**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

кандидат физ.-мат. наук,

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](#_Toc358578051)

[ВВЕДЕНИЕ 3](#_Toc358578052)

[ГЛАВА 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc358578053)

[1.1. Сравнение с аналогами. Обзор существующих разработок 5](#_Toc358578054)

[1.2. Краткий обзор существующих технологий разработки приложения 8](#_Toc358578055)

[1.3. Проектирование приложения: модель 15](#_Toc358578056)

[1.4. Детализация проекта 17](#_Toc358578057)

[ГЛАВА 2 ПРАКТИЧЕСКАЯ ЧАСТЬ 19](#_Toc358578058)

[2.1. Описание разработки практической части 19](#_Toc358578059)

[2.2. Методология тестирования. 42](#_Toc358578060)

[ГЛАВА 3 ОРГАНИЗАЦИОННАЯ ЧАСТЬ 45](#_Toc358578061)

[3.1. Перспективы развития 45](#_Toc358578062)

[3.2. Выводы 45](#_Toc358578063)

[СПИСОК ЛИТЕРАТУРЫ 47](#_Toc358578064)

[ПРИЛОЖЕНИЕ 48](#_Toc358578065)

[Серверная часть 48](#_Toc358578066)

[Клиентская часть 65](#_Toc358578067)

[REST API 82](#_Toc358578068)

[АННОТАЦИЯ 125](#_Toc358578069)

[Abstract. 125](#_Toc358578070)

[Keywords. 125](#_Toc358578071)

# 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-математический факультет. 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), [Windows Phone](http://ru.wikipedia.org/wiki/Windows_Phone) или [Android](http://ru.wikipedia.org/wiki/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.

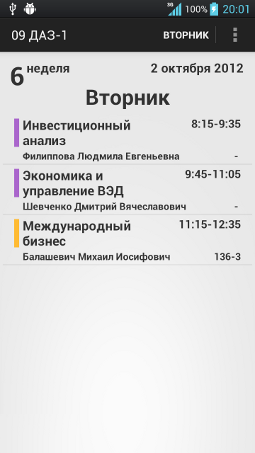
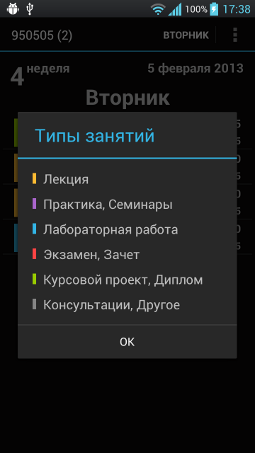
На данный момент «Студхелпер» поддерживает работуf,jne только с двумя ВУЗами (БГУИР и БГЭУ). Основным недостатком является отсутствие БГУ среди этих ВУЗов.

Рисунок 1. Аналог. Приложение «Студхелпер».

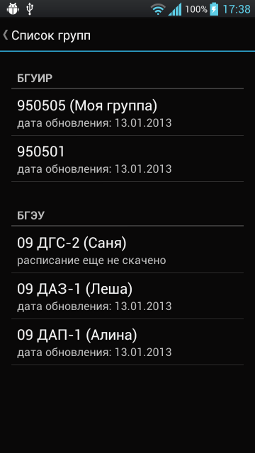
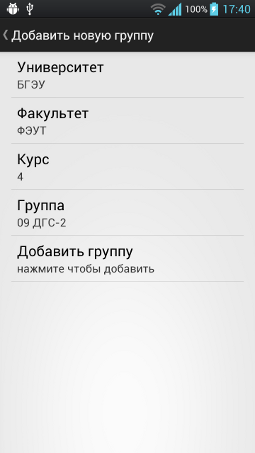


Рисунок 2. Аналог. Приложение «Студхелпер».

