Министерство образования и науки Российской Федерации

ФГАОУ ВО «Северо-Восточный федеральный университет имени М.К. Аммосова»

Институт математики и информатики

Кафедра информационных технологий

ДОПУСТИТЬ К ЗАЩИТЕ

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_/С.Д.Мордовской/

Протокол №\_\_\_ от «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

мобильное Приложение «Справочник студента» для платформы android

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

направление: 02.03.02 «Фундаментальная информатика и информационные технологии»

Выполнил: студент IV курса

группы ФИИТ-14, ИМИ СВФУ

Анисимов Анатолий Юрьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель: доцент кафедры ИТ,   
к.ф.-м.н. Леверьев Владимир Семенович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Якутск

2018

**Содержание**

[Введение 3](#_Toc515445170)

[Глава 1. Теоретическая 5](#_Toc515445171)

[1.1 Выбор ПО для разработки приложения 5](#_Toc515445172)

[1.2 Выбор языка программирования 7](#_Toc515445173)

[1.3 Django 9](#_Toc515445174)

[1.4 REST 10](#_Toc515445175)

[1.5 Django REST Framework 11](#_Toc515445176)

[1.6 Retrofit 2 12](#_Toc515445177)

[1.7 Требования к приложению 12](#_Toc515445178)

[Выводы по первой главе 13](#_Toc515445179)

[Глава 2. Разработка приложения 14](#_Toc515445180)

[2.1. Описание приложения 14](#_Toc515445181)

[Заключение 20](#_Toc515445182)

[Список использованной литературы 21](#_Toc515445183)

[Приложения 22](#_Toc515445184)

# Введение

Тема данной выпускной квалификационной работы: мобильное приложение «Справочник студента» для платформы Android.

В последние годы платформа Android обрела широкую популярность во всем мире, у всех от мала до велика, имеется по смартфону с этой операционной системой.

**Актуальность:** во время пар бывает необходимо сверяться со справочниками для уточнения нужной информации, например, на сайте yagu.s-vfu.ru, а переключаться с программы в браузер начинает надоедать. В этом случае удобно использовать сторонний девайс для просмотра справочника на нем. Смартфон для этого как раз и подходит.

Приложение разработано и оптимизировано специально для смартфона, отображается лучше, чем сайт Moodle отображается в браузере, где надо увеличивать масштаб и попадать на ссылки. Благодаря кэшированию может работать без подключения к интернету, но для этого обязательно должна быть выполнена предварительная загрузка данных минимум 1 раз.

*Объектом исследования* является разработка мобильного приложения «Справочник студента» и веб-сервера для хранения данных. *Предметом исследования* выступает изучение среды разработки на Android, сопутствующего программного обеспечения, изучение веб-разработки, и изучение предметной области справочника.

**Цель работы**: разработать приложение «Справочник студента», работающее с веб-сервером и предоставляющее полную или достаточную информацию по выбранной теме.

Данная цель будет достигаться путем решения следующих **задач**:

1. Проанализировать и изучить среду разработки для Android, веб-разработку, предметную область выбранной темы.
2. Разработать веб-сервер и приложение по выбранной теме.

Выпускная квалификационная работа содержит введение, две главы, заключение, список использованной литературы и приложения. В первой главе рассмотрен выбор программных средств разработки: Android Studio, язык программирования Java, веб-фреймворк Django, язык програмиирования Python. Во второй главе описан процесс разработки веб-сервера и приложения.

# Глава 1. Теоретическая

## 1.1 Выбор ПО для разработки Android приложения

Здесь будут рассмотрены ПО для разработки Android приложения от разных компаний. Они делятся на кроссплатформенные (код написанный с использованием фреймворка переводится в нативный код для ОС, таким образом, экономится время при разработке приложений, один и тот же код подходит для нескольких ОС) и нативные (в которых используются инструменты и оригинальные языки ОС), а также игровые движки.

К кроссплатформенным относятся:

**PhoneGap** – это бесплатный фреймворк для разработки мобильных приложений. Код пишется на JavaScript, и компилируется в готовые приложения для любой мобильной ОС. Необходимость в знаниях родных языков программирования каждой из мобильных ОС отпадает, но требуется знание JavaScript. Приложение запускает активность с WebView и посредством него вызывает машинные команды устройства.

**Xamarin** – это тоже бесплатный фреймворк для разработки приложений, но не только мобильных, но и для Windows и Mac OS. Используя язык, основанный на базе C#, разработчики могут использовать инструменты Xamarin для написания нативных Android, iOS и Windows приложения с нативным пользовательским интерфейсом и распространять код на нескольких ОС, включая Windows и macOS. В 2016 году куплен Microsoft и полностью интегрирован в Visual Studio.

**React.Native** – это фреймворк, который строит иерархию компонентов пользовательского интерфейса для построения JavaScript кода. Он имеет набор компонентов, для обеих платформ iOS и Android для создания мобильных приложений с оригинальным внешним видом и ощущением. React.Native фокусируется на интерфейсе пользователя, что делает приложения, созданные с использованием его, быстрозагружаемыми и плавными. Для программирования на нем необходимо знание HTML, CSS и JavaScript.

**NativeScript** – это фреймворк с открытым исходным кодом, основанный на использовании языков JavaScript и TypeScript. Позволяет создавать кроссплатформенные приложения с пользовательским интерфейсом, написанным на платформо-независимом языке XML, который с помощью этих абстракций вызывает нативные компоненты интерфейса определенной платформы. Обеспечивает прямой доступ к API платформы, не требует компилирования кода на платформенном языке, он транслируется как есть в WebView на устройстве и запускается на нативном уровне.

К нативным относится:

**Android Studio** (созданная на базе IntelliJ IDEA Community Edition) в настоящее время является основной интегрированной средой, рекомендуемой для разработки приложений Android (исходные средства разработки Android работали на базе Eclipse IDE). Среда Android Studio в сочетании с бесплатным пакетом Android Software Development Kit (SDK) и бесплатным пакетом Java Development Kit (JDK) предоставляет все необходимое для создания, запуска и отладки приложений Android, поддержки их распространения (например, отправки в магазин Google Play) и т. д.

**Android SDK** – это набор инструментов для разработки программного обеспечения. Включает в себя отладчик, библиотеки, эмулятор, документацию, образцы кода и учебное пособие. Имеет возможности, отличающие его от других редакторов, среди которых: широкие функции теста и отладки кода, наблюдение результатов в реальном времени, оценивание работоспособности и совместимости приложения на разных версиях Android,

**Android NDK** – это набор инструментов, позволяющий включать код на нативных языках, таких как C и C++, и предоставляющий библиотеки платформ, которые помогают использовать нативные активности и получать доступ к компонентам устройства. Google не советует прибегать к NDK, если не требуется совершать сложных вычислений, но он может быть полезен для обеспечения быстроты и плавности работы приложения, либо повторном использовании библиотек, написанных на C или С++. При работе в Android Studio код, написанный на С или С++ компилируется в нативные библиотеки с помощью NDK, затем код Java вызывает функции библиотек NDK через Java Native Interface фреймворк

Также есть возможность создавать приложения с помощью игровых движков, например на **Unreal engine 4**, **Unity** и **LibGDX.**

**Unreal Engine 4 –** это на данный момент самый мощный игровой движок, разработанный, поддерживаемый и бесплатно распространяемый компанией Epic Games. Вовсю используется в разработке для iOS и Android в midcore-жанре, а многие студии рассматривают UE4 для веб-разработки. В частности, это касается мультиплатформенных проектов. В комплекте с ним идет набор инструментов разработки Unreal SDK. Использует язык программирования С++, есть функция графического программирования через Blueprint. Предоставляет высокий уровень портируемости, осуществляемый модульной системой зависимых компонентов. Можно писать приложения для Microsoft Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, Magic Leap One, and virtual reality (SteamVR/HTC Vive, Oculus Rift, PlayStation VR, Google Daydream, OSVR and Samsung Gear VR).

**Unity** – это кроссплатформенный игровой движок, разработанный Unity Technologies, который, в первую очередь, используется для разработки трехмерных и двумерных видеоигр и симуляций для компьютеров, консолей и мобильных девайсов. Впервые анонсирован только для macOS X, и с тех пор был расширен до 27 целевых платформ. Поддерживает 2D и 3D-графику, функциональность перетаскивания и программирования языком C#, Boo и JavaScript.

**LibGDX** – это бесплатный open source фреймворк разработки игр, написанный на Java c некоторыми С и С++ компонентами для производительного кода. Он позволяет разрабатывать десктопные и мобильные игры, используя одну и ту же основу кода. Кроссплатформенный, поддерживает Windows, Linux, Mac OS X, Android, iOS, Blackberry и веб-браузеры с поддержкой OpenGL. LibGDX дает разработчикам возможность писать, тестировать и отлаживать приложение на их ПК и использовать тот же код на Android. Он абстрагирует отличия между Windows, Linux приложениями и Android приложениями.

Для разработки была выбрана среда Android Studio. Она была выбрана потому, что знакомство с этой средой уже было налажено, было написано несколько тестовых приложений, в общем, имелся опыт работы на ней. В других средах надо было бы изучить язык С# либо JavaScript. Если первый еще знаком, на парах изучен более-менее, то второй надо было бы изучать почти с нуля. По языку Java достаточные знания есть, и плюс он является нативным языком, значит программирование будет вестись так, как оно подразумевается разработчиками Androida. Еще отчасти потому, что было решено довериться лицензионному ПО от Google, кто, как не они, знают, как правильно работать со своим детищем, да и они вроде все что делают, делают хорошо. Среда основательная, дизайн уютный, весь нужный функционал прикручен: присутствуют понятные и своевременные подсказки, и, например, программа может сама находить устаревший вариант кода и преобразовывать его в актуальный, можно оформлять внешний вид, окна, формы, тут же начать их программировать на языке Java, и сразу построить и запустить получившееся приложение на создаваемом через менеджер виртуальных устройств виртуальном устройстве, либо на своем смартфоне, предварительно подключив его через USB кабель в режиме отладки.

## 1.2 Выбор языка программирования

В Android Studio поддерживаются 2 языка программирования: Java и Kotlin.

Kotlin – это статический типизированный язык программирования. Работающий на Java Virtual Machine и также может быть скомпилирован в код JavaScript. Разработан команда программистов JetBrains из Санкт-Петербурга. Хотя его синтакс не совместим с Java, JVM-ная интерпретация стандартной библиотеки Kotlin предназначена для взаимодействия и совместимости с Java. Kotlin использует автоматическое определение типа для определения типа значений и выражений, для которых тип был оставлен неустановленным. Это снижает его похожесть с Java, который часто использует полностью избыточные спецификации типов. Начиная с Android Studio 3.0 Kotlin полностью поддерживаемый Google’ом язык программирования на Android ОС, включен в пакет Android Studio IDE как альтернативна стандартному компилятору Java. Первая версия вышла в 2016-м году, т.е. этот язык еще совсем молод. В основе его лежат простота, малое количество кода, и полная совместимость с Java. Это сделано для безболезненного и постепенного перехода разработчиков с Java на Kotlin.

