**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Механико-математический факультет

Кафедра Веб-технологий и компьютерного моделирования

**ВОЙТЕХ СВЕТАЛАНА ГЕННАДЬЕВНА**

**ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ RESTFUL ВЕБ-СЕРВИСОВ И АНДРОИД-КЛИЕНТА ДЛЯ СИСТЕМЫ УПРАВЛЕНИЯ УЧЕБНЫМ РАСПИСАНИЕМ**

Дипломная работа

студентки V курса специальности Математик

(информационные технологии)

Руководитель

СУЗДАЛЬ Станислав Валерьевич

доцент кафедры ВТиКМ

Рецензент

КРЫЛОВ Евгений Вячеславович

старший преподаватель кафедры ВТиКМ

«Допустить к защите»

**Зав. Кафедрой** ВТ и КМ,

канд. физ.-мат. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_ Романчик В.С.

«\_\_\_» \_\_\_\_\_\_\_\_ 2013 г.

Минск 2013

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc326322909)

[ВВЕДЕНИЕ 3](#_Toc326322910)

[ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc326322911)

[1.1 Сравнение с аналогами. Обзор существующих разработок. 5](#_Toc326322912)

[1.2 Краткий обзор существующих технологий разработки приложения. 8](#_Toc326322913)

[1.3 Проектирование приложения. Модель. 15](#_Toc326322914)

[1.4 Детализация проекта. 17](#_Toc326322915)

[ГЛАВА 2 ПРАКТИЧЕСКАЯ ЧАСТЬ 1](#_Toc326322916)9

[2.1 Описание разработки практической части 16](#_Toc326322917)

[2.2 Методология тестирования. Результаты 35](#_2.2_Методология_тестирования.)

[ГЛАВА 3 ОРГАНИЗАЦИОННАЯ ЧАСТЬ 20](#_Toc326322919)

[3.1 Перспективы развития 36](#_3.1_Перспективы_развития)

[3.2 Выводы 36](#_Toc326322921)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc326322922)

[ПРИЛОЖЕНИЕ 37](#_Toc326322922)

[АННОТАЦИЯ 37](#_Toc326322922)

[Abstract 36](#_Toc326322921)

[Keywords 36](#_Toc326322921)

# ВВЕДЕНИЕ

Человек за свою жизнь неоднократно сталкивается с расписанием разного рода событий. Например, такие события как, транспортные, учебные, производственные и другие, которые организованы периодичным образом. Зная расписание, человек может с умом тратить свое время, распоряжаться им рационально. Но так как все в нашей жизни изменчиво, то и расписание не является исключением. Наверное, не стоит объяснять, что, если вовремя не оповестить людей об изменениях в расписании, то это может привести к неблагоприятным последствиям. Поэтому очень важно, чтобы люди, непосредственно имеющие отношение к расписанию, всегда знали о последних изменениях.

Данная дипломная работа посвящена учебному расписанию в высших учебных заведениях, в частности БГУ, механико-математическому факультету. Неотъемлемой частью любого учебного года является расписание занятий. Каждый раз в начале нового учебного семестра студент задает себе вопрос: «А какое у меня расписание? И какие пары сегодня?». Учитывая, что процесс обучения в университете является довольно динамичным и изменчивым, хотелось бы дать студентам возможность узнавать об изменениях в расписании быстро и легко.

В наше время, время развития интернет технологий, наверное, уже все ВУЗы имеют собственные веб-сайты. И на многих из них размещается расписание учебных занятий. Но, к сожалению, существует ряд недостатков:

1. Сайт факультета может не предоставлять возможность просмотра расписания занятий.
2. На сайте может быть расположена устаревшая информация, неактуальная на текущий день.
3. Не всегда есть доступ к компьютеру (например, в дороге).
4. Не всегда есть доступ к интернету.
5. Открытие сайта, не имеющего мобильного аналога, с телефона или планшета влечет за собой ряд неудобств в процессе пользования.

Отметим, что в наше время сложно уже встретить студента, у которого не было бы смартфона с возможностью выхода в интернет, работающего под управлением операционной системы [Apple iOS](http://ru.wikipedia.org/wiki/Apple_iOS" \o "Apple iOS), [Windows Phone](http://ru.wikipedia.org/wiki/Windows_Phone" \o "Windows Phone) или [Android](http://ru.wikipedia.org/wiki/Android" \o "Android). Учитывая этот факт, целью данной дипломной работы является разработка приложения для андроид для просмотра учебного расписания механико-математического факультета БГУ. А также разработка RESTful веб-сервисов, которыми будет пользоваться приложение для получения данных.

Основная задача – проанализировать существующие аналоги данного приложения, собрать их достоинства, избавиться от недостатков, добавить дополнительный функционал. Приложение предоставляет следующие возможности:

1. Просмотр расписания для студента.
2. Просмотр расписания для преподавателя.
3. Возможность оставлять пометки в учебном расписании.
4. Поддержка офлайн режима для просмотра расписания.

# ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

### 1.1. Сравнение с аналогами. Обзор существующих разработок

**1.1.1. Приложение «Студхелпер»**

Приложение «Студхелпер» – один из существующих аналогов. «Студхелпер» позволяет быть в курсе последних событий: иметь под рукой актуальное расписание, список выполненных тобой работ, всегда знать, как зовут преподавателей и моментально получать информацию от старосты группы.

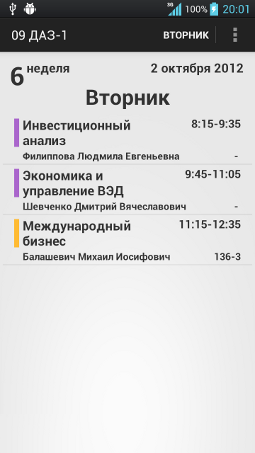
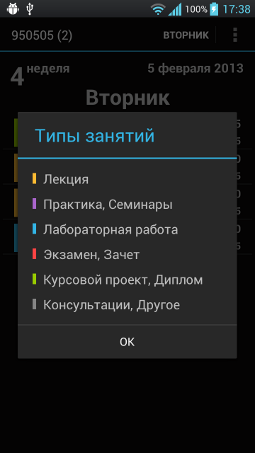
На данный момент «Студхелпер» поддерживает только 2 ВУЗа (БГУИР и БГЭУ). Основным недостатком является отсутствие БГУ среди этих ВУЗов.

Рисунок 1. Аналог. Приложение «Студхелпер».

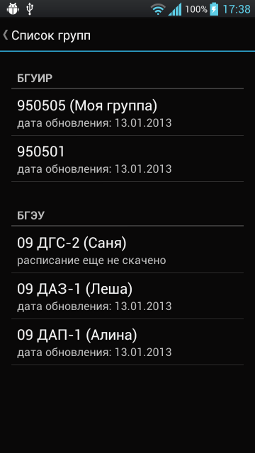
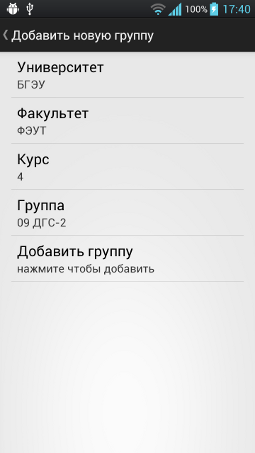


Рисунок 2. Аналог. Приложение «Студхелпер».

**1.1.2. Приложение «Расписание для ВУЗов»**

Приложение для студентов, преподавателей и университетов.  
Существуют версии предложения, как для Android, так и для iOS.

***Возможности:***

* автоматически обновляемое расписание;
* избранное, в которое можно добавлять расписание группы, преподавателя и друзей из других групп и вузов;
* планировщик заданий с возможностью прикрепления фотографий конспектов;
* проверка адреса аудитории;
* оповещение студентов через приложение (для университетов).