**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 работу только с ВУЗами России.

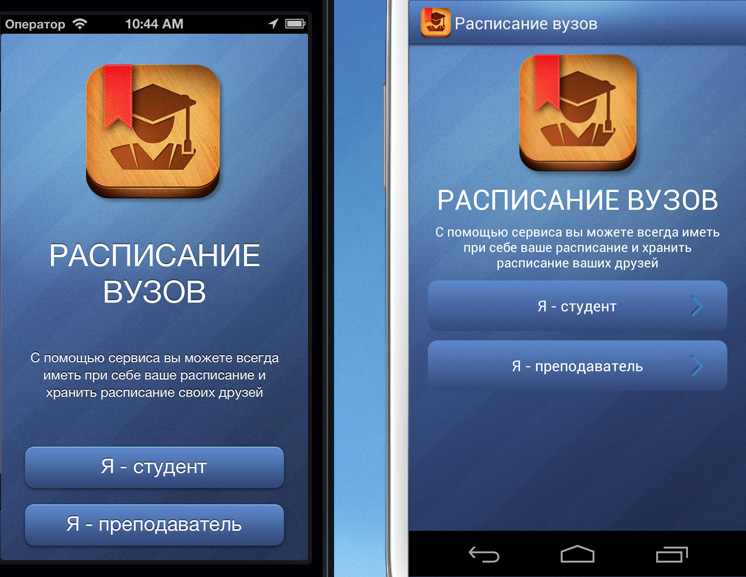


Рисунок 3. Аналог.Приложение «Расписание для ВУЗов».

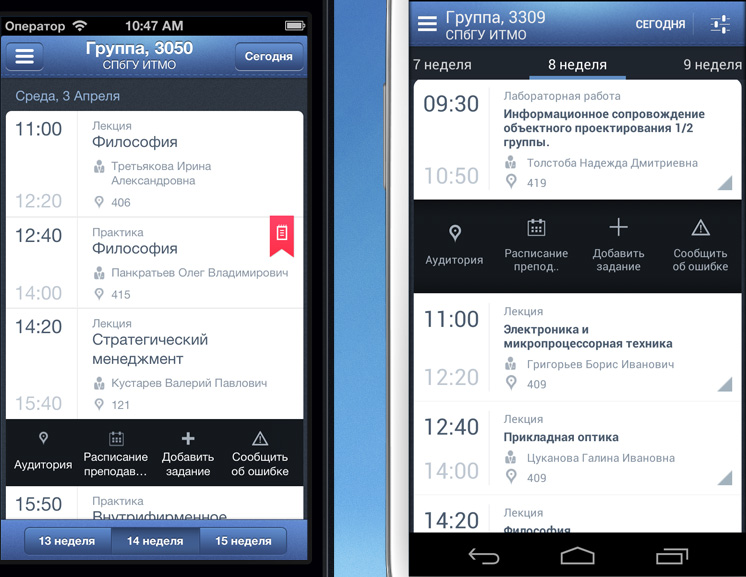


Рисунок 4. Аналог. Приложение **«**Расписание для ВУЗов».

### 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, Java Persistence API 2.0 (JPA 2.0), 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 JPA 2.0, а точнее его Hibernate реализация. Использование JPA освобождает разработчика от значительного объема сравнительно низкоуровневого программирования по обеспечению хранения объектов в реляционной БД. JPA содержит обширный набор аннотаций для описания отображений: связи, объединения, таблицы и колонки БД, генераторы последовательности для БД и другое. Благодаря JPA 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). 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).

#### 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.

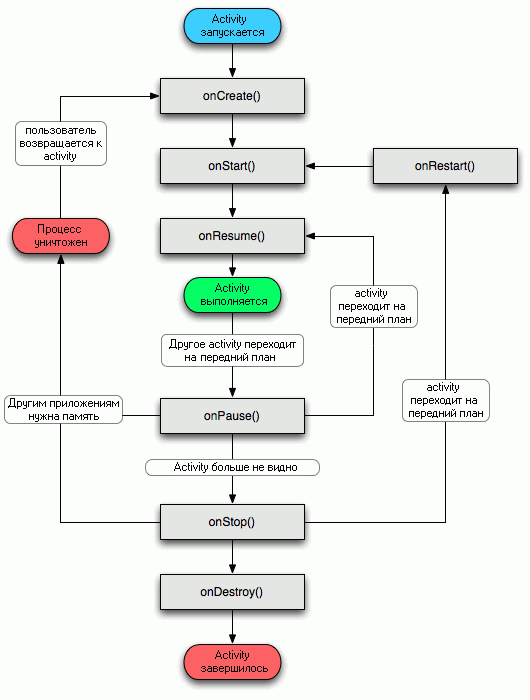


Рисунок 5. Жизненный цикл activity.

