Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Құлманов Тайыр Тасқынбайұлы

Название: Разработка мобильного приложения – «Личный кабинет студента»

Координатор:Жибек Алибиева

Коэффициент подобия 1:1,6

Коэффициент подобия 2:1

Замена букв:2

Интервалы:0

Микропробелы:0

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

- ☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;
- ☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;
- ☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

……

…… ……

Дата *Подпись Научного руководителя*

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ

ОТЗЫВ

НАУЧНОГО РУКОВОДИТЕЛЯ

на дипломный проект
(наименование вида работы)
Кулманов Тайыр
(Ф.И.О. обучающегося)
5В070400 – Вычислительная техника и программное обеспечение
(шифр и наименование специальности)

Тема: "Разработка мобильного приложения – личный кабинет студента"

Студент Кулманов Т. выполнил дипломную проект на актуальную тему, связанную с разработкой мобильного приложения под операционную систему Android, предоставляющего полную информацию, необходимую для студентов «Satbayev University», которая включает не только просмотр расписания занятий и экзаменов, но и мониторинг учебной успеваемости. Несомненным достоинством проекта является то, что мобильное приложение, разработанное Кулмановым Т., является решением, которое позволит предоставлять актуальные сведения удобным и понятным для студентов образом. Мобильное приложение может улучшить качество получаемых студентами данных. Дипломный проект состоит из 4-х логически связанных глав, включая введение, заключение и приложение.

Дипломный проект выполнен в срок и соответствует всем требованиям, предъявляемым к студентам, обучающимся по специальности 5В070400 – «Вычислительная техника и программное обеспечение». Студент Кулманов Т. рекомендован к защите дипломного проекта и заслуживает отличной оценки при условии удачной защиты проекта, а также присвоения академической степени «бакалавра» по специальности 5В070400 – «Вычислительная техника и программное обеспечение».

Научный руководитель
маг. техн. наук
(должность, уч. степень, звание)
К.Маргулан
(подпись)
«___»_____2020г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

Кулманов Т.Т.

"Разработка мобильного приложения – личный кабинет студента"

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассоциированный -профессор
_____Р.Юнусов

"____" _____2020 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

На тему: "Разработка мобильного приложения – личный кабинет студента"

по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнил Кулманов Т.Т.

Научный руководитель
маг. техн. наук

_____ К.Маргулан

"____" _____ 2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассоциированный-профессор
_____ Р. Юнусов
"_____" _____ 2020 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Кулманову Тайыру Таскынбайулы*
Тема: *Корпоративная система регистрации посетителей*
Утверждена *приказом проректора по академической работе №_____*
от "" _ 2019 г.
Срок сдачи законченного проекта *"_" __ 2020 г.*
Исходные данные к дипломному проекту: *Паспорт проекта, техническая документация, техническое задание, описание БД для хранения информации в виде ER-диаграммы.*
Перечень подлежащих разработке в дипломном проекте вопросов:
а) Проектирование архитектуры приложения в соответствии с паттерном проектирования MVVM;
б) Разработка пользовательского интерфейса приложения;
в) Реализация аутентификации на основе JWT;
г) Тестирование и отладка приложения;
Перечень графического материала (с точным указанием обязательных чертежей): *представлены 15 слайдов презентации.*
Рекомендуемая основная литература: *из 8 наименований.*

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Разработка технического задания и анализ программного комплекса.		
2. Подробное проектирование архитектуры приложения и создание бизнес-логики.		
3. Разработка пользовательского интерфеса.		
4. Создание временного веб-сервера. Подключение приложения к серверу.		
5. Тестирование и отладка всех частей приложения.		
6. Написание пояснительной записки к дипломному проекту.		

Подписи

консультантов и нормоконтролера на законченный дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	**Консультанты, И.О.Ф. (уч. степень, звание)**	**Дата подписания**	**Подпись**
Программное обеспечение	А.М.Каирбеков магистр техн. наук, лектор		
Нормоконтролер	К.Маргулан магистр техн. наук, лектор		

Научный руководитель _____К.Маргулан

Задание принял к исполнению обучающийся _____Т.Т Кулманов

Дата "__" _____ 2020г.

АННОТАЦИЯ

Данный дипломный проект представляет собой студенческое мобильное приложение на базе операционной системы Android. Разработанное решение позволит студентам “Satbayev University” получить полный доступ к отслеживанию учебной успеваемости, просмотру расписания занятий и экзаменов, проверке транскрипта и подробностей текущей аттестации.

Дипломная работа поделена на несколько логических частей.

Первая часть описывает общее описание и цели разработки данного приложения. Вторая часть представляет собой полное техническое описание мобильного приложения, включая описание стека технологий, подробное описание архитектуры. Третья часть описывает проектные особенности приложения. Заключительная часть дипломной работы описывает функциональные возможности приложения.

АҢДАТПА

Дипломдық жоба - Android операциондық жүйесіне негізделген студенттік мобильді қосымша. Әзірленген шешім “Satbayev University”студенттеріне академиялық үлгерімді бақылауға, сабақ кестесі мен емтихандарды қарауға, транскрипт пен ағымдағы үлгерімді тексеруге толық мүмкіндік береді.

Дипломдық жоба бірнеше логикалық бөлімге бөлінеді.

Бірінші бөлімде осы қосымшаның жалпы бейнесі мен даму мақсаттары сипатталалған. Екінші бөлім - мобильді қосымшаның толық техникалық сипаттамасы, оның ішінде технологиялық стек сипаттамасы және архитектураны толық суреттейді. Үшінші бөлімде қосымшаның жобалық ерекшеліктері бейнеленген. Диссертацияның қорытынды бөлімі қосымшаның функционалдығын сипаттайды.

ANNOTATTION

This graduation project is a student mobile application based on the Android operating system. The developed solution will allow students of the University of Satbayev to get full access to tracking academic performance, viewing education schedule and exams schedule, checking the transcript and current attestation details.

The graduate work is divided into several logical parts.

The first part is a general description and development goals for this mobile application. Second part is detailed architecture description. The third part is the design features of the application. The final part of the thesis is determined by the functionality of the application.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Образование – неотъемлемая часть развитого государства, от качества подготовки специалистов зависит будущее благосостояние страны. Улучшение подготовки кадров должно находится в одном из приоритетных направлений.
На данный момент одним из вариантов улучшения образовательного сектора является внедрение информационных систем.

В целях цифровизации сферы образования, разрабатываются информационные системы позволяющие комплексно автоматизировать процессы кредитной системы и дистанционной технологии обучения. Такие системы добавляют удобства преподавательскому составу, а также студентам в отслеживании учебной деятельности. Автоматизация учебного процесса ускоряет и упрощает взаимодействие ВУЗа, педагогов и студентов.

Основные задачи образовательных информационных систем:

– Комфортный и эффективный контроль академического процесса преподавателем;
– Предоставление удобного мониторинга успеваемости студентам;
– Возможность просматривать необходимый учебный материал;

На момент написания дипломной работы Вуза “Satbayev University” имеет готовую систему контроля академической деятельности. Система обладает всеми необходимыми элементами для полноценной работы.

Клиентская часть системы состоит:

– Веб-приложения “Stud.satbayev.university”
– Мобильного приложения на платформе IOS “Satbayev University”

Практическая польза данного дипломного проекта заключается в создании мобильного приложения “Satbayev University” под операционную систему Android. Мобильное приложение предоставляет удобный доступ ко всей необходимой учебной информации владельцам устройств, работающих на платформе Android.

1 Общее описание

1.1 Цель разработки системы

Мобильное приложение “Satbyaev University” разработанное под операционную систему Android, обеспечит студентам университета, владеющим устройствами под управлением данной платформы, комфортный доступ к учебной информации со смартфона.

Основные задачи, предъявляемые приложению:

– Отображение текущей успеваемости: количество пропусков, количество баллов по курсу;

– Просмотр расписания занятий, экзаменов;

– Возможность просмотра текущей аттестации;

– Предоставление полной информации по транскрипту: GPA, балл и оценку по предмету за семестр и т.д.;

– Доступ к учебно-методическому комплексу дисциплины:

1.2 Определения, термины, сокращения

В таблице 1 сформулированы все сокращения и термины, используемые в предметной области разрабатываемого проекта, а также специфические термины, связанные с программной реализацией проекта и используемыми технологиями при разработке.

Таблица 1 – Сокращения, термины и их определения

Сокращение или термин	Определение
ОС	Операционная система
ПО	Программное обеспечение
БД	База данных
DAO	Data Access Object
DI	Dependency Injection
JSON	JavaScript Object Notation
API	Application Programming Interface
IDE	Integrated Development Environment
XML	eXtensible Markup Language
SDK	Software Development Kit
MVVM	Model-View-ViewModel
UI	User Interface

1.3 Состояние вопроса

На момент начала разработки мобильного приложения под платформу Android, ВУЗ “Satbayev University” обладал помимо веб-приложения для студентов и преподавателей, мобильным студенческим приложением на платформе IOS. На рисунке 1.1 предоставлен главный экран приложения.

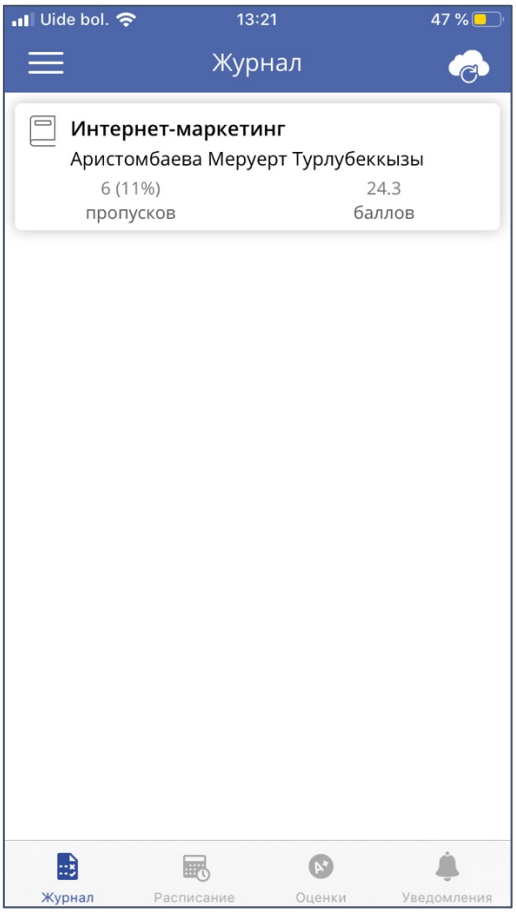

Рис 1.1 – Пример мобильного приложения “Satbayev University” на IOS

Приложение на IOS обладает приятным UI, высокой скоростью работы и стабильностью приложения. Данный проект является основой для разработки мобильного приложения под ОС Android.

2 Технологический раздел

2.1 Технологии, выбранные при создании программного обеспечения

2.1.1 Язык программирования

Android ОС официально поддерживает два языка программирования:
– Java
– Kotlin

Приложение полностью написано на языке Kotlin. Связано это с несколькими важными причинами:

Google, компания поддерживающая ОС, объявила Kotlin первостепенным языком для разработчики ПО на Android. В связи с этим язык Kotlin обладает первичной поддержкой от компании, новые API и обновление библиотек Android Jetpack будут выпускаться на Kotlin и только затем на Java.

Также язык программирования Kotlin обладает возможностями, облегчающими разработку ПО, и позволяет уменьшить количество написанного кода.

2.1.2 IDE

Google официально утвердила основной средой разработки под ОС Android – Android Studio. Данная IDE разработана специально под разработку приложений на Android. И обладает всеми необходимыми инструментами для создания приложений под Android из базовой комплектации.

К встроенным инструментам относятся: эмулятор-менеджер ОС – AVDM; SDK-менеджер, позволяющий управлять SDK; расширенный редактор макетов – позволяет работать с UI компонентами в режиме конструктора.

Также к преимуществам Android Studio можно отнести, основную систему сборки Gradle. Gradle обладает высокой скоростью, и простой системой добавления зависимостей. Android Studio и встроенный Gradle поддерживают Kotlin без необходимости установки дополнительных плагинов.

Остальные IDE требуют устанавливать расширения для работы с Kotlin и встроенными функциями в Android Studio.

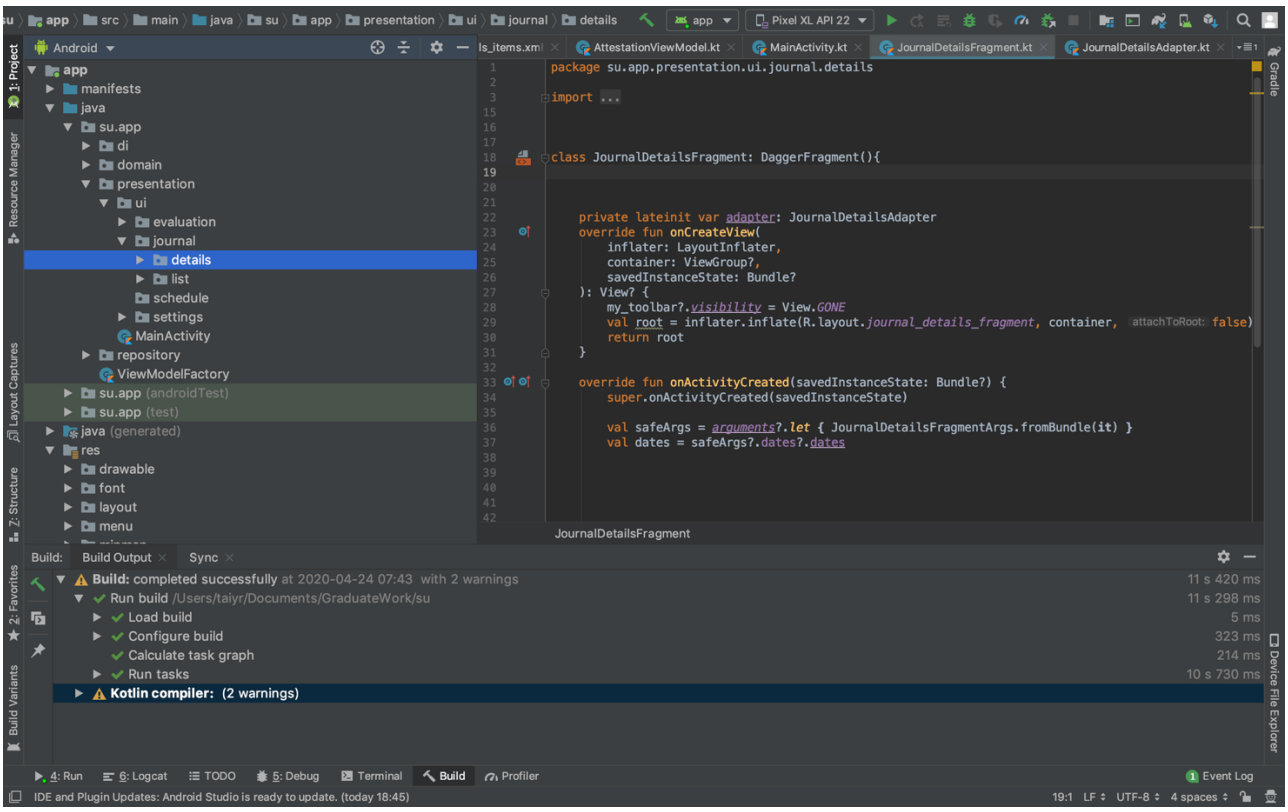

Рис. 2.1 – Среда разработки Android Studio

2.1.3 Android Jetpack

Android Jetpack - это компоненты и библиотеки, инструменты и руководства, облегчающие производство программного обеспечения. Вместе с Jetpack был введен новый стандарт библиотек на androidx.*. Это API отделенные от платформы. Это означает, обеспечение полной совместимости с предыдущими версиями Android.