Java — сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года.

Java — это программная платформа, включающая в себя мощную библиотеку, большой объем кода, пригодного для повторного использования, а также среду для выполнения программ, которая обеспечивает безопасность, независимость от операционной системы и автоматическую сборку "мусора".

Создатели Java составили официальное техническое описание, в котором объяснялись цели и достоинства нового языка. В этом документе приведено одиннадцать характерных особенностей Java. Этот язык:

* простой;
* объектно-ориентированный;
* распределенный;
* надежный;
* безопасный;
* не зависящий от архитектуры компьютера;
* переносимый;
* интерпретируемый;
* высокопроизводительный;
* многопоточный;
* динамичный.

Синтаксис Java, по существу, представляет собой упрощенный вариант синтаксиса C++. В этом языке отсутствует потребность в файлах заголовков, арифметике (и даже в синтаксисе) указателей, структурах, объединениях, перегрузке операций, виртуальных базовых классах и т.п.

Языком программирования был выбран Java, т.к. есть опыт работы на нем и на форумах и документации должно быть больше информации по возникающим вопросам.

## 1.3 Django

Согласно книге Владимира Дронова «Django: Практика создания веб-сайтов на Python», Django – это библиотека, созданная специально для разработки сайтов, ставшая самой популярной библиотекой такого рода из всех, написанных для использования с Python. Она реализует базовую функциональность веб-сайта и включает в себя все необходимое для программирования сайта, не занимаясь созданием всей инфраструктуры. Django содержит также средства для взаимодействия с базами данных, инструментами отладки, административный сайт, который можно использовать при работе с хранящимися в базе данными, и даже полнофункциональный веб-сервер, нужный для отладки сайта.

В возможности Django входит:

* реализация архитектуры «Model-View-Controller»;
* унифицированные средства для работы с БД любых поддерживаемых форматов;
* мощный шаблонизатор, основанный на специальных тегах, с возможностью наследования шаблонов;
* средства для работы с формами;
* инструменты для реализации разграничения доступа;
* встроенный административный сайт с возможностями настройки, который можно использовать для работы с данными;
* простая и ясная структура создаваемых сайтов: каждый раздел сайта представляет собой отдельное приложение, которое можно использовать в другом сайте, а их совокупность – сам сайт.

## 1.4 Язык программирования Python

Согласно книге Владимира Дронова «Django: Практика создания веб-сайтов на Python», Python – высокоуровневый, объектно-ориентированный, тьюринг-полный язык программирования, одинаково хорошо подходящий для разработки как простейших командных скриптов, так и сложных настольных и Web-приложений. В комплекте с ним поставляется богатейшая стандартная библиотека, включающая мощные средства для обработки текстов, поддержки шифрования, работы с файлами, реализации обмена данных через Интернет и многое другое.

## 1.5 REST

REST - (сокр. от англ. REpresentational State Transfer - «передача состояния представления») – это стиль архитектурного взаимодействия элементов распределенного приложения в сети. Основан на использовании ресурсов, а не действий посредством GET или POST запросов (Rest-запрос). Принимает запрос и посылает представление состояния ресурса в формате JSON или XML.

Требует соблюдения шести ограничений, направленных на общее улучшение работоспособности системы:

1. Унифицированный интерфейс.

Это основное требование, описывает интерфейсы клиента и сервера. Подразумевает использование HTTP запросов (GET, PUT, POST, DELETE), унифицированный идентификатор ресурса и прием ответов HTTP(status, body).

1. Отсутствие состояния.

Сервер не должен иметь информации о состоянии клиента, каждое сообщение должно быть самоописываемым и иметь достаточную информацию для ее обработки на сервере. Каждое состояние должно хранится на клиенте.

1. «Клиент-серверная» модель.

Клиент должен быть отделен от сервера, не иметь прямой доступ к ресурсам сервера, и наоборот. Мостом, связующим их, служит унифицированный интерфейс, использующий HTTP запросы.

1. Возможность кэширования.

Все, что приходит с сервера, должно быть кэшируемым, явно, неявно, либо по договоренности

1. Расслоенная система.

Клиент не должен знать, напрямую он обращается к серверу, либо к какому-то промежуточным элементом. Например, клиент не должен знать, принимает он данные с сервера, или с кэша. Это улучшает масштабируемость системы, обеспечивает многоуровневость.

1. Код по требованию.

Уникальное требование, означает возможность расширения клиента, путем отправки некоторой логики сервера на клиент как сценарий или апплет. Это единственное выборочное требование.

Соблюдение REST требований придает системе следующие свойства:

* масштабируемость;
* простоту;
* модифицируемость;
* прозрачность;
* мобильность компонентов;
* надежность.

## 1.6 Django REST Framework

Django REST Framework – это мощный и гибкий фреймворк, использующий подход REST (Representational State Transfer, передача состояния представления), для построения веб-API (от англ. application programming interface - интерфейс прикладного программирования), т.е. API для сервера.

Имеет возможности:

* просматриваемый в веб браузере API, что очень полезно для разработчиков;
* политика аутентификации, включающая пакеты OAuth1 и OAuth2;
* поддержка сериализации ORM и не-ORM источников данных;
* полностю настраиваемый – просто используйте обычные views, основанные на функциях, если не нуждаетесь в более мощных фичах;
* исчерпывающая документация и отличное сообщество, которое поможет вам советом;
* используется и доверен такими известными международными компаниями, как Mozilla, Red Hat и Eventbrite.

## 1.7 Retrofit 2

Документация описания Retrofit 2 крайне скупа, она гласит, что Retrofit 2 – это типобезопасный HTTP клиент для Android и Java, разработанный компанией Square. В связке с библиотекой Okhttp, предоставляющей возможность кэширования ответов сервера и управления header’ами, тоже, кстати, их детищем, - это необходимое приспособление для написания клиентского API для приложения, сообщающегося с сервером. Следует заветам REST архитектуры, поэтому его можно назвать RESTful API.

Работа с ним заключается в присоединении его к коду, создании классов, описывающих типы принимаемых данных и создании интерфейсов, содержащих в себе описание запроса с использованием команд HTTP.

## 1.8 Требования к приложению

Требования к приложению, учитывая специфику, следующие:

* быстрота;
* простота и понятность пользовательского интерфейса;
* получение данных с сервера и управление принятыми данными;
* кэширование полученных данных;
* показ списка дисциплин;
* показ списка лекций;
* четкое и разборчивое отображение текста лекции;

## Выводы по первой главе

В первой главе были приведены описания инструментов разработки ПО, выбор Android Studio и языка программирования Java. Описание веб-фреймворка Django, язык программирования Python, описание архитектуры REST, фреймворков RESTful API Django REST Framework, Retrofit 2.

Глава 2. Разработка приложения

## 2.1 Описание приложения

Темой справочника было решено сделать сборник курсов, доступных на сайте «yagu.s-vfu.ru», поэтому приложение названо «Справочник студента». Оно основано на теоретической базе лекций с сайта «yagu.s-vfu.ru», так же известном как «мудл». Приложение отправляет запрос серверу на отправку списков дисциплин, курсови тегов, по приему кэширует их и далее работает с кэшем. Выводит список имеющихся дисциплин. при нажатии на элемент списка, в новой активности выводит список лекций по выбранному предмету. При выборе лекции открывает новую активность с полным текстом лекции. По клавишам «Пред. лекция» и «След. лекция» можно перейти к предыдущей или следующей лекции из списка. В целях удобности и быстроты было решено не делать ненужных функций, нагромождений на экране. В итоге приложение вышло понятным, удобным, быстрым насколько возможно, с простором для дальнейших улучшений.

## 2.2 Разработка веб-сервера

В начале разработки первым делом необходимо создать условия для работы. На моем ПК в имени учетной записи пользователя имелись кириллические символы, что создавало бы конфликты при работе программ, поэтому была создана новая учетная запись, содержащая только буквы латинского алфавита.

Для разработки веб-сервера были нужны Python и Django. Сначала был установлен Python версии 3.6.5 с официального сайта, прописан путь в переменной PATH, и установлен IDE PyCharm от JetBrains для программирования. После установок в командной строке создана директория проекта. Создано виртуальное окружение(venv) проекта командой «python -m venv myvenv». Это нужно для того, чтобы огородить ПО этого проекта от конфликтов с ПО других проектов, если они будут. Потом этот venv нужно запустить командой «Scripts\activate». Затем в запущенном venv установлены фреймворки и библиотеки, нужные для разработки веб-сервера, командой «pip install -r requirements.txt», который устанавливает требуемые библиотеки, предварительно записанные в txt файл. На этом подготовка к началу работ над веб-сервером закончена.



Рис. 1 Запущено виртуальное окружение.

Создание проекта в командной строке «django-admin.exe startproject backend .», эта команда создаст папку проекта и необходимый минимум файлов для функционирования сервера. Затем нужно поправить файл settings.py: указать часовой пояс, язык, добавить ip в allowed hosts и создать базу данных командой «manage.py migrate». Затем создание администратора сервера командой «manage.py createsuperuser». После того, как все настроено и база данных создана, создается приложение командой «manage.py startapp backend» и заносится в список приложений в файле settings.py. Запуск приложения командой «manage.py runserver 0.0.0.0:8000», ip в конце нужно, чтобы сервер был виден в сети.

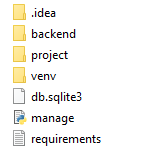


Рис. 2 Внутренности проекта, создаются автоматически.

В файле models.py создаем классы, которые будут служить таблицами в базе данных, определяем поля и их типы, задаем метаданные. Полный код models.py будет представлен в приложении (приложение 11).

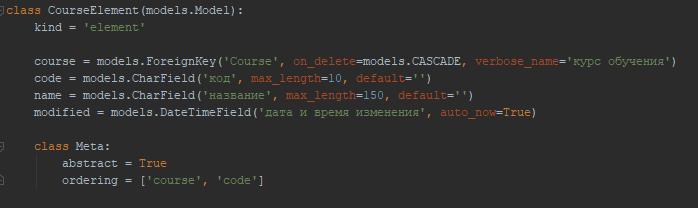


Рис. 3 Пример класса в файле models.py.