***Скоро появятся:***

* веб-сервис для университетов;
* виджет приложения для сайтов университетов;
* возможность синхронизировать задания с одногруппниками;
* и другие удобные способы обмена информацией среди студентов и преподавателей.

Основным недостатком является то, что это приложение поддерживает только ВУЗы России.

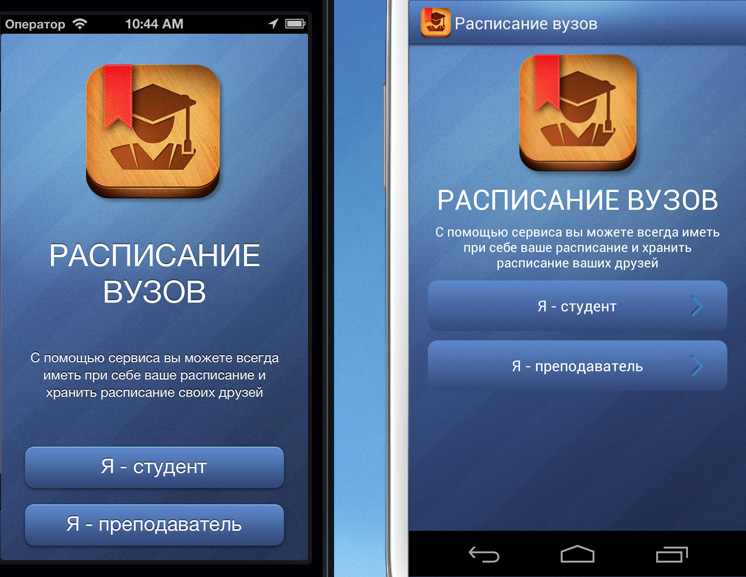


Рисунок 3. Аналог.Приложение «Расписание для ВУЗов».

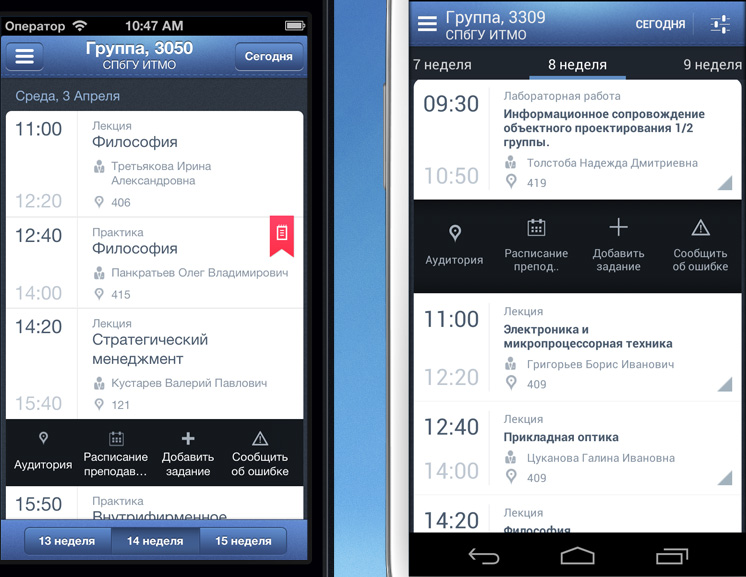


Рисунок 4. Аналог. Приложение **«**Расписание для ВУЗов».

### 1.2. Краткий обзор существующих технологий разработки приложения

Система разрабатывается на платформе Java EE. Java EE представляет собой набор спецификаций и соответствующей документации для языка Java, описывающий архитектуру серверной платформы для задач средних и крупных предприятий.

Java EE является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость и гибкость.

Для Java и Java EE существует большое количество свободных (open source) фреймворков, серверов приложений и отдельных Servlet/JSP контейнеров.

Для данного проекта решено использовать контейнер сервлетов Apache Tomcat, библиотеки Hibernate, Spring фреймворк. Все перечисленные средства являются свободными и доступны для загрузки с сайтов разработчиков.

Контейнер Apache Tomcat написан на Java и реализует спецификацию сервлетов и JSP, которые являются стандартами для разработки web-приложений на языке Java. Tomcat используется в качестве самостоятельного web-сервера, достаточно легко устанавливается и конфигурируется.

Для хранения данных выбрана реляционная СУБД MySQL. Данная СУБД так же является свободной, является решением для малых и средних приложений.

Для связи Java классов с таблицами БД выбран Hibernate. Hibernate освобождает разработчика от значительного объема сравнительно низкоуровневого программирования по обеспечению хранения объектов в реляционной БД. Благодаря Hibernate существенно сокращается время разработки части приложения, отвечающего за сохранение и извлечение данных из БД, а также облегчается сопровождение кода.

Spring Framework обеспечивает решения многих задач, с которыми сталкиваются Java разработчики и организации, которые хотят создать информационную систему, основанную на платформе [Java](http://ru.wikipedia.org/wiki/Java" \o "Java). Spring Framework  может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако, они обеспечивают большую функциональность при совместном их использовании. Решено использовать следующие его элементы:

* [***Inversion of Control***](http://ru.wikipedia.org/wiki/Inversion_of_Control)***контейнер*:** конфигурирование компонент приложений и управление жизненным циклом Java объектов.
* ***Фреймворк управления***[***транзакциями***](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))***:*** координация различных [API](http://ru.wikipedia.org/wiki/API) управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java.
* ***Фреймворк***[***аутентификации***](http://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F)***и***[***авторизации***](http://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)***:*** конфигурируемый инструментарий процессов аутентификации и авторизации, поддерживающий много популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект [Spring Security](http://ru.wikipedia.org/wiki/Spring_Security" \o "Spring Security).

#### 1.2.1. REST-сервис, JAX-RS

REST – это набор архитектурных принципов и стиль проектирования приложений, ориентированный на создание сетевых систем, в основе которых лежат механизмы для описания и обращения к ресурсам. Примером такой системы может служить World Wide Web.

В REST определяется строгое разделение ответственности между компонентами клиент-серверной системы, облегчающее реализацию необходимых актеров (actors). Другой целью REST является упрощение семантики взаимодействия компонентов сетевых систем, что позволяет улучшить масштабируемость и повысить производительность. В основу REST заложен принцип автономности запросов, означающий, что запросы, обрабатываемые клиентом или сервером, должны включать всю контекстную информацию, необходимую для их понимания. При работе REST-систем для обмена данными стандартных медиа-типов используется минимальное количество запросов.

REST-системы используют URI (универсальные идентификаторы ресурсов) для поиска и получения доступа к представлениям необходимых ресурсов.

В течение последних нескольких лет разработчики создавали REST-сервисы для своих Java-приложений, используя самые разнообразные технологии. Архитектура REST отличается своей простотой, требуя от приложений обеспечить только возможность приема сообщений с HTTP-заголовками. Эта функция легко реализуется простыми веб-контейнерами для Java-приложений.

REST-приложения часто создаются на основе сервлетов. Сервлеты не предписывают какие-либо конкретные походы к разработке. Как правило, сервлеты получают на обработку запросы, анализируют их заголовки, в том числе URI, чтобы определить, к какому ресурсу выполняется обращение. Ряд API был создан на основе этой простой модели сервлетов. Несмотря на все усилия по формализации, ни один из этих API не превратился в официальный стандарт.

Учитывая рост популярности архитектуры REST, в итоге появился документ JSR-311, а также спецификация JAX-RS 1.0, описывающая подход к созданию REST-сервисов на основе аннотаций. В отличие от модели на основе сервлетов, аннотации JAX-RS позволяют разработчикам сосредоточиться на прикладных ресурсах и данных, не отвлекаясь на вопросы, связанные с обменом информацией (как в случае сервлетов).

JAX-RS (JSR-311) – это спецификация, описывающая сервисы, работающие по принципам REST, в среде Java EE. Подобные сервисы представляют собой реализуемую на практике альтернативу традиционным веб-сервисам, использующим протокол SOAP.

