МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_Павленко Г. Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка мобильного приложения коммуникации «преподаватель-студент»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 19-1  Кудряшов Иван  Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Содержание

|  |  |
| --- | --- |
| Введение | 4 |
| 1 Описание предметной области | 5 |
| 1.1 Аналоги разрабатываемого приложения | 5 |
| 1.2 Техническое задание | 7 |
| 1.3 Описание структуры приложения | 8 |
| 2 Описание разработки приложения | 8 |
| 2.1 Обоснование средств разработки | 9 |
| 2.2 Разработка интерфейса | 10 |
| 2.3 Разработка Логики работы приложения. Схема взаимодействия компонентов проекта | 11 |
| 2.4 Описание переменных, компонентов, классов и подрограмм | 13 |
| 3.Тестирование и установка приложения | 22 |
| Заключение | 23 |
| Список использованной литературы | 25 |
| Приложение А. Листинг рабочего кода с комментариями | 26 |

Введение

В связи с актуальными технологиями, позволяет создавать информационные системы, которые позволяют облегчить взаимодействия работы с человеком. Данный проект разработан для удобного и оперативного получения важной информации учеников и преподавателей, с функционалом о связи между ними, подходящий для базовых информационных систем учебных учреждений с базовым функционалом.

Всё сильнее школы стараются автоматизировать информационными технологиями работу с данными обучающихся в учебных заведениях и предоставлять всё большее данных удалённым доступом, от чего было решено создать проект «Журнал и чат», позволяющий не только изучить базовые основы создания web-приложений, но и изучить рынок получше и язык Kotlin.

Данная версия проекта ещё находится в альфа-версии, от чего некоторый функционал не доступен, но многое готово для показа будущим покупателям новый проект.

1 Описание предметной области

Актуальность данного проекта высока по причине роста используемых информационных технологий в повседневной жизни, в числе которой входит обучение в учебных заведениях.

* 1. Аналоги разрабатываемого приложения

Для точного составления технического задания (ТЗ) с актуальным функционалом, нужно обратиться к аналогам, чтобы учесть ошибки и избежать их при разработке.

Для анализа было рассмотрено два проекта:

ЭлЖур

Разработчики: ООО “Веб-Мост”

Платформа: Android

Актуальная версия: 3.2.11 (11.04.22)

Количество установок: 1.000.000+

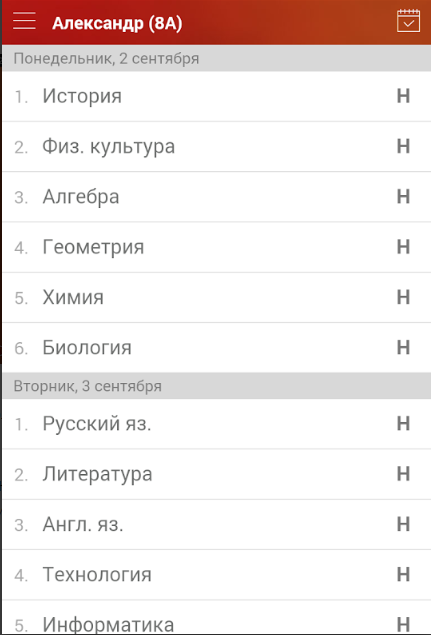
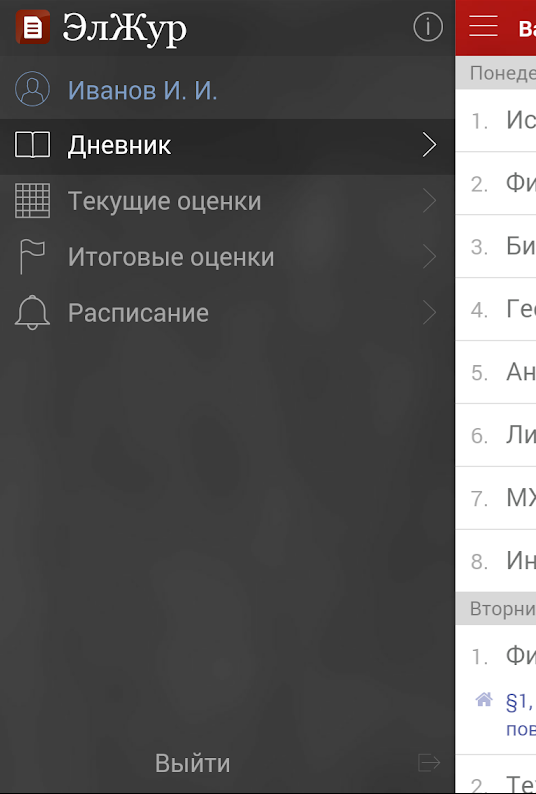
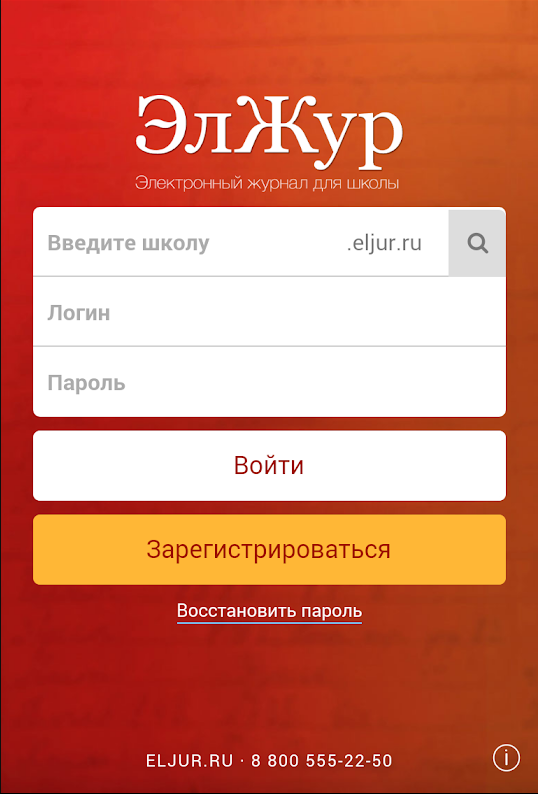


Рисунок 1 – Приложение «ЭлЖурнал»

Данный проект даёт в аренду учебным заведениям сервер со системой и базой данных, которую могут использовать. Обычные пользователи могут ознакомится с информацией, принадлежащей к нему и создан исключительно для школьников.

Проблема данного проекта в его поддержке, так-как многие жалуются, что оно в последнее время имеет много багов и для их исправления требуют деньги за исправленную версию (если судить комментариям пользователей). От чего система показала нам, что брать под контроль все школьные системы не выгодно как коммерчески, так и в плане поддержки.

Хоть данное приложения на русском рынке востребовано, больше как аналог с хорошим функционалом, с плохой поддержкой, что можно убедиться в картинке с оценками:

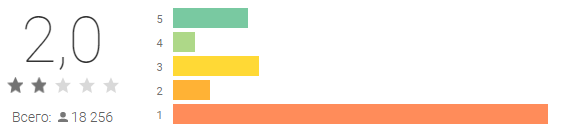


Рисунок 2 – Оценка приложения «ЭлЖурнал»

Дневничок - электронный дневник МЭШ, Спб, СГО, РТ

Разработчик: InkOut

Платформа: Android

Актуальная версия: 14.0.0(25.04.21)

Количество установок: 1.000.000+

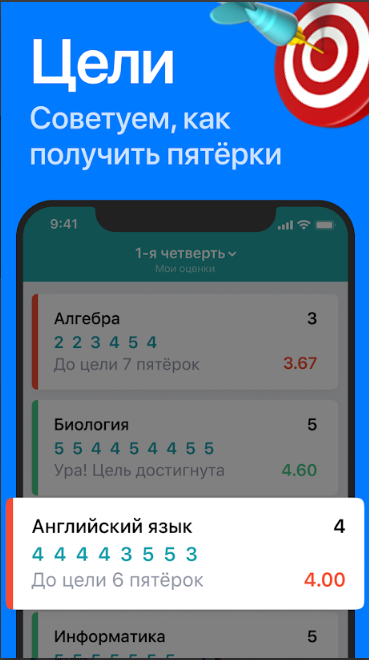
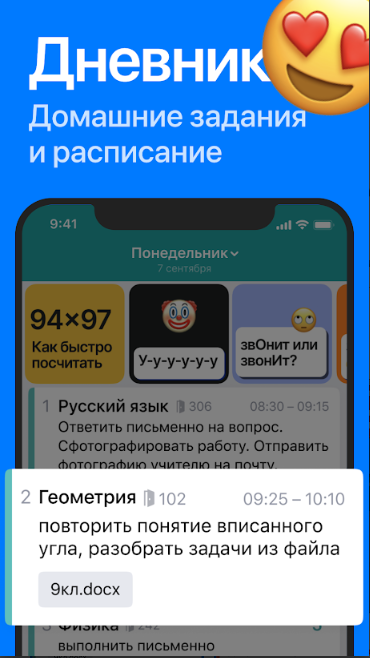
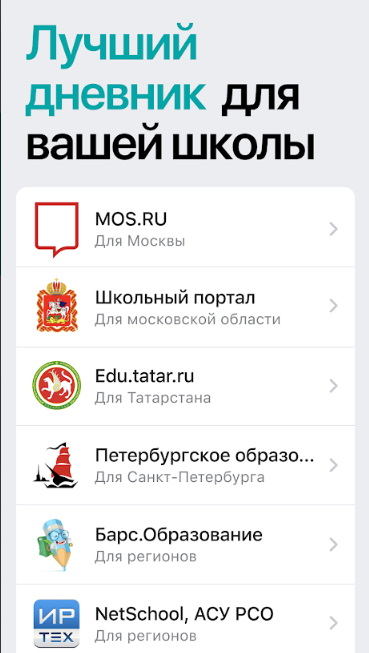


Рисунок 3 – Приложение «Дневничок»

Данный продукт полностью имеет коммерческую цель, предоставляя данные нужные только от тех учебных заведений, которые предоставляют доступ к базе данных данному проекту.