В файле admin.py регистрируем классы, созданные в models.py, чтобы они были видны на административной странице сервера. Полный код admin.py будет представлен в приложении (приложение 12).

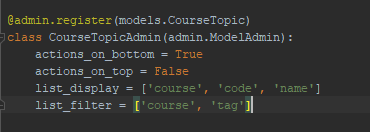


Рис. 4 Пример регистрации класса в файле admin.py.

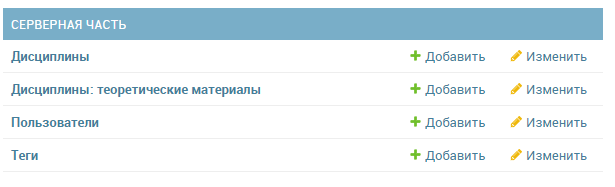


Рис. 5 Административная страница сервера.

Заполнение базы данных данными осуществляется через административный сайт, который имеет нужные инструменты для добавления, изменения, удаления и просмотра данных. И самое крутое, что не надо делать ничего сложного, эта функция вшита в Django, надо лишь указать, что мы хотим видеть там.

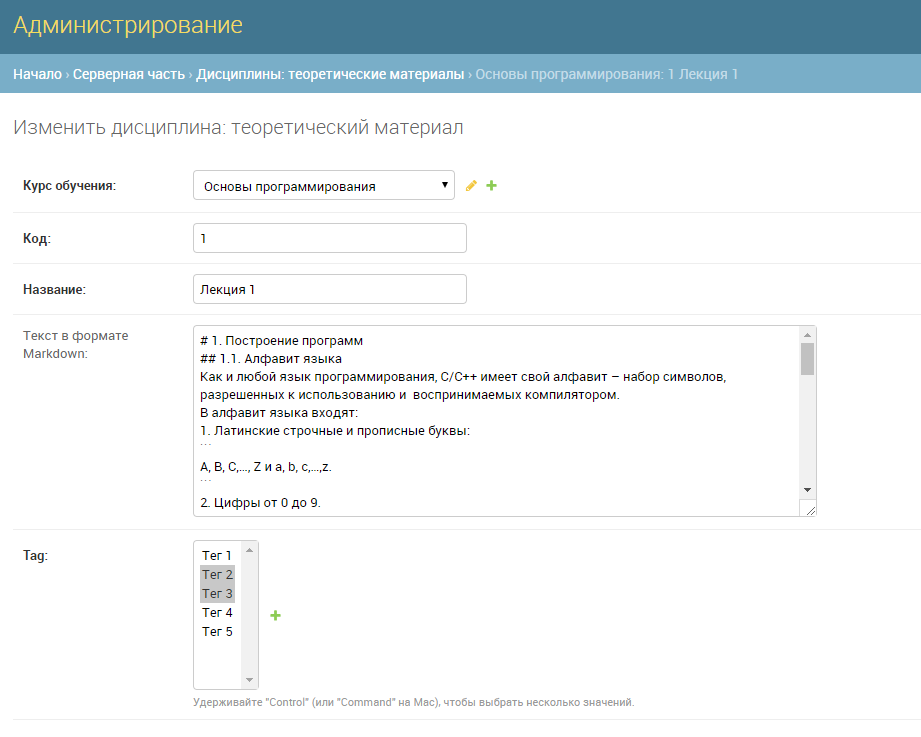


Рис. 6 Администрирование записей.

Когда база данных заполнена, нужно сделать API для сообщения с клиентом. Добавляем в список установленных приложений в файле settings.py название фреймворка для создания API, Django REST framework и добавляем запись в urls.py.

Затем создается файл api.py, в нем и будет описываться API. Добавляем в созданный файл сериализаторы и ViewSet’ы на каждый класс в models.py, указываем поля, которые хотим передавать клиенту. Таким образом, данные будут сериализовываться в JSON файл, который будет отправлен клиенту в теле ответа. Полный код api.py будет представлен в приложении (приложение 13).

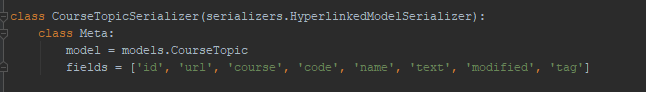


Рис. 7 Пример сериализатора в api.py.

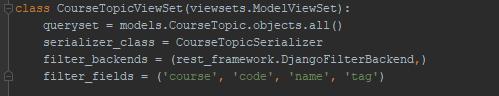


Рис. 8 Пример ViewSet’a в api.py.

Также стоит упомянуть о фильтрации, которая использовалась для фильтрации данных таблиц по значению поля, передавая данные в GET запросе. За нее ответственна библиотека Django filters. Устанавливается так же, как и Django REST Framework. Чтобы его использовать, нужно в файле api.py добавить строчку «filter\_fields» и указать в ней поля, по которым делать фильтрацию, как показано на рис. 8.



Рис. 9 Пример GET запроса с фильтром по дисциплинам.

В Django REST Framework использует функцию веб просматриваемого API, обращаясь к localhost’у можно видеть данные так, как их видит API.

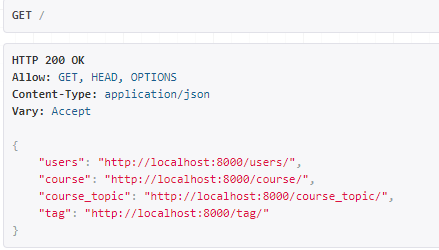


Рис. 10 Страница API.

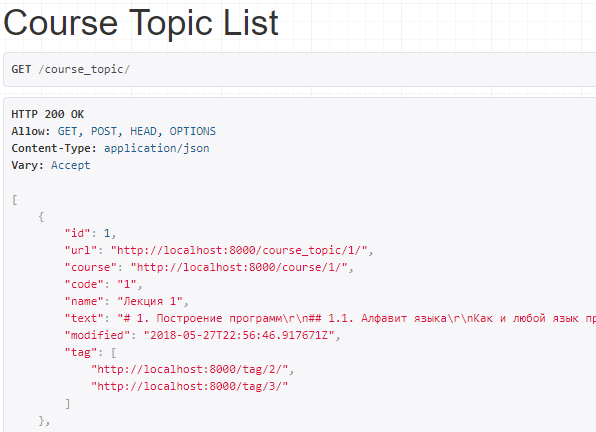


Рис. 11 Страница API, лекции.

## 2.3 Структура базы данных

База данных расположена на веб-сервере, используется СУБД Django по умолчанию - MySQL. Состоит из таблиц:

* users, имеет поля id, username, email, is\_staff, last\_name, first\_name, second\_name;
* courses, имеет поля id, url, name;
* tags, имеет поля id, name;
* course\_topics, имеет поля id, course, tag, url, code, name, text, modified;

Таблицы course\_topics и courses связаны по id course, таблицы course\_topics и tags связаны по id tags.

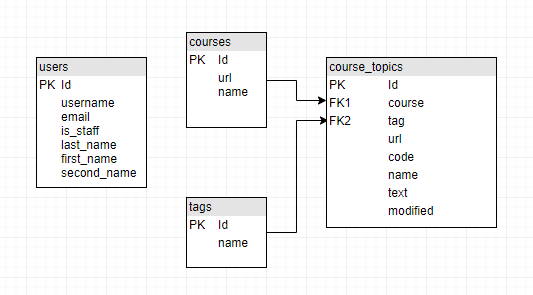


Рис. 12 Структура БД.

## 2.4 Диаграмма классов

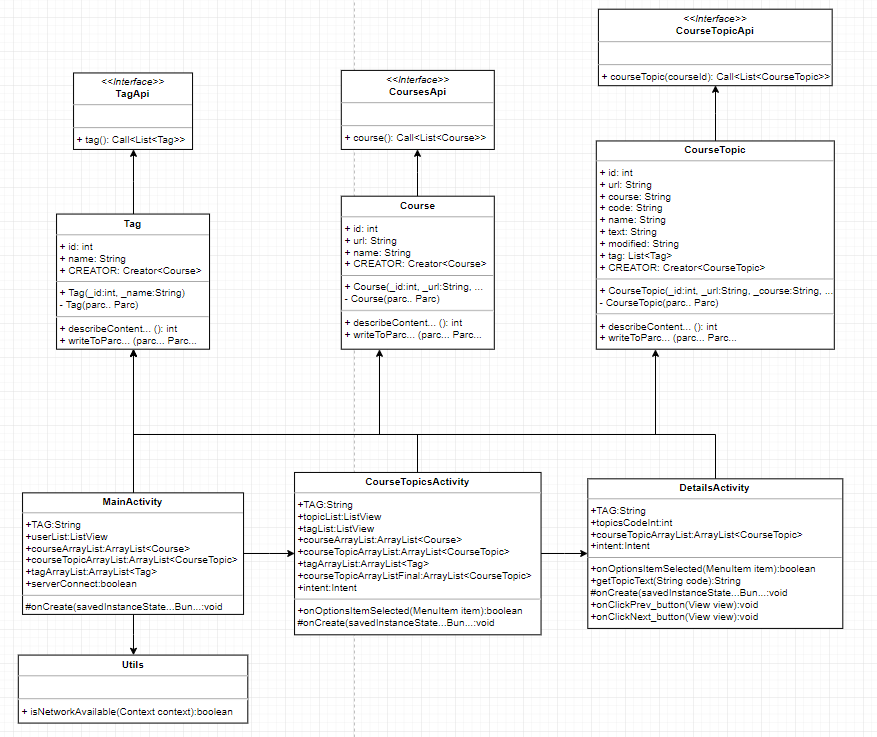


Рис. 13 Диаграмма классов

## 2.5 Модель взаимодействия сервера и клиента

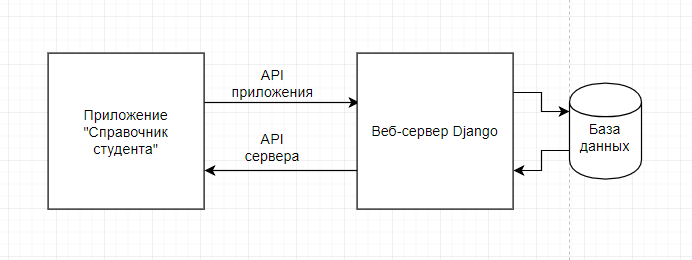


Рис. 14 Модель взаимодействия сервера и клиента.

## 2.6 Разработка приложения

Сначала был установлен IDE Android Studio, затем на него установлен и подключен Retrofit, с помощью которого будет писаться API клиента, и библиотека Markwon, с помощью которой текст лекций с разметкой markdown будет отображаться в TextView.

Первая активность. Сначала был сделан внешний вид активности, файл layout в формате xml. Внутрь него поместили список ListView и текст TextView. Код layout файла первой активности будет приведен в приложении (приложение 1).