### 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.

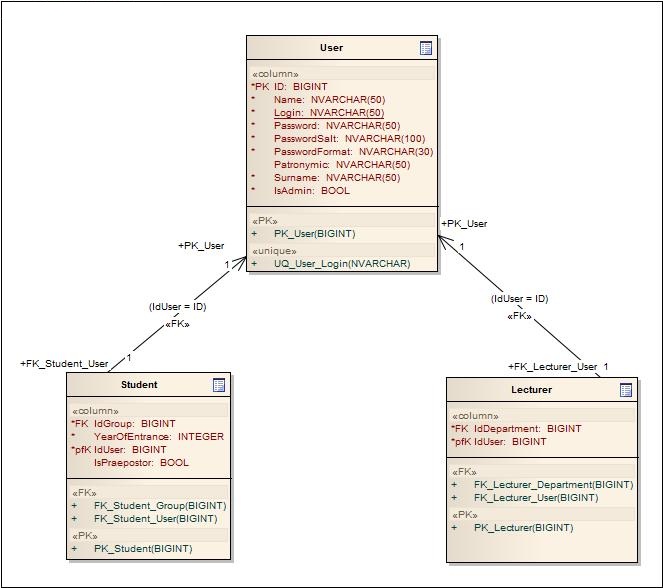


Рисунок 7. Информационная модель.

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.

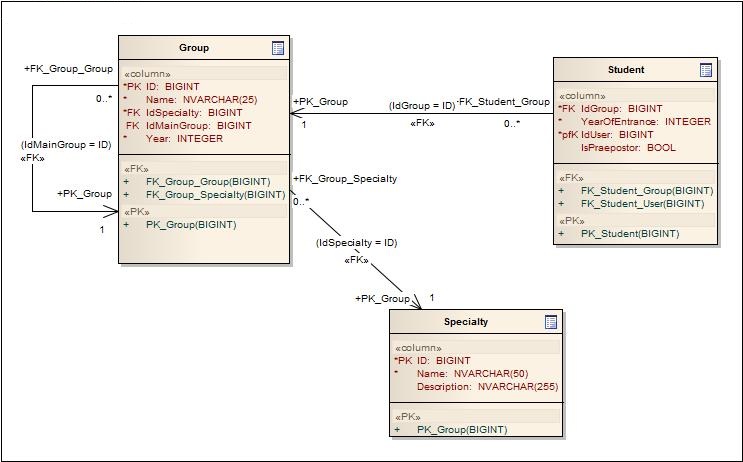


Рисунок 8. Информационная модель.

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.

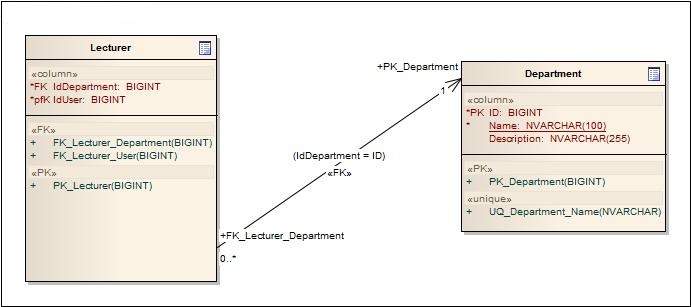


Рисунок 9. Информационная модель.

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).