Приложение имеет большой функционал, который кроме основных данных, предоставляет сколько осталось до нужного итогового балла, материалы к обучению и многое другое.

Но из-за авантюрной основы, поддержка проекта слабая и имеет много рекламы и внутренние покупки, что может от пугать пользователей от использования данным приложением.

Оценки данного проекта средние и имеет перспективы:

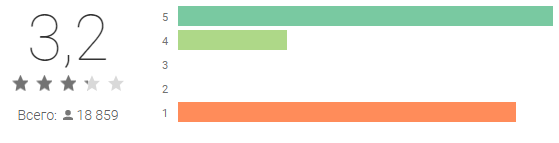


Рисунок 4 – Оценка приложения «Дневничок»

Так-же было замечено, что компания ООО “Веб-Мост” заняла основной рынок создания web-приложений электронного журнала, делая под заказ в учебные заведения. Но оценки и комментарии к ним низкие, от чего есть шанс в данном рынке приобрести клиентуру к нашему шаблонному приложению.

* 1. Техническое задание

Следуя с двух причин актуальности, пришла идея создать версию шаблонного web-приложения под кодовым названием «Журнал и чат», который показывает актуальную и основную информацию ученику и преподавателю, и имеет особенность «чат», этим сокращая создания лишних и временных акантов в разных социальных площадках и менеджеров, предоставляя всем одну площадку с уже учебными аккаунтами.

Для использования данного приложения, обучающийся должен приобрести свой аккаунт, который будет создаваться, в зависимости от покупателя данного шаблонного приложения и при вводе логина и пароля он получает информацию в таких модулях, как:

* новости, показывающий актуальные события, происходящие в учебном заведении, с показом времени и приложение с картинкой;
* оценки, которые показывают актуальные оценки за актуальный учебный период по всем дисциплинам, которые привязаны к данной учебной группе;
* расписание, показывающий актуальное расписание на актуальную неделю с дисциплинами к принадлежащей учебной группе пользователя;
* группа, показывающий участников группы пользователя, для последующей разработки функционала;
* чат, главная фишка данного проекта, позволяющий связаться с пользователем данной платформы в любое время, используя текст.

Приложение должно получать данные с базы данных (БД) на сервере, связываясь через API модуль, поэтому нужно сделать функционал обработки JSON-запросов и их вывода на экран, для более удобной интеграции проекта в учебную информационную систему (ИС).

1.3 Описание структуры приложения

Проект состоит из двух активностей, первая которая запускает приложение, показывая шапку журнала и предлагая авторизоваться, после авторизации, сохраняя токен пользователя в кэше и переходя на основную активность.

В основной активности есть фрагменты, которые сконструированы по модулям, перечисленные выше, имеющие свою логику. Запросы к серверу, на получение данных, обрабатываются службой парсера JSON, который конвертирует данные формата JSON на язык программы.

Фрагментов должно быть минимум 5 штук, каждый из которых будет вывод данных, а на последнем фрагменте обеспечить ввод данных, и отправку запроса с сообщением в JSON-файл.

Обеспечить модуль обработки данных с API запросов, независимо делая его от других модулей и так-же несколько конструкторов, выполняющие роль создания кэша.

2. Описание разработки приложения

2.1 Обоснование средств разработки

Для обеспечения разработки приложения, используется IDE (среда разработки) Android Studio. Программист, ответственный за разработку имеет хороший опыт.

IDE ещё имеет хороший функционал и нужные базовые библиотеки, для ускоренного проектирования приложения. Инструментарий для разработки для интерфейса гибок и даёт большой спектр вариаций разработки.

Язык программирования был выбран Kotlin, занимающий большой рынок мобильных приложений из-за синтаксиса, который заточён для разработки мобильного приложения и политика упрощена, позволяя выпускать на рынок проекты, юридически ускоряя работу и уменьшая трата деньги на лицензирование использования, в отличии от Java, с которым увеличивается инцидент с юридической стороны.

Поддержка данного языка большой, предоставляя большую базу удобных для программиста библиотек и функционала, с каждым днём предоставляя все более оптимизированную версию библиотек и функционала языка.

2.2 Разработка интерфейса

Интерфейс должен быть сделан базово, в геометрических плане объектах, и иметь наименьшее число кислотных цветов, для комфортного использования и понятности пользователю информации и навигации.

Было решено использовать основную тему, состоящую из мягко синих цветов и ровных фигур, с округлёнными концами, чтобы не занимать лишней визуальной информацией пользователю. Данные фигуры с грандиентовым фоном описаны в файлах формата xml, для автоматического масштабирования под любой формат разрешения смартфонов.

Шрифт использовать чёрного цвета, для подчёркивания основной и важной информации, а серой дополнительную информацию, позволяя на мягко синем фоне увидеть чётко буквы.

Контейнеры снабжать LinerLayout для адаптивности интерфейса к экрану пользователя, позволяя элементам подстраиваться под нужное разрешение, а вывод данных под общий формат использовать Recyle View, позволяя сделать адаптер, для автоматизации подставления данных в интерфейс.

Использование контейнера фрагмент, предоставит возможности не создавать много видов с общими элементы по новой создаваться, экономя память и облегчая обработку графических данных, позволяя сделать функцию пролистывания в материнском виде, дочерние виды.

Далее показан альфа-версии интерфейса приложения:

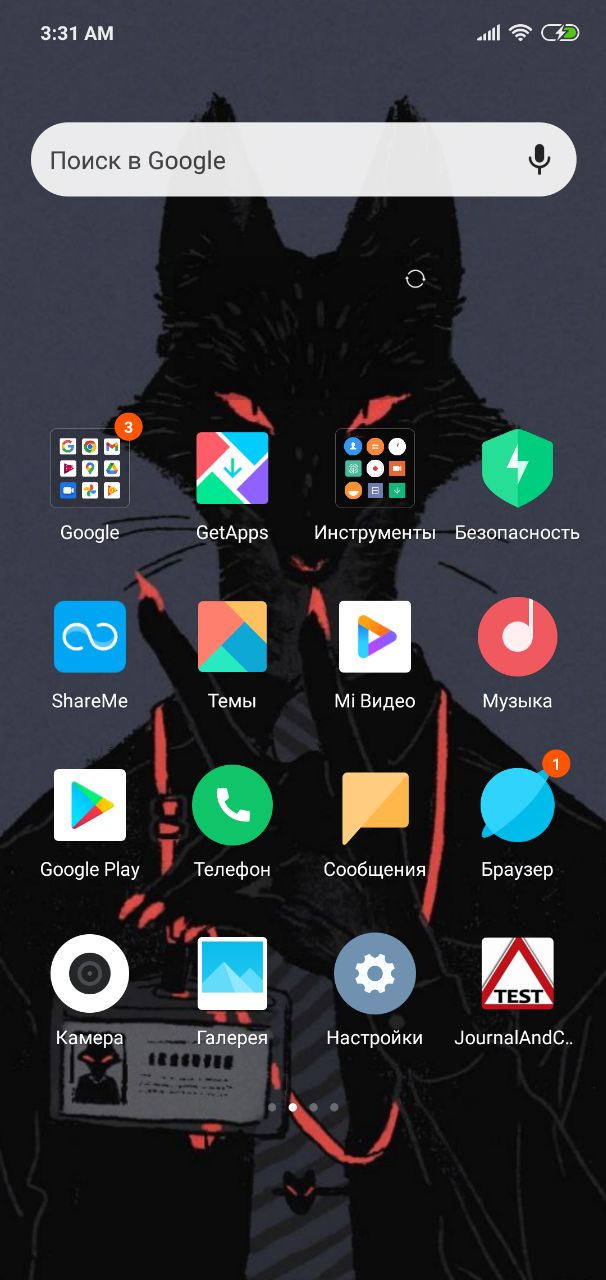


Рисунок 5 – Ярлык приложения «Журнал и чат»