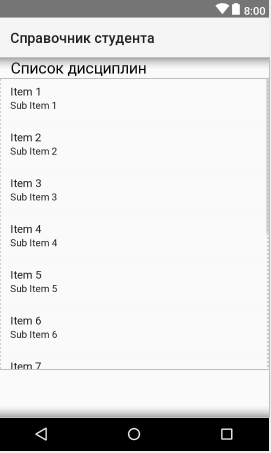


Рис. 15 Вид первой активности.

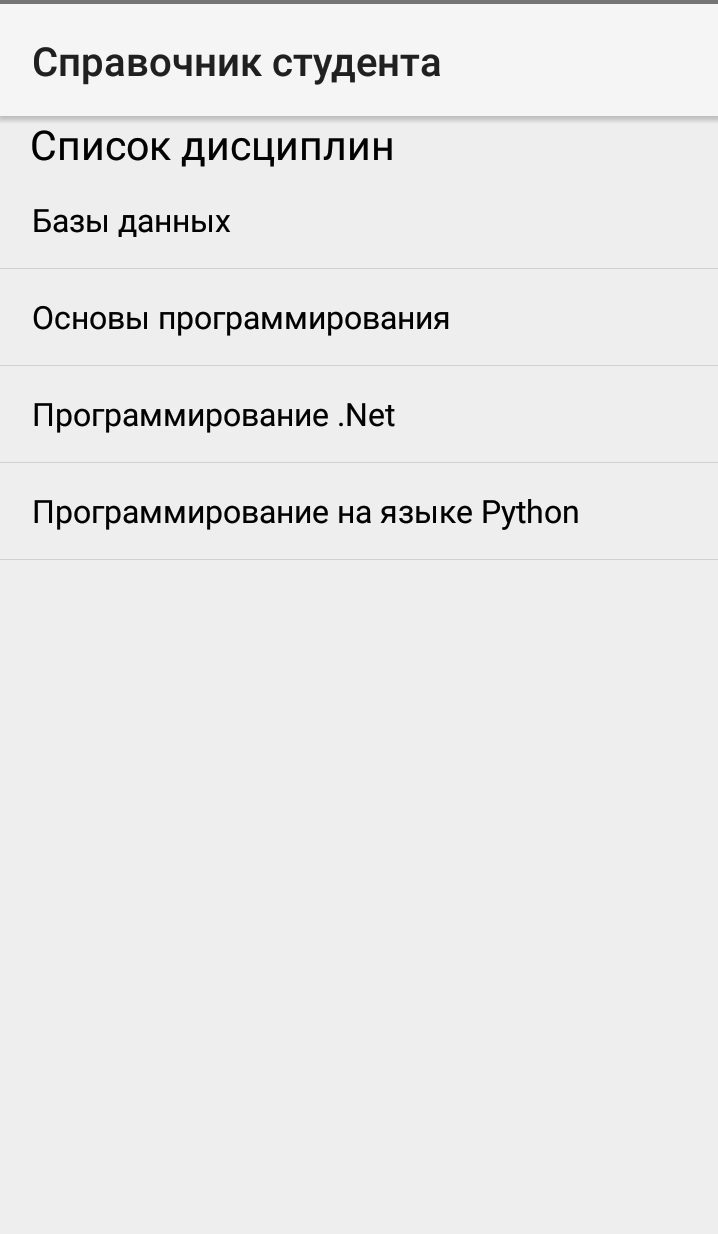


Рис. 16 Вид первой активности.

Затем был создан java файл, который будет содержать код первой активности. В области кода, который запускается при создании активности, была записана следующая логика: происходит проверка связи с сервером, при помощи метода класса Utils, если связь есть, то приложение, используя фреймворк Retrofit, отправляет запрос на прием списка дисциплин (List<Course>), списка лекций по дисциплинам (List<CourseTopic>), списка тегов (List<Tag>) и создает новый кэш. В дальнейшем работа с данными будет происходить только с использованием кэша. В ListView отображается список дисциплин, и при нажатии на элемент списка, срабатывает триггер, по которому через класс intent передается данные следующей активности. Данные включают в себя список дисциплин, список лекций, список тегов, и выбранную дисциплину. Полный код будет приведен в приложении (приложение 2).

Вторая активность была создана по тому же порядку, что и первая: сначала layout файл xml, потом java класс. Она показывает список лекций по выбранной дисциплине. Layout включает в себя список ListView и текст TextView. Код layout файла второй активности будет приведен в приложении (приложение 3).

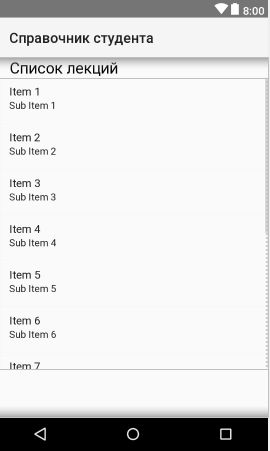


Рис. 17 Вид второй активности.

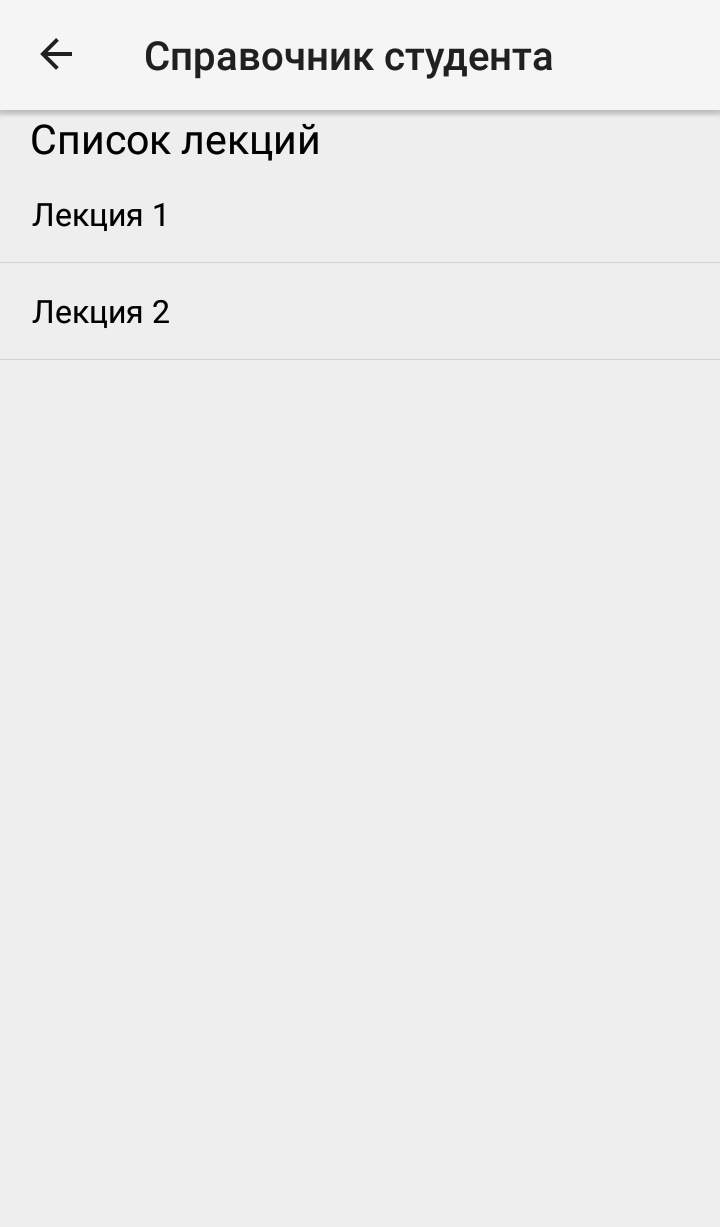


Рис. 18 Вид второй активности.

Логика этой активности: принимает данные, отправленные первой активностью через класс Intent, и размещает их на своих переменных для работы. Производится поиск лекций выбранной дисциплины в списках лекций по полю course. При нахождении лекция записывается в отдельный список, который будет выводиться в ListVew. При выборе лекции запускается третья активность, и через intent ей передаются данные о выбранной лекции и ранее отсортированный список лекций. Полный код второй активности будет приведен в приложении (приложение 4).

Третья активность предназначена для отображения текста лекции типа String и имеющей разметку markdown. Для отображения разметки markdown была использована сторонняя библиотека Markwon, которая позволяет разместить текст в формате markdown на любом элементе интерфейса, имеющей текст. Вид третьей активности состоит из TextView и двух кнопок. Код layout файла третьей активности будет приведен в приложении (приложение 5).

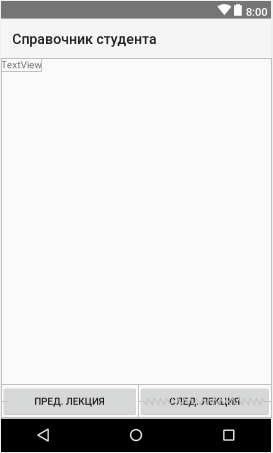


Рис. 19 Вид третьей активности.

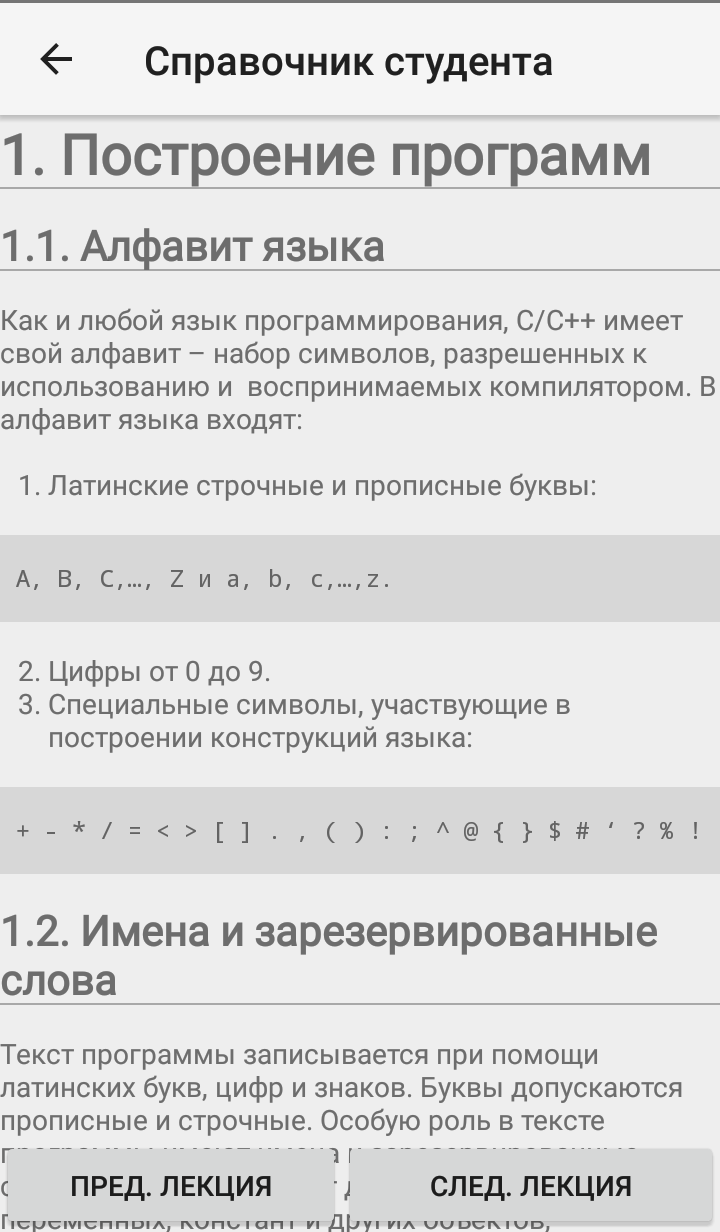


Рис. 20 Вид третьей активности.