**Ресурсы в Java**

JAX-RS задает унифицированный способ описания ресурсов на основе своей модели программирования. Он включает пять основных компонентов: корневые ресурсы, дочерние ресурсы, методы ресурсов, методы дочерних ресурсов и локаторы дочерних ресурсов.

**Корневые ресурсы**

Корневыми ресурсами являются Java-классы, отмеченные аннотацией @Path. Эта аннотация включает атрибут value, задающий путь к ресурсу. Его значением могут быть строка символов, переменные, а также переменные в сочетании с регулярным выражением. Пример приведен ниже:

package com.mmf.rest.impl;

import ...

**@Path("schedule")**

public class ScheduleResource

extends CrudResource<Schedule, ScheduleService>{

…

}

**Методы ресурсов**

Методами ресурсов называются методы Java-классов, представляющих собой корневые или дочерние ресурсы. Эти методы привязаны к типам HTTP-запросов при помощи аннотаций, например аннотации @GET.

**@GET**

**@Produces**(MediaType.APPLICATION\_JSON)

public Response getSchedule(

**@QueryParam**("course") int course,

**@QueryParam**("group") int group,

**@QueryParam**("subGroup") **@DefaultValue**("") String subGroup, **@QueryParam**("lecturerId") Long lecturerId) {

if (lecturerId == null && (course == 0 || group == 0)) {

throw new RestServiceException(

Response.Status.BAD\_REQUEST.getStatusCode());

}

if (lecturerId != null && (course != 0 || group != 0)) {

throw new RestServiceException(

Response.Status.BAD\_REQUEST.getStatusCode());

}

if (lecturerId == null) {

return getScheduleForStudent(course, group, subGroup);

} else {

return getScheduleForLecturer(lecturerId);

}

}

**Методы дочерних ресурсов**

Методы дочерних ресурсов аналогичны методам ресурсов за тем исключением, что они дополнительно отмечены аннотацией @Path, уточняющей, в каких случаях их следует вызывать.

**@GET**

**@Path("/list")**

**@Produces**(MediaType.APPLICATION\_JSON)

public Response list(){

try {

List<ScheduleResponse> scheduleResponses

= new LinkedList<ScheduleResponse>();

for(Schedule schedule : getService().list()){

scheduleResponses.add(new ScheduleResponse(schedule));

}

return Response.ok(scheduleResponses).

header("Content-Encoding", "utf-8").build();

} catch (BusinessServiceException e) {

throw new RestServiceException(e.getErrorCode());

}

}

**Аннотации @GET, @POST, @PUT, @DELETE, @HEAD**

Аннотации @GET, @POST, @PUT, @DELETE и @HEAD соответствуют типам HTTP-запросов. Их можно использовать для привязки методов корневых и дочерних ресурсов к запросам соответствующих типов. Запросы типа GET передаются на обработку методам, аннотированным @GET, запросы типа @POST – методам с аннотацией @POST и т.д.

**Аннотации @Consumes и @Produces**

Аннотация @Consumes задает типы содержимого MIME, принимаемые ресурсом, а @Produces – типы MIME, возвращаемые ресурсом. Этими аннотациями могут отмечаться ресурсы, дочерние ресурсы, методы ресурсов и дочерних ресурсов, а также локаторы дочерних ресурсов.

**@POST**

**@Path("/add")**

**@Consumes**(MediaType.APPLICATION\_JSON)

**@Produces**(MediaType.APPLICATION\_JSON)

public Response add(T domain) {

try {

DomainUtil.checkingForNull(domain.getId());

validate(domain);

getService().create(domain);

return Response.ok(domain).header("Content-Encoding", "utf-8").build();

} catch (BusinessServiceException e) {

throw new RestServiceException(e.getErrorCode());

} catch (NotNullPropertyException e) {

throw new RestServiceException(

Response.Status.BAD\_REQUEST.getStatusCode());

}

}

#### 1.2.2. Android

Android – это платформа, предназначенная для мобильных устройств. Если говорить более точно, то Android можно охарактеризовать как программный стек, одной из составляющих которого является операционная система, построенная на ядре Linux. Также в Android входят набор промежуточного программного обеспечения, пользовательский интерфейс и приложения, обеспечивающие базовый функционал.

В отличие от приложений в большинстве других систем, приложения Android не имеют единой точки входа (например, отсутствует функция main()). Вместо этого существуют четыре типа основных компонентов, из которых строятся андроид-приложения и которые система может запускать по мере необходимости. Это **Activity**, **Service**, **Broadcast receiver** и **Content provider**.

**Activity** представляет собой визуальный интерфейс (отдельный экран) для одного действия, которое пользователь может совершить. Например, в нашем приложении существуют для просмотра расписания, логина и др.

Визуальный интерфейс строится на основе иерархии визуальных компонентов — объектов, производных от базового класса View. Android имеет ряд готовых к использованию компонентов, включая кнопки, текстовые поля, полосы прокрутки, меню, флажки и многое другое.

Для того чтобы подключить интерфейс к activity, нужно вызвать метод Activity.setContentView(view). Параметром этого метода является экземпляр класса, расширяющего класс View.

Жизненный цикл activity состоит из трёх вложенных циклов:

* Жизненный цикл activity начинается с вызова метода onCreate(), в котором производится первоначальная настройка глобального состояния, и завершается вызовом метода onDestroy(), в котором оно освобождает занятые ресурсы. Например, в onCreate() можно создать поток, загружающий данные из сети в фоновом режиме, и затем остановить его в onDestroy().
* Видимая часть жизненного цикла происходит между вызовами onStart() и onStop(). В течение этого времени пользователь может видеть activity на экране, хотя оно может быть не на переднем плане и не взаимодействовать с пользователем. Между этими двумя методами вы можете выделять ресурсы, необходимые для отображения activity пользователю. Методы onStart() и onStop()могут вызываться столько раз, сколько activity становится видимым или скрытым для пользователя.
* На переднем плане activity находится между вызовами onResume() и onPause(). В течение этого времени activity находится поверх других и взаимодействует с пользователем. Activity может часто переходить в состояние паузы и выходить из него. Например, метод onPause() может быть вызван, когда устройство переходит в спящий режим или когда запускается другое activity, а методonResume() — при получении результата от закрывающегося activity. Таким образом, код в этих двух методах должен быть довольно легким.

На диаграмме ниже показаны эти циклы. Цветные овалы являются основными состояниями, в которых может находиться activity. Прямоугольники представляют колбеки, которые можно реализовать для выполнения каких-либо операций при изменении состояния activity.

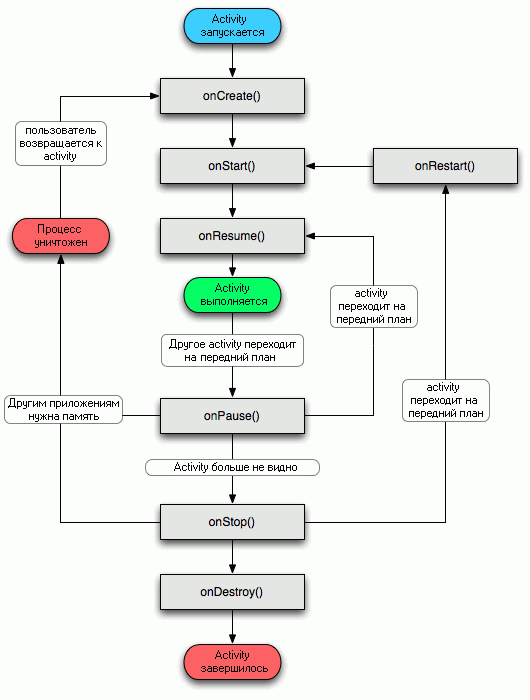


Рисунок 5. Жизненный цикл activity.