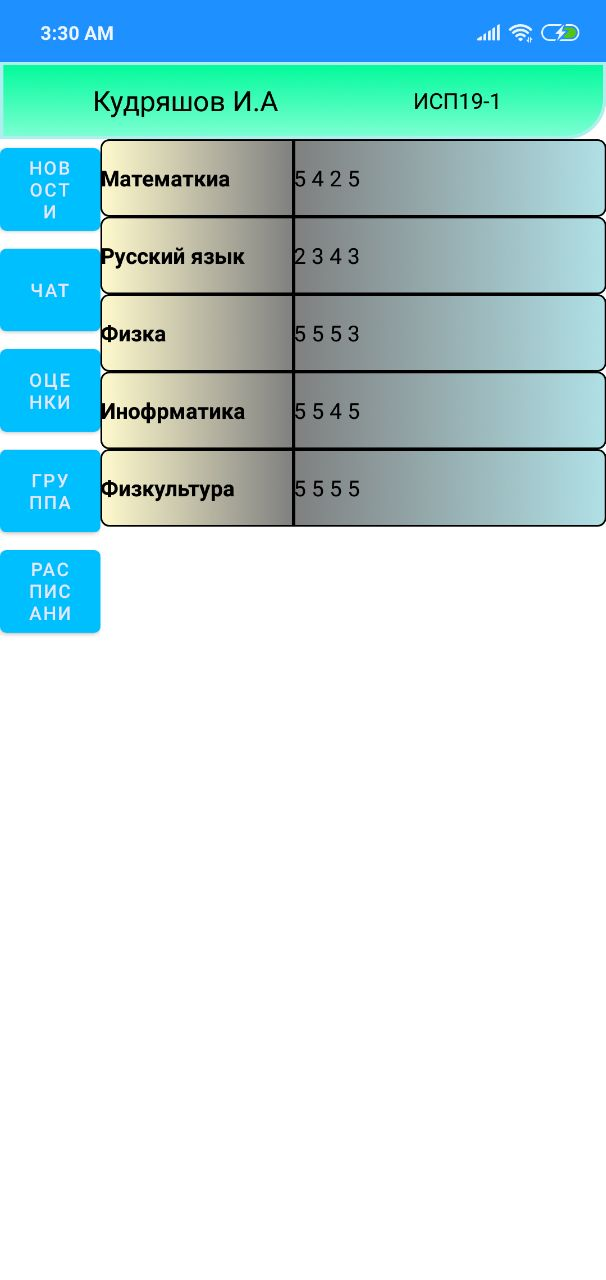
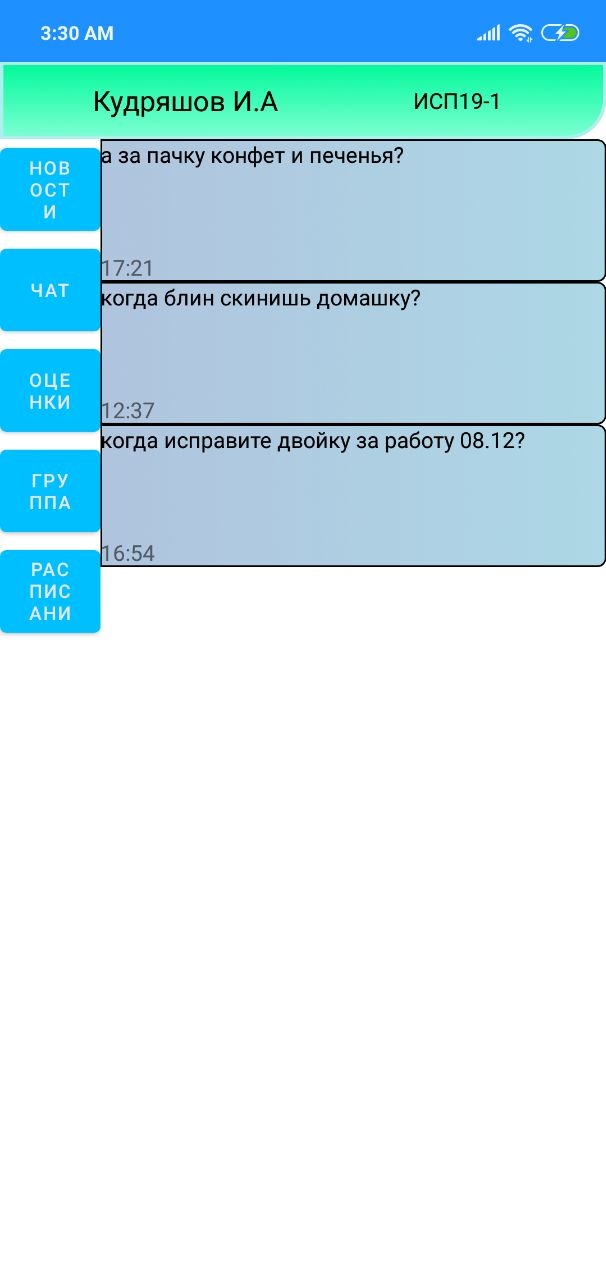
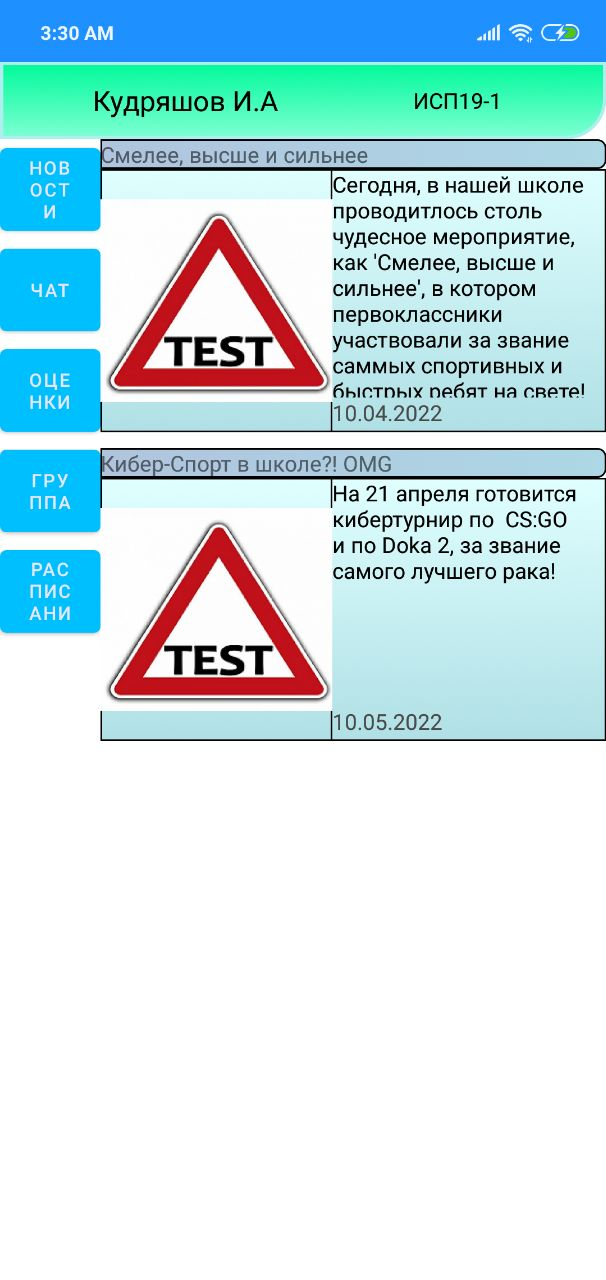