Логика этой активности: принимает данные, отправленные второй активностью через класс Intent, и размещает их на своих переменных для работы. Производится поиск выбранной лекции в принятом списке лекций по полю code. При нахождении лекции, ее текст транслируется в TextView. При нажатии на кнопку «Пред. Лекция» проверяется code выбранной лекции, и если он не равен 1, то intent’ом передается код текущей лекции минус 1 и активность перезапускается, а если он равен1, то выводится сообщение «Это первая лекция». При нажатии на кнопку «След. Лекция» проверяется code выбранной лекции, и если он не равен размеру списка лекций, то intent’ом передается код текущей лекции плюс 1 и активность перезапускается, а если он равен размеру списка, то выводится сообщение «Это последняя лекция». Полный код второй активности будет приведен в приложении (приложение 6).

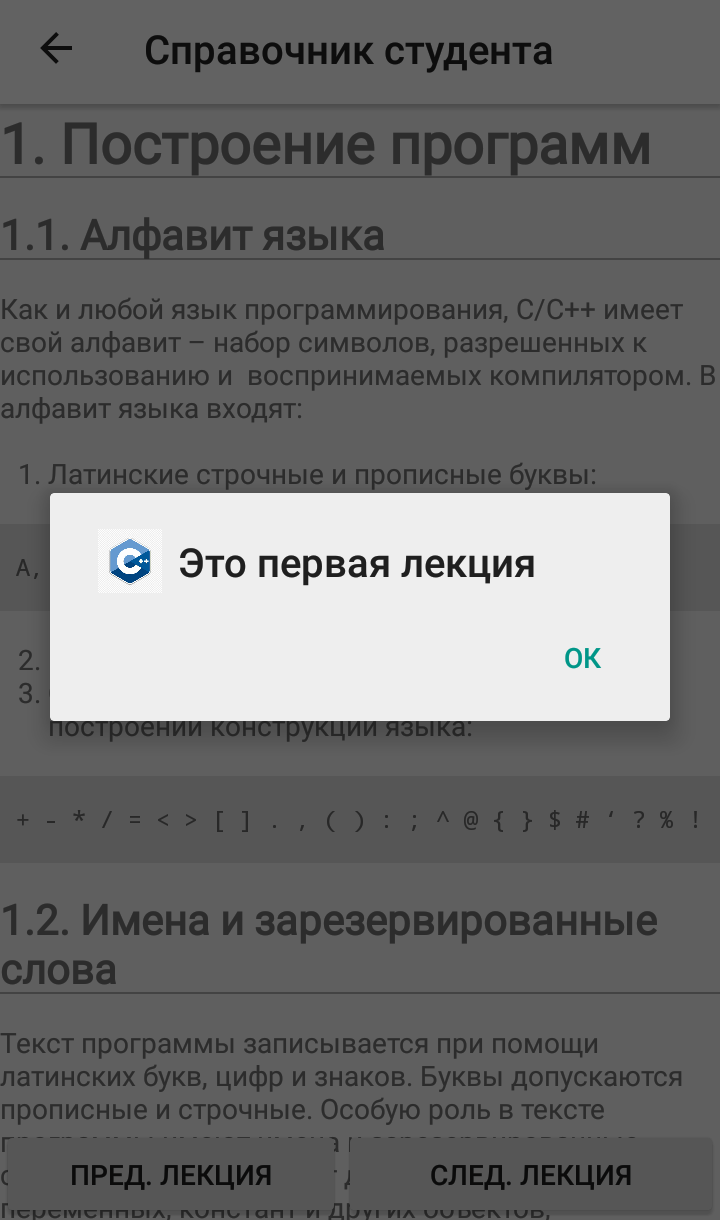


Рис. 21.

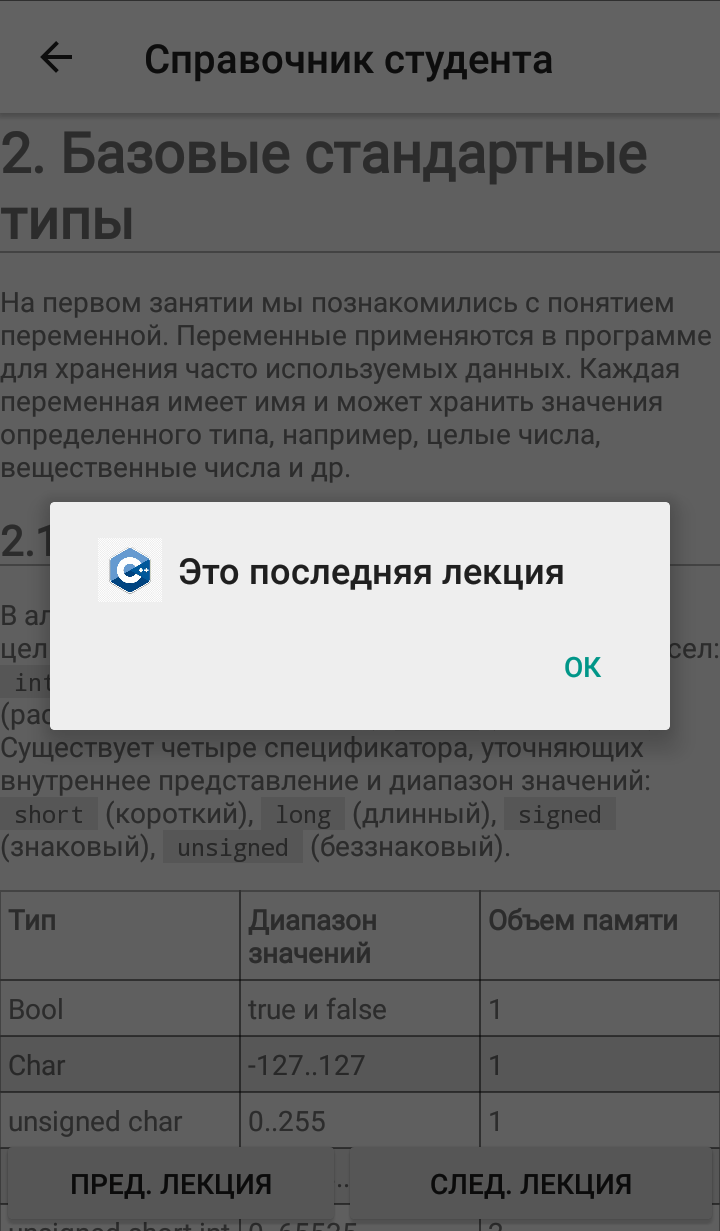


Рис. 22.

Кроме активностей были созданы: класс Utils, проверяющий соединение с Интернетом (код которого будет приведен в приложении (приложение 7)), классы, описывающие типы принимаемых данных, т.е. таблицы базы данных, - Course, CourseTopic, Tag, и интерфейсы API, для каждого класса. Классы типов были созданы с учетом того, что списки их типа надо будет передавать через intent, поэтому они наследуют класс Parcelable, который может передаваться через intent. Имеют такие же поля, какие указаны в API сервера, как отправляемые.

Интерфейсы имеют GET запрос, принимающий список класса, подходящего под таблицу БД. Их код короткий, поэтому приведу его здесь. Коды классов Course.java, CourseTopic.java, Tag.java будут представлены в приложении (приложение 8, приложение 9, приложение 10 соответственно).

Код CoursesApi:

**package** com.example.anisi.metanit;  
  
**import** java.util.List;  
  
**import** retrofit2.Call;  
**import** retrofit2.http.GET;  
  
**public interface** CoursesApi {  
 @GET(**"course/"**)  
 Call<List<Course>> course();  
}

Код CourseTopicApi:

**package** com.example.anisi.metanit;  
  
**import** java.util.List;  
  
**import** retrofit2.Call;  
**import** retrofit2.http.GET;  
**import** retrofit2.http.Query;  
  
**public interface** CourseTopicApi {  
*// @GET("course\_topic/")  
// Call<List<CourseTopic>> courseTopics();* @GET(**"course\_topic/"**)  
 Call<List<CourseTopic>> courseTopic(@Query(**"course"**) **int** courseId);  
}

Код TagApi:

**package** com.example.anisi.metanit;  
  
**import** java.util.List;  
  
**import** retrofit2.Call;  
**import** retrofit2.http.GET;  
  
**public interface** TagApi {  
 @GET(**"tag/"**)  
 Call<List<Tag>> tag();  
}

В дальнейшем планируется создание главной формы, откуда можно перейти в другие формы, создание активностей для новых функций, реализация вставки картинок. Из функциональных возможностей в планах сортировка по критериям, тексту, создание тегов, разбиение по тегам, картинки, заметки. Воплощен в коде пока только общий список и статья.

# Заключение

Первая часть данной работы посвящена обзору технологий разработки мобильных приложений, а также методик создания серверной части.

Вторая часть работы является практической и описывает процесс создания мобильного приложения и серверной части.

В ходе работы создано мобильное приложение, позволяющее студенту просматривать справочные, учебные и иные материалы по изучаемым дисциплинам. Отличительной особенностью является автоматическое кеширование передаваемых данных, что позволяет использовать приложение в режиме оффлайн.

На самом деле писать приложение оказалось делом не простым, несмотря на то, что Android Studio понятный в использовании, и он максимально пытается облегчить жизнь программисту. Язык Java, благодаря предмету ООП, прошлому опыту разработки приложений и сходству с языком С#, был использован хорошо и немного углублены знания о нем.