Библиотеки Jetpack разделены на четыре основных вида:

Foundation – базовые компоненты, позволяющие добиться совместимости с предыдущими версиями ОС, тестирование и поддержку языка Kotlin: AppCompat, Test, Android KTX

Architecture – компоненты, упрощающие построение архитектуры приложения: DataBinding, LiveData, ViewModel, Room, Navigation.

Behavior – компоненты поведения, помогающие вашему приложению интегрироваться со стандартными службами Android.

UI – компоненты, помогающие в построении пользовательского интерфейса, добавляя новые возможности для Fragment, Layout, Animation & Transitions

2.1.3.1 ViewModel

ViewModel — представляет собою компонент помогающий управлять, хранить необходимые данные, связанные с отрисовкой пользовательского интерфейса, учитывая Lifecycle представления. Класс ViewModel позволяет хранить данные при изменении конфигурации представления (прим.Изменение ориентации).

Операционная система Android управляет Lifecycle контроллеров пользовательского интерфейса, такими как Activity и Fragment.

В зависимости от событий платформа может уничтожить или пересоздать контроллер пользовательского интерфейса. В этот момент, любые данные из UI хранящиеся в нем будут потеряны.

ViewModel будет находиться в памяти до момента, пока жизненный цикл к которому он относится, не уничтожится навсегда. Пример жизненного цикла ViewModel представлен на рисунке. 2.2

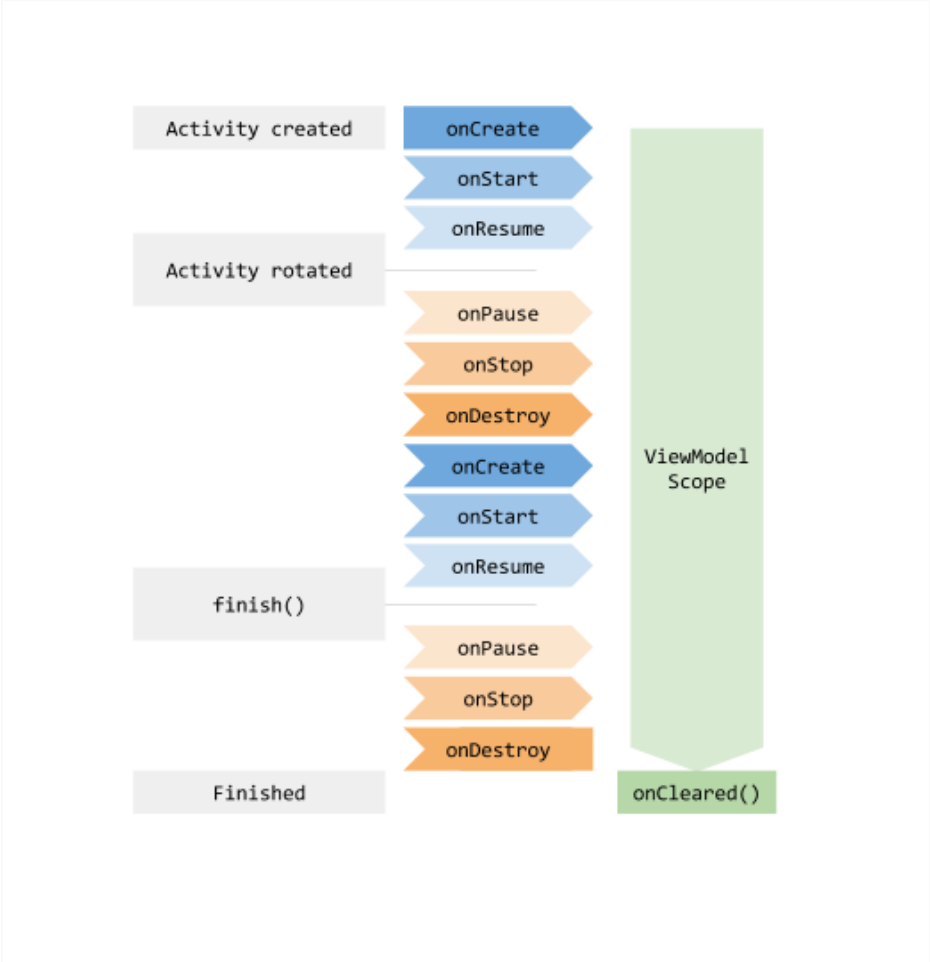

Рис. 2.2 – Жизненный цикл ViewModel

Свойства ViewModel позволяют эффективнее и проще отделять хранимые данные от контроллера пользовательского интерфейса. В связи с этим Activity и Fragment содержат минимум бизнес-логики.

2.1.3.2 LiveData

LiveData – является наблюдаемым объектом хранящим данные, иначе говоря это Observable обертка. Отличием от обычного Observable является то, что LiveData смотрит на жизненный цикл компонентов приложения, например Activity, Fragment или Service. Эта означает, что LiveData отправляет обновления наблюдателям (Observer), находящимся в состоянии активного Lifecycle, в состоянии **STARTED** и **RESUMED**. Схема работы LiveData представлена на рисунке 2.3.

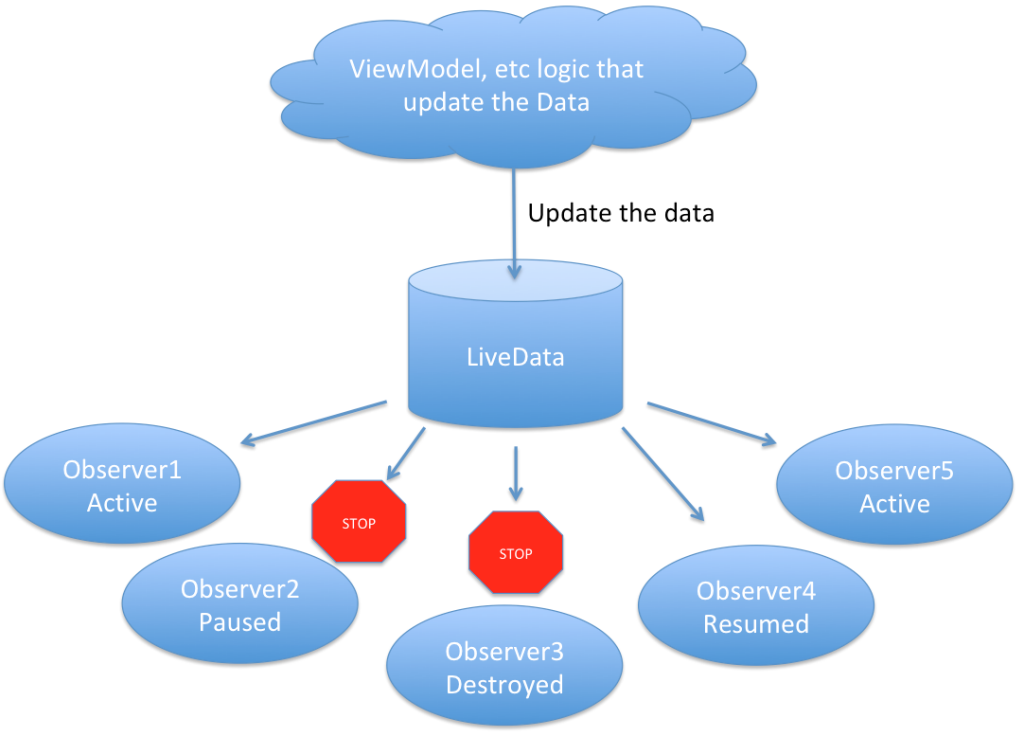

Рис. 2.3 – Схема работы LiveData

Преимуществами использования LiveData, являются уменьшение утечек памяти, наблюдатели отписываются при необходимости; нет необходимости постоянного обновления UI, наблюдатель может обновить UI при изменении данных; не надо обрабатывать жизненный цикл вручную.

2.1.3.3 DataBinding

DataBinding – библиотека, позволяющая передавать необходимые данные прямо в пользовательский интерфейс, не используя сторонних методов.

Data Binding заметно упрощает взаимодействие с передачей данных в макет. Без нее взаимодействие с компонентами производилось через контроллер UI при помощи ***findViewById( )*** или аналогичным, либо прямым обращением к ресурсу макета**,** использование данных методов показаны на рисунках 2.4, 2.5

```
course_name.text = journal.title
course_teacher.text = journal.teacher
pass_text_view.text = journal.getMissed.toString()
point_text_view.text = journal.getGrade.toString()
```

Рис. 2.4 – Обращение напрямую к ресурсу макета

```
findViewById(R.id.main_area)
```

Рис. 2.5 – Обращение к ресурсу через *findViewById( )*

DataBinding уменьшает количество исполняемого кода, связывая модели приложения с макетом. Библиотека самостоятельно генерирует необходимые элементы. Пример привязки данных отображен на рисунках 2.6, 2.7

```
class JournalHolder(val binding: JournalDetailsItemsBinding)

    fun bind(date: Dates) {
        binding.setDate(date)
    }

}
```

Рис. 2.6 – Отправка данных в макет

```
<data>
    <variable
        name="date"
        type="su.app.repository.model.journal.Dates"
        />
</data>

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="10dp"
    android:orientation="vertical">

    <TextView
        android:id="@+id/grade"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@{`P=` + date.grade}" />
```

Рис. 2.7 – Вид макета при использовании DataBinding

2.1.3.4 Room

Room – является библиотекой, обеспечивающей уровень абстракции над встроенной в Android SQLite, чтобы обеспечить более надежный и комфортный доступ к БД.

Room выполняет большую часть работы с БД, убирая необходимость вручную инициализировать БД, работать с ***Cursor*** и т.д.

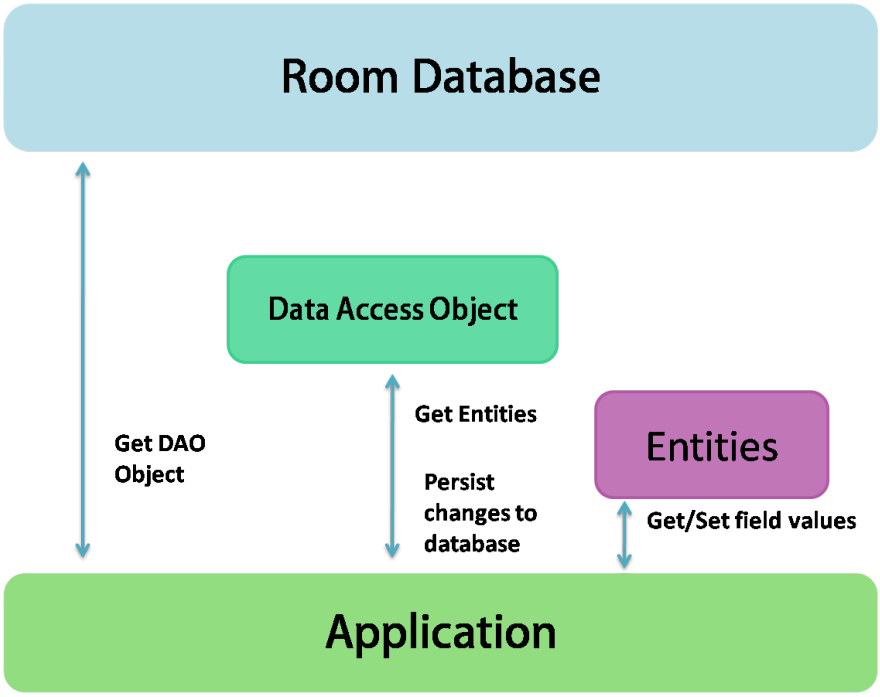

Рис. 2.8 – Абстрактная схема работы Room

Как показано на рисунке 2.8, вся необходимая информация для создания таблиц находится в Entity, но фактически нет возможности добавлять, запрашивать и т.д. данные из БД. Для этой задачи необходим объект доступа к данным(DAO). DAO предоставляет возможность управления данными хранящимися в Entity.

2.1.3.5 Navigation

Navigation – библиотека позволяющая упростить контроль навигации при использовании приложения. Navigation забирает работу по настройке навигации между представлениями. Позволяя больше не писать код по контролю за переходами.

Navigation имеет три ключевых компонента, отвечающих за навигацию:

Navigation Graph: это макет XML, содержащий все ссылки на View, а также их связи между собой

NavHost: работает в роли элемента, отображающего необходимою View

NavController: контроллер, отвечающий за управление всей навигацией в проекте

2.1.4 Retrofit 2

Retrofit 2 – типобезопасный HTTP клиент для Android. Позволяет упростить работу с REST API. Retrofit использует библиотеку OkHttp для HTTP-запросов.