Рисунок 6 – Интерфейс мобильного приложения

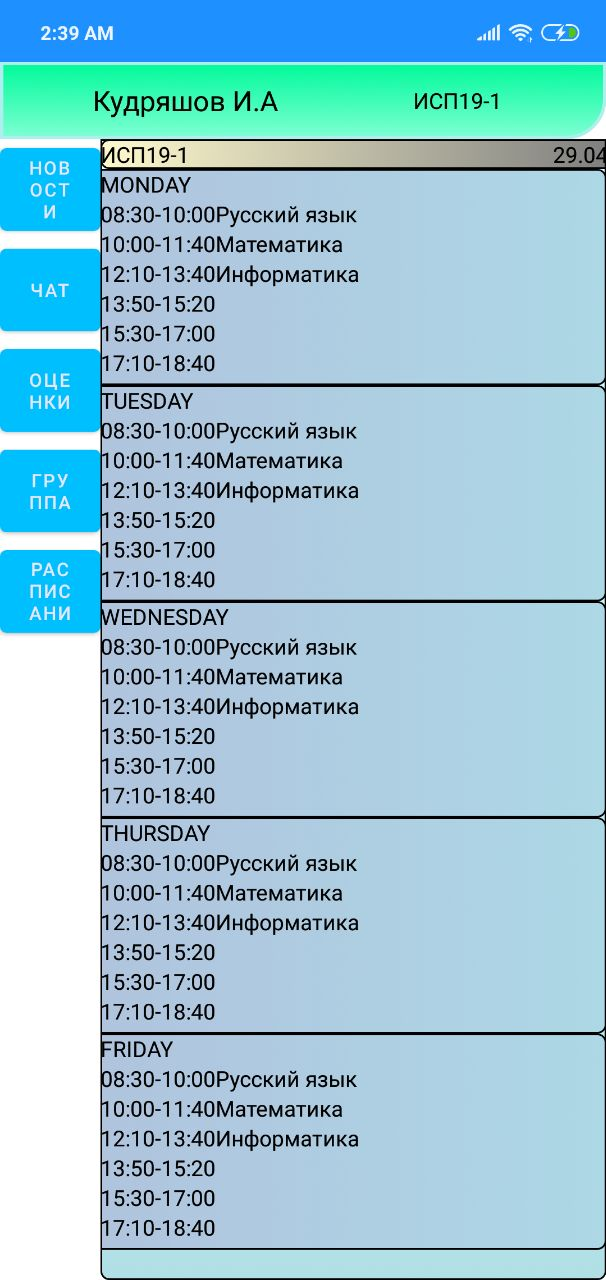
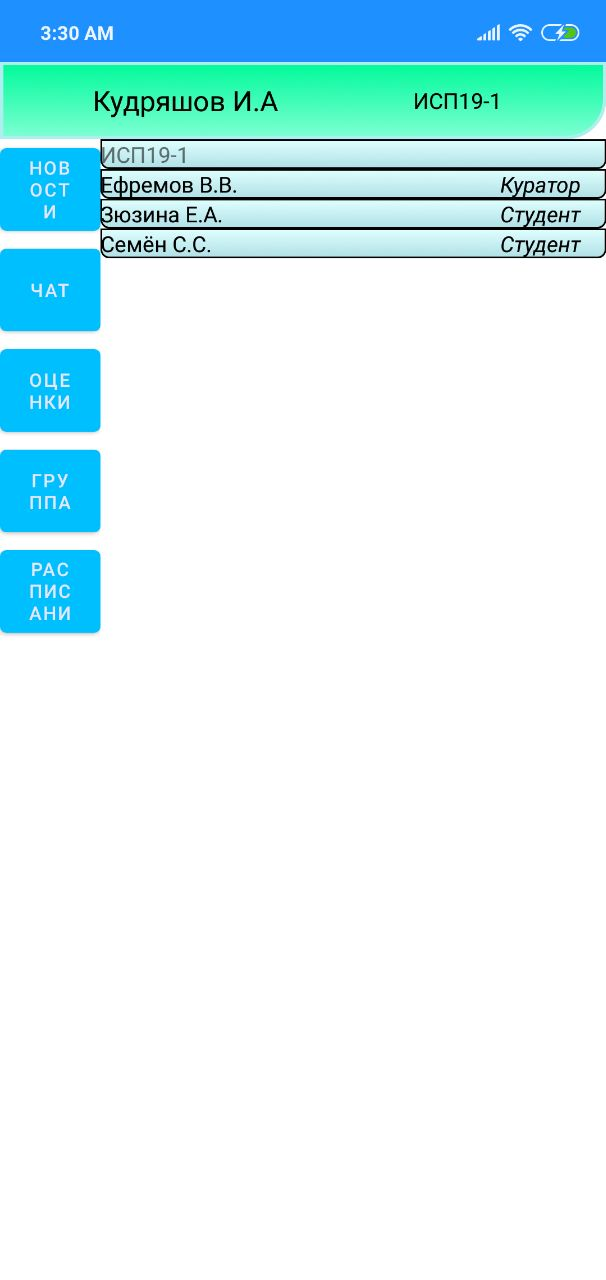


Рисунок 7 – Интерфейс мобильного приложения

2.3 Разработка Логики работы приложения

В приложении разработаны два класса: один для вывода шапки с логотипа компании и перехода к основному, второму классу, который предоставляет доступ к фрагментам и их взаимодействию. При помощи элемента ViwePager, идёт вывод фрагментов, которые отображают информацию через элемент RecyclerView, отвечающий за вывод данных в списке, логика которой описана в классах адаптерах, имеющий свой вид и класс с конструктором, парсером и кешом. Далее будет описан каждый тип модуля программы, с подробным описанием и х работы.

Парсер – Content Provider проекта, отвечающий за обработку JSON-запросов и отправки данных в адаптеры, для последующего вывода. Когда идёт отправка запроса, запрос первым делом отправляется в эту службу, которая на фоне переводит данные в более понятный формат для языка программирования Kotlin и уже разбивается на данные в массиве, который имеет описанную структуру в Content Provider конструктор, отправляя в адаптер данные для кеширования и вывода пользователю.

Адаптер – конструктор элементов списка, представляющий RecyclerView формат элементов для заполнение списков. Парсер, отдавая свои данные на хранение и использования адаптеру, использующие информацию для размещение по наименованиям данных в массиве, определяет, в какой элемент вида устанавливать данные для отображение. RecycleView получая информацию, создаёт список с элементами, созданные по правилам адаптера.

Фрагменты – основной элемент интерфейса, отображающий данные, при помощи RecycleView, пользователю. Фрагменты имеют вызов парсинга для обработки данных и отправки их в адаптер. При помощи Broadcast Receiver, ViewPager, позволяет переходить с одного фрагмента, на другой, для оперативного перехода свайпом.

Далее будет представлена схема работы приложения

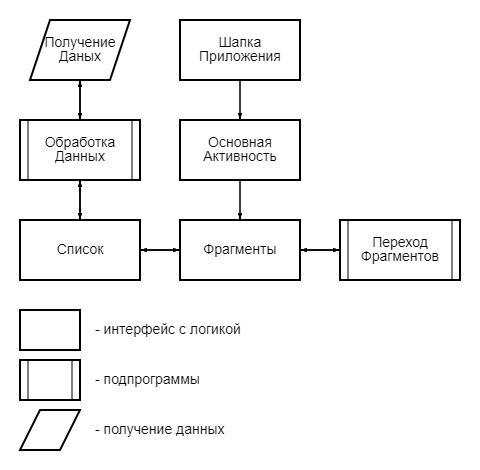


Рисунок 8 – Схема структуры приложения

2.4 Описание переменных, компонентов, классов и подрограмм

Таблица 1 – Список классов

|  |  |  |  |
| --- | --- | --- | --- |
| Название компонента | Тип компонента | Тип процесса | Описание |
| MainActivity.kt | Activity | Активный | Шапка приложения с лого |
| MainMenuActivity.kt | Activity | Активный | Основной класс и активность, выполняющий роль интерфейса, хранящий фрагменты в себе |
| FragmentChat.kt | Activity | Видимый | Фрагмент, отображающий информацию о чатах |
| FragmentGrabel.kt | Activity | Видимый | Фрагмент, отображающий оценки пользователя |
| FragmentGroup.kt | Activity | Видимый | Фрагмент, отображающий список обучающихся в группе пользователя |
| FragmentMainMenu.kt | Activity | Видимый | Фрагмент, отображающий новости учебного заведения пользователя. Открывается самой первой |
| FragmentTimetable.kt | Activity | Видимый | Фрагмент, отображающий расписание уроков/пар пользователя |
| GradeAdapter.kt | Service | Фоновый | Адаптер, для создание элементов списка оценок |
| GroupAdapter.kt | Service | Фоновый | Адаптер, для создание элементов списка учащихся в группе пользователя |
| NewsAdapter.kt | Service | Фоновый | Адаптер, для создание элементов списка новостей |
| ChatSelectAdapter.kt | Service | Фоновый | Адаптер, для создания элементов списка чатов |
| GradeBD.kt | Content Provider | Пустой | Конструктор, отвечающий за получение данных о оценках по их наименованию из запросов |
| FroupBD.kt | Content Provider | Пустой | Конструктор, отвечающий за получение данных о оценках по их наименованию из запросов |
| NewsBD.kt | Content Provider | Пустой | Конструктор, отвечающий за получение данных о оценках по их наименованию из запросов |
| ChatSelectBD.kt | Content Provider | Пустой | Конструктор, отвечающий за получение данных о оценках по их наименованию из запросов |
| TimetableBD.kt | Content Provider | Пустой | Конструктор, отвечающий за получение данных о оценках по их наименованию из запросов |
| ParserJSON.kt | Content Provider | Сервисный | Служба, обработки запросов JSON под более удобный формат |
| ViewPager | Boardcast Receiver | Фоновый | Функция, командных обрабатывающий жесты перехода фрагментов |