### 1.3. Проектирование приложения: модель

В основе разрабатываемой системы лежит архитектура «клиент-сервер», в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. В качестве среды взаимодействия клиента с сервером используется интернет.



Рисунок 6. Общая архитектура приложения (концепция взаимодействия).

Основными достоинствами архитектуры «клиент-сервер» являются:

* Возможность, в большинстве случаев, распределить функции вычислительной системы между несколькими независимыми компьютерами в сети. Это позволяет упростить обслуживание вычислительной системы. В частности, замена, ремонт, модернизация или перемещение сервера, не затрагивают клиентов.
* Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.
* Позволяет объединить различные клиенты. Использовать ресурсы одного сервера часто могут клиенты с разными аппаратными платформами, операционными системами и т.п.

Основные недостатки:

* В случае использования централизованной системы, неработоспособность основного сервера может сделать неработоспособным всё приложение.
* Администрирование данной системы требует квалифицированного профессионала;
* Высокая стоимость оборудования.

В ходе выбора аппаратной платформы будут предложены и реализованы решения, позволяющие минимизировать вероятность выхода из строя серверной части приложения, а также позволяющие снизить стоимость оборудования до оплаты минимально необходимого уровня производительности.

Клиентская часть приложения должна поддерживать следующие технологии:

* Доступ к сети интернет.
* Возможность работы по протоколу HTTP.
* Поддержка устройством взаимодействия с человеком для ввода данных.

### 1.4. Детализация проекта

В основе разрабатываемого приложения лежит расписание занятий для студентов механико-математического факультета. Каждый элемент расписания характеризуется следующей информацией:

* Время и день проведения занятия
* Место (аудитория) проведения занятия
* Группы, для которых проводится данное занятие
* Преподаватель, который ведет данное занятие
* Пометки, оставленные пользователем

В качестве пользователей системы выступают 3 основные группы: администратор, преподаватель, студент, гость. Так же отдельно выделяется роль «Староста группы».

Гость приложения имеет возможность просматривать расписание для студентов и преподавателей. Никакая информация о нем в системе не хранится.

Студент характеризуется следующей информацией:

* ФИО пользователя
* Логин и пароль
* Учебная группа
* Специальность

Студент обладает следующими правами:

* Просматривать расписание своей группы и любой другой на факультете
* Просматривать расписание для преподавателей факультета
* Залогинившись, оставлять пометки к занятиям
* Делать пометки публичными для своей группы
* Если студент обладает ролью «Староста группы», то он имеет возможность делать пометки публичными для нескольких групп, а также и для преподавателей

Преподаватель характеризуется следующей информацией:

* ФИО пользователя
* Логин и пароль
* Кафедра

Преподаватель обладает теми же правами, что и студент.

Администратор характеризуется следующей информацией:

* ФИО пользователя
* Логин и пароль

Администратор обладает следующими правами:

* Создание/редактирование/удаление пользователей
* Создание/редактирование/удаление учебных групп
* Создание/редактирование/удаление кафедр
* Создание/редактирование/удаление специальностей
* Создание/редактирование/удаление предметов
* Создание/редактирование/удаление аудиторий
* Создание/редактирование/удаление учебных программ
* Создание/редактирование/удаление расписания

При первом запуске приложения требуется доступ к интернету. После приложение может работать в офлайн режиме. При работе приложения в онлайн режиме для получения данных используются REST-сервисы. Имея доступ к интернету, данные можно обновлять.

# ГЛАВА 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

### 2.1. Описание разработки практической части

#### 2.1.1. Постановка задачи

1. Создание информационной модели
2. Реализация RESTful веб-сервисов.
3. Создание прототипа приложения
4. Создание андроид-клиента для просмотра учебного расписания в режимах онлайн и офлайн.

Целью разработки является создание системы, включающей в себя серверную часть, обрабатывающую поступающие запросы пользователей системы; клиентскую часть, к которой относятся интерфейсы пользователей; и базу данных для хранения всей необходимой информации.

#### 2.1.2. Описание архитектуры информационной модели

**Серверная часть**

Информационная модель представляет собой совокупность следующих сущностей:

* Classroom
* Curriculum
* Department
* Discipline
* DisciplineTime
* DisciplineType
* Group
* Lecturer
* Note
* Schedule
* Specialty
* Student
* Study
* User

Сущности Student и Lecturer связаны с сущностью User связью «один-к-одному» (Рисунок 7): один пользователь может быть только или студентом, или преподавателем; он не может быть одновременно и студентом, и преподавателем.

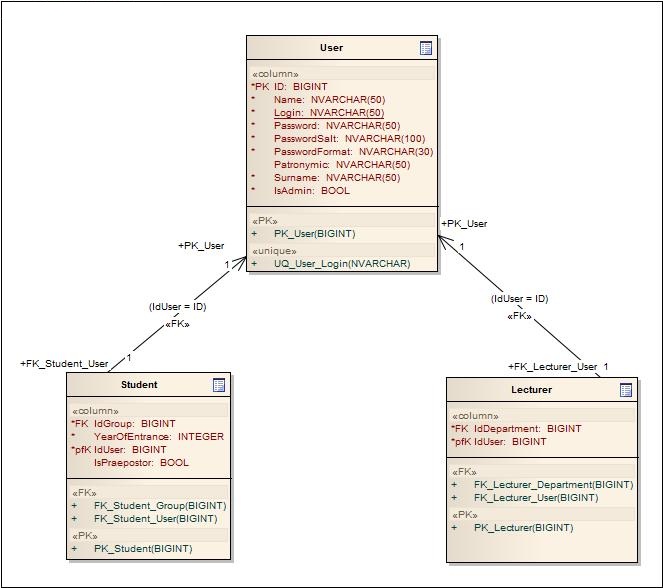


Рисунок 7. Информационная модель.

Сущность Group связана с сущностью Student «один-ко-многим» (Рисунок 8): учебная группа состоит из множества студентов; студент может учиться только в одной группе.

Сущность Group состоит из подгрупп, поэтому она имеет связь сама на себя «один-ко-многим» (Рисунок 8).

Сущность Specialty связана с сущностью Group связью «один-ко-многим» (Рисунок 8): каждая группа может принадлежать только к одной специальности; к одной специальности могут относиться несколько групп.

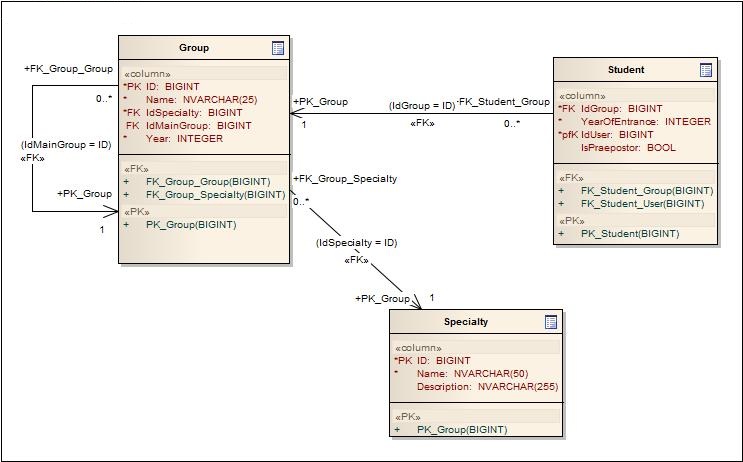


Рисунок 8. Информационная модель.

Сущность Department связана с сущностью Lecturer связью «один-ко-многим» (Рисунок 9): один преподаватель может принадлежать только одной кафедре; одна кафедра состоит из множества преподавателей.