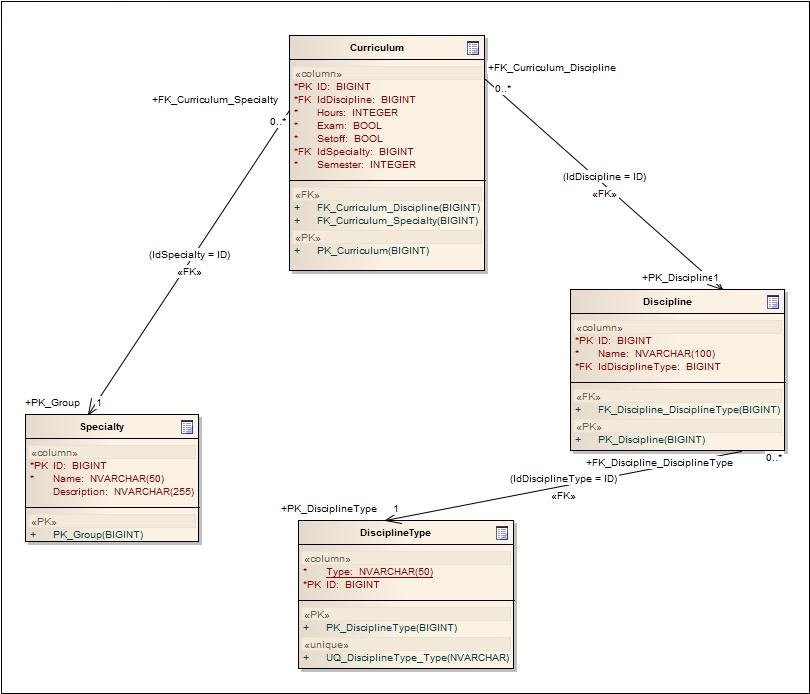


Рисунок 10. Информационная модель.

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.

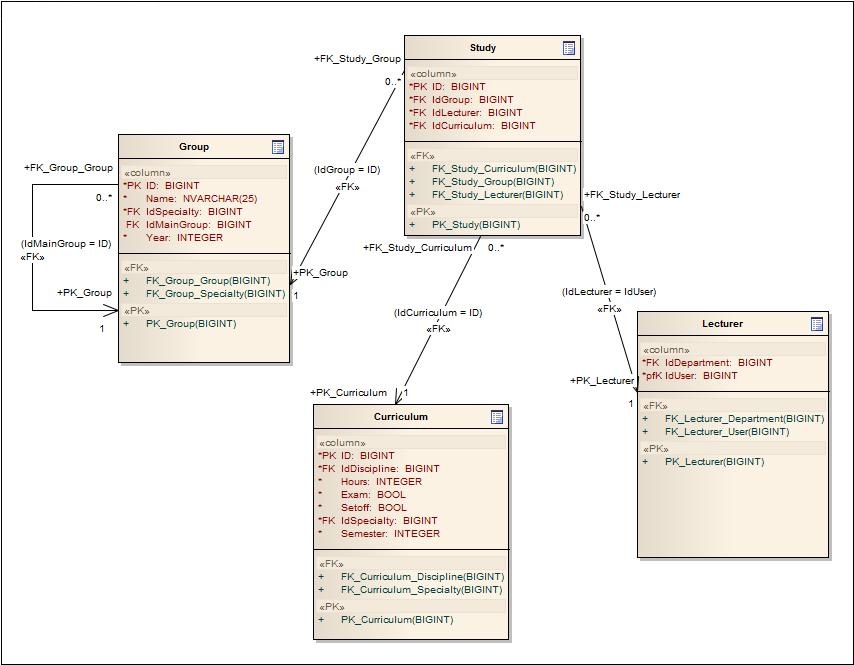


Рисунок 11. Информационная модель.

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.