Таблица 2 – Таблица список методов и переменных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название метода и его класс | Тип метода | Описание метода | Переменные метода | | |
| Название | Тип | Описание |
| MainActivity.  onCreate | Процедура | Генерирующий шапку и переход на MainMenuActivity | - | - | - |
| MainMenuActivity.onCreate | Процедура | Генерирует вид и | viewPager | ViewPager2 | Ссылка на объект ViewPager2 вида и его свойства |
| adapter | ViewPager | Ссылка на класс ViewPager |
| FragmentChat.  loadChatList | Процедура | Загружает данные об чатах в RecyclerAdapter | adapter | ChatSelectAdapter | Ссылка на конструктор ChatSelectAdapter |
| FragmentChat.  onCreateView | Функция | Загружает вид и службы фрагмента | list | ArrayList<ChatSelectBD> | Массив, хранящие данные типа ChatSelectBD |
| parser | ParserJSON | Ссылка на функцию парсировки JSON |
| FragmentChat.  onViewCreated | Процедура | Отвечающий за логику фрагмента | recycleView | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |
| FragmentGrade.  loadGrable | Процедура | Загружает данные об оценках в RecyclerAdapter | adapter | GradeAdapter | Ссылка на конструктор GradeAdapter |
| FragmentGrade.  onCreateView | Функция | Загружает вид и службы фрагмента | list | ArrayList<GradeBD> | Массив, хранящие данные типа GradeBD |
| parser | ParserJSON | Ссылка на функцию парсировки JSON |
| FragmentGrade.  onViewCreated | Процедура | Отвечает за логику фрагмента | recycleView | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FragmentGroup.  loadGroup | Процедура | Загружает данные об участниках группы в RecyclerAdapter | adapter | ArrayList<GroupBD> | Массив, хранящие данные типа GroupBD |
| FragmentGroup.  onCreateView | Функция | Загружает вид и службы фрагмента | list | ParserJSON | Ссылка на функцию парсировки JSON |
| parser | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |
| FragmentGroup.  onViewCreated | Процедура | Отвечает за логику фрагмента | recycleView | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |
| FragmentMainMenu.loadNews | Процедура | Загружает данные о новостей в RecyclerAdapter | adapter | ArrayList<NewsBD> | Массив, хранящие данные типа NewsBD |
| FragmentMainMenu.onCreateView | Функция | Загружает вид и службы фрагмента | list | ParserJSON | Ссылка на функцию парсировки JSON |
| parser | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |
| FragmentMainMenu.onViewCreated | Процедура | Отвечает за логику фрагмента | recycleView | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FragmentTimetable.- | - | - | monText | List(Int) | Список ссылок TextView о кол-ва предметов учебных в понедельник |
| tuesText | List(Int) | Список ссылок TextView о кол-ва предметов учебных во вторник |
| wedText | List(Int) | Список ссылок TextView о кол-ва предметов учебных в среду |
| thurText | List(Int) | Список ссылок TextView о кол-ва предметов учебных в четверг |
| friText | List(Int) | Список ссылок TextView о кол-ва предметов учебных в пятницу |
| FragmentTimetable.enterDayWeek | Процедура | Отвечает за ввод данных о предметах в будних днях | les\_text | TextView | Ссылка на объект вида, отвечающие за предмет в будни день |
| time\_text | TextView | Ссылка на объект вида, отвечающие за дату буднего дня |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FragmentTimetable.onViewCreated | Функция | Отвечает за логику фрагмента | list | ParserJSON | Ссылка на функцию парсировки JSON |
| parser | RecyclerView | Ссылка на объект RecyclerView вида и его свойства |
| FragmentTimetable.onCreateView | Процедура | Загружает вид и службы фрагмента |  |  |  |
| ChatSelectAdapter.  ViewHolderChatSelect | Конструктор | Отвечает за сохранение массив ссылок на объекты вида элемента списка | userbane\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение ввода имени пользователя чата |
| message\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение собщения пользователя чата |
| date\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение времени отправленного последнего сообщения |
| ChatSelectAdapter.  onCreateViewHolder | Функция | Отвечает за создания вида элемента списка | itemView | View | Вид элемента списка |
| ChatSelectAdapter.  getItemCount | Функция | Возвращает количество элементов в виде числа | - | - | - |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ChatSelectAdapter.  onBindViewHolder | Метод | Отвечает за подставленные данных в элемент списка | userbane\_text | \* | \* |
| message\_text | \* | \* |
| date\_text | \* | \* |
| GradeAdapter.  ViewHolderGrade | Конструктор | Отвечает за сохранение массив ссылок на объекты вида элемента списка | subject\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение наименование предмета |
| grade\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение оценок предмета |
| GradeAdapter.  onCreateViewHolder | Функция | Отвечает за создания вида элемента списка | itemView | View | Вид элемента списка |
| GradeAdapter.  getItemCount | Функция | Возвращает количество элементов в виде числа | - | - | - |
| GradeAdapter.  onBindViewHolder | Метод | Отвечает за подставленные данных в элемент списка | subjectname\_text | \* | \* |
| grade\_data | ArrayList<GradeBD> | Массив, хранящий оценки типа GradeBD |
| grade\_text | \* | \* |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GroupAdapter.  ViewHolderGroup | Конструктор | Отвечает за сохранение массив ссылок на объекты вида элемента списка | name\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение имени участника группы |
| type\_text | TextView | Ссылка на элемент TextView, отвечающий за отображение роли участника группы |
| group\_data | ArrayList<GroupBD> | Массив, хранящий оценки типа GroupBD |
| GroupAdapter.  onCreateViewHolder | Функция | Отвечает за создания вида элемента списка | itemView | View | Вид элемента списка |
| GroupAdapter.  getItemCount | Функция | Возвращает количество элементов в виде числа | - | - | - |
| GroupAdapter.  onBindViewHolder | Метод | Отвечает за подставленные данных в элемент списка | name\_text | \* | \* |
| type\_text | \* | \* |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NewsAdapter.  ViewHolderNews | Конструктор | Отвечает за сохранение массив ссылок на объекты вида элемента списка | nameNews | TextView | Ссылка на элемент TextView, отвечающий за отображение заголовка новости |
| textNews | TextView | Ссылка на элемент TextView, отвечающий за отображение текст новости |
| dataNews | TextView | Ссылка на элемент TextView, отвечающий за отображение времени новости |
| imageNews | ImageView | Ссылка на элемент TextView, отвечающий за отображение обложки новости |
| NewsAdapter.  onCreateViewHolder | Функция | Отвечает за создания вида элемента списка | itemView | View | Вид элемента списка |
| NewsAdapter.  getItemCont | Функция | Возвращает количество элементов в виде числа | - | - | - |
| NewsAdapter.  onBindViewHolder | Метод | Отвечает за подставленные данных в элемент списка | nameNews | \* | \* |
| textNews | \* | \* |
| dataNews | \* | \* |
| imageNews | \* | \* |
| news\_data | ArrayList<NewsBD> | Массив с данными, типа NewsBD |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ChatSelectBD.- | - | - | id\_chat | String | Хранит ID чата |
| user\_name | String | Хранит имя пользователя чата |
| last\_message | String | Хранит текст последнего сообщения |
| data\_message | String | Хранит дату последнего сообщения |
| GroupBD.- | - | - | id\_account | String | Хранит ID аккаунта |
| name | String | Хранит имя пользователя группы |
| type\_account | String | Хранит роль участника группы |
| GradeBD.- | - | - | name | String | Хранит наименование предмета |
| grade | List<String> | Список, типа строковой, хранящий данные о оценках по предмету |
| NewsBD.- | - | - | name\_news | String | Хранит наименование заголовка |
| string\_text | String | Хранит текст новости |
| news\_date | String | Хранит дату новости |
| TimetableBD.- | - | - | time\_text | String | Хранит дату расписания |
| day\_week | String | Хранит дату будних дней |
| name\_les | List<String> | Список, типа строковой, хранящий набор предметов на будний день |
| ViewPager.  getItemCount | Функция | Возвращает количество фрагментов в виде числа | - | - | - |

Продолжение таблицы 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ViewPager.  createFragment | Функция | Возвращает фрагмент на отображение в активности | fragment | Fragment | Ссылка на класс фрагмента, для его отображения |
| ParserJSON.  getJSonArray | Функция | Возвращает обработанный набор | jsonString | String | Хранит обработанный запрос JSON |
| ParserJSON.  getArray\* | Функция | Возвращает массив данных на отображение в интерфейс | list\* | Class | Получает тип класса для обработки обработанного запроса |
| \*\_Array | ArrayList<Class> | Хранит распределённые данные в массиве, для отображение в интерфайсе |

3.Тестирование и установка приложения

Таблица.3 – Тесты приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование теста | Алгоритм выполнения теста | Предполагаемый результат | Фактический результат | Примечание |
| Большой ответ запроса | Сделать запрос (файл с новостями) большое количество объектов для обработки приложения | Приложение спокойно обработает данные и будет корректное отображение | Данные корректно отображаются | - |
| Запрос другого формата | Попытаться сделать запрос, который невозможен обработке приложением | Программа проигнорирует это и интерфейс ничего не отобразит | Приложение вывело ошибку и вылетело | Требуется обработка ошибок |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Длинный новостной текст | Получить новости с длинным текстом, неподходящего формата | Текст не влезет и пролистывание не будет | Текст не влез и нельзя пролистывать его | Формат хоть текста будет краток и передавать суть, но сделано для проверки корректного отображении, при таких случаях |
| Буквенные данные оценок | Отправить вместо оценок, случайный набор оценок | Расположит в список оценок, буквы | Расположил в список оценок, буквы | Нужно будет сделать проверку на отбор информации, чтобы принимались цифры от 1 до 5 и буквы н, у, б |

Алгоритм установки приложения:

1. В данный момент, так-как проект ещё в разработке, получить APK у разработчиков или доверенных лиц.
2. Полученный APK файла нажать и подтвердить установку.
3. Дождаться установки и запустить приложение.

Заключение

Хоть и данный проект ещё на стадии разработки, выполнение данного приложение приближенно на 85%.

Разработка данного проекта была интересной, показав проблемы фрагментов на языке Kotlin и приоткрыв мобильный рынок журнала и разработки базовых веб-приложений. При анализе приложений было выдвинуто предположение, что из-за малой финансовой поддержке и потребности только базового функционала, данные проекты не подлежат к долгому жизненному циклы, от чего такие проекты либо переходят в полукоммерческую основу или полностью закрываются. Выгоднее всего продавать готовый проект и контракт на поддержку, предположительно, 2-4 лет, чтобы получить выгоду с продажи проекта, от чего нужно сделать универсальный проект, для распространение приложения.