Процесс разработки веб-сервера был тяжел для понимания. Но в этом проекте нужен простой сервер, поэтому веб-сервер был успешно сделан. Во время разработки сервера был глубже изучен язык Python, и изучен с нуля веб-фреймворк Django. Также изучен процесс обмена сообщениями сервера и клиента, архитектура REST, и основы написания API клиентской и серверной стороны.

Таким образом, поставленные задачи были решены, выпускная квалификационная работа окончена.

# Список использованной литературы

1. Android Studio // Википедия. [2017—2017]. Дата обновления: 18.05.2017. URL: http://ru.wikipedia.org/?oldid=85473651 (дата обращения: 18.05.2017).
2. PhoneGap // Википедия. [2018—2018]. Дата обновления: 15.05.2018. URL: https://ru.wikipedia.org/?oldid=92666334 (дата обращения: 15.05.2018).
3. Xamarin // Википедия. [2017—2017]. Дата обновления: 07.09.2017. URL: https://ru.wikipedia.org/?oldid=87495961 (дата обращения: 07.09.2017).
4. React Native // Википедия. [2017—2018]. Дата обновления: 24.05.2018. URL: https://en.wikipedia.org/w/index.php?title=React\_(JavaScript\_library)&oldid=842830736 (дата обращения: 30.05.2018).
5. NativeScript // Википедия. [2018—2018]. Дата обновления: 23.01.2018. URL: https://ru.wikipedia.org/?oldid=90474846 (дата обращения: 23.01.2018).
6. LibGDX // Википедия. [2018—2018]. Дата обновления: 22.04.2018. URL: https://ru.wikipedia.org/?oldid=92235569 (дата обращения: 22.04.2018).
7. Android software development // Википедия. [2018—2018]. Дата обновления: 28.05.2018. URL: https://ru.wikipedia.org/?oldid= 843295879 (дата обращения: 30.05.2018).
8. Герберт Шилдт – Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. — М.: «Вильямс», 2015. — 1376 с. — ISBN 978-5-8459-1918-2.
9. Гриффитс Дон, Гриффитс Дэвид – Head First. Программирование для Android. — СПб.: Питер, 2016. — 704 с.: ил. — (Серия «Head First O’Reilly»).
10. Дейтел П., Дейтел Х., Уолд А. – Android для разработчиков. 3-е изд. — СПб.: Питер, 2016. — 512 с.: ил. — (Серия «Библиотека программиста»)
11. Хорстманн, Кей С. – Java. Библиотека профессионала, том 1. Основы. 10-е изд.: Пер. с англ. — М .: ООО "И.Д. Вильямс", 2016. — 864 с .: ил. — Парал. тит. англ.
12. Дронов В. – Django: Практика создания Web-сайтов на Python. – СПб.: БХВ-Петербург, 2016. – 528 с.: ил. – (Профессиональное программирование)

# Приложения

Приложение 1 – код layout первой активности, activity\_main.xml.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:weightSum="1"**>  
  
 <**TextView  
 android:id="@+id/header"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text=" Список дисциплин"  
 android:textColor="#000000"  
 android:textSize="10pt"** />  
  
 <**ListView  
 android:id="@+id/list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="0.86"  
 android:textColor="#000000"** />  
  
</**LinearLayout**>

Приложение 2 – код класса первой активности, MainActivity.java.

**package** com.example.anisi.metanit.Activity;  
  
**import** android.content.Intent;  
**import** android.graphics.Color;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.TabHost;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.example.anisi.metanit.Course;  
**import** com.example.anisi.metanit.CourseTopic;  
**import** com.example.anisi.metanit.CourseTopicApi;  
**import** com.example.anisi.metanit.CoursesApi;  
**import** com.example.anisi.metanit.R;  
**import** com.example.anisi.metanit.Tag;  
**import** com.example.anisi.metanit.TagApi;  
**import** com.example.anisi.metanit.Utils;  
  
**import** okhttp3.Cache;  
**import** okhttp3.Interceptor;  
**import** okhttp3.OkHttpClient;  
**import** okhttp3.Request;  
**import** retrofit2.Response;  
**import** retrofit2.Retrofit;  
**import** retrofit2.Call;  
**import** retrofit2.Callback;  
**import** retrofit2.converter.gson.GsonConverterFactory;  
  
**import** java.io.IOException;  
**import** java.net.Inet4Address;  
**import** java.net.InetAddress;  
**import** java.security.KeyStore;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**import** javax.crypto.NullCipher;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 String **TAG** = **"MainActivity"**;  
 ListView **userList**;  
 ArrayList<Course> **courseArrayList** = **new** ArrayList<>();  
 ArrayList<CourseTopic> **courseTopicArrayList** = **new** ArrayList<>();  
 ArrayList<Tag> **tagArrayList** = **new** ArrayList<>();  
 **boolean serverConnect**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Log.*d*(**TAG**,**"onCreate"**);  
  
 *//проверка связи с сервером, если нет, то клиенту Okhttp будет передано, чтобы он в этом случае использовал кэш* **final** Retrofit proverkaServera = **new** Retrofit.Builder()  
 .baseUrl(**"http://192.168.0.107:8000/"**)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
 CoursesApi coursesApiProverka = proverkaServera.create(CoursesApi.**class**);  
 Call<List<Course>> coursesProverka = coursesApiProverka.course();  
  
 coursesProverka.enqueue(**new** Callback<List<Course>>() {  
 @Override  
 **public void** onResponse(Call<List<Course>> call, Response<List<Course>> response) {  
 **if** (response.isSuccessful()) {  
 **serverConnect** = **true**;  
 Log.*d*(**TAG**,**"Connection to server - success"**);  
 } **else** {  
 Log.*d*(**TAG**,**"Connection to server - not success"**);  
 **serverConnect** = **false**;  
 }  
 }  
 @Override  
 **public void** onFailure(Call<List<Course>> call, Throwable t) {  
 Log.*d*(**TAG**,**"Connection to server - failure"**);  
 **serverConnect** = **false**;  
 }  
 });  
  
 *//прием данных и кэширование* OkHttpClient client = **new** OkHttpClient  
 .Builder()  
 .cache(**new** Cache(getApplicationContext().getCacheDir(), 50\*1024\*1024)) *// 10 MB* .addInterceptor(**new** Interceptor() {  
 @Override  
 **public** okhttp3.Response intercept(Chain chain) **throws** IOException {  
 Request request = chain.request();  
 **if**(Utils.*isNetworkAvailable*(getApplicationContext()) || !(InetAddress.*getByName*(**"192.168.0.107:8000"**)).isReachable(500)) {  
 request = request.newBuilder().header(**"Cache-Control"**, **"public, max-age="** + 60).build();  
 } **else** {  
 request = request.newBuilder().header(**"Cache-Control"**, **"public, only-if-cached, max-stale="** + 60 \* 60 \* 24 \* 7).build();  
 }  
 **return** chain.proceed(request);  
 }  
 })  
 .build();  
  
 **final** Retrofit retrofit = **new** Retrofit.Builder()  
 .baseUrl(**"http://192.168.0.107:8000/"**)  
 .client(client)  
 .addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
  
 *//прием данных дисциплин* CoursesApi coursesApi = retrofit.create(CoursesApi.**class**);  
 Call<List<Course>> courses = coursesApi.course();  
  
 courses.enqueue(**new** Callback<List<Course>>() {  
 @Override  
 **public void** onResponse(Call<List<Course>> call, Response<List<Course>> response) {  
 **if** (response.isSuccessful()) {  
 *//создание массива дисциплин* **courseArrayList**.addAll(response.body());  
 *//создание массива с именами дисциплин* ArrayList<String> ar = **new** ArrayList<String>();  
 **for** (Course object: **courseArrayList**){  
 ar.add(object.**name**);  
 }  
 ArrayAdapter<String> adapter;  
 adapter = **new** ArrayAdapter<String>(getApplicationContext(), android.R.layout.***simple\_list\_item\_1***, ar){  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent){  
 *// Get the Item from ListView* View view = **super**.getView(position, convertView, parent);  
  
 *// Initialize a TextView for ListView each Item* TextView tv = (TextView) view.findViewById(android.R.id.***text1***);  
  
 *// Set the text color of TextView (ListView Item)* tv.setTextColor(Color.***BLACK***);  
  
 *// Generate ListView Item using TextView* **return** view;  
 }  
 };  
 **userList**.setAdapter(adapter);  
  
 *//Прием данных лекций в цикле* **for** (**int** i = 1; i <= **courseArrayList**.size(); i++) {  
 CourseTopicApi courseTopicApi = retrofit.create(CourseTopicApi.**class**);  
 Call<List<CourseTopic>> courseTopics = courseTopicApi.courseTopic(i);  
  
 courseTopics.enqueue(**new** Callback<List<CourseTopic>>() {  
 @Override  
 **public void** onResponse(Call<List<CourseTopic>> call, Response<List<CourseTopic>> response) {  
 **if** (response.isSuccessful()) {  
 **courseTopicArrayList**.addAll(response.body());  
 Log.*d*(**TAG**, **"Topic - success"**);  
 } **else** {  
 Log.*d*(**TAG**, **"Topic - not success"**);  
 }  
 }  
  
 @Override  
 **public void** onFailure(Call<List<CourseTopic>> call, Throwable t) {  
 Log.*d*(**TAG**, **"Topics - failure"**);  
 }  
 });  
 }  
  
 Log.*d*(**TAG**,**"Courses - success"**);  
 } **else** {  
 Log.*d*(**TAG**,**"Courses - not success"**);  
 }  
 }  
 @Override  
 **public void** onFailure(Call<List<Course>> call, Throwable t) {  
 Log.*d*(**TAG**,**"Courses - failure"**);  
 }  
 });  
  
 *//прием данных тегов* TagApi tagApi = retrofit.create(TagApi.**class**);  
 Call<List<Tag>> tag = tagApi.tag();  
  
 tag.enqueue(**new** Callback<List<Tag>>() {  
 @Override  
 **public void** onResponse(Call<List<Tag>> call, Response<List<Tag>> response) {  
 **if** (response.isSuccessful()) {  
 **tagArrayList**.addAll(response.body());  
 Log.*d*(**TAG**,**"Tag - success"**);  
 } **else** {  
 Log.*d*(**TAG**,**"Tag - not success"**);  
 }  
 }  
 @Override  
 **public void** onFailure(Call<List<Tag>> call, Throwable t) {  
 Log.*d*(**TAG**,**"Tag - failure"**);  
 }  
 });  
  
 **userList** = (ListView)findViewById(R.id.***list***);  
 **userList**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 **public void** onItemClick(AdapterView<?> a, View v, **final int** position, **long** id) {  
 Intent intent = **new** Intent();  
 intent.setClass(MainActivity.**this**, CourseTopicsActivity.**class**);  
 String courseUrl = **courseArrayList**.get(position).**url**;  
 String chosenTag = **"НетТега"**;  
 intent.putExtra(**"ChosenCourseUrl"**, courseUrl);  
 intent.putParcelableArrayListExtra(Course.**class**.getCanonicalName(), **courseArrayList**);  
 intent.putParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName(), **courseTopicArrayList**);  
 intent.putParcelableArrayListExtra(Tag.**class**.getCanonicalName(), **tagArrayList**);  
 intent.putExtra(**"ChosenTag"**, chosenTag);  
 *//запускаем активность лекций* startActivity(intent);  
 }  
 });  
 }  
*}*

Приложение 3 – код layout второй активности, activity\_course\_topics.xml.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:weightSum="1"**>  
  
 <**TextView  
 android:id="@+id/header"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text=" Список лекций"  
 android:textColor="#000000"  
 android:textSize="10pt"** />  
  
 <**ListView  
 android:id="@+id/listTopic"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="0.86"  
 android:textColor="#000000"**/>  
</**LinearLayout**>

Приложение 4 – код класса второй активности, CourseTopicsActivity.java.