Одним из основных преимуществ этой библиотеки является простота конфигурирования. Также позволяет использовать различные конвертеры, например в JSON. Обладает хорошей документацией по использованию.

Основной причиной выбора данного клиента является хорошая интеграция с Room и архитектурой приложения. Состоит отметить, что Retrofit является самым быстрым HTTP клиентов, обгоняя аналогичную ему Volley.

3 Проектный раздел

3.1 Архитектура приложения

3.1.1 Архитектура MVVM

Мобильное приложение разработано по архитектурному шаблону MVVM. Данное решение было принято по нескольким причинам:

– Google официально начала рекомендовать данную парадигму;

– MVVM лучше остальных подходит для Android Jetpack и позволяет использовать его преимущества, в частности Data Binding;

– MVVM, обладает большей модульностью по сравнению с шаблонами MVC и MVP;

– Данный паттерн идеально подходит для реализации архитектуры Android Repository,

описание в другой главе;

MVVM – это паттерн(шаблон) проектирования, Model-View-ViewModel. Он разделяет приложение на 3 слоя:

– Model – абстрактное название слоя, содержащего всю бизнес-логику приложения, доступ к данным с сервера, реализацию доступа к БД. Model ничего не знает про ViewModel;

– ViewModel – слой, который содержит данные. ViewModel должна иметь доступ к Model и все данные получает оттуда, но не должна знать откуда данные были получены из БД или с веб-сервера. Также ViewModel не знает о существовании View.

– View – представляет собою слой, отвечающий за отображение макета. View не содержит логики отвечающей за данные, а также и сами данные. Она должна иметь, только экземпляр ViewModel откуда и приходят данные.

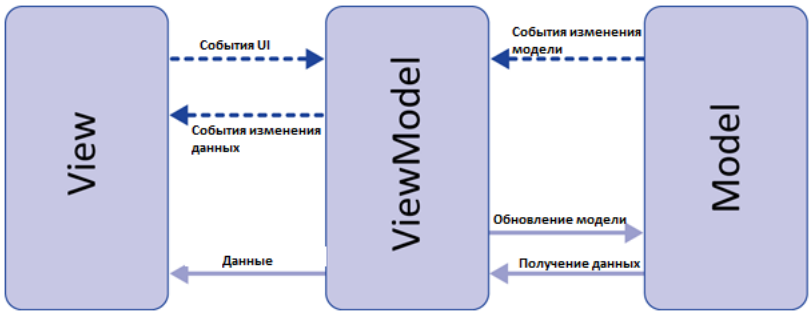

Рис. 3.1 – Шаблон работы MVVM

3.1.2 Android Repository

Данная архитектура добавляет дополнительную бизнес-логику в слой Model паттерна MVVM.

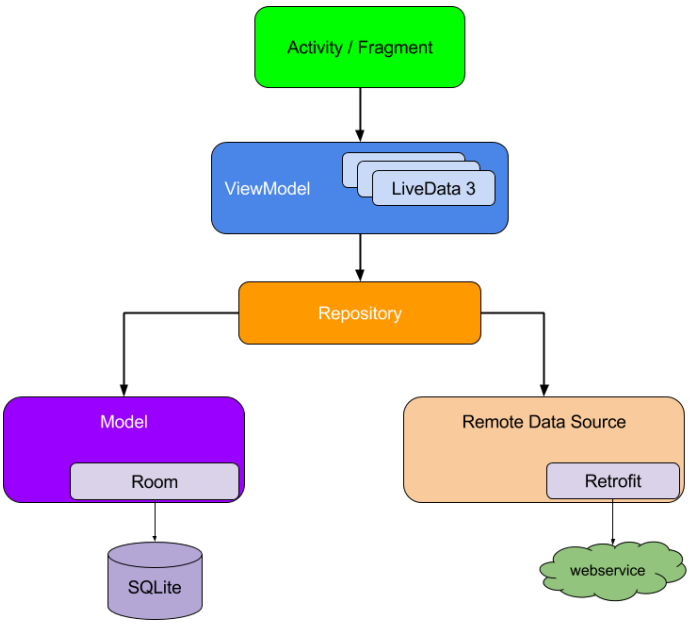

Рис. 3.2 – Архитектура Android Repository

Как видно, на рисунке 2.10. В бизнес-логику добавляются новые элементы: Model, отвечающая за хранение данных локально, не путать со слоем Model из MVVM, и Remote Data Source контролирующий запрос данных из веб-сервера. Repository управляет сохранением данных в БД, а также отвечает за информацию передаваемую во ViewModel. Сохраняется иерархия, при которой сущность, лежащая ниже не имеет доступа к сущности, стоящей над нее. В данной архитектуре во ViewModel передается только экземпляр Repository.

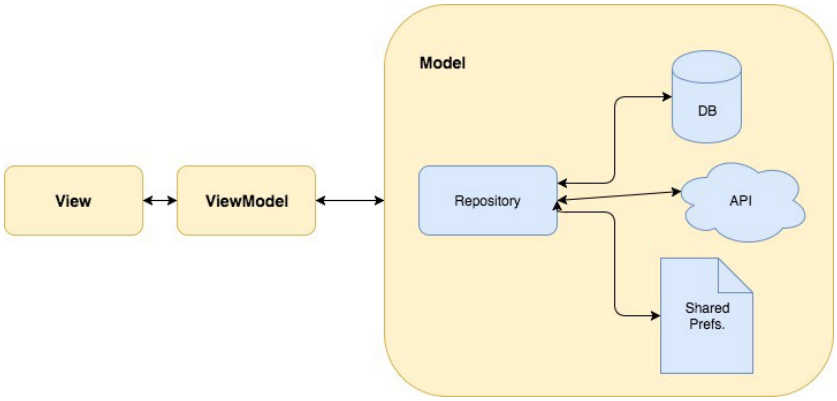

Рис. 3.3 – Вид слоев MVVM, с использованием Android Repository

3.1.3 Архитектура UI

UI был построен с использованием архитектуры Single-Activity. Она представляется в виде одного Activity и использованием множества Fragment. Activity в данном виде является базовым элементом на котором меняются Fragment-ы. Google рекомендует использовать данный подход, так как он дает возможность использовать Navigation в полном объеме.

3.1.4 Архитектура клиент-серверной системы

Разрабатываемое приложение работает на клиент-серверной архитектуре. В качестве серверной части выступает университетский REST API сервер.
На стороне сервера происходит большая часть обработки данных, клиенту отправляются уже готовые данные в формате JSON. В приложении за отправку запросов на сервер, по протоколу HTTP отвечает библиотека Retrofit 2. Приложение выполняет только два вида запросов на сервер: GET и POST(для изменения параметров аккаунта), приложение предназначено только для просмотра информации, но не добавления или изменения ее. На рисунке 2.12 показано взаимодействие клиента и сервера.

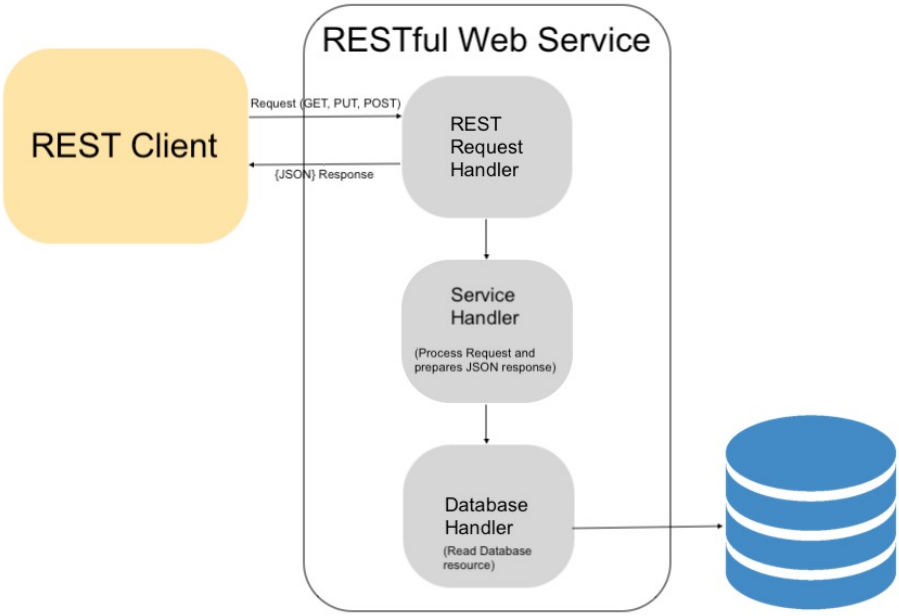

Рис. 3.4 – Архитектура клиент-серверных приложений

3.2 Диаграмма прецедентов

Диаграмма прецедентов(вариантов использования) демонстрирует взаимодействие пользователя с прецедентами. Варианты использования – это функции или сервисы, выполняемые системой.

Разрабатываемая система, представляет собою студенческое мобильное приложение, и единственным пользователем является студент. Студент может посмотреть полную учебную успеваемость, получить доступ к УМКД, либо узнать расписание, поменять язык, получить уведомление.

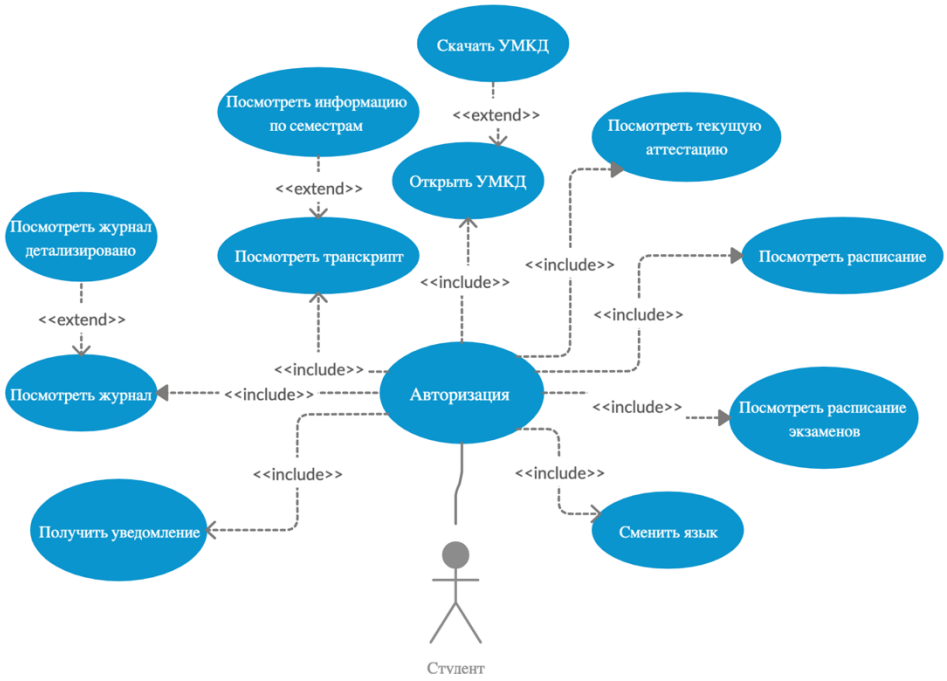

Рис. 3.5 – Диаграмма прецедентов

3.3 ER-диаграммы используемых моделей

Архитектура приложения подразумевает работу как с данными, полученными с веб-сервера, так и использование локальной базы данных. Для этих целей используются модели данных, позволяющие упростить работу с данными.

ER-диаграмма – является представлением Entity-Relationship. В данном случае она позволит показать концептуальную модель данных.

Модель Journal:

- journalId - Id предмета
- journalTitle - Название предмета
- teacherFullName - Полное имя преподавателя
- dates - Список прошедших занятий

Модель Dates:
– date - Дата проведения занятия
– grade - Выставленный балл
– attended - Посещение занятия
Подробности и связь моделей Journal и Dates отображены на рисунке 3.6

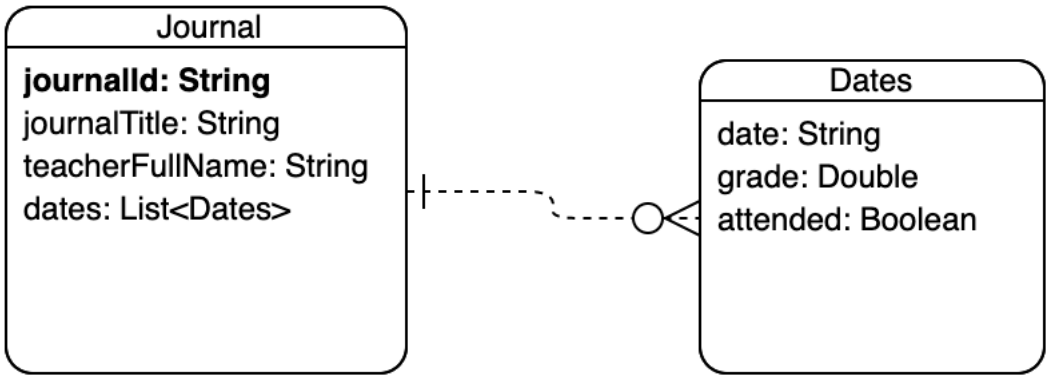

Рис. 3.6 – ER-диаграмма моделей Journal и Dates

Модель Attestation:
– a - Id аттестации
– subjectTitle - Название предмета
– att1 - Балл за первую аттестацию
– att2 - Балл за вторую аттестацию
– final - Балл за экзамен
– total – Общий балл по предмету
Подробности и связь моделей Attestation отображены на рисунке 3.7