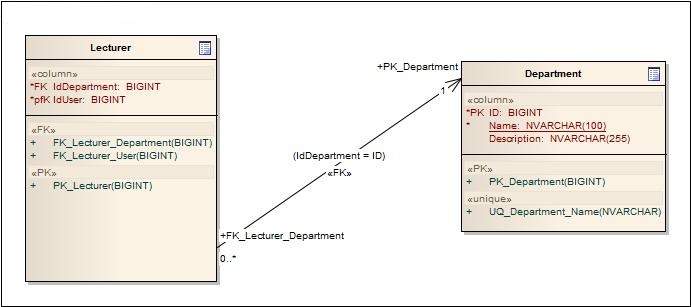


Рисунок 9. Информационная модель.

Сущность Curriculum связана с сущностями Specialty и Discipline связью «один-ко-многим» (Рисунок 10): учебная программа составляется отдельно для каждой специальности по каждому предмету.

Сущность DisciplineType связана с сущностью Discipline связью «один-ко-многим» (Рисунок 10).

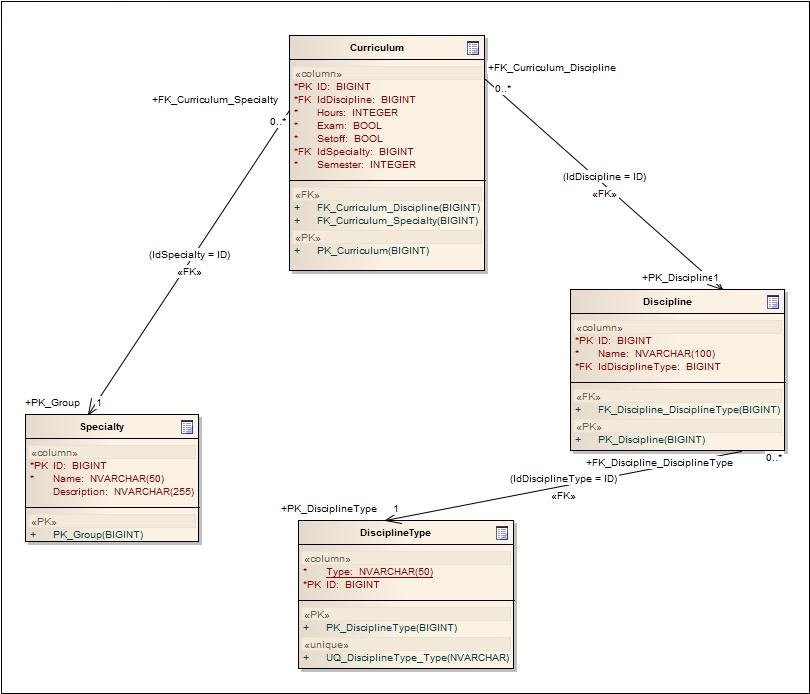


Рисунок 10. Информационная модель.

Сущность Study реализует связь «многие-ко-многим» попарно между сущностями Group, Lecturer и Curriculum (Рисунок 11):

* Один преподаватель ведет у множества групп, и у одной группы ведут занятия множество преподавателей;
* Одной группе преподается множество дисциплин в соответствие с учебной программой, и одна учебная программа относится к нескольким группам;
* Один преподаватель ведет множество дисциплин в соответствие с учебной программой, и одну дисциплину могут везти несколько преподавателей.

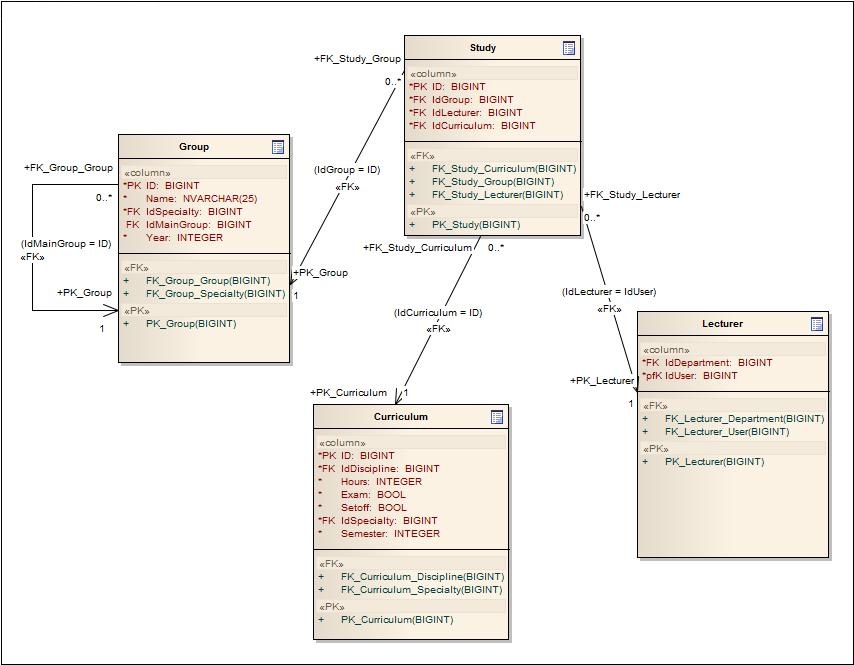


Рисунок 11. Информационная модель.

Сущности Study, DisciplineTime, Classroom связаны с сущностью Schedule связью «один-ко-многим» (Рисунок 12): каждый элемент расписания характеризуется конкретным учебным занятием, аудиторией, в которой проводится занятие, и временем, когда проводится занятие.

Сущность Schedule связана с сущностью Note связью «один-ко-многим» (Рисунок 12): множество пометок может быть оставлено пользователями, относящихся к различным элементам расписания.

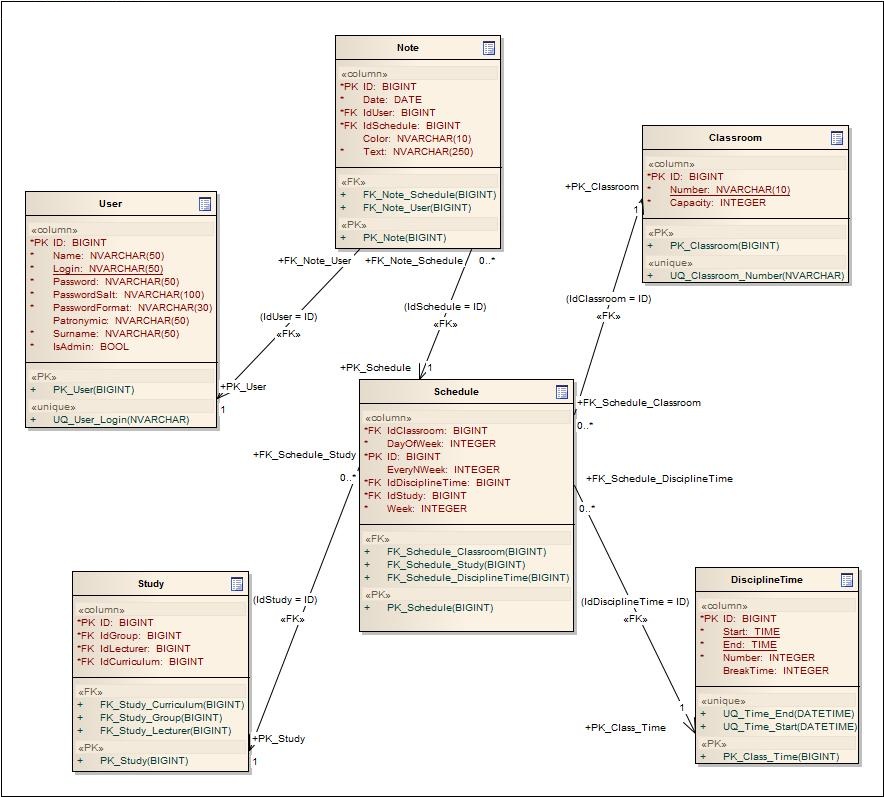


Рисунок 12. Информационная модель.