**package** com.example.anisi.metanit.Activity;  
  
**import** android.content.Intent;  
**import** android.graphics.Color;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
  
**import** com.example.anisi.metanit.Course;  
**import** com.example.anisi.metanit.CourseTopic;  
**import** com.example.anisi.metanit.R;  
**import** com.example.anisi.metanit.Tag;  
  
**import** java.util.ArrayList;  
  
**import static** com.example.anisi.metanit.R.id.***listTopic***;  
  
**public class** CourseTopicsActivity **extends** AppCompatActivity {  
  
 String **TAG** = **"CourseTopicActivity"**;  
 ListView **topicList**;  
 ListView **tagList**;  
 ArrayList<Course> **courseArrayList** = **new** ArrayList<>();  
 ArrayList<CourseTopic> **courseTopicArrayList** = **new** ArrayList<>();  
 ArrayList<Tag> **tagArrayList** = **new** ArrayList<>();  
 ArrayList<CourseTopic> **courseTopicArrayList1** = **new** ArrayList<>();  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** android.R.id.***home***:  
 CourseTopicsActivity.**this**.onBackPressed();  
 **return true**;  
 **default**:  
 **return super**.onOptionsItemSelected(item);  
 }  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_course\_topics***);  
 getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);  
 **topicList** = (ListView)findViewById(***listTopic***);  
 Log.*d*(**TAG**,**"onCreate: create"**);  
  
 *//получение id выбранного курса* **final** Intent intent = getIntent();  
 String courseUrl = intent.getStringExtra(**"ChosenCourseUrl"**);  
 **courseArrayList** = getIntent().getParcelableArrayListExtra(Course.**class**.getCanonicalName());  
 **courseTopicArrayList** = getIntent().getParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName());  
 **tagArrayList** = getIntent().getParcelableArrayListExtra(Tag.**class**.getCanonicalName());  
  
 *//фильтр топиков по курсам* **for** (CourseTopic object: **courseTopicArrayList**) {  
 **if** (object.**course**.equals(courseUrl)){  
 **courseTopicArrayList1**.add(object);  
 }  
 }  
  
 ArrayList<String> courseTopicNames = **new** ArrayList<>();  
 **for** (CourseTopic object : **courseTopicArrayList1**) {  
 courseTopicNames.add(object.**name**);  
 }  
 ArrayAdapter<String> adapter;  
 adapter = **new** ArrayAdapter<String>(getApplicationContext(), android.R.layout.***simple\_list\_item\_1***, courseTopicNames){  
 @Override  
 **public** View getView(**int** position, View convertView, ViewGroup parent){  
 *// Get the Item from ListView* View view = **super**.getView(position, convertView, parent);  
  
 *// Initialize a TextView for ListView each Item* TextView tv = (TextView) view.findViewById(android.R.id.***text1***);  
  
 *// Set the text color of TextView (ListView Item)* tv.setTextColor(Color.***BLACK***);  
  
 *// Generate ListView Item using TextView* **return** view;  
 }  
 };  
 **topicList**.setAdapter(adapter);  
  
 **topicList**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **int** position, **long** l) {  
 intent.setClass(CourseTopicsActivity.**this**, DetailsActivity.**class**);  
 **int** chosenTopicsCode = position + 1;  
 intent.putExtra(**"ChosenTopicsCode"**, chosenTopicsCode);  
 intent.putParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName(), **courseTopicArrayList1**);  
 *//запускаем активность лекции* startActivity(intent);  
 }  
 });  
 }}

Приложение 5 – код layout третьей активности, activity\_details.xml.

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** >  
  
 <**TextView  
 android:id="@+id/textView6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 android:text="TextView"  
 android:layout\_alignParentLeft="true"** />  
  
 <**Button  
 android:id="@+id/prev\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="10dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentLeft="true"  
 android:layout\_toLeftOf="@+id/next\_button"  
 android:onClick="onClickPrev\_button"  
 android:text="Пред. лекция"** />  
  
 <**Button  
 android:id="@+id/next\_button"  
 android:layout\_width="189dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_x="260dp"  
 android:layout\_y="469dp"  
 android:onClick="onClickNext\_button"  
 android:text="След. лекция"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentEnd="true"** />  
  
</**RelativeLayout**>

Приложение 6 – код класса третьей активности, DetailsActivity.java.

**package** com.example.anisi.metanit.Activity;  
  
**import** android.app.AlertDialog;  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.util.Log;  
**import** android.widget.TextView;  
  
**import** com.example.anisi.metanit.CourseTopic;  
**import** com.example.anisi.metanit.R;  
  
**import** java.util.ArrayList;  
  
**import** ru.noties.markwon.Markwon;  
  
**public class** DetailsActivity **extends** AppCompatActivity {  
  
 String **TAG** = **"DetailsActivity"**;  
 **int topicsCodeInt**;  
 ArrayList<CourseTopic> **courseTopicArrayList** = **new** ArrayList<>();  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** android.R.id.***home***:  
 DetailsActivity.**this**.onBackPressed();  
 **return true**;  
 **default**:  
 **return super**.onOptionsItemSelected(item);  
 }  
 }  
  
 **public** String getTopicText(String code){  
 String topicText = **"Ошибка загрузки текста"**;  
 **courseTopicArrayList** = getIntent().getParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName());  
 **for** (**int** i = 0; i < **courseTopicArrayList**.size(); i++) {  
 **if** (**courseTopicArrayList**.get(i).**code**.equals(code)) {  
 topicText = **courseTopicArrayList**.get(i).**text**;  
 }  
 }  
 **return** topicText;  
 }  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_details***);  
 getSupportActionBar().setDisplayHomeAsUpEnabled(**true**);  
 TextView textView = (TextView)findViewById(R.id.***textView6***);  
 Log.*d*(**"DetailsActivity"**, **"onCreate: create"**);  
 Intent intent = getIntent();  
 **topicsCodeInt** = intent.getIntExtra(**"ChosenTopicsCode"**,0);  
 String topicsCodeString = Integer.*toString*(**topicsCodeInt**);  
 Log.*d*(**TAG**,**"ChosenTopicCode = "** + topicsCodeString);  
 String MDtext = getTopicText(topicsCodeString);  
 Markwon.*setMarkdown*(textView, MDtext);  
 }**public void** onClickPrev\_button(View view) {  
 **if**(**topicsCodeInt** == 1){  
 AlertDialog.Builder builder = **new** AlertDialog.Builder(DetailsActivity.**this**);  
 builder.setTitle(**"Это первая лекция"**)  
 .setIcon(R.drawable.***n4***)  
 .setCancelable(**false**)  
 .setNegativeButton(**"ОК"**,  
 **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 AlertDialog alert = builder.create();  
 alert.show();  
 *//this.finish();* }  
 **else** {  
 Intent intent = **new** Intent();  
 intent.setClass(DetailsActivity.**this**, DetailsActivity.**class**);  
 intent.putExtra(**"ChosenTopicsCode"**, **topicsCodeInt** - 1);  
 intent.putParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName(), **courseTopicArrayList**);  
 startActivity(intent);  
 **this**.finish();  
 }  
 }  
  
 **public void** onClickNext\_button(View view) {  
 Intent intent = **new** Intent();  
 **if**(**topicsCodeInt** == **courseTopicArrayList**.size()){  
 AlertDialog.Builder builder = **new** AlertDialog.Builder(DetailsActivity.**this**);  
 builder.setTitle(**"Это последняя лекция"**)  
 .setIcon(R.drawable.***n4***)  
 .setCancelable(**false**)  
 .setNegativeButton(**"ОК"**,  
 **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 AlertDialog alert = builder.create();  
 alert.show();  
 *//this.finish();* }  
 **else** {  
 intent.setClass(DetailsActivity.**this**, DetailsActivity.**class**);  
 intent.putExtra(**"ChosenTopicsCode"**, **topicsCodeInt** + 1);  
 intent.putParcelableArrayListExtra(CourseTopic.**class**.getCanonicalName(), **courseTopicArrayList**);  
 startActivity(intent);  
 **this**.finish();  
 }  
 }  
}

Приложение 7 – код класса Utils, Utils.java.

**package** com.example.anisi.metanit;  
  
**import** android.content.Context;  
**import** android.net.ConnectivityManager;  
**import** android.net.NetworkInfo;  
  
**public class** Utils {  
 **public static boolean** isNetworkAvailable(Context context){  
 ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.***CONNECTIVITY\_SERVICE***);  
 NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();  
 **if** ((activeNetworkInfo != **null**)&&(activeNetworkInfo.isConnected())){  
 **return true**;  
 }**else**{  
 **return false**;  
 }  
 }  
}

Приложение 8 – код класса Course, Course.java.

**package** com.example.anisi.metanit;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
**import** android.util.Log;  
  
  
**public class** Course **implements** Parcelable {  
 **public int id**;  
 **public** String **url**;  
 **public** String **name**;  
  
 **public** Course(**int** \_id, String \_url, String \_name) {  
 **id** = \_id;  
 **url** = \_url;  
 **name** = \_name;  
 }  
  
 **public int** describeContents() {  
 **return** 0;  
 }  
  
 **public void** writeToParcel(Parcel parcel, **int** flags) {  
 parcel.writeInt(**id**);  
 parcel.writeString(**url**);  
 parcel.writeString(**name**);  
 }  
  
 **public static final** Parcelable.Creator<Course> ***CREATOR*** = **new** Parcelable.Creator<Course>() {  
 *// распаковываем объект из Parcel* **public** Course createFromParcel(Parcel in) {  
 **return new** Course(in);  
 }  
  
 **public** Course[] newArray(**int** size) {  
 **return new** Course[size];  
 }  
 };  
  
 **private** Course(Parcel parcel) {  
 **id** = parcel.readInt();  
 **url** = parcel.readString();  
 **name** = parcel.readString();  
 }  
}

Приложение 9 – код класса CourseTopic, CourseTopic.java.

