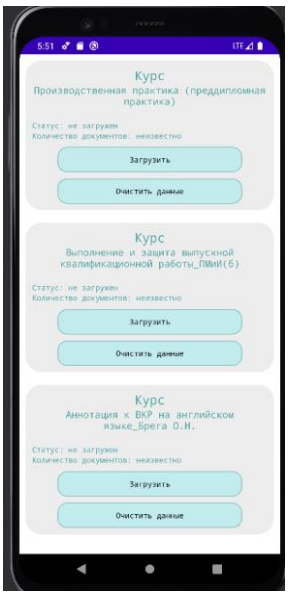


Рисунок 28 – Список курсов

Документы каждого курса можно отдельно загрузить и очистить. Проверим это на примере курса «Производственная практика» (представлено на рисунках 29 и 30).

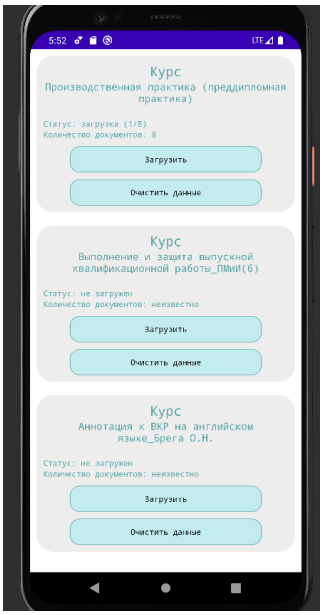


Рисунок 29 – Загрузка

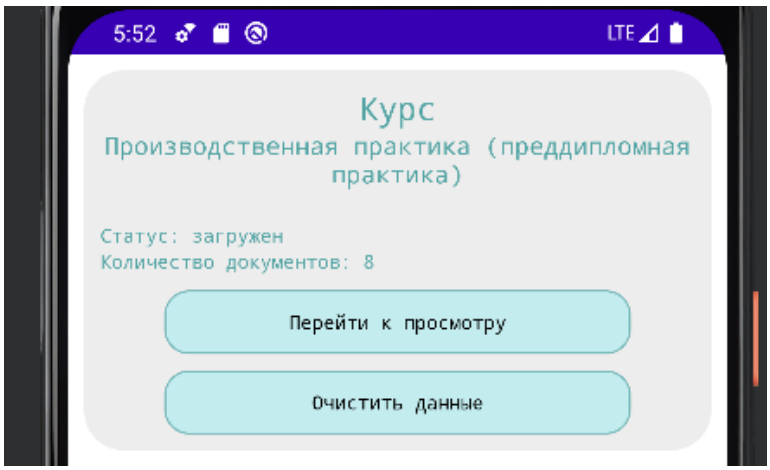


Рисунок 30 – Выбор действия

Далее переходим к просмотру учебных материалов (рисунок 31).

Рисунок 31 – Переход к просмотру

Перед нами открылся список всех документов этого курса, попробуем открыть первый (рисунок 32).

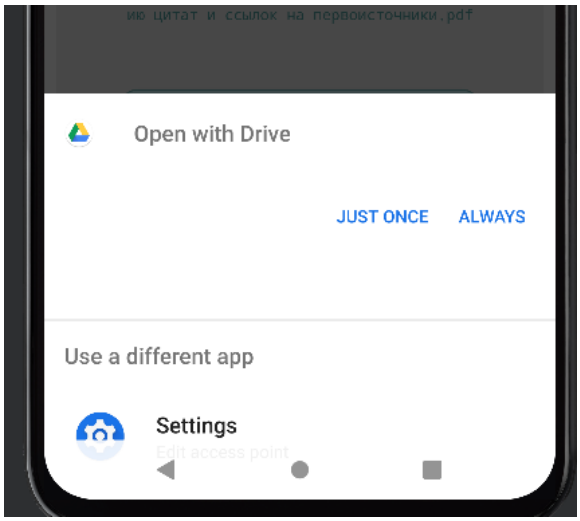


Рисунок 32 – Открытие документа

Система нам предлагает выбор приложения, с помощью которого мы откроем этот PDF-файл. Открываем с помощью предлагаемого Google Drive (рассмотрим на рисунке 33).

Рисунок 33 – Документ

Как мы видим, изначально файл был корректно загружен и его возможно прочитать.

Так-как перед нами стоит задача разработки приложения оффлайн-просмотра документов, проверим, работает ли система без интернета.

При входе в приложение с выключенным интернет-соединением мы получаем следующее сообщение (рисунок 34).