**Клиентская часть**

Центральной сущностью является сущность Schedule (Рисунок 13): каждый элемент расписания характеризуется конкретным учебным занятием, аудиторией, в которой проводится занятие, и временем, когда проводится занятие.

Сущность Schedule связана с сущностью Note связью «один-ко-многим» (Рисунок 12): множество пометок может быть оставлено пользователем, относящихся к различным элементам расписания.

Сущность Filter является вспомогательной: упрощает реализацию синхронизации.

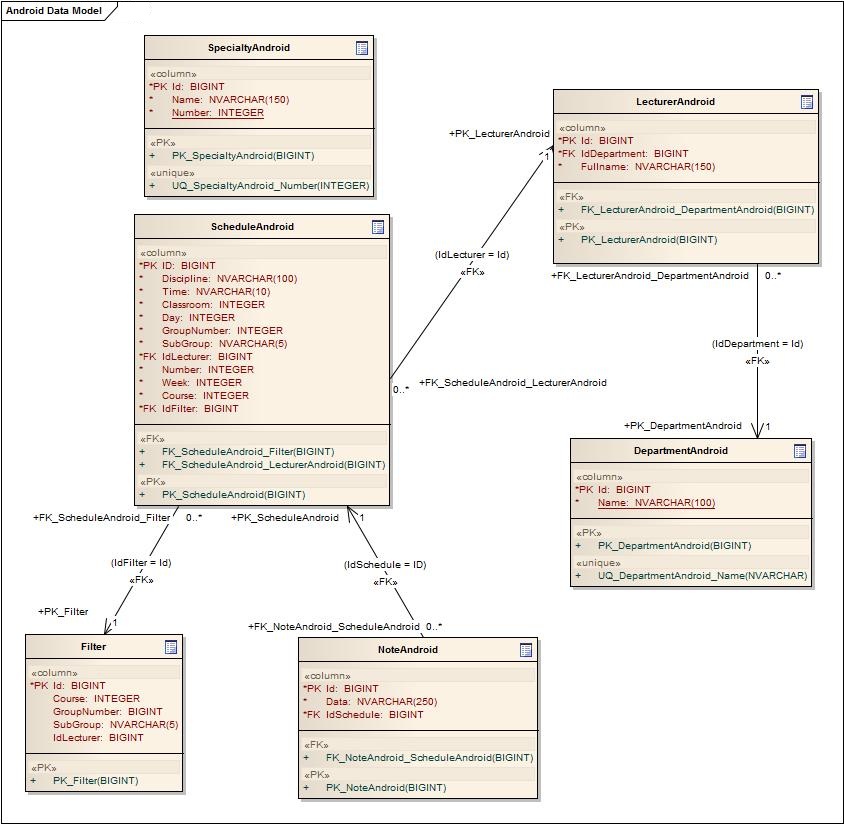
****

Рисунок 13. Информационная модель.

#### 2.1.3. Реализация RESTful веб-сервисов

**Ресурсы**

**Пользователь** (User)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор пользователя |
| Name | (required) | String(50) | Имя пользователя |
| Surname | (required) | String(50) | Фамилия пользователя |
| Patronymic | (required) | String(50) | Отчество пользователя |
| FullName |  | String(150) | ФИО |
| Login | (required) | String(50) | Логин пользователя |
| Password | (required) | String(50) | Пароль пользователя (номер зачетки) |
| IsAdmin | (required) | Boolean | Является ли пользователя админом системы |
| Notes |  | Array[] | Список всех пометок, оставленных пользователем |

Таблица 1. Ресурс пользователь.

**Студент** (Student)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор пользователя. |
| Name | (required) | String(50) | Имя пользователя |
| Surname | (required) | String(50) | Фамилия пользователя |
| Patronymic | (required) | String(50) | Отчество пользователя |
| FullName |  | String(150) | ФИО |
| Login | (required) | String(50) | Логин пользователя |
| Password | (required) | String(50) | Пароль пользователя (номер зачетки) |
| IsAdmin | (required) | Boolean | Является ли пользователь админом системы |
| IsPraepostor | (required) | Boolean | Является ли студент старостой группы |
| Group | (required) | Group | Группа, в которой учится студент |
| YearOfEntrance | (required) | Integer | Год поступления в университет |
| Notes |  | Array[] | Список всех пометок, оставленных студентом |

Таблица 2. Ресурс студент.

**Группа** (Group)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор группы |
| Number | (required) | Integer | Номер группы |
| Subgroup | (required) | String(1) | Название подгруппы (a,b) или null. При создании группы значение поля должно быть null, а при создании подгруппы – “a” или “b”. Нельзя создать подгруппу, не создав перед этим группу. |
| Course | (required) | Integer | Номер курса |
| Year | (required) | Integer | Год создания группы (год поступления) |
| Specialty | (required) | Specialty | Специальность группы |
| Students |  | Array[] | Список студентов, учащихся в данной группе |

Таблица 3. Ресурс учебная группа.

**Специальность** (Specialty)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор специальности |
| Name | (required) | String(50) | Название специальности |
| Description | (required) | String(255) | Описание специальности |
| Groups |  | Array[] | Группы данной специальности |

Таблица 4. Ресурс специальность.

**Преподаватель** (Lecturer)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор преподавателя |
| Name | (required) | String(50) | Имя преподавателя |
| Surname | (required) | String(50) | Фамилия преподавателя |
| Patronymic | (required) | String(50) | Отчество преподавателя |
| FullName |  | String(150) | ФИО |
| Login | (required) | String(50) | Логин преподавателя |
| Password | (required) | String(50) | Пароль преподавателя (номер зачетки) |
| IsAdmin | (required) | Boolean | Является ли преподаватель админом системы |
| Department | (required) | Department | Кафедра, которой принадлежит преподаватель |
| Notes |  | Array[] | Список всех пометок, оставленных преподавателем |

Таблица 5. Ресурс преподаватель.

**Кафедра** (Department)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор кафедры |
| Name | (required) | String(50) | Название кафедры |
| Description | (required) | String(255) | Описание кафедры |
| Lecturers |  | Array[] | Список преподавателей, принадлежащих кафедре |

Таблица 6. Ресурс кафедра.

**Учебная дисциплина** (Discipline)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор учебной дисциплины |
| Name | (required) | String(50) | Название дисциплины |
| DisciplineType | (required) | DisciplineType | Тип дисциплины |

Таблица 7. Ресурс учебная дисциплина.

**Тип учебной дисциплины** (DisciplineType)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор типа учебной дисциплины |
| Name | (required) | String(50) | Тип учебной дисциплины |

Таблица 8. Ресурс тип учебной дисциплины.

**Учебная программа** (Curriculum)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор учебной программы |
| Discipline | (required) | Discipline | Дисциплина, для которой составлена учебная программа |
| Hours | (required) | Integer | Количество часов |
| Semester | (required) | Integer | Номер семестра, в котором будет применена данная учебная программа |
| Specialty | (required) | Specialty | Специальность, для которой составлена данная учебная программа |
| IsExam | (required) | Boolean | Будет ли экзамен по данной учебной дисциплине |
| IsSetoff | (required) | Boolean | Будет ли зачет по данной учебной дисциплине |

Таблица 9. Ресурс учебная программа.

**Аудитория** (Classroom)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор аудитории |
| Number | (required) | String(10) | Номер аудитории |
| Capacity | (required) | Integer | Вместимость аудитории |

Таблица 10. Ресурс аудитория.

**Время занятий** (DisciplineTime)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор |
| StartTime | (required) | Date | Время начала пары (формат "HH:mm") |
| EndTime | (required) | Date | Время окончания пары (формат "HH:mm") |
| BreakTime | (required) | Integer | Перерыв между парами (в минутах) |
| Number | (required) | Integer | Номер пары |

Таблица 11. Ресурс время занятий.

