Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский политехнический университет»

Кафедра «Инфокогнитивные технологии»

Образовательная программа «Корпоративные информационные системы»

Отчет по курсовому проекту

по дисциплине «Инженерное проектирование»

Тема: «Доставка продуктов»

**Выполнил:**

Студент группы 191-361

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Афанасьев К.М.

подпись, дата

**Принял:**

Старший преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Васильев Д. Б.

подпись, дата

Москва 2022

ВВЕДЕНИЕ

**Тема курсового проекта:** Видео картотека.

В рамках курсового проекта было реализовано мобильное приложение для поиска и просмотра видеофайлов различного формата, хранящихся на устройстве.

Используемый стек технологий:

* Java
* Android SDK
* SQLite
* Gradle

ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ

В состав основных функций приложения входят:

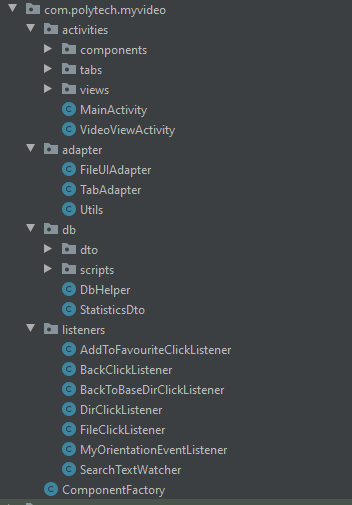
* поиск видеофайлов по проводнику;
* навигация по проводнику;
* просмотр видеофайлов, с возможностью разворота видео на полный экран;
* добавление папок и видеофайлов в избранное с возможностью последующего их просмотра;
* поиск по избранным.

В состав дополнительных возможностей приложения входят:

* ведение и просмотр истории просмотров c информацией о дате обращения;
* поиск по истории;
* ведение и просмотр статистики использования приложения с информацией о сессиях использования приложения, количеством просмотренных видео, времени использования приложения.

СТРУКТУРА ПРОЕКТА

Структура проекта выглядит следующим образом:



Основные пакеты приложения:

* activities – содержит классы отвечающие за логику работы всех основных страниц приложения, фрагментов и других компонентов осуществляющих взаимодействие с пользователем;
* adapter – вспомогательные классы для заполнения и обработки данных для activities;
* db – содержит логику создания и взаимодействия с локальной базой данных;
* listeners – содержит кастомные классы различных слушателей, которые навешиваются на различные пользовательские компоненты (например, кнопки)

Помимо этого, в проекте дополнительно присутствует папка ресурсов приложения (res), которая содержит файлы разметки пользовательских страниц и других компонентов, а также стили, изображения, строки и цвета, используемые в приложении.

Также в проекте присутствует файл манифеста, определяющий информацию о приложении - название, версию, иконки, какие разрешения приложение использует, регистрирует все используемые классы activity, сервисы и т.д. Данный файл можно найти в проекте в папке manifests.

СТРУКТУРА БАЗЫ ДАННЫХ

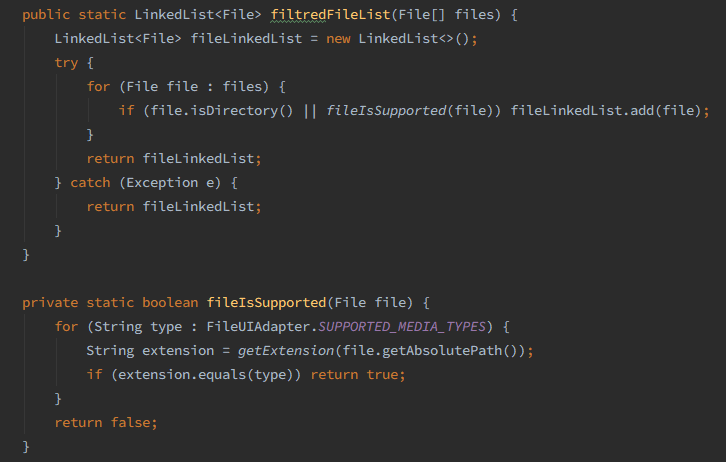
Для хранения информации о избранных файлах, истории использования приложения, а также статистики использования приложения была создана локальная база данных, состоящая из 3 таблиц:

* favourite – состоящая из id, названия избранного файла и абсолютного пути к этому файлу;
* history – состоящая из id, названия посещаемого файла и абсолютного пути к этому файлу и даты обращения;
* logs – состоящая из id и дат начала и конца сессии.

АЛГОРИТМЫ

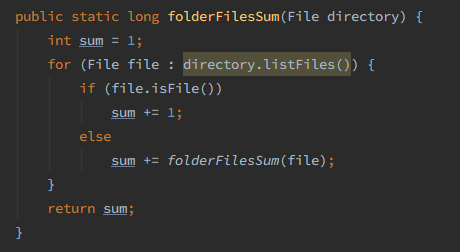
*Алгоритм фильтрации по видео файлам и папкам (метод filtredFileList):*

Изначально поступает массив файлов из локального хранилища, далее инициализируется новый список для отфильтрованных файлов, в цикле проходим по исходному массиву и добавляем при условии, что файл – папка или расширение файла соответствует поддерживаемому формату (в состав поддерживаемых видео-форматов входят: .mp4, .avi и .mkv) видеофайла (метод fileIsSupported), чтобы определить, что файл поддерживается необходимо пройтись по массиву поддерживаемых форматов и если формат файла, соответствует текущему элементу массива, то возвращается true, по итогу цикла если не один из поддерживаемых файлов не подошел – делается вывод, что файл не поддерживается и возвращается false.



*Алгоритм подсчета файлов в папке (метод* *folderFileSum):*

Инициализируется переменная для подсчета количества файлов в папке, далее в цикле проходимся по каждому файлу из директории и проверяем, если файл – директория рекурсивно вызываем *folderFileSum* и передаем текущий файл из цикла иначе накапливаем количество файлов.



*Алгоритм хранения компонентов, требующих независимого доступа из разных мест программы*

Для хранения таких компонентов был создан отдельный класс (ComponentFactory), который осуществляет статичное создание и доступ ко всем заданным ресурсам. При первом доступе к компоненту – осуществляется его создание, при последующих доступах, когда компонент уже существует, класс просто отдает уже созданый.

ЗАКЛЮЧЕНИЕ

По итогу курсовой работы, проект полностью работоспособен и готов к использованию.

Разработка велась с использованием git-репозитория, полную историю коммитов и весь исходных код можно увидеть по ссылке:

<https://github.com/kirilla111/MyVideo>