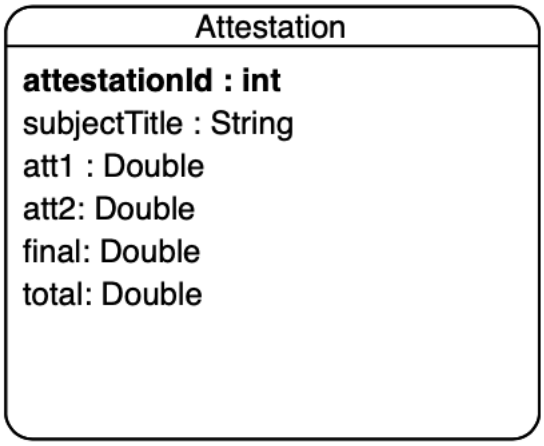

Рис. 3.7 – ER-диаграмма моделей Attestation

Модель Transcript:
– id - Id транскрипта
– semesters - список всех пройденных семестров

Модель Semester:
– semesterPeriod - период семестра
– semesterCredits - кол-во кредитов за семестр
– semesterECTS - ECTS за семестр
– confirmedSemesterCredits - засчитанные кредиты за семестр
– semesterGPA - GPA за семестр
– yearGPA - GPA за год
– cumulativeGPA - кумулятивный GPA
– subjects - список предметов за семестр
Модель subject:
– subjectTitle - название предмета
– subjectPoint - балл за предмет
– subjectLetter - буквенная оценка за предмет
– subjectCredits - кол-во кредитов за предмет

Подробности и связь моделей Transcript, Semester и Subject отображены на рисунке 3.8

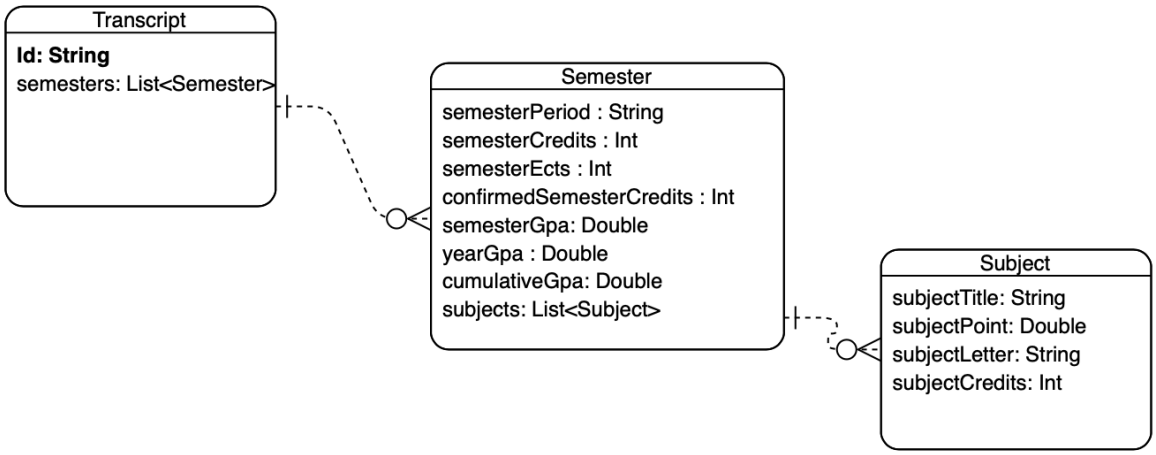

Рис. 3.8 – ER-диаграмма моделей Transcript, Semester и Subject

Модель Schedule:
– id - Id занятия
– lessonTitle - название занятия
– lessonTutor - преподаватель занятия
– lessonType - тип занятия
– lessonBuilding - название здания
– lessonClass - номер класса
– lessonDay - день проведения занятия
– lessonStart - начало занятия
– lessonEnd - конец занятия

Подробности модели Schedule отображены на рисунке 3.9

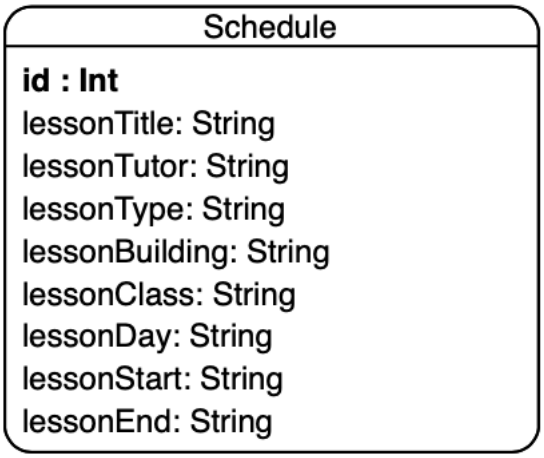

Рис. 3.9 – ER-диаграмма модели Schedule

Модель Exam:
- id - Id проводимого экзамена
- examTitle - название занятия
- examinerFullname - экзаменатор
- proctorrFullname - проктор
- examBuilding - название здания
- examClass - номер класса
- examDay - день проведения экзамена
- examStart - начало экзамена
- examEnd - конец экзамена

Подробности модели Exam отображены на рисунке 3.10

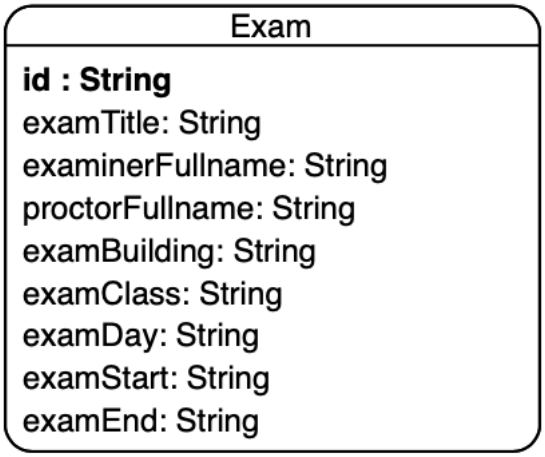

Рис. 3.10 – ER-диаграмма модели Exam

Модель News:
– id - Id новости
– title - название новости
– body - информация по новости
Подробности модели News отображены на рисунке 3.11

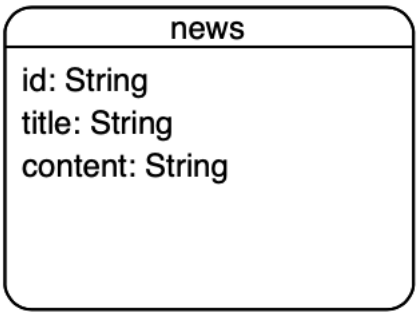

Рис. 3.11 – ER-диаграмма модели News

Все модели представлены специальным классом: Kotlin data class, упрощающим доступ к информации из модели.

В связи с тем, что модели также представляют Entity, Room создает аналогичные таблицы в локальной БД.

4 Экспериментальный раздел

4.1 Пользовательский интерфейс (UI)

В связи с вышеописанной Single-Activity Architecture. Все View с которыми взаимодействует пользователь, являются Fragment. Также этот подход позволяет уменьшить количество повторяющегося кода, путем создание общих для всех Fragment элементов. Таковыми являются большинство элементов компонента Navigation: ActionBar, BottomNavigationBar, Navigation Layout.

Приложение поделено на три части:

– Успеваемость
– УМКД
– Настройки

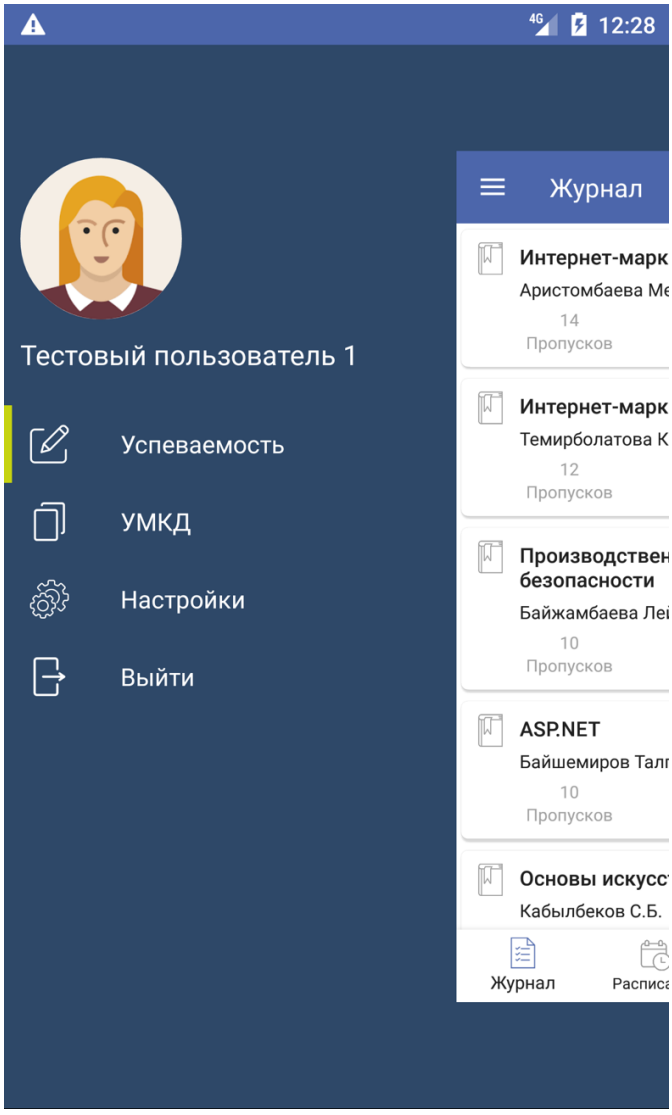

Рис 4.1 – Вид бокового меню, отображающего все разделы приложения

4.1.1 Успеваемость

“Успеваемость” является основным разделом приложения.

Раздел успеваемость имеет 4 основных Fragment: JournalFragment – страница “Журнал”, MainScheduleFragment – страница “Расписание”, EvaluationFragment – страница “Оценки” и NotificationFragment – страница “Уведомления”.

В странице “Журнал” отображается текущая успеваемость по каждому предмету текущего семестра. Журнал представляет из себя список карточек с данными по предмету:

– Название предмета;
– ФИО преподавателя ведущего занятие;
– Кол-во пропусков;
– Кол-во полученных баллов;

Из данного фрагмента можно попасть во фрагмент SubjectDetailsFragment, который отображает подробности по выбранному предмету.

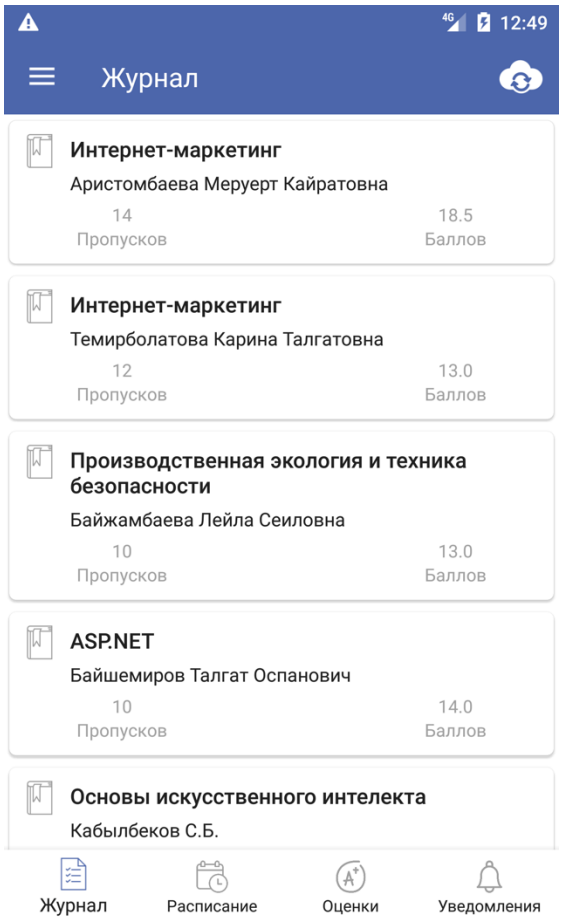

Рис 4.2 – Окно, отображающее журнал

Во фрагменте SubjectDetailsFragment можно посмотреть оценку, посещаемость и дату каждого пройденного занятия.

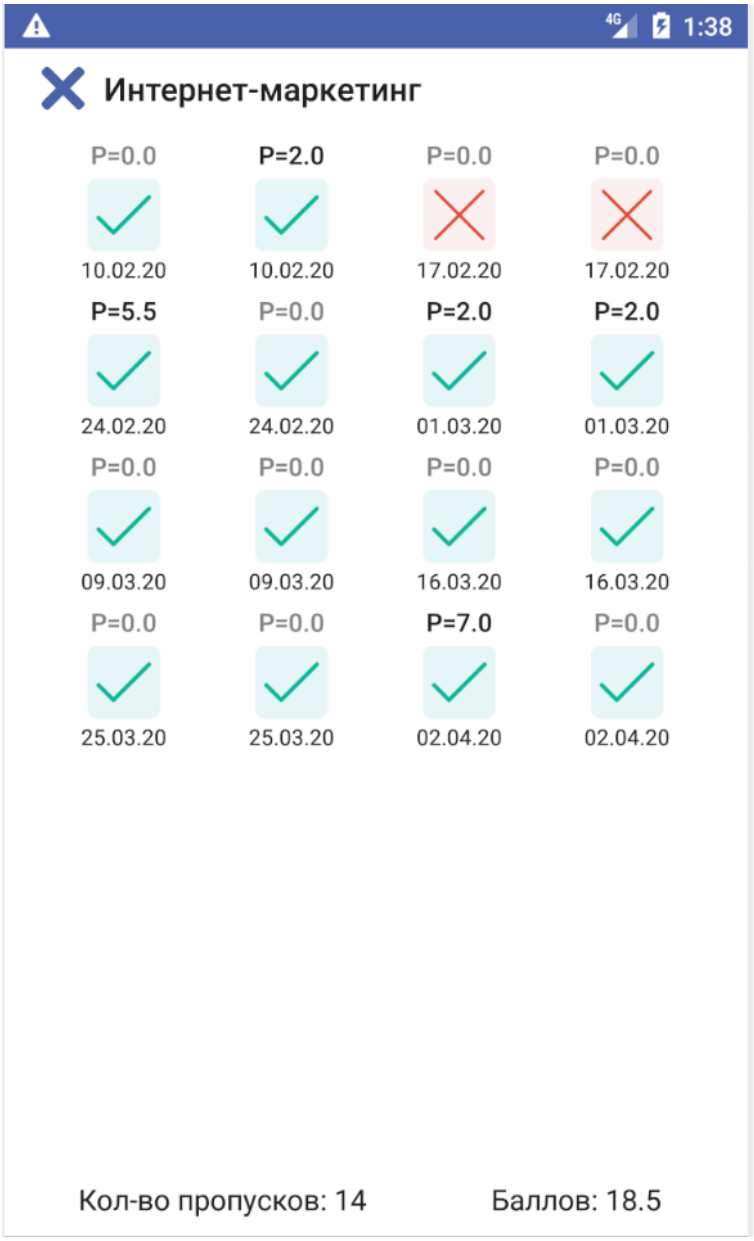

Рис 4.3 – Окно, отображающее детализацию предмета

Страница “Расписание” состоит из двух фрагментов: MainScheduleFragment и ExamScheduleFragment. Окно отображающее расписание предметов выполнено в виде календаря и списка предметов по выбранной дате. Можно использовать свайп для перехода по дням недели, а также выбор определенной даты. Список предметов отображает:

– Время начала и конец занятия;
– Название предмета;
– ФИО преподавателя;
– Тип занятия;
– Корпус и кабинет;

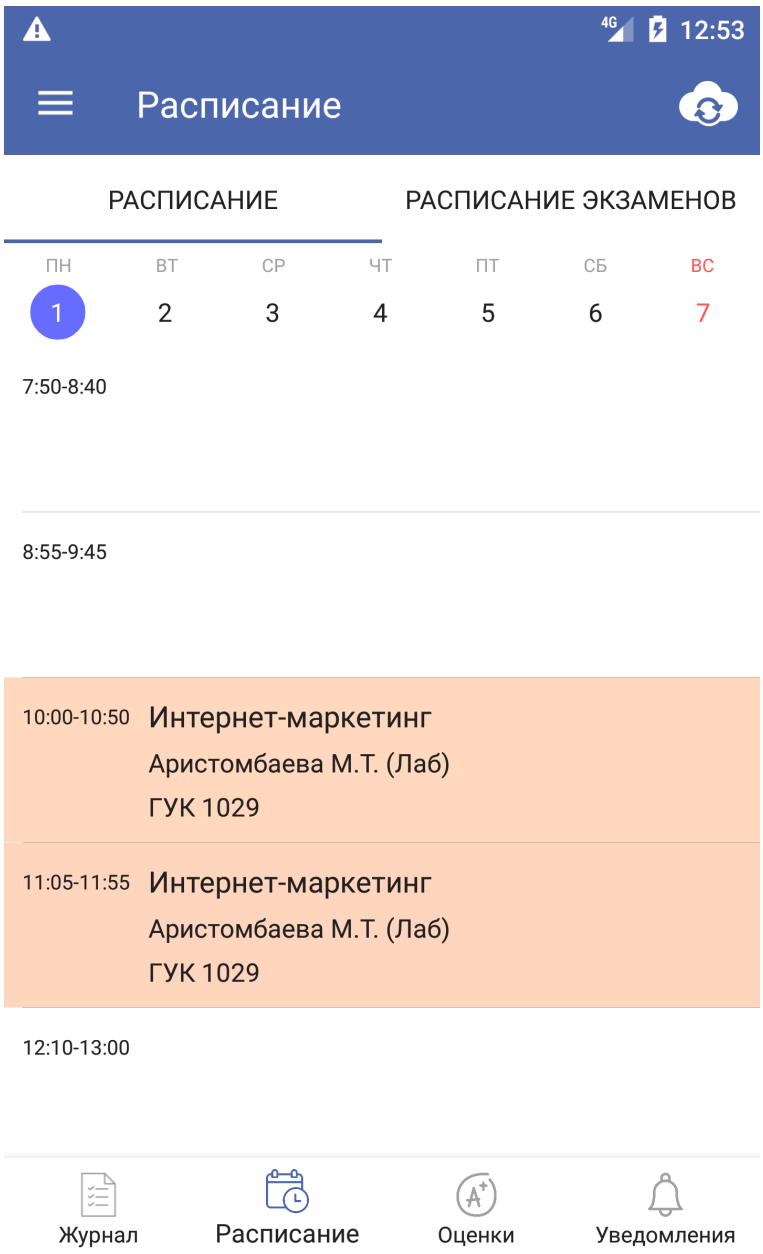

Рис 4.4 – Окно, расписания предмета

В случае если в выбранную дату нет занятий , то вместо списка предметов , отображается иконка отдыха.

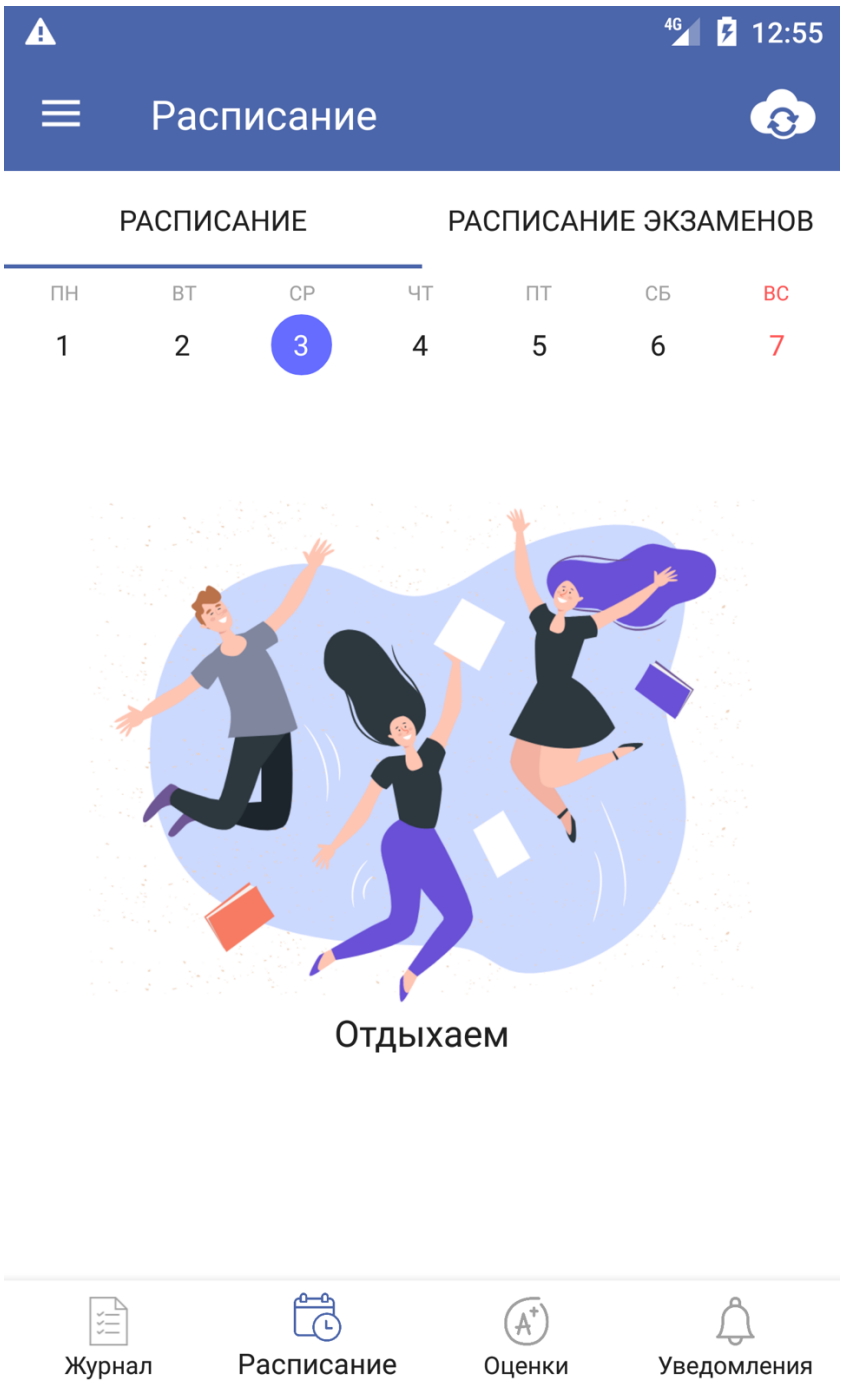

Рис 4.5 – Вид окна при отсутствии занятий

Окно расписания экзаменов выглядит в виде списка карточек с информацией о экзамене:

– Название предмета;
– Дата и время проведения экзамена;
– Кабинет;
– Экзаменатора и проктора;

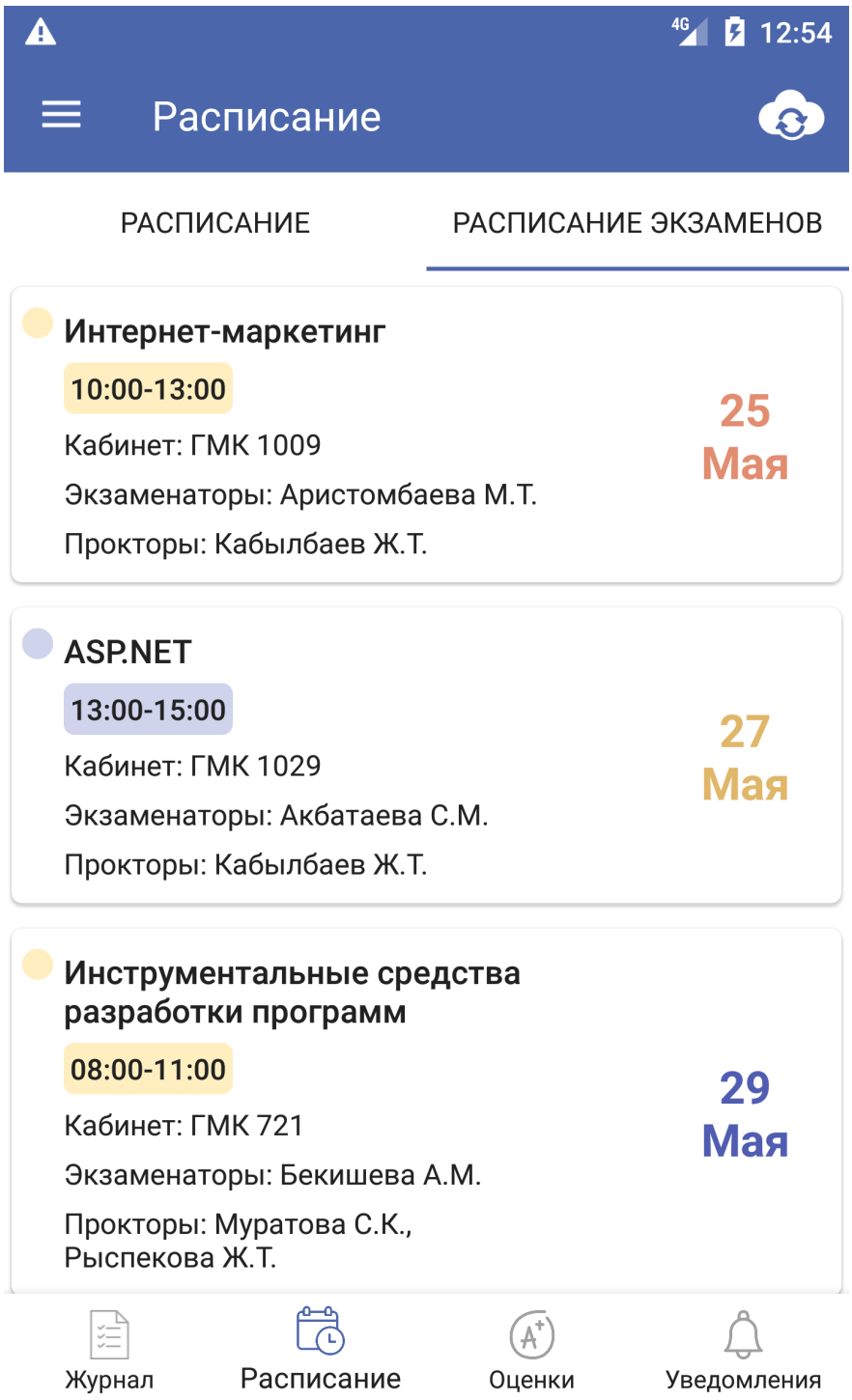

Рис 4.6 – Окно, отображающее проводимые экзамены

Страница “Оценки” позволяет смотреть текущую аттестацию с помощью AttestationFragment, в котором отображается полная информация о текущей аттестации:

– Балл за первую и вторую аттестацию;
– Балл за экзамен;
– Итоговый балл по предмету;

Рис 4.7 – Окно, отображающее информацию по аттестации

Также при свайпе страницы можно просмотреть данные по транскрипту за весь период обучения, за это отвечает TranscriptFragment.