Kotlin во многом показал себя с хорошей стороны удобства создания мобильных проектов, хотя из-за новизны и уменьшение трения юридической и коммерческой работы, поддержка самого языка затруднена. Функционал фрагментов нужно доработать, чтобы не было проблем с работой их логикой.

Список использованной литературы

1. «Head First Kotlin»; Дон Гриффитс, Дэвид Гриффитс; 2019
2. «Kotlin. Программирование для профессионалов»; Джош Скин, Дэвид Гринхол; 2019
3. «Волшебство kotlin»; Пьер-Ив Симон; 2020
4. «Kotlin в действии»; Дмитрий Жемеров, Светлана Исакова; 2017
5. «Колтин. Программирование на примерах»; Ияну Аделекан; 2020
6. «Effective Kotlin: Best practices»; Марчин Москала, 2021
7. «Atomic Kotlin»; Брус Экел, Светлана Исакова; 2021
8. «Kotlin and Android Development featuring Jetpack: Build Better, Sager Android Apps»;Мичел Фазио, 2021
9. «Beginning Android Development With Kotlin»; Грег Лим, 2020
10. «How to Bulid Android Apps with Kotlin»; Алекс Форрестер, Эран Воуджах, Александр Думбрав, Джорман Тигал; 2021

Приложение А. Листинг рабочего кода с комментариями

MainActivity.kt

package com.example.journalandchat  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.os.Handler  
import androidx.core.os.HandlerCompat.postDelayed  
  
@Suppress("DEPRECATION")  
class MainActivity : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
  
 Handler().postDelayed({  
 val intent = Intent(this, MainMenuActivity:: class .java)  
 startActivity(intent)  
 finish()  
 }, 3000)  
 }  
}

MainMenuActivity.kt

package com.example.journalandchat  
  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.viewpager2.widget.ViewPager2  
  
/\*\*  
 \* Класс, имеющий основной вид и фрагменты  
 \* \*\*/  
class MainMenuActivity: AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.mainmenu\_student)  
 var viewPager: ViewPager2 = findViewById(R.id.view\_pager)  
 var adapter: ViewPager = ViewPager(this)  
 viewPager.adapter = adapter  
 }  
}

FragmentChat.kt

package com.example.journalandchat  
  
import android.content.Context  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
/\*\*  
 \* Класс фрагмента чата, отображая список чатов пользователя,  
 \* отображающийся в элеменетне viewPager, находящийся в MainMenuActivity  
 \* \*\*/  
  
class FragmentChat: Fragment(){  
  
 //Статичный объект, передающий данные фрагмента в viewPager  
 companion object{  
 fun newInstate(): FragmentChat{  
 return FragmentChat()  
 }  
 }  
  
 /\*\*  
 \* adapter - адаптер, для получения элементов и возвращение их в список  
 \* recyclerView - список элементов чата  
 \* parser - парсер, обрабатывающий запросов, для вывода данных в списке  
 \* list - массив, типа класса ChatSelectBD, для получение данных с парсера и  
 \* их дальнейшего использования  
 \* \*\*/  
 var adapter: ChatSelectAdapter? = null  
 private lateinit var recyclerView: RecyclerView  
 var parser :ParserJSON = ParserJSON()  
 private lateinit var list: ArrayList<ChatSelectBD>  
  
 //Функция создания view fragment\_chat  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 list = parser.getArrayChatSelect(requireContext())  
 return inflater.inflate(R.layout.fragment\_chat, container, false)  
 }  
  
 //Процедура создания элементов view fragment\_chat, после его создания  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 recyclerView = view.findViewById<RecyclerView>(R.id.list\_chat)  
 loadChatList(recyclerView, requireContext())  
 }  
  
 //Процедура внесение данных о чатах в список  
 fun loadChatList(recyclerAdapter: RecyclerView, context: Context){  
 recyclerAdapter.layoutManager = LinearLayoutManager(context)  
 recyclerAdapter.adapter = ChatSelectAdapter(list)  
 }  
}

FragmentGrabel.kt

package com.example.journalandchat  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
/\*\*  
 \* Класс фрагмента чата, отображая список оценок по предметам,  
 \* отображающийся в элеменетне viewPager, находящийся в MainMenuActivity  
 \* \*\*/  
  
class FragmentGrabel: Fragment() {  
  
 //Статичный объект, передающий данные фрагмента в viewPager  
 companion object {  
 fun newInstate(): FragmentGrabel {  
 return FragmentGrabel()  
 }  
 }  
  
 /\*\*  
 \* adapter - адаптер, для получения элементов и возвращение их в список  
 \* recyclerView - список элементов чата  
 \* parser - парсер, обрабатывающий запросов, для вывода данных в списке  
 \* list - массив, типа класса GrabeBD, для получение данных с парсера и  
 \* их дальнейшего использования  
 \* \*\*/  
 var adapter: GradeAdapter? = null  
 lateinit var recyclerView: RecyclerView  
 var parser: ParserJSON = ParserJSON()  
 private lateinit var list: ArrayList<GradeBD>  
  
 //Функция создания view fragment\_grade  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 list = parser.getArrayGrade(requireContext())  
 return inflater.inflate(R.layout.fragment\_grade, container, false)  
 }  
  
 //Процедура создания элементов view fragment\_grade, после его создания  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 recyclerView = view.findViewById(R.id.list\_grade)  
 loadGrable(recyclerView)  
 }  
  
 //Процедура внесение данных о оценках в список  
 fun loadGrable(recyclerAdapter: RecyclerView) {  
 recyclerAdapter.layoutManager = LinearLayoutManager(context)  
 recyclerAdapter.adapter = GradeAdapter(list)  
 }  
}

FragmentGroup.kt

package com.example.journalandchat  
  
import android.content.Context  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
/\*\*  
 \* Класс фрагмента чата, отображая список пользователей в группе,  
 \* отображающийся в элеменетне viewPager, находящийся в MainMenuActivity  
 \* \*\*/  
  
  
class FragmentGroup: Fragment() {  
  
 //Статичный объект, передающий данные фрагмента в viewPager  
 companion object{  
 fun newInstate(): FragmentGroup{  
 return FragmentGroup()  
 }  
 }  
  
  
 /\*\*  
 \* adapter - адаптер, для получения элементов и возвращение их в список  
 \* recyclerView - список элементов чата  
 \* parser - парсер, обрабатывающий запросов, для вывода данных в списке  
 \* list - массив, типа класса GroupBD, для получение данных с парсера и  
 \* их дальнейшего использования  
 \* \*\*/  
 var adapter: GroupAdapter? = null  
 lateinit var recyclerView: RecyclerView  
 var parser :ParserJSON = ParserJSON()  
 private lateinit var list: ArrayList<GroupBD>  
  
 //Функция создания view fragment\_group  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 list = parser.getArrayGroup(requireContext())  
 return inflater.inflate(R.layout.fragment\_group, container, false)  
 }  
  
 //Процедура создания элементов view fragment\_group, после его создания  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 recyclerView = view.findViewById(R.id.list\_group)  
 loadGroup(recyclerView, requireContext())  
 }  
  
 //Процедура внесение данных пользователей группы в список  
 fun loadGroup(recyclerAdapter: RecyclerView, context: Context){  
 recyclerAdapter.layoutManager = LinearLayoutManager(context)  
 recyclerAdapter.adapter = GroupAdapter(list)  
 }  
}

FragmentMainMenu.kt

package com.example.journalandchat  
  
import android.content.Context  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
  
/\*\*  
 \* Класс фрагмента чата, отображая список новостей,  
 \* отображающийся в элеменетне viewPager, находящийся в MainMenuActivity  
 \* \*\*/  
  
class FragmentMainMenu: Fragment() {  
  
 //Статичный объект, передающий данные фрагмента в viewPager  
 companion object{  
 fun newInstate(): FragmentMainMenu{  
 return FragmentMainMenu()  
 }  
 }  
  
 /\*\*  
 \* adapter - адаптер, для получения элементов и возвращение их в список  
 \* recyclerView - список элементов чата  
 \* parser - парсер, обрабатывающий запросов, для вывода данных в списке  
 \* list - массив, типа класса NewsBD, для получение данных с парсера и  
 \* их дальнейшего использования  
 \* \*\*/  
 var adapter: NewsAdapter? = null  
 lateinit var recyclerView: RecyclerView  
 var parser :ParserJSON = ParserJSON()  
 private lateinit var list: ArrayList<NewsBD>  
  
 //Функция создания view fragment\_news  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 list = parser.getArrayNews(requireContext())  
 return inflater.inflate(R.layout.fragment\_news, container, false)  
 }  
  
 //Процедура создания элементов view fragment\_news, после его создания  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 recyclerView = view.findViewById(R.id.list\_news)  
 loadNews(recyclerView, requireContext())  
 }  
  
 //Процедура внесение данных новостей в список  
 fun loadNews(recyclerAdapter: RecyclerView, context: Context){  
 recyclerAdapter.layoutManager = LinearLayoutManager(context)  
 recyclerAdapter.adapter = NewsAdapter(list)  
 }  
}