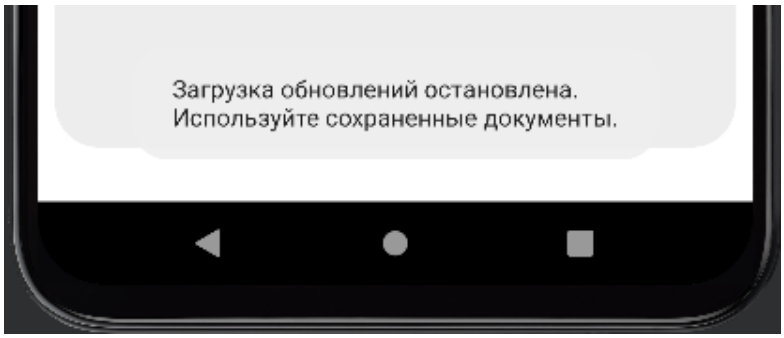


Рисунок 34 – Сообщение об остановке обновлений

3.2 Сравнительный анализ

Для проведения сравнительного анализа мы использовали 5 учебников с Росдистанта. Сравнение проводилось на мобильных телефонах Honor P30 Lite и Samsung A21 через Android Studio.

Скорость загрузки учебников с Росдистанта до и после сжатия зависит от нескольких характеристик телефонов Honor P30 Lite и Samsung A21:

- процессор: Оба телефона оснащены процессорами среднего уровня. Honor P30 Lite имеет 8-ядерный процессор Kirin 710, а Samsung A21 оборудован 8-ядерным процессором Exynos 9611. Более мощный процессор может обеспечить более быструю загрузку учебников;
- память: Оперативная память (RAM) и внутренняя память (ROM) также влияют на скорость загрузки учебников. Honor P30 Lite имеет 4 ГБ оперативной памяти и 128 ГБ встроенной памяти, а Samsung

A21 имеет 3 ГБ оперативной памяти и 32 ГБ встроенной памяти. Большая оперативная память и встроенная память могут ускорить загрузку учебников;

- технология связи: Honor P30 Lite поддерживает LTE и Wi-Fi 802.11 a/b/g/n/ac, в то время как Samsung A21 поддерживает только LTE и Wi-Fi 802.11 b/g/n. Технология связи может влиять на скорость загрузки учебников в зависимости от того, какой тип связи используется;
- разрешение экрана: Honor P30 Lite имеет экран с разрешением 1080 x 2312 пикселей, а Samsung A21 - 720 x 1600 пикселей. Более высокое разрешение экрана может потребовать больше времени на загрузку изображений и других элементов учебника;
- версия операционной системы: Оба телефона работают на операционной системе Android, но Honor P30 Lite работает на более новой версии Android 10, в то время как Samsung A21 работает на Android 9. Более новая версия операционной системы может обеспечить более быструю загрузку учебников.

Таким образом, скорость загрузки учебников с Росдистанта до и после сжатия зависит от многих факторов, включая процессор, память, технологию связи, разрешение экрана и версию операционной системы.

Для измерения времени загрузки учебника мы воспользовались методом System.currentTimeMillis(). Чтобы определить объем учебника был использован метод length() класса File, который возвращает размер файла в байтах, далее перевели размер файла в киллобайты.

Сравнение объема и времени загрузки учебников до сжатия и после сжатия алгоритмом GZIP описано в таблице 1.

Таблица 1 – Сравнение объема и времени загрузки учебников до сжатия и после сжатия алгоритмом GZIP.

Учебник	Размер до сжатия (в Кб)	Размер после сжатия (в Кб)	Время загрузки без сжатия на Honor P30 Lite (в секундах)	Время загрузки без сжатия на Samsung A21 (в секундах)	Время загрузки после сжатия на Honor P30 Lite (в секундах)	Время загрузки после сжатия на Samsung A21 (в секундах)
1	765	245	25	30	12	15
2	657	218	21	26	10	13
3	876	290	28	32	15	18
4	234	106	12	15	6	9
5	567	188	18	22	9	11

В таблице указаны размеры учебников до и после сжатия, время загрузки на каждом телефоне до и после сжатия, а также процентное сжатие каждого учебника. В данном примере, размер каждого учебника уменьшился после сжатия алгоритмом GZIP. Время загрузки также уменьшилось на каждом телефоне после сжатия.