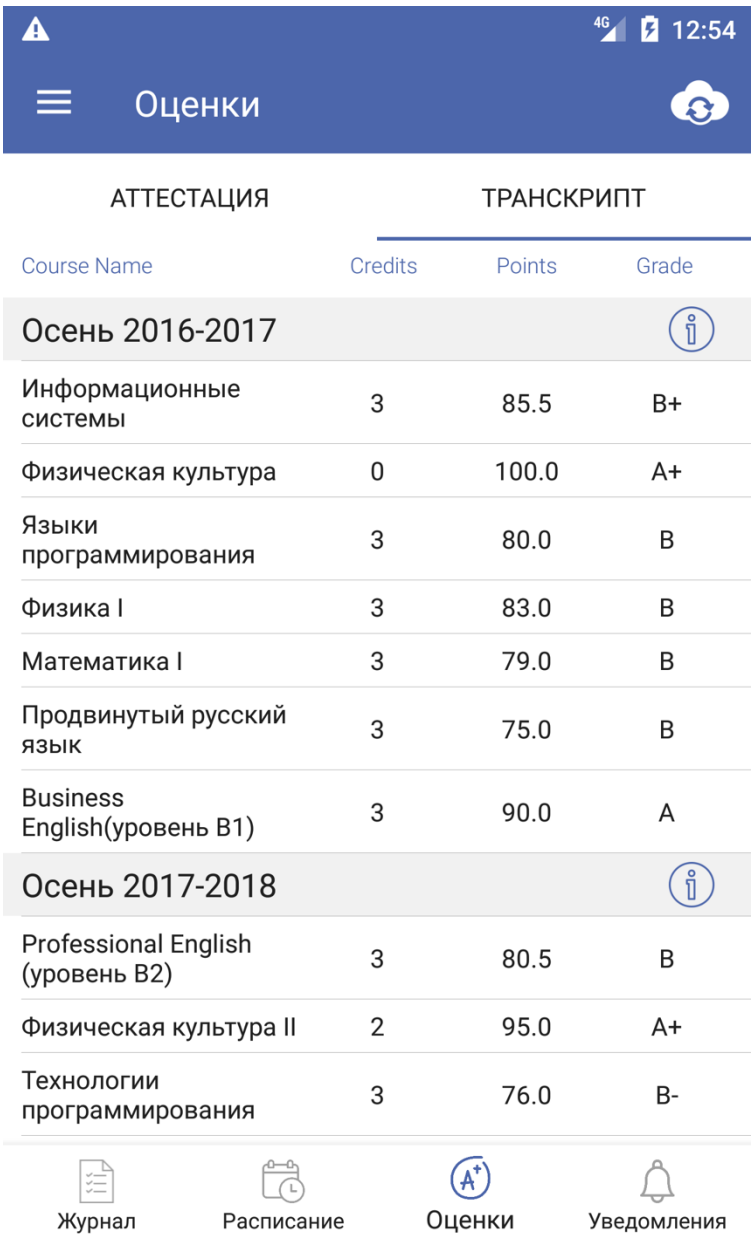

Рис 4.8 – Окно, отображающее данные по транскрипту

Транскрипт разделен на семестры обучения. В каждом семестре отображается информация по пройденным предметам. У каждого семестра при нажатии иконки “Info” можно посмотреть информацию по необходимому семестру:

- Кредиты за семестр;
- Засчитанные кредиты;
- GPA за семестр;
- годовой GPA;
- Кумулятивный GPA;

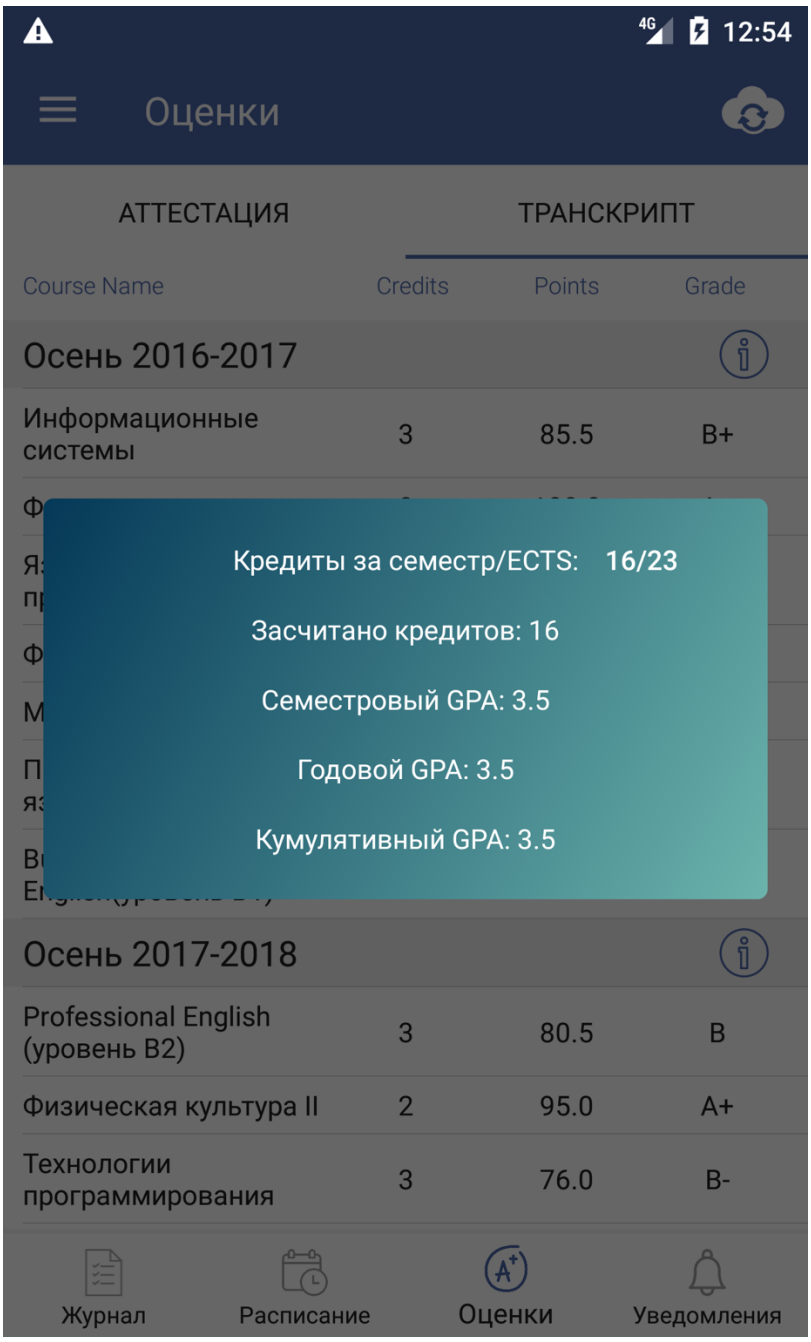

Рис 4.9 – Диалоговое окно, отображающее данные по семестру

На странице “Уведомления” можно узнать о новостях, связанных с университетской деятельностью. Данная страница выглядит как список названий всех предложенных новостей.

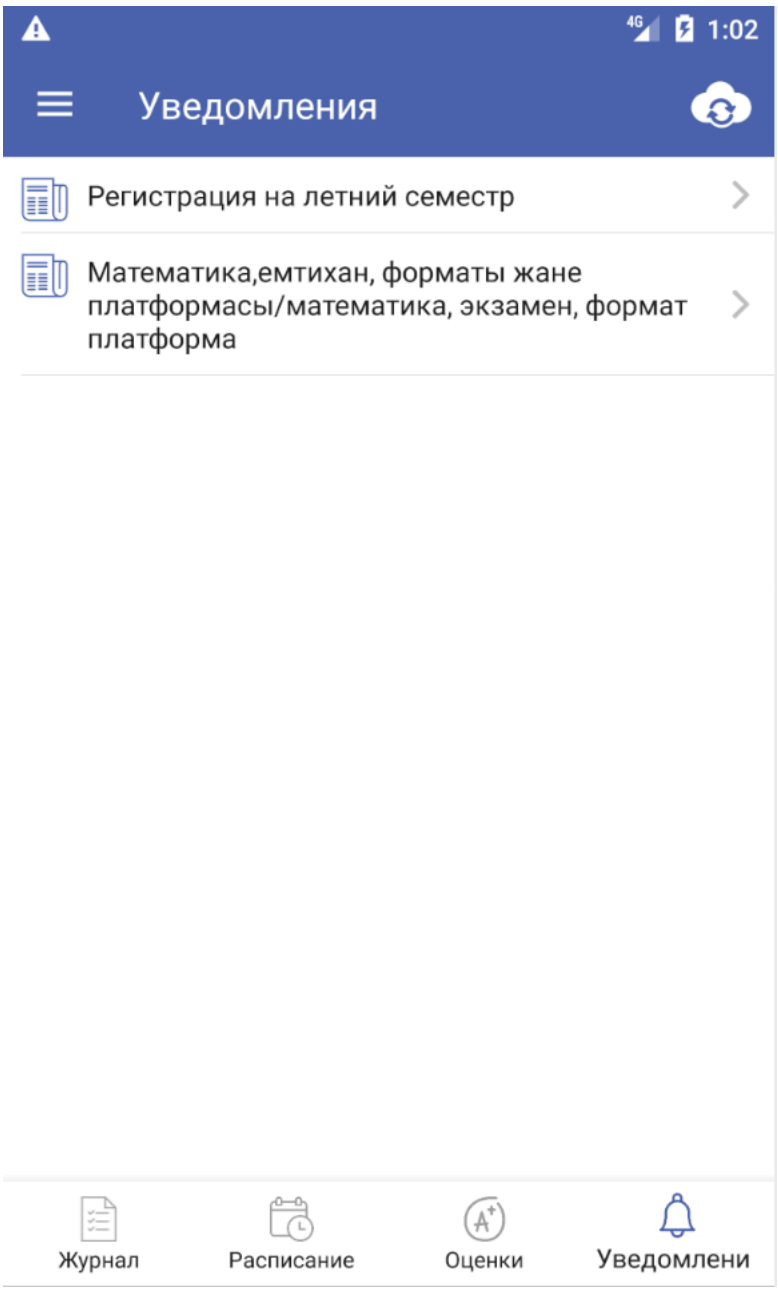

Рис 4.10 – Окно, отображающее все передоложенные новости

При нажатии на элемент списка происходит переход на страницу отображающую всю информацию по выбранной новости.

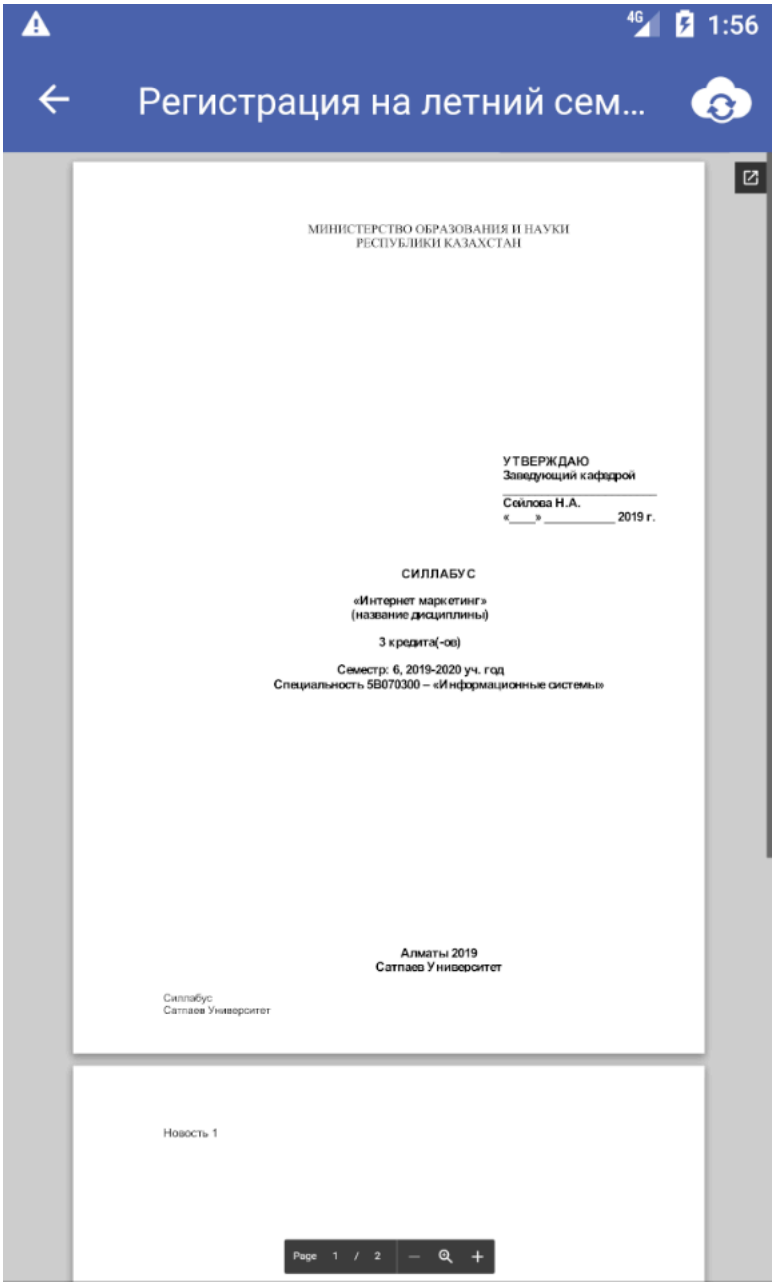

Рис 4.11 – Окно, отображающее информацию по выбранному элементу

В случае если в выбранной странице нет предоставляемых данных, то отображается иконка, предупреждающая пользователя об отсутствии информации.