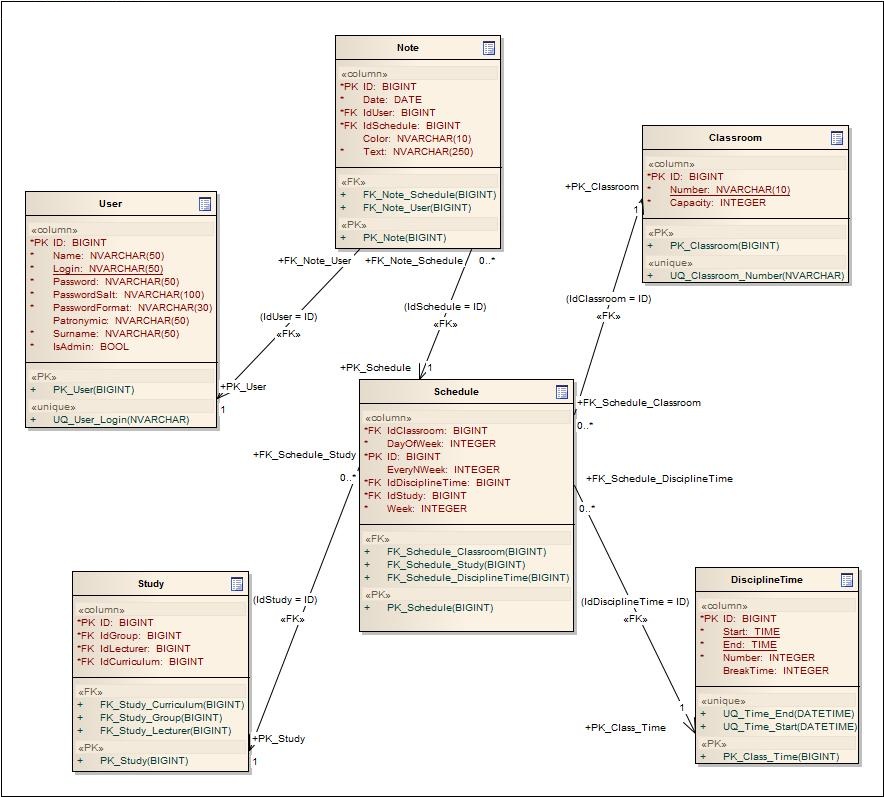


Рисунок 12. Информационная модель.

**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.

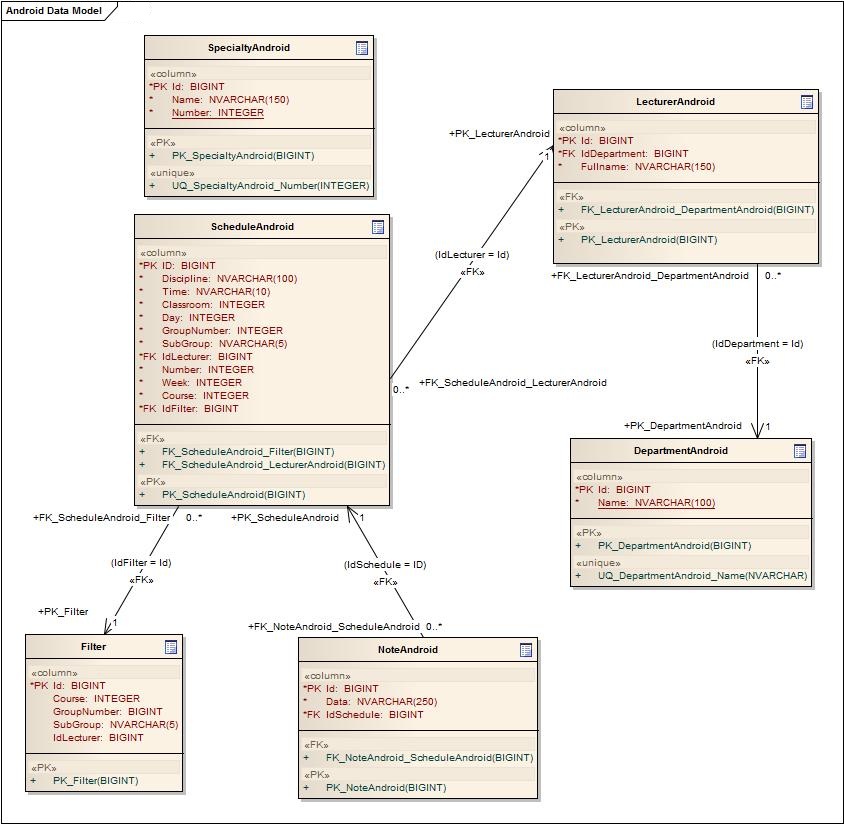
****

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.