Для вычисления общего объема занимаемой памяти, воспользуемся формулой:

$$S = s_1 + s_2 + \ldots + s_n. \quad (14)$$

Общий объем занимаемой памяти до сжатия:

$$S = 765 + 657 + 876 + 234 + 567 = 3099 \text{ Кб}. \quad (15)$$

Общий объем занимаемой памяти после сжатия алгоритмом GZIP:

$$S = 245 + 218 + 290 + 106 + 188 = 1047 \text{ Кб}. \quad (16)$$

Мы видим, что объем занимаемой памяти после сжатия алгоритмом GZIP, уменьшился в ≈ 2,96 раза, что является хорошим показателем.

Для вычисления общего времени загрузки документов, воспользуемся формулой:

$$T_{general} = T_1 + T_2 + \ldots + T_l. \quad (17)$$

Общее время загрузки документов на Honor P30 Lite до сжатия:

$$T_{general} = 25 + 21 + 28 + 18 + 20 = 112 \text{ секунд}. \quad (18)$$

Общее время загрузки документов на Honor P30 Lite после сжатия алгоритмом GZIP:

$$T_{general} = 12 + 10 + 15 + 6 + 9 = 52 \text{ секунд}. \quad (19)$$

Мы видим, что время загрузки документа после сжатия алгоритмом GZIP, уменьшилось в ≈ 2,15 раза.

Общее время загрузки документов на Samsung A21 до сжатия:

$$T_{general} = 30 + 26 + 32 + 15 + 22 = 125 \text{ секунд}. \quad (20)$$

Общее время загрузки документов на Samsung A21 после сжатия алгоритмом GZIP:

$$T_{general} = 15 + 13 + 18 + 9 + 11 = 66 \text{ секунд}. \quad (21)$$

Время загрузки документа после сжатия алгоритмом GZIP, уменьшилось в ≈ 1,9 раза, что так же является хорошим показателем.

Таким образом, мы применили математическую модель офлайн просмотра учебников Росдистанта для вычисления, занимаемого устройством места и времени загрузки.

Выводы по третьему разделу

В данном разделе был описан запуск программы и выполнение его функций в виде руководства пользователя по разработанному приложению. Далее проведен сравнительный анализ объема занимаемой памяти и времени загрузки учебников до сжатия и после сжатия алгоритмом GZIP, с помощью математических формул. Мы видим, что применение алгоритма GZIP значительно уменьшает объем используемой памяти и время загрузки учебников.

Заключение

В рамках выпускной квалификационной работы был разработан алгоритм оффлайн просмотра учебников Росдистанта, который позволит пользователям просматривать образовательные материалы, даже если они не имеют доступа к сети Интернет. По данному алгоритму было создано мобильное приложение.

Была выбрана операционная система мобильного устройства для работы приложения, а также среда разработки.

В ходе выполнения выпускной квалификационной работы были решены следующие задачи:

- проанализирована предметная область;
- проанализированы исходные данные с сайта вуза и подготовить эскизы и проект продукта;
- рассмотрены алгоритмы защиты данных от копирования;
- рассмотрены алгоритмы сжатия данных;
- реализовано мобильное приложение под управлением выбранной операционной системы;
- применен выбранный алгоритм защиты данных от копирования;
- применен выбранный алгоритм сжатия данных;
- наполно данными приложение, взятыми с официального сайта;
- протестировано разработанное приложение.

Результатом работы является создание мобильного приложения для просмотра учебников в кабинете Росдистанта.

Разработанное приложение выполняет все основные требования, которые были представлены выше, и может быть использовано студентами Росдистанта для просмотра учебников онлайн и оффлайн. Таким образом, следует считать, что задачи выпускной квалификационной работы полностью решены и цель исследования достигнута.