**package** com.example.anisi.metanit;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** CourseTopic **implements** Parcelable{  
 **public int id**;  
 **public** String **url**;  
 **public** String **course**;  
 **public** String **code**;  
 **public** String **name**;  
 **public** String **text**;  
 **public** String **modified**;  
 **public** List<String> **tag**;  
  
 **public** CourseTopic(**int** \_id, String \_url, String \_course, String \_code, String \_name, String \_text, String \_modified, List<String> \_tag) {  
 **id** = \_id;  
 **url** = \_url;  
 **course** = \_course;  
 **code** = \_code;  
 **name** = \_name;  
 **text** = \_text;  
 **modified** = \_modified;  
 **tag** = \_tag;  
 }  
  
 **public int** describeContents() {  
 **return** 0;  
 }  
  
 **public void** writeToParcel(Parcel parcel, **int** flags) {  
 parcel.writeInt(**id**);  
 parcel.writeString(**url**);  
 parcel.writeString(**course**);  
 parcel.writeString(**code**);  
 parcel.writeString(**name**);  
 parcel.writeString(**text**);  
 parcel.writeString(**modified**);  
 parcel.writeStringList(**tag**);  
 }  
  
 **public static final** Parcelable.Creator<CourseTopic> ***CREATOR*** = **new** Parcelable.Creator<CourseTopic>() {  
 *// распаковываем объект из Parcel* **public** CourseTopic createFromParcel(Parcel in) {  
 **return new** CourseTopic(in);  
 }  
  
 **public** CourseTopic[] newArray(**int** size) {  
 **return new** CourseTopic[size];  
 }  
 };  
  
 **private** CourseTopic(Parcel parcel) {  
 **id** = parcel.readInt();  
 **url** = parcel.readString();  
 **course** = parcel.readString();  
 **code** = parcel.readString();  
 **name** = parcel.readString();  
 **text** = parcel.readString();  
 **modified** = parcel.readString();  
 **tag** = parcel.createStringArrayList();  
 }  
}

Приложение 10 – код класса Tag, Tag.java.

**package** com.example.anisi.metanit;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
  
**import** java.util.List;  
  
**public class** Tag **implements** Parcelable {  
 **public int id**;  
 **public** String **name**;  
  
 **public** Tag(**int** \_id, String \_url, String \_name) {  
 **id** = \_id;  
 **name** = \_name;  
 }  
  
 **public int** describeContents() {  
 **return** 0;  
 }  
  
 **public void** writeToParcel(Parcel parcel, **int** flags) {  
 parcel.writeInt(**id**);  
 parcel.writeString(**name**);  
 }  
  
 **public static final** Parcelable.Creator<Tag> ***CREATOR*** = **new** Parcelable.Creator<Tag>() {  
 *// распаковываем объект из Parcel* **public** Tag createFromParcel(Parcel in) {  
 **return new** Tag(in);  
 }  
  
 **public** Tag[] newArray(**int** size) {  
 **return new** Tag[size];  
 }  
 };  
  
 **private** Tag(Parcel parcel) {  
 **id** = parcel.readInt();  
 **name** = parcel.readString();  
 }  
}

Приложение 11 – код models.py.

from django.db import models  
  
from django.contrib.auth.models import AbstractUser  
  
  
class User(AbstractUser):  
 second\_name = models.CharField('отчество', max\_length=30, blank=True, default='')  
  
 class Meta:  
 verbose\_name = 'пользователь'  
 verbose\_name\_plural = 'пользователи'  
 ordering = ['last\_name', 'first\_name', 'second\_name']  
  
 def \_\_str\_\_(self):  
 parts = [self.last\_name or self.username]  
 if self.first\_name:  
 parts.append(self.first\_name[0] + '.')  
 if self.second\_name:  
 parts.append(self.second\_name[0] + '.')  
 return ' '.join(parts)  
  
  
class Course(models.Model):  
 name = models.CharField('наименование', max\_length=100)  
 # course\_type = models.IntegerField('тип курса', choices=COURSE\_TYPE)  
 # semester = models.ForeignKey('Semester', on\_delete=models.IGNORE, verbose\_name='семестр')  
 # lecture\_hours = models.IntegerField('часы лекций', blank=True, default=0)  
 # lab\_work\_hours = models.IntegerField('часы лаб. работ', blank=True, default=0)  
 # practice\_hours = models.IntegerField('часы практ. работ', blank=True, default=0)  
 # student\_work\_hours = models.IntegerField('часы СРС', blank=True, default=0)  
 # control\_hours = models.IntegerField('часы КСР', blank=True, default=0)  
 # total\_hours = models.IntegerField('часов всего', blank=True, default=0)  
  
 class Meta:  
 verbose\_name = 'дисциплина'  
 verbose\_name\_plural = 'дисциплины'  
 ordering = ['name']  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 # def get\_total\_hours(self):  
 # return (self.lecture\_hours + self.lab\_work\_hours + self.practice\_hours +  
 # self.student\_work\_hours + self.control\_hours)  
  
  
class CourseElement(models.Model):  
 kind = 'element'  
  
 course = models.ForeignKey('Course', on\_delete=models.CASCADE, verbose\_name='курс обучения')  
 code = models.CharField('код', max\_length=10, default='')  
 name = models.CharField('название', max\_length=150, default='')  
 modified = models.DateTimeField('дата и время изменения', auto\_now=True)  
  
 class Meta:  
 abstract = True  
 ordering = ['course', 'code']  
  
  
class Tag(models.Model):  
 name = models.CharField('название', max\_length=150, default='')  
  
 class Meta:  
 verbose\_name = 'тег'  
 verbose\_name\_plural = 'теги'  
 ordering = ['name']  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class CourseTopic(CourseElement):  
 kind = 'topic'  
  
 text = models.TextField('текст в формате Markdown', blank=True, default='')  
 tag = models.ManyToManyField(Tag)  
  
 class Meta:  
 verbose\_name = 'дисциплина: теоретический материал'  
 verbose\_name\_plural = 'дисциплины: теоретические материалы'  
  
 def \_\_str\_\_(self):  
 return '%s: %s %s' % (self.course.name, self.code, self.name)

Приложение 12 – код admin.py.

from django.contrib import admin  
from django.contrib.auth.admin import UserAdmin  
  
from . import models  
  
  
@admin.register(models.User)  
class UserAdmin(UserAdmin):  
 actions\_on\_bottom = True  
 actions\_on\_top = False  
 # fields = [  
 # 'username', 'password', 'last\_name', 'first\_name', 'second\_name', 'email',  
 # 'is\_superuser', 'is\_staff', 'is\_active',  
 # ]  
 list\_display = [  
 'username', 'last\_name', 'first\_name', 'second\_name',  
 'is\_superuser', 'is\_staff', 'is\_active'  
 ]  
 list\_display\_links = [  
 'username', 'last\_name', 'first\_name', 'second\_name'  
 ]  
 search\_fields = [  
 'username', 'last\_name', 'first\_name', 'second\_name'  
 ]  
  
 def \_\_init\_\_(self, \*argv, \*\*kwargs):  
 self.fieldsets[1][1]['fields'] = (  
 'last\_name', 'first\_name', 'second\_name', 'email'  
 )  
 super().\_\_init\_\_(\*argv, \*\*kwargs)  
  
  
@admin.register(models.Course)  
class CourseAdmin(admin.ModelAdmin):  
 actions\_on\_bottom = True  
 actions\_on\_top = False  
 list\_display = ['name']  
 search\_fields = ['name']  
  
  
@admin.register(models.CourseTopic)  
class CourseTopicAdmin(admin.ModelAdmin):  
 actions\_on\_bottom = True  
 actions\_on\_top = False  
 list\_display = ['course', 'code', 'name']  
 list\_filter = ['course', 'tag']  
  
  
@admin.register(models.Tag)  
class TagTopicAdmin(admin.ModelAdmin):  
 actions\_on\_bottom = True  
 actions\_on\_top = False  
 list\_display = ['name']  
  
  
admin.site.site\_header = 'Администрирование'  
admin.site.site\_title = 'Администрирование'  
admin.site.index\_title = 'Таблицы'

Приложение 13 – код api.py.

from django\_filters import rest\_framework  
from rest\_framework import routers, serializers, viewsets  
  
from backend import models  
  
  
class UserSerializer(serializers.HyperlinkedModelSerializer):  
 class Meta:  
 model = models.User  
 fields = ['id', 'url', 'username', 'email', 'is\_staff', 'last\_name', 'first\_name', 'second\_name']  
  
  
class CourseSerializer(serializers.HyperlinkedModelSerializer):  
 class Meta:  
 model = models.Course  
 fields = ['id', 'url', 'name']  
  
  
class CourseTopicSerializer(serializers.HyperlinkedModelSerializer):  
 class Meta:  
 model = models.CourseTopic  
 fields = ['id', 'url', 'course', 'code', 'name', 'text', 'modified', 'tag']  
  
  
class TagSerializer(serializers.HyperlinkedModelSerializer):  
 class Meta:  
 model = models.Tag  
 fields = ['id', 'name']  
  
  
class UserViewSet(viewsets.ModelViewSet):  
 queryset = models.User.objects.all()  
 serializer\_class = UserSerializer  
  
  
class CourseViewSet(viewsets.ModelViewSet):  
 queryset = models.Course.objects.all()  
 serializer\_class = CourseSerializer  
  
  
class CourseTopicViewSet(viewsets.ModelViewSet):  
 queryset = models.CourseTopic.objects.all()  
 serializer\_class = CourseTopicSerializer  
 filter\_backends = (rest\_framework.DjangoFilterBackend,)  
 filter\_fields = ('course', 'code', 'name', 'tag')  
  
  
class TagViewSet(viewsets.ModelViewSet):  
 queryset = models.Tag.objects.all()  
 serializer\_class = TagSerializer  
  
  
router = routers.DefaultRouter()  
router.register(r'users', UserViewSet)  
router.register(r'course', CourseViewSet)  
router.register(r'course\_topic', CourseTopicViewSet)  
router.register(r'tag', TagViewSet)