Рассмотрим описание некоторых сервисов. Полное описание всех сервисов находится в разделе «ПРИЛОЖЕНИЕ».

Сервис, отвечающий за ***логин***.

**/rest/bsu/mmf/login**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_USER  ROLE\_STUDENT  ROLE\_LECTURER  ROLE\_ADMIN |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пользователе (ресурс **student** или **lecturer**)  **JSON student:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"1e3960302be7c4f7e1b360fadf320afbb20bdf96",  "groupId":2,  "yearOfEntrance":2011,  "praepostor":false,  "authorities":[  {"authority":"ROLE\_STUDENT"},  {"authority":"ROLE\_USER"},  {"authority":"ROLE\_ADMIN"}  ],  "admin":true  }  **JSON lecturer:**  {  "id":4,  "name":"Станислав",  "surname":"Суздаль",  "patronymic":"Валерьевич",  "fullName":"Суздаль Станислав Валерьевич",  "login":"suzdal",  "password":"1H8M7ArYttfoBrQipfJDkvXpwrI=",  "departmentId":4,  "authorities":[  {"authority":"ROLE\_LECTURER"},  {"authority":"ROLE\_USER"}  ],  "admin":false  } |
| 401 |  | Вернется, если введены неверные логин или пароль |

Сервисы, связанные с управлением ресурса ***user***.

**/rest/bsu/mmf/user/{userId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пользователе (ресурс **user**)  **JSON:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"70d0b1fbf6e5cb5fd4486bc295b99ac4149d23f6",  "authorities":[  {"authority":"ROLE\_USER"},  {"authority":"ROLE\_ADMIN"}  ],  "admin":true  } |
| 204 |  | Вернется, если такого пользователя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Actions**

**/rest/bsu/mmf/user/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "login":"yasvedko",  "password":"12345",  "admin":true  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет создан |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/{userId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id": 20,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "login":"yasvedko",  "password":"12345",  "admin":true  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет изменен |
| 204 | Вернется, если такого пользователя не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/{userId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет удален |
| 204 | Вернется, если такого пользователя не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пользователей (ресурс **Array[]** **users**) |
| 401 |  | Вернется, если пользователь не авторизован |

#### 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 (Рисунки 14-16) и pdf. Добавив ссылки на макеты, с помощью экспорта в pdf можно получить интерактивный pdf-файл. (смотри приложение).