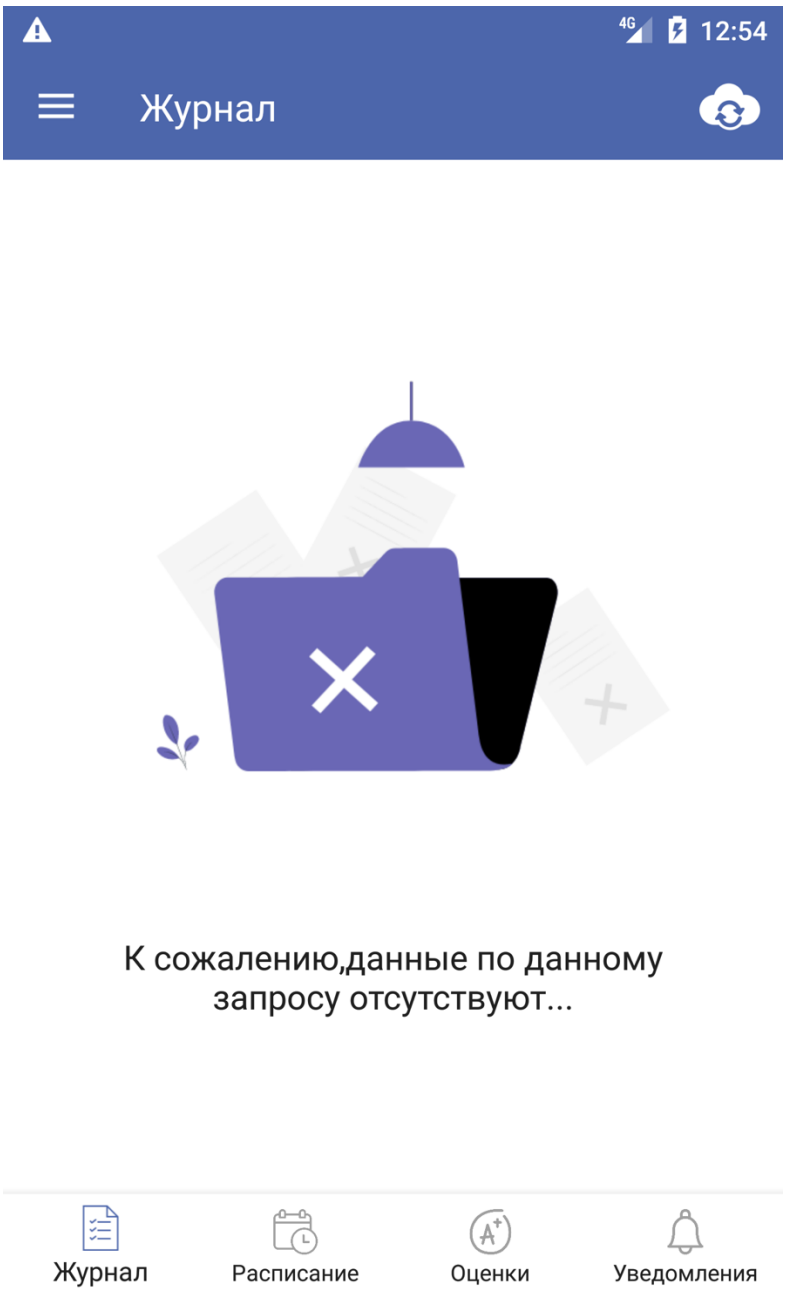

Рис 4.12 – Предупреждение об отсутствии необходимой информации

4.1.2 УМКД

Данный раздел предназначен для доступа ко всем учебно-методическим комплексам дисциплин за период текущего семестра обучения. Раздел построен на основе UmkdFragment.

Начальный экран УМКД представлен в виде списка текущих предметов и ФИО преподавателя ведущего предмет.

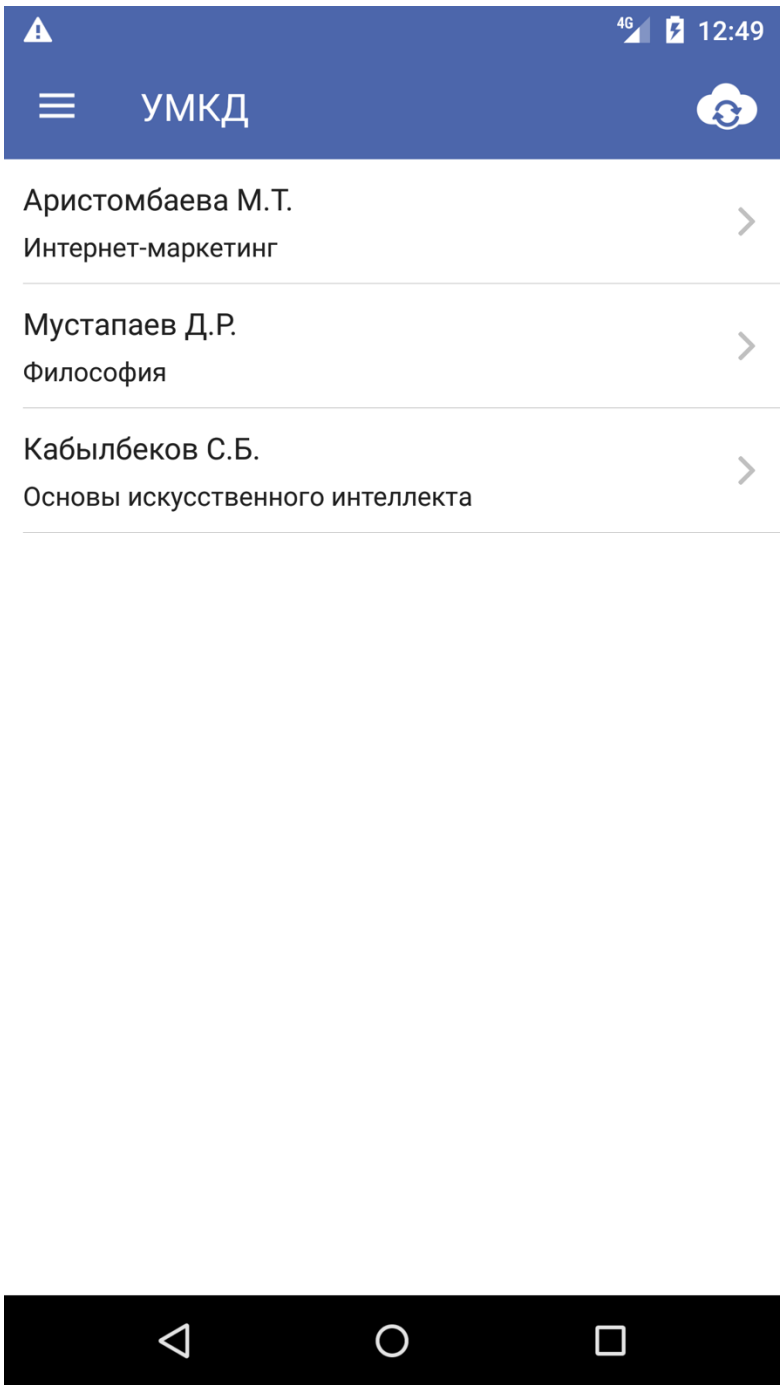

Рис 4.13 – Список текущих предметов студента

При нажатии на необходимый предмет, открывается окно с предоставляемыми УМКД по предмету:

- Лекции;
- Лабораторные работы;
- Силлабус;
- Экзамен;

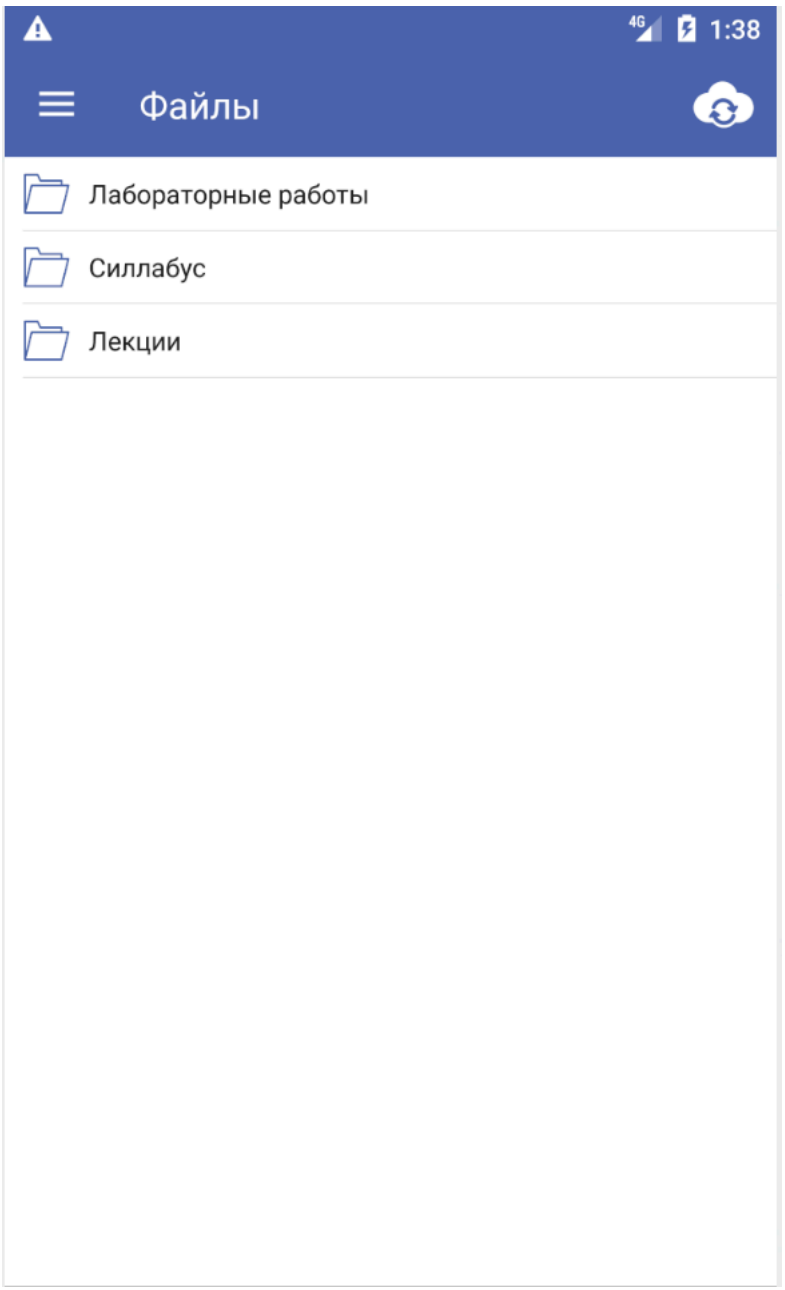

Рис 4.14 – Список предоставляемых разделов по предмету

После выбора необходимого раздела осуществляется переход к списку всех имеющихся документов, с возможностью просмотреть необходимый материал или скачать его.

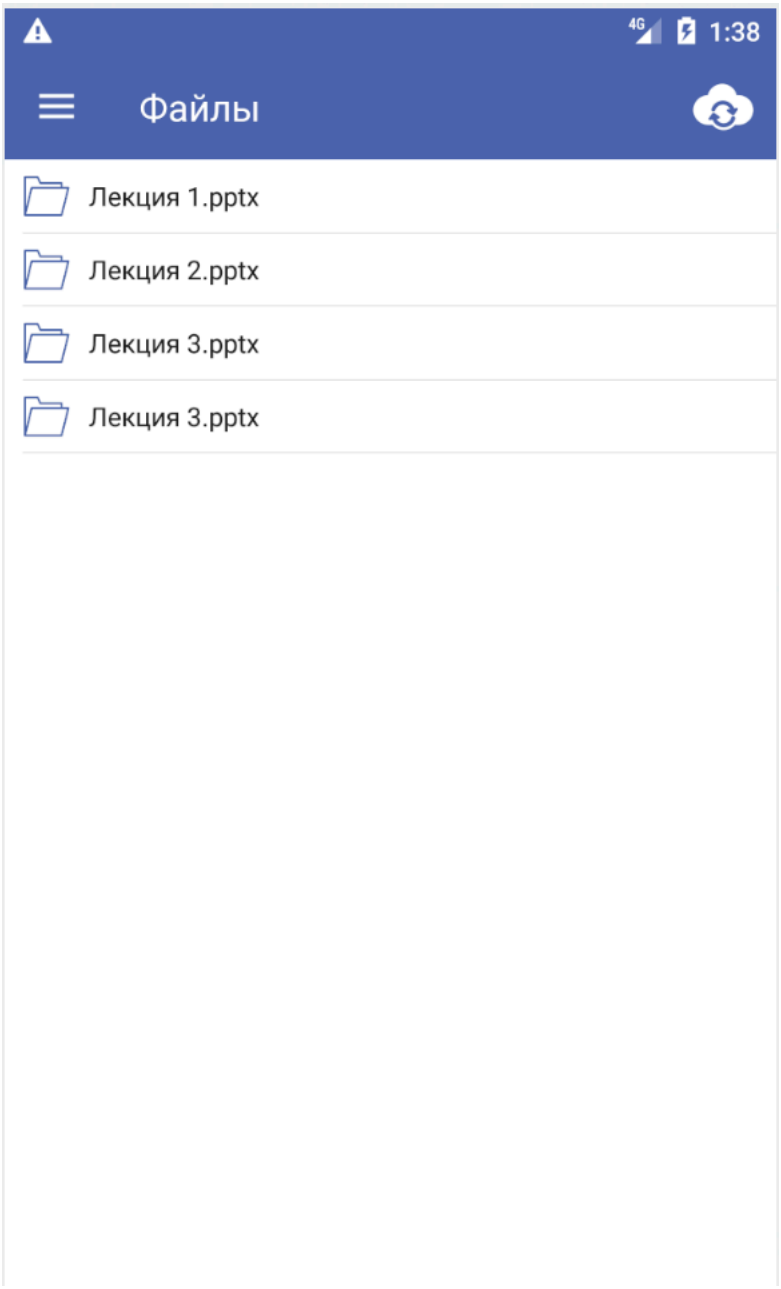

Рис 4.15 – Список всех доступных документов по дисциплине

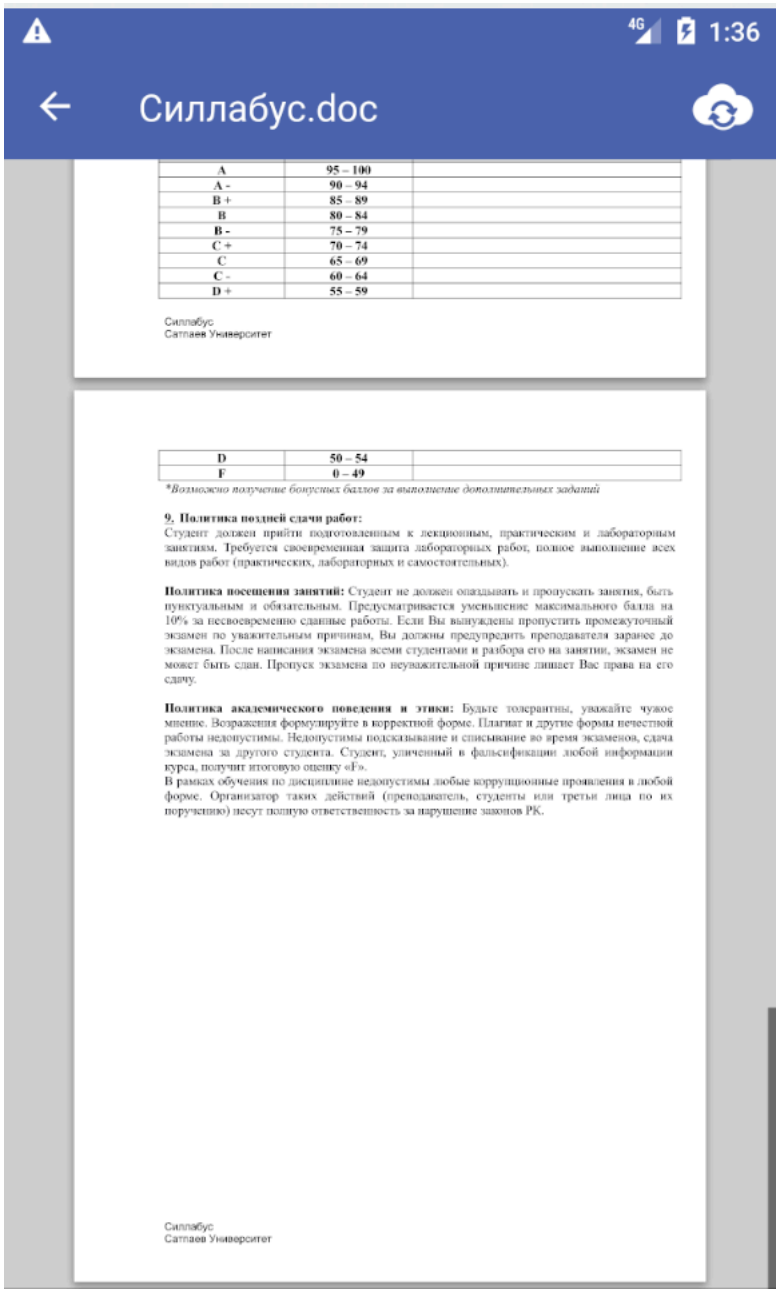

Рис 4.16 – Окно просмотра выбранного документа

4.1.2 Настройки

Данный раздел построен в виде личного кабинета пользователя. Основой данного раздела является SettingsFragement. Окно предоставляет из себя фотографию и ФИО пользователя. Также предоставляется возможность изменить язык приложения. И выйти из текущей сессии.