FragmentTimetable.kt

package com.example.journalandchat  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.fragment.app.Fragment  
  
/\*\*  
 \* Класс фрагмента чата, отображая рассписания группы,  
 \* отображающийся в элеменетне viewPager, находящийся в MainMenuActivity  
 \* \*\*/  
  
class FragmentTimetable : Fragment() {  
  
 //Статичный объект, передающий данные фрагмента в viewPager  
 companion object{  
 fun newInstate(): FragmentTimetable{  
 return FragmentTimetable()  
 }  
 }  
  
 /\*\*  
 \* parser - парсер, обрабатывающий запросов, для вывода данных в списке  
 \* list - массив, типа класса TimetabelBD, для получение данных с парсера и  
 \* их дальнейшего использования  
 \* \*\*/  
 var parser :ParserJSON = ParserJSON()  
 private lateinit var list: ArrayList<TimetableBD>  
  
 //данные списки хранят ссылки о элементах TextView, буднях, для оптимизированного заполнеия  
 val monText: List<Int> = listOf(R.id.monLes\_text1, R.id.monLes\_text2, R.id.monLes\_text3,  
 R.id.monLes\_text4, R.id.monLes\_text5, R.id.monLes\_text6)  
 val tuesText: List<Int> = listOf(R.id.tuesLes\_text1, R.id.tuesLes\_text2, R.id.tuesLes\_text3,  
 R.id.tuesLes\_text4, R.id.tuesLes\_text5, R.id.tuesLes\_text6)  
 val wedText: List<Int> = listOf(R.id.wesLes\_text1, R.id.wesLes\_text2, R.id.wesLes\_text3,  
 R.id.wesLes\_text4, R.id.wesLes\_text5, R.id.wesLes\_text6)  
 val thurText: List<Int> = listOf(R.id.thurLes\_text1, R.id.thurLes\_text2, R.id.thurLes\_text3,  
 R.id.thurLes\_text4, R.id.thurLes\_text5, R.id.thurLes\_text6)  
 val friText: List<Int> = listOf(R.id.friLes\_text1, R.id.friLes\_text2, R.id.friLes\_text3,  
 R.id.friLes\_text4, R.id.friLes\_text5, R.id.friLes\_text6)  
  
 //Функция создания view fragment\_timetable  
 override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
 savedInstanceState: Bundle?): View? {  
 list = parser.getArrayTimetable(requireContext())  
 return inflater.inflate(R.layout.fragment\_timetable, container, false)  
 }  
  
 //Процедура создания элементов view fragment\_timetable, после его создания  
 override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
 super.onViewCreated(view, savedInstanceState)  
 for(i in 0..list.size - 1){  
 when(list[i].day\_week){  
 "MONDAY" -> enterDayWeek(list[i].name\_les, monText,  
 list[i].day\_week, R.id.monTime\_text, view)  
 "TUESDAY" -> enterDayWeek(list[i].name\_les, tuesText,  
 list[i].day\_week, R.id.tuesTime\_text, view)  
 "WEDNESDAY" -> enterDayWeek(list[i].name\_les, wedText,  
 list[i].day\_week, R.id.wesTime\_text, view)  
 "THURSDAY" -> enterDayWeek(list[i].name\_les, thurText,  
 list[i].day\_week, R.id.thurTime\_text, view)  
 "FRIDAY" -> enterDayWeek(list[i].name\_les, friText,  
 list[i].day\_week, R.id.friTime\_text, view)  
 }  
 }  
 }  
  
 //Процедура внесение данных рассписания в список  
 private lateinit var time\_text: TextView  
 private lateinit var les\_text: TextView  
 private fun enterDayWeek(listDay: List<String>, listText: List<Int>, dayWeek: String, textDayWeek: Int, view: View){  
 time\_text = view.findViewById(textDayWeek)  
 time\_text.text = dayWeek  
 for(i in 0..listText.size - 1)  
 {  
 les\_text = view.findViewById(listText[i])  
 les\_text.text = listDay[i]  
 }  
 }  
}

ChatSelectAdapter.kt

package com.example.journalandchat  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import java.util.ArrayList  
  
/\*\*  
 \* Данный класс, отвечает за построенние элемента в списке выбора чата,  
 \* который будет отображаться в фрагменте FragmentChat  
 \* \*\*/  
  
class ChatSelectAdapter(private val chatSelect\_data: ArrayList<ChatSelectBD>):  
 RecyclerView.Adapter<ChatSelectAdapter.ViewHolderChatSelect>() {  
 //Класс-конструктор, хранящие ссылки элементов view адаптера  
 class ViewHolderChatSelect(itemView: View): RecyclerView.ViewHolder(itemView){  
 val fc: FragmentChat = FragmentChat()  
 val username\_text :TextView = itemView.findViewById(R.id.username\_text)  
 val message\_text :TextView = itemView.findViewById(R.id.message\_text)  
 val date\_text :TextView = itemView.findViewById(R.id.date\_text)  
 }  
  
 //Функция, создающий виды каждого элемента, возвращая в список  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):  
 ChatSelectAdapter.ViewHolderChatSelect {  
 val itemView = LayoutInflater.from(parent.context)  
 .inflate(R.layout.adapter\_selecchat, parent, false)  
 return ChatSelectAdapter.ViewHolderChatSelect(itemView)  
 }  
  
 //Подсчёт кол-ва элементов, возвращая их число  
 override fun getItemCount(): Int = chatSelect\_data.size  
  
 //Процедура, вносящие данные в элементы вида элемента  
 override fun onBindViewHolder(holder: ViewHolderChatSelect, position: Int) {  
 holder.username\_text.text = chatSelect\_data[position].user\_name  
 holder.message\_text.text = chatSelect\_data[position].last\_message  
 holder.date\_text.text = chatSelect\_data[position].date\_message  
 }  
}

GradeAdapter.kt

package com.example.journalandchat  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import java.util.ArrayList  
  
/\*\*  
 \* Данный класс, отвечает за построенние элемента в списке оценок,  
 \* который будет отображаться в фрагменте FragmentAdapter  
 \* \*\*/  
  
class GradeAdapter(private val grade\_data: ArrayList<GradeBD>): RecyclerView.Adapter<GradeAdapter.ViewHolderGrade>() {  
 //Класс-конструктор, хранящие ссылки элементов view адаптера  
 class ViewHolderGrade(itemView: View): RecyclerView.ViewHolder(itemView){  
 val subjectname\_text: TextView = itemView.findViewById(R.id.subjectname\_text)  
 val grade\_text: TextView = itemView.findViewById(R.id.grade\_text)  
 }  
  
 //Функция, создающий виды каждого элемента, возвращая в список  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GradeAdapter.ViewHolderGrade {  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.adapter\_grabel, parent, false)  
 return GradeAdapter.ViewHolderGrade(itemView)  
 }  
  
 //Процедура, вносящие данные в элементы вида элемента  
 override fun onBindViewHolder(holder: GradeAdapter.ViewHolderGrade, position: Int) {  
 var string\_array: String = ""  
 holder.subjectname\_text.text = grade\_data[position].name  
 grade\_data[position].grade.forEach { element -> string\_array = string\_array + element + " "}  
 holder.grade\_text.text = string\_array  
 }  
  
 //Подсчёт кол-ва элементов, возвращая их число  
 override fun getItemCount(): Int = grade\_data.size  
}

GroupAdapter.kt

package com.example.journalandchat  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import java.util.ArrayList  
  
  
/\*\*  
 \* Данный класс, отвечает за построенние элемента в списке участников группы,  
 \* который будет отображаться в фрагменте FragmentGroup  
 \* \*\*/  
class GroupAdapter (private val group\_data: ArrayList<GroupBD>):  
 RecyclerView.Adapter<GroupAdapter.ViewHolderGroup>() {  
 //Класс-конструктор, хранящие ссылки элементов view адаптера  
 class ViewHolderGroup(itemView: View): RecyclerView.ViewHolder(itemView){  
 val name\_text: TextView = itemView.findViewById(R.id.namePerson\_text)  
 val type\_text: TextView = itemView.findViewById(R.id.type\_text)  
 }  
  
 //Функция, создающий виды каждого элемента, возвращая в список  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):  
 GroupAdapter.ViewHolderGroup {  
 val itemView = LayoutInflater.from(parent.context)  
 .inflate(R.layout.adapter\_group, parent, false)  
 return GroupAdapter.ViewHolderGroup(itemView)  
 }  
  
 //Процедура, вносящие данные в элементы вида элемента  
 override fun onBindViewHolder(holder: GroupAdapter.ViewHolderGroup, position: Int) {  
 holder.name\_text.text = group\_data[position].name  
 holder.type\_text.text = group\_data[position].type\_account  
 }  
  
 //Подсчёт кол-ва элементов, возвращая их число  
 override fun getItemCount() = group\_data.size  
}

NewsAdapter.kt

package com.example.journalandchat  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.ImageView  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
  