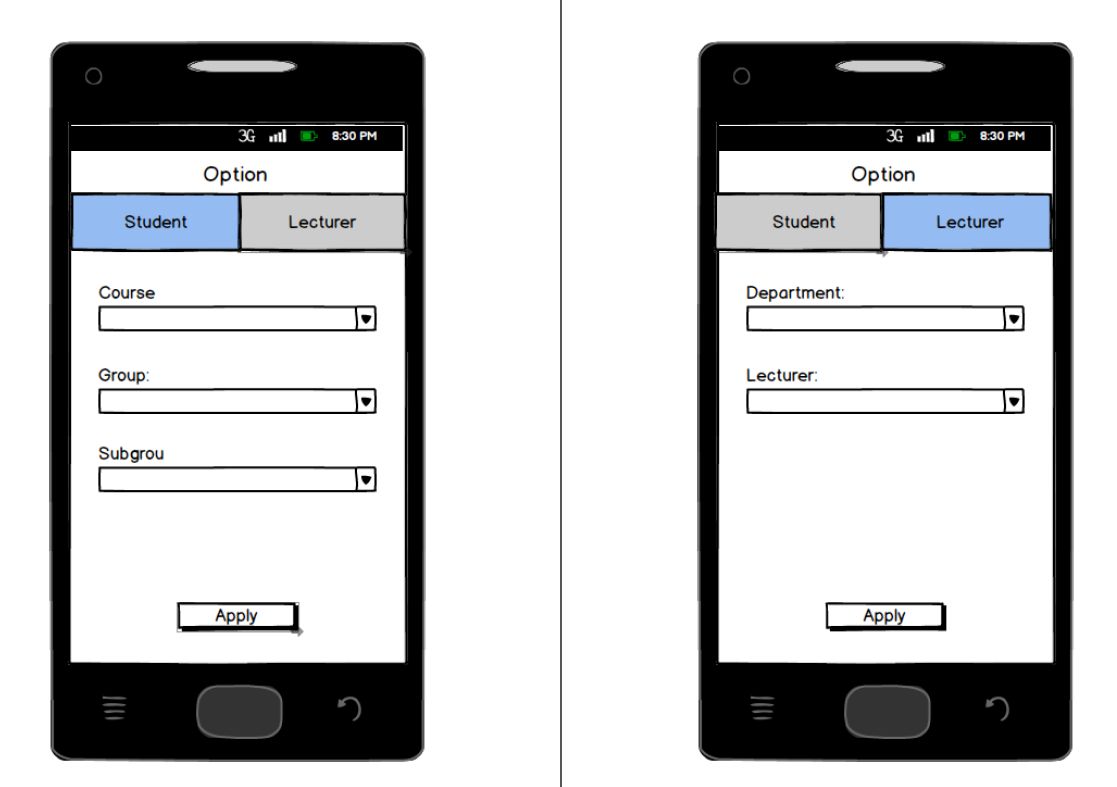
****

Рисунок 14. Экран настроек.

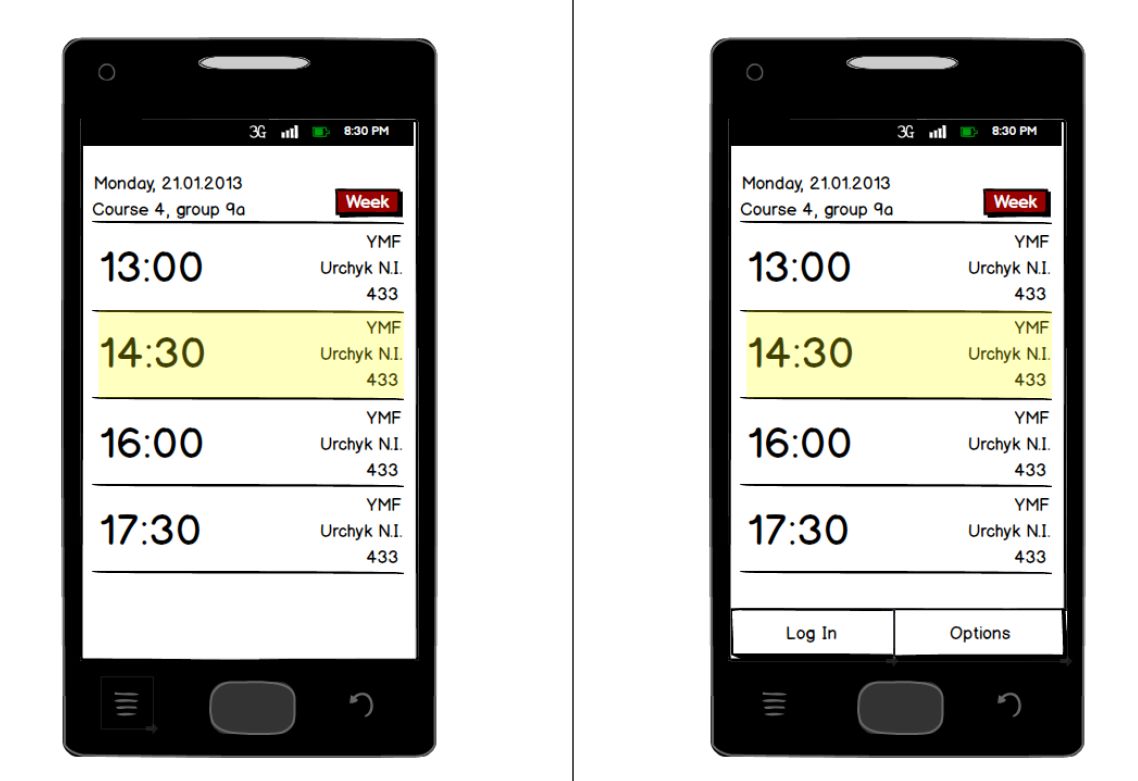


Рисунок 15. Режим просмотра расписания на день.