**Занятие** (Study)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор занятия |
| Group | (required) | Group | Группа, идущая на данное занятие |
| Lecturer | (required) | Lecturer | Преподаватель, который ведет данное занятие |
| Curriculum | (required) | Curriculum | Учебная программа по данному занятию |

Таблица 12. Ресурс занятие.

**Пометка** (Note)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор пометки |
| User | (required) | User | Пользователь, оставивший пометку |
| Date | (required) | Date | Дата, на которую добавлена пометка |
| Schedule | (required) | Schedule | Элемент расписания, на который была добавлена заметка |
| Text | (required) | String(250) | Текст пометки |
| Color |  | String(10) | Цвет пометки (#afafaf) |

Таблица 13. Ресурс пометка.

**Элемент расписания** (Schedule)

|  |  |  |  |
| --- | --- | --- | --- |
| Название |  | Тип | Описание |
| Id |  | Long | Уникальный идентификатор элемента расписания |
| Classroom | (required) | Classroom | Аудитория, в которой будет проводиться занятие |
| Study | (required) | Study | Занятие, которое будет проводиться |
| DisciplineTime | (required) | DisciplineTime | Время, в которое будет проводиться занятие |
| DayOfWeek | (required) | Integer | День недели, в который будет проводиться занятие.  2 – понедельник  3 – вторник  4 – среда  5 – четверг  6 – пятница  7 - суббота |
| Week | (required) | Integer | По каким неделям будет занятие.  0 – каждую неделю  1 – по нечетным неделям  2 – по четным неделям |
| Notes |  | Array[] | Список пометок для данного занятия |

Таблица 14. Ресурс элемент расписания.

**API Endpoints**

Schedule API обеспечивает доступ к таким ресурсам, как schedule, discipline, curriculum, student, lecturer и другие. Например, информация о ресурсе учебная дисциплина может быть получена, вызвав URL

http://api.schedule.by/rest/bsu/mmf/discipline/{disciplineId}.

Получив ресурс, можно получить информацию об аспекте этого ресурса, например http://api.schedule.by/rest/bsu/mmf/discipline/{disciplineId}/discilinetype.

Каждый полученный disciplinetype ресурс имеет свой собственный id, который соответствует URL для ресурса, например

http://api.schedule.by/rest/bsu/mmf/disciplinetype/{disciplineTypeId}.

Также каждый ресурс имеет список действий (actions). Например, вызывая http://api.schedule.by/rest/bsu/mmf/discipline/{disciplineId}/delete, удалится данная учебная дисциплина.

Вызов некоторых endpoints требует, чтобы пользователь был авторизован. Для таких запросов должен быть добавлен заголовок:

Authorization: Basic “{username}:{password}”,

строка “{username}:{password}” должна быть зашифрована, используя Base64.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Resource | Aspects | Actions | Resource | Aspects | Actions |
| user |  | add  edit  delete  list | **discipline** | disciplineType | add  edit  delete  list |
| student | group  notes | add  edit  delete  list  scheduleForDay  schedule | **disciplineType** |  | add  edit  delete  list |
| lecturer | department  notes | add  edit  delete  list  scheduleForDay  schedule | **disciplineTime** |  | add  edit  delete  list |
| group | specialty  students | add  edit  delete  list | **curriculum** | specialty  discipline | add  edit  delete  list |
| specialty | groups | add  edit  delete  list | **classroom** |  | add  edit  delete  list |
| department | lecturers | add  edit  delete  list | **study** | lecturer  group  curriculum | add  edit  delete  list |
| note | schedule  user | add  edit  delete  list | **schedule** | classroom  study  disciplineTime  notes  group  lecturer  discipline | add  edit  delete  list  schedule |

Таблица 15. API Endpoints.

#### 2.1.4. Создание прототипа андроид-приложения

# В качестве средства для создания прототипа приложения была выбрана программа Balsamiq Mockups.

Макеты, получаемые с помощью Balsamiq Mockups, относятся к так называемым *макетам с низкой степенью детализации*. Предполагается, что именно скорость создания макетов является ключевым преимуществом Balsamiq Mockups.

Приложение поддерживает также такие стандартные действия как группировку элементов, отмену предыдущего шага, блокировку элементов от случайного перемещения и размещение элементов слоями.

Balsamiq Mockups предоставляет возможность экспорта нарисованного макета в форматы png (Рисунок 14-16) и pdf. Добавив ссылки на макеты, с помощью экспорта в pdf можно получить интерактивный pdf-файл. (смотри приложение).

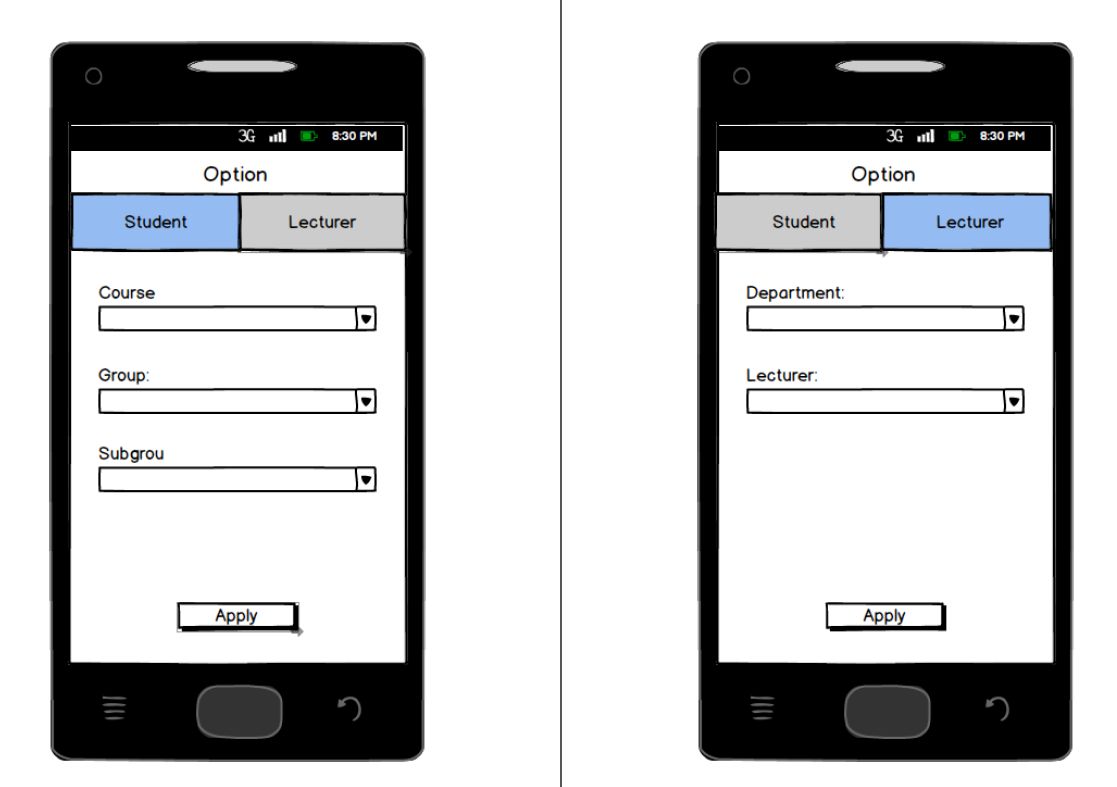
****

Рисунок 14. Экран настроек.