/\*\*  
 \* Данный класс, отвечает за построенние элемента в списке новостей,  
 \* который будет отображаться в фрагменте FragmentMainMenu  
 \* \*\*/  
  
class NewsAdapter(private val news\_data: ArrayList<NewsBD>) : RecyclerView.Adapter<NewsAdapter.ViewHolderNews>() {  
 //Класс-конструктор, хранящие ссылки элементов view адаптера  
 class ViewHolderNews(itemView: View): RecyclerView.ViewHolder(itemView){  
 val nameNews: TextView = itemView.findViewById(R.id.text\_nameNews)  
 val textNews: TextView = itemView.findViewById(R.id.text\_textNews)  
 val dataNews: TextView = itemView.findViewById(R.id.text\_dataNews)  
 val imageNews: ImageView = itemView.findViewById(R.id.image\_imageNews)  
 }  
  
 //Функция, создающий виды каждого элемента, возвращая в список  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolderNews {  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.adapter\_news, parent, false)  
 return ViewHolderNews(itemView)  
 }  
  
 //Процедура, вносящие данные в элементы вида элемента  
 override fun onBindViewHolder(holder: ViewHolderNews, position: Int) {  
 holder.nameNews.text = news\_data[position].name\_news  
 holder.textNews.text = news\_data[position].string\_text  
 holder.dataNews.text = news\_data[position].news\_date  
 holder.imageNews.setImageResource(R.drawable.test\_pictuer)  
 }  
  
 //Подсчёт кол-ва элементов, возвращая их число  
 override fun getItemCount() = news\_data.size  
}

ChatSelectBD.kt

package com.example.journalandchat  
  
/\*\*  
 \* Конструктор типа для получение и дальнешего использования данных о чатах  
 \* Переменные названы, как и свойства объекта JSON для правильного внесения данных  
 \* id\_chat - id чата  
 \* user\_name - имя пользователя  
 \* last\_message - последнее сообщение  
 \* date\_message - дата последнего сообщения  
 \* \*\*/  
data class ChatSelectBD (val id\_chat: String, val user\_name: String,  
 val last\_message: String, val date\_message: String) {}

GradeBD.kt

package com.example.journalandchat  
  
/\*\*  
 \* Конструктор типа для получение и дальнешего использования данных о оценках  
 \* Переменные названы, как и свойства объекта JSON для правильного внесения данных  
 \* name - наименование предметов  
 \* grade - оценки предмета  
 \* \*\*/  
data class GradeBD (val name: String, val grade: List<String>) {}

GroupBD.kt

package com.example.journalandchat  
  
/\*\*  
 \* Конструктор типа для получение и дальнешего использования данных о учасиниках группы  
 \* Переменные названы, как и свойства объекта JSON для правильного внесения данных  
 \* id\_account - id пользователя  
 \* name - имя пользователя  
 \* type\_account - роль пользователя в группе  
 \* \*\*/  
data class GroupBD(val id\_account: String, val name: String, val type\_account: String) {  
}

NewsBD.kt

package com.example.journalandchat  
  
/\*\*  
 \* Конструктор типа для получение и дальнешего использования данных новостей  
 \* Переменные названы, как и свойства объекта JSON для правильного внесения данных  
 \* name\_news - наименования загаловка новостей  
 \* string\_text - текст новости  
 \* news\_date - дата новости  
 \* \*\*/  
  
data class NewsBD(val name\_news: String, val string\_text:String, val news\_date: String) {}

Timetable.BD

package com.example.journalandchat  
  
/\*\*  
 \* Конструктор типа для получение и дальнешего использования данных о рассписании группы  
 \* Переменные названы, как и свойства объекта JSON для правильного внесения данных  
 \* time\_text - дата написания рассписания  
 \* day\_week - дата буднего дня  
 \* name\_less - список наименования предметов в будний день  
 \* \*\*/  
  
data class TimetableBD(val time\_text: String, val day\_week: String, val name\_les: List<String>) {}

ParserJSON.kt

package com.example.journalandchat  
  
import android.content.Context  
import android.util.Log  
import androidx.appcompat.app.AppCompatActivity  
import com.google.gson.Gson  
import com.google.gson.reflect.TypeToken  
  
/\*\*  
 \* Класс, отвечающий за парсировку данных Json и последующего  
 \* использования в списках фрагментов  
 \* \*\*/  
  
class ParserJSON: AppCompatActivity() {  
 //Данная переменная вызывает функцию библиотеки Gson, для парсировки дданных  
 val gson = Gson()  
  
 //Функция, обрабатывающий запрос под более понятный формат приложению  
 fun getJsonArray(context: Context, fileName: String):String?{  
 Log.i("data", "Идёт парсировка")  
 val jsonString: String = context.assets.open(fileName).bufferedReader().use{it.readText()}  
 Log.i("data", jsonString)  
 return jsonString  
 }  
  
 /\*\*  
 \* Дальнейшие методы обрабатывают jsonString,рабирая его на данные для массива и возвращая его в  
 \* списки  
 \* list\* - класс-тип  
 \* \*\_Array - массив, заполняющий данными для использования в списках фрагментов  
 \* \*\*/  
 fun getArrayNews(context: Context): ArrayList<NewsBD>{  
 val listNews = object : TypeToken<ArrayList<NewsBD>>() {}.type  
 val news\_Array: ArrayList<NewsBD> =  
 gson.fromJson(getJsonArray(context, "NewsJson.json"), listNews)  
 news\_Array.forEachIndexed { id, tut -> Log.i("data", "> Item ${id}:\n ${tut}") }  
 return news\_Array  
 }  
  
 fun getArrayChatSelect(context: Context): ArrayList<ChatSelectBD>{  
 val listChatSelect = object : TypeToken<ArrayList<ChatSelectBD>>() {}.type  
 val chatSelect\_Array: ArrayList<ChatSelectBD> =  
 gson.fromJson(getJsonArray(context, "ChatSelectJson.json"), listChatSelect)  
 chatSelect\_Array.forEachIndexed {  
 id, tut -> Log.i("data", "> Item ${id}:\n ${tut}") }  
 return chatSelect\_Array  
 }  
  
 fun getArrayGrade(context: Context): ArrayList<GradeBD>{  
 val listGrade = object : TypeToken<ArrayList<GradeBD>>() {}.type  
 val grade\_Array: ArrayList<GradeBD> =  
 gson.fromJson(getJsonArray(context, "GradeJson.json"), listGrade)  
 grade\_Array.forEachIndexed { id, tut -> Log.i("data", "> Item ${id}:\n ${tut}") }  
 return grade\_Array  
 }  
  
 fun getArrayGroup(context: Context): ArrayList<GroupBD>{  
 val listGroup = object : TypeToken<ArrayList<GroupBD>>() {}.type  
 val group\_Array: ArrayList<GroupBD> =  
 gson.fromJson(getJsonArray(context, "GroupJson.json"), listGroup)  
 group\_Array.forEachIndexed { id, tut -> Log.i("data", "> Item ${id}:\n ${tut}") }  
 return group\_Array  
 }  
  
 fun getArrayTimetable(context: Context): ArrayList<TimetableBD>{  
 val listTimetable = object : TypeToken<ArrayList<TimetableBD>>() {}.type  
 val timetable\_Array: ArrayList<TimetableBD> =  
 gson.fromJson(getJsonArray(context, "TimeTableJson.json"), listTimetable)  
 timetable\_Array.forEachIndexed {  
 id, tut -> Log.i("data", "> Item ${id}:\n ${tut}") }  
 return timetable\_Array  
 }  
 /\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*/  
}

ViewPager.kt

package com.example.journalandchat  
  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.FragmentActivity  
import androidx.viewpager2.adapter.FragmentStateAdapter  
  
/\*\* Класс, отвечающий за смену и отображение фграментов \*\*/  
  
class ViewPager(fragmentManager: FragmentActivity): FragmentStateAdapter(fragmentManager) {  
  
 //Функция о получение списка фрагментов, вовзращая числом  
 override fun getItemCount(): Int = 5;  
  
 //Функция, возвращающая фрагмент выбраный для отображения  
 override fun createFragment(position: Int): Fragment {  
 val fragment: Fragment  
 when(position){  
 0 -> fragment = FragmentMainMenu()  
 1 -> fragment = FragmentChat()  
 2 -> fragment = FragmentGrabel()  
 3 -> fragment = FragmentGroup()  
 4 -> fragment = FragmentTimetable()  
 else -> fragment = FragmentMainMenu()  
 }  
 return fragment  
 }  
  
}

Рецензия

Курсовой проект Кудряшов И.А составлен в соответствии с ГОСТ'ом «ГОСТ Р 7.0.97-2016», ошибок в составленном документе не нашлось.

Тема курсового проекта имеет специфичную для студента идею. Создание электронного журнала не имеет на данный момент актуальности, с учётом распространённости.

Не до конца раскрыта тема данного проекта. Анализ данного проекта и выводы были раскрыты хорошо, с обоснованными аргументами, подтверждая хорошие знания аналитической способности и систематизации информации и самостоятельного составления вывода.

Логическая связь, переходы главы и входящие компоненты, составлены правильно.

Проект храните в GitHub по данной ссылке: [BadCaTez/JournalAndChat: Курсовой проект ИСП19-1 Кудряшов И.А (github.com)](https://github.com/BadCaTez/JournalAndChat)