Список использованной литературы и использованных источников

1. 10 принципов разработки мобильных интерфейсов. – CMS magazine [Электронный ресурс] – URL: https://cmsmagazine.ru/journal/items-10-principles- mobile-interface-design/ (дата обращения: 22.03.2023).
2. Skillbox [Электронный ресурс] - URL: https://skillbox.ru/media/design/ (дата обращения: 16.03.2023).
3. Гриффитс Р. Д. Head First. Программирование для Android [Текст]
4. Дерси Л. Создание приложений для Android за 24 часа [Текст] / Л. Л. Дерси – Москва: Эксмо, 2015. – 464 с.
5. Дональд Н. – Дизайн привычных вещей – 2006, 2-е изд., С. 28.
6. Дэрси Л. Разработка приложений для Android-устройств. Базовые принципы [Текст] /Л. Дэрси, Ш. Кондер – Том 1. – Москва: Эксмо, 2014. – 598 с.
7. Использование мобильных телефонов в разных странах [Электронный ресурс]. – Режим доступа: http://www.cossa.ru/152/37433/ (дата обращения: 15.03.2023).
8. Использование мобильных устройств [Электронный ресурс]. – Режим доступа: http://www.wi-life.ru/stati/wi-fi/marketingovye-stati-2/mobile-devices-use-aruba-research-results (дата обращения: 15.03.2023).
9. История возникновения интернета. – Web Building [Электронный ресурс] – URL: https://webbuilding.com.ua/rus/blog/istoriya-interneta (дата обращения: 16.03.2021).
10. Лебедев А.А. – Ководство – Москва: Студия Артемия Лебедева, 6-е изд., 2020, §136.
11. Майер Р. Программирование приложений для планшетных компьютеров и смартфонов [Текст] / Р. Майер – Москва: Эксмо, 2013. – 816 с.
12. Мобильное приложение, что такое, определение, новости, статьи, видео [Электронный ресурс]. – Режим доступа: https://indicator.ru/tags/ mobilnoe-prilozhenie/ (дата обращения: 15.03.2023).

13. Мобильное приложение. – Calltouch [Электронный ресурс] – URL: https://www.calltouch.ru/glossary/chto-takoe-mobilnoe-prilozhenie-i-zachem-ono-mozhet-potrebovatsya/ (дата обращения: 22.03.2023).

14. Опрос: более 60% жителей городов-миллионников в России пользуются доставкой еды. – Тасс [Электронный ресурс] – URL: https://tass.ru/obschestvo/7309655 (дата обращения: 22.03.2023).

15. Основные этапы разработки мобильных приложений [Электронный ресурс] – Режим доступа: https://spark.ru/startup/componentix/blog/4499/ osnovnie-etapi-razrabotki-mobilnih-prilozhenij (дата обращения: 15.03.2023).

16. Панина А.А., Сычева Л.А. – Психология цвета – Cyber Leninka [Электронный ресурс] – URL: https://cyberleninka.ru/article/n/psihologiya-tsveta (дата обращения: 26.03.2023).

17. Правильно выбираем конкурентов. – POWERBRANDING [Электронный ресурс] – URL: http://powerbranding.ru/competition/kak-opredelit-konkurentov (дата обращения: 05.03.2023).

18. Приложения в Google Play – MSU-Life [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.springlabs.msulife&hl=ru (дата обращения: 15.03.2023).

19. Приложения в Google Play – НГУЭУ [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=burov.nsuem (дата обращения: 15.03.2023).

20. Приложения в Google PlayMTI Mobile [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=ru.edu.mti.mtimobile (дата обращения: 15.03.2023).

21. Принципы дизайна мобильных приложений от Google. – IGOR JAZOV [Электронный ресурс] – URL: https://www.blog.jazov.com/mobile-design/principy-dizajna-mobilnyx-prilozhenij-ot-google.html (дата обращения: 23.03.2023).

22. Сравнительная характеристика технологий .Net и .JAVA [Электронный источник]. Режим доступа: https://studfile.net/preview/3675377/, дата доступа 08.06.2023.

23. Android Studio. [Электронный источник]. Режим доступа: https://deviceadvice.ru/kto-razrabatyvaet-android-studio/, дата доступа 08.06.2023.

24. Beginner’s Guide To Android App Development. [Электронный источник]. Режим доступа: https://medium.com/software-incubator/beginners-guide-to-android-app-development-6f3100e7968, дата доступа 12.04.2023.

25. ChaCha20 and Poly1305 for IETF Protocols. [Электронный источник]. Режим доступа: https://www.rfc-editor.org/rfc/rfc7539, дата доступа 09.05.2023.

26. Cohen Ryan, Wang Tao. GUI Design for Android Apps. NY: Apress, 2016. – 147 p.

27. Franceschi Herve J. Android App Development. NY: Jones & Bartlett Learning, 2018. – 374 p.

28. How to Get Started with Android Development. [Электронный источник]. Режим доступа: https://www.freecodecamp.org/news/intro-to-android-development/, дата доступа 12.04.2023.

29. Learn Java For Android App Development – A Complete Guide. [Электронный источник]. Режим доступа: https://www.geeksforgeeks.org/learn-java-for-android-app-development-a-complete-guide/, дата доступа 01.04.2023

30. Spath Peter. Pro Android with Kotlin: Developing Modern Mobile Apps with Kotlin and Jetpack, 2nd Edition. NY: Apress, 2022. - 881 p.