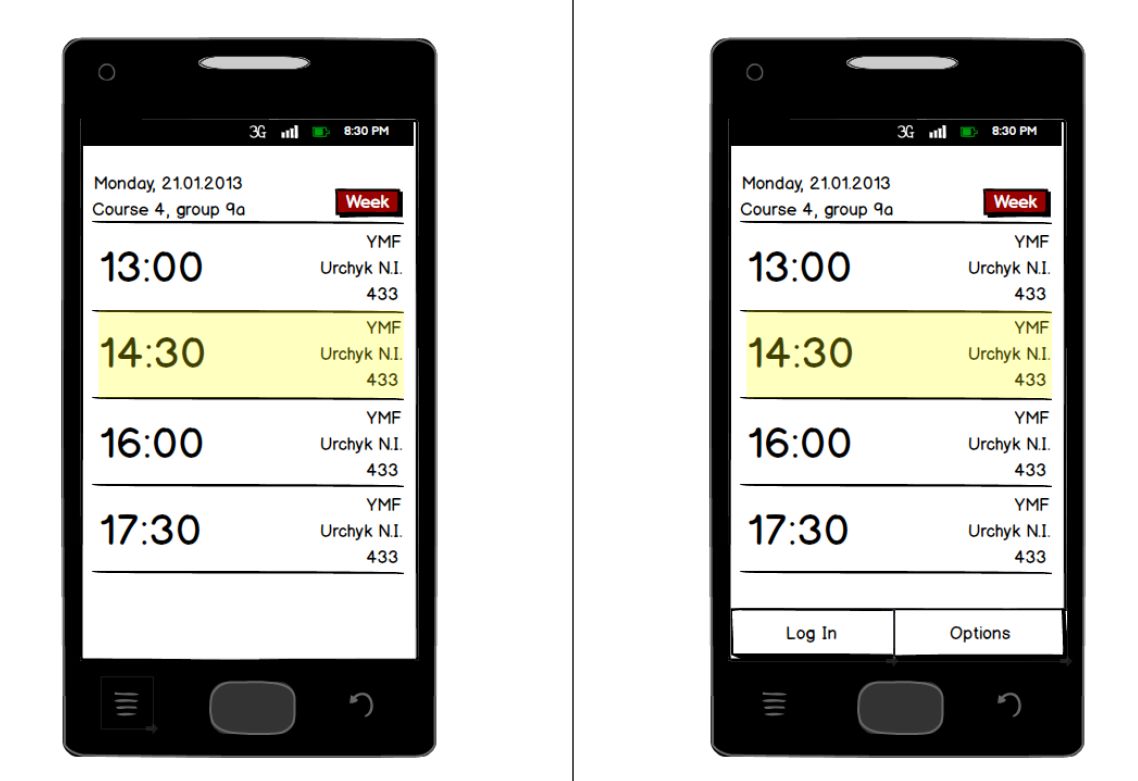


Рисунок 15. Режим просмотра расписания на день.

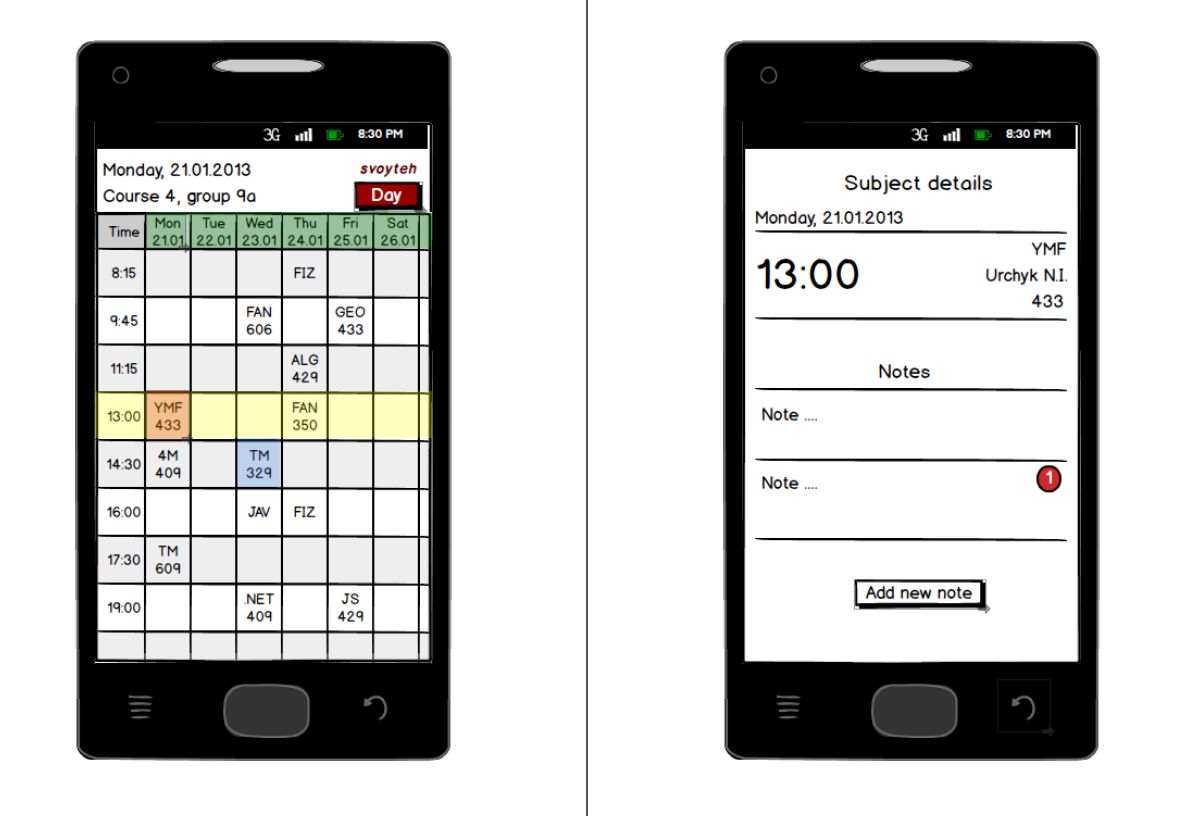


Рисунок 16. Режим просмотра расписания на неделю. Добавление заметок.

#### 2.1.5. Описание функционала

### 2.2. Методология тестирования. Результаты

# ГЛАВА 3 ОРГАНИЗАЦИОННАЯ ЧАСТЬ

### 3.1. Перспективы развития

### 3.2. Выводы

В ходе выполнения дипломной работы была изучена методология написания REST сервисов и приложения для Android. Были изучены и использованы такие технологии, как Hibernate, Spring, JAX-RS.

В итоге была спроектирована архитектура серверной и клиентской частей. Была разработана серверная часть с использованием технологий JEE. Было разработано андроид-приложение, позволяющее просматривать расписание занятий, как в режиме онлайн, так и офлайн. Также реализована возможность пользователям оставлять пометки в учебном расписании, например, чтобы не забыть о лабораторной работе или о дате контрольной.

Социально-практическая значимость дипломной работы заключается в том, что на данный момент не существует аналогов разработанному приложению. Не смотря на то, что существуют приложения на мобильные устройства для просмотра учебного расписания в ВУЗах, ни одно из них не поддерживает работу с БГУ.

# СПИСОК ЛИТЕРАТУРЫ

[1] Эккель, Б. Философия Java. Библиотека программиста. 4-е изд. – СПб.: Питер, 2009. – 640 с.

[2] Блинов, И.Н. Java. Промышленное программирование: практическое пособие /И.Н. Блинов, В.С. Романчик. – Минск: УниверсалПресс, 2007. – 704 с.

[3] Эрик Фримен, Элизабет Фримен. Паттерны проектирования. /Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс – СПб: Питер, 2012. – 656 с.

[4] MySQL [Электронный ресурс] – Электронные данные. – Режим доступа: <http://www.mysql.com/>

[5] Fast Object Database for Java - with JPA/JDO support [Электронный ресурс] – Электронные данные. – Режим доступа: <http://www.objectdb.com/api/java/jpa>

[6] Spring Source Community [Электронный ресурс] – Электронные данные. – Режим доступа: <http://www.springsource.org/documentation>

[7] Google Android [Электронный ресурс] – Электронные данные. – Режим доступа: <http://startandroid.ru/uroki/vse-uroki-spiskom.html>