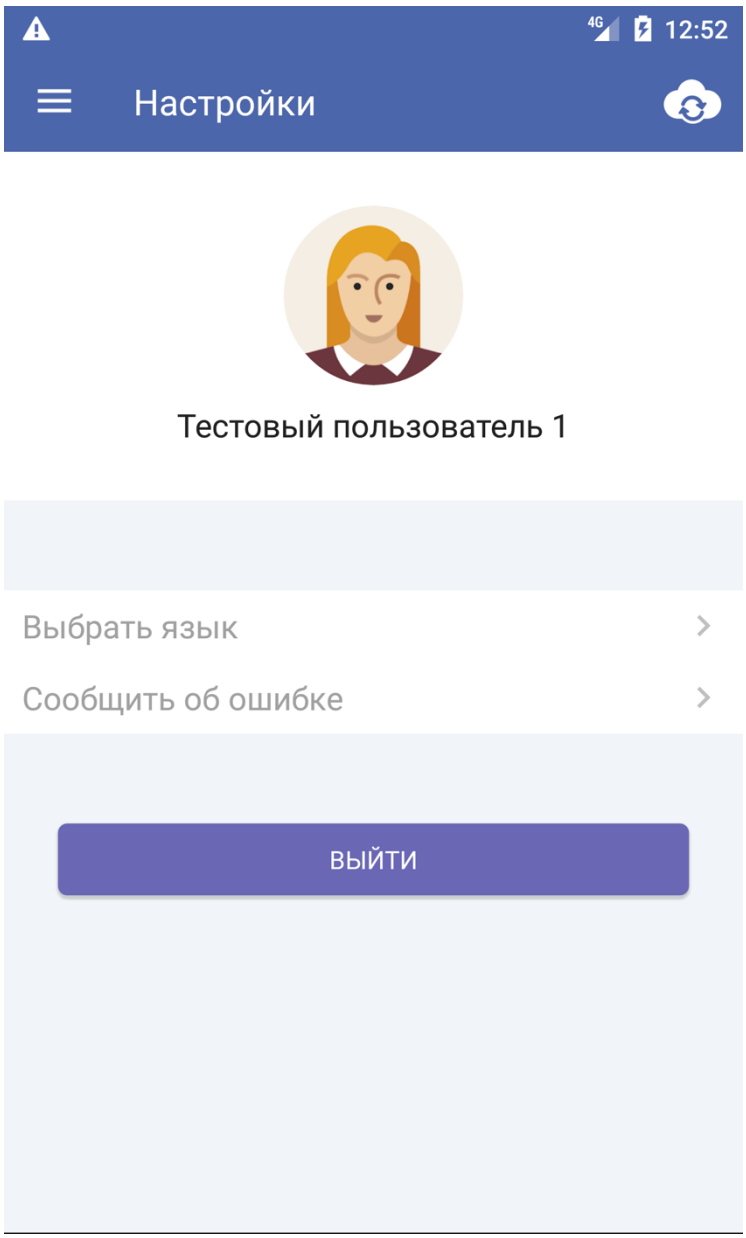

Рис 4.17 – Окно, отображающее раздел настройки

ЗАКЛЮЧЕНИЕ

В результате выполнения данной дипломной работы было разработано готовое мобильное приложение “Satbayev University” под ОС Android. Основными пользователями получившегося программного обеспечения должны стать студенты обучающиеся в Вузе “Satbayev University”. Приложение позволяет получить комфортный доступ к интересующей информации в любое время, через портативное устройство.

В ходе работы над мобильным приложением была полностью разработана подходящая в данном случае архитектура приложения, позволяющая дорабатывать, расширять уже имеющийся функционал, и предоставляющая возможность дополнять новыми компонентами. Проведен анализ для подбора оптимального стека технологий. Также, была подробно проделана работа над бизнес-логикой приложения, с момента получения данных до вывода ее пользователю.

Был досконально изучен пользовательский интерфейс уже имеющегося приложения на базе IOS. В дальнейшем, с которого был, обрисован интерфейс мобильного приложения на Android.

Финальный дипломный проект имеет возможность быть размещенным в Google Play Market, дополняя имеющуюся информационную систему университета “Satbayev University” мобильным приложением под Android.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Adroid Jetpack // Электронная версия на сайте https://developer.android.com/jetpack

2 Android Architecture components // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture

3 Basics of Kotlin Coroutines // Электронная версия на сайте https://kotlinlang.org/docs/reference/coroutines/basics.html

4 Data Binding Library // Электронная версия на сайте https://developer.android.com/topic/libraries/data-binding

5 Navigation component // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture/navigation

6 Room Persistence Library // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture/room

7 ViewModel Overview // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture/viewmodel

8 LiveData Overview // Электронная версия на сайте https://developer.android.com/topic/libraries/architecture/livedata

Приложение А

Техническое задание

А.1.1 Техническое задание на разработку мобильного приложения

Техническое задание распространяется на разработку мобильного приложения на ОС Android, предназначенного для отслеживания учебного процесса. Предполагается, что пользователями данного приложения будут студенты обучающиеся в ВУЗе “Satbayev University”. Мобильное приложение позволит владельцам устройств на платформе Android отслеживать динамику учебного процесса, позволит иметь мобильный доступ к УМКД.

А.1.1.1 Основание на разработку

Приложение разрабатывается на основании дипломной работы

А.1.1.2 Назначение

Приложение предназначено для отображения сведений об учебной успеваемости студентов обучающихся в “Satbayev University”.

А.1.1.3 Требования к функциональным характеристикам

Мобильное приложение должно обеспечивать возможность выполнения следующего функционала:

– отображение текущего журнала студента;
– отображение расписания обучающегося;
– отображение расписания экзаменов обучающегося;
– отображение текущей аттестации обучающегося;
– отображение подробного транскрипта обучающегося за весь период обучения;
– отображение уведомлений;
– отображение УКМД текущего семестра;
– возможность смены языка приложения;
– возможность авторизации;

А.1.1.4 Требования к надежности

Предусмотреть возможные исключительные ситуации при работе пользователя с мобильным приложением. Предусмотреть устойчивою работу приложения. Обеспечить целостность данных.

А.1.1.5 Требования к составу и параметрам технических средств

Для полноценной работы приложения требуется подключение к сети интернет.

А.1.1.6 Требования к информационной и программной совместимости

Система должна работать на устройствах под управлением операционной системы Android, начиная с версии 4.0.3

А.1.1.7 Требования к программной документации

Тексты программ должны быть содержать все необходимые комментарии.

Приложение Б

Текст программы

```
//Данный пример кода описывает компонент “журнал”, остальные компоненты строятся аналогично

//Модель Journal

package su.app.repository.model.journal

import androidx.room.Entity
import androidx.room.PrimaryKey
import java.io.Serializable

@Entity(tableName = "journal")
data class Journal(
    @PrimaryKey
    var journalId: String = "",
    var journalTitle: String = "",
    var teacherFullName: String = "",
    var dates: List<Dates>

):Serializable{

    val getGrade: Double
        get() {
            return this.dates.sumByDouble { it.grade }
        }

    val getMissed: Int
        get() {
            var n: Int = 0
            for(i in this.dates){
                if(i.attended) n+=1
            }
            return n
        }
}

//Модель Dates
package su.app.repository.model.journal

data class Dates(
    var date: String = "",
```

```
    var attended: Boolean = false,
    var grade: Double = 0.0
)



//Реализация http- клиента

package su.app.repository.network

import kotlinx.coroutines.Deferred
import retrofit2.http.GET
import su.app.repository.model.evaluation.attestation.Attestation
import su.app.repository.model.journal.Journal

interface ApiService {
    @GET("/journal")
    fun getJournal(): Deferred<List<Journal>>
}


//Реализация RemoteDataSource

package su.app.repository.network.datasources.Journal

import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import com.google.android.material.snackbar.Snackbar
import su.app.repository.model.journal.Journal
import su.app.repository.network.ApiService
import java.lang.Exception
import javax.inject.Inject

class JournalRemoteDataSourceImpl @Inject constructor(private val apiService:
ApiService): JournalRemoteDataSource {
    private val _downloadedJournal = MutableLiveData<List<Journal>>()
    override val downloadedJournal: LiveData<List<Journal>>
        get() = _downloadedJournal
    override suspend fun retrieveJournal(){
        try{
            val fetchedJournal = apiService.getJournal().await()
            _downloadedJournal.postValue(fetchedJournal)
        }catch (e: Exception){
```

```
            Log.e("Connection","No internet connection")
        }
    }
}


        //Реализация Journal DAO
package su.app.repository.room.dao

import androidx.lifecycle.LiveData
import androidx.room.*
import su.app.repository.model.journal.Journal

@Dao
interface JournalDao {
    @Query("SELECT * FROM journal")
    fun getJournal(): LiveData<List<Journal>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insert(journals: List<Journal>)

    @Query("DELETE FROM journal")
    fun deleteJournals()
}

//Реализация Journal Repository

package su.app.repository

import androidx.lifecycle.LiveData
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import su.app.repository.model.journal.Journal
import su.app.repository.network.datasources.Journal.JournalRemoteDataSource
import su.app.repository.room.dao.JournalDao
import javax.inject.Inject

class JournalRepositoryImpl @Inject constructor(
    private val journalRemoteDataSource: JournalRemoteDataSource,
    private val journalDao: JournalDao) : JournalRepository{
init{
```

```
        journalRemoteDataSource.apply {
            downloadedJournal.observeForever { JournalData ->
                persistRetrievedJournal(JournalData)
            }
        }
    }
    override suspend fun getJournal(): LiveData<List<Journal>> {
        journalDao.deleteJournals()
        return withContext(Dispatchers.IO){
            retrieveJournal()
            return@withContext journalDao.getJournal()
        }
    }

    private suspend fun retrieveJournal() {
        journalRemoteDataSource.retrieveJournal()
    }

    private fun persistRetrievedJournal(retrievedData: List<Journal>){
        GlobalScope.launch(Dispatchers.IO) {
            journalDao.insert(retrievedData)
        }
    }
}

// реализация JournalViewModel
package su.app.domain

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.*
import su.app.repository.JournalRepositoryImpl
import javax.inject.Inject

class JournalViewModel @Inject constructor(
    private val journalRepository: JournalRepositoryImpl
    ) : ViewModel() {

    val journal by lazyDeferred {
        journalRepository.getJournal()
    }

}
```

Продолжение приложения Б

```
//Реализация фрагмента JournalFragment
package su.app.presentation.ui.journal.list

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.lifecycle.Observer
import androidx.navigation.Navigation
import androidx.recyclerview.widget.LinearLayoutManager
import dagger.android.support.DaggerFragment
import kotlinx.android.synthetic.main.journal_fragments.*
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import su.app.R
import su.app.ViewModelFactory
import su.app.di.module.injectViewModel
import su.app.domain.JournalViewModel
import su.app.repository.model.journal.Journal
import javax.inject.Inject

class JournalFragment: DaggerFragment(),ItemClickListener{


    @Inject
    lateinit var viewModelFactory: ViewModelFactory

    private  lateinit var  journalViewModel: JournalViewModel

    private  val adapter: JournalAdapter by lazy{
        JournalAdapter(arrayListOf(),this)
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        journalViewModel = injectViewModel(viewModelFactory)
```

```
        val root = inflater.inflate(R.layout.journal_fragments, container, false)
        return root
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)

        journal_recyclerview.layoutManager = LinearLayoutManager(activity)
        journal_recyclerview.adapter = adapter
        observeData()
    }


    private fun drawRecyclerView(lessons: List<Journal>) {
        adapter.refresh(lessons)
        adapter.notifyDataSetChanged()
    }


    private fun observeData() = GlobalScope.launch(Dispatchers.Main) {
        val journals = journalViewModel.attestation.await()

        journals.observe(this@JournalFragment, Observer { journalList ->
            if (journalList == null) return@Observer
            drawRecyclerView(journalList)
        })
    }

    private fun showJournalDetails(journal:Journal,view: View){
        val actionDetails = JournalFragmentDirections?.journalDetails(journal)
        Navigation.findNavController(view).navigate(actionDetails)
    }

    override fun onItemClicked(journal: Journal,view: View) {
        showJournalDetails(journal,view)
    }

}
```