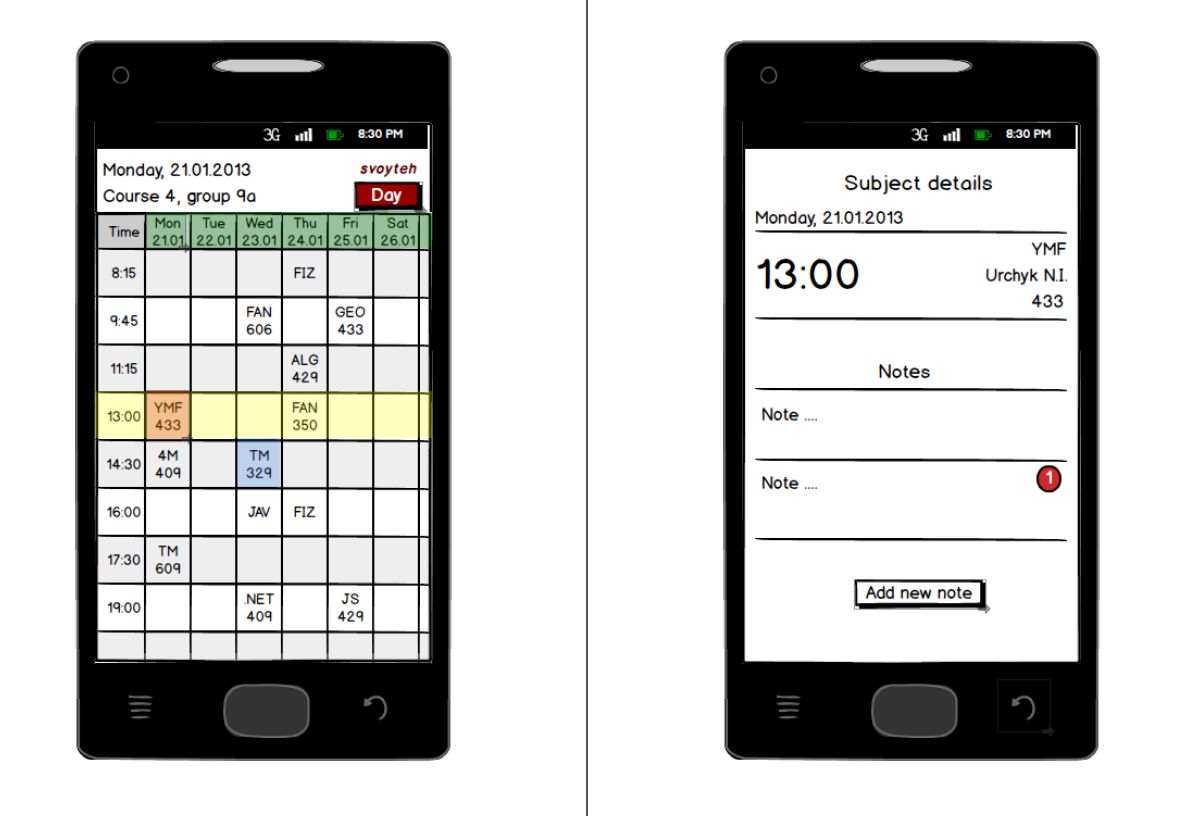


Рисунок 16. Режим просмотра расписания на неделю. Добавление заметок.

#### 2.1.5. Описание функционала

При первом запуске приложения требуется доступ к интернету. После приложение может работать в офлайн режиме.

Запустив приложение впервые, пользователь увидит экран с настройками. Пользователь имеет возможность просмотреть расписание, как для студента, так и преподавателя. Для просмотра расписания для студента ему необходимо выбрать интересующие его курс, группу и подгруппу (Рисунок 17). Для просмотра расписания для преподавателя ему необходимо выбрать интересующие его кафедру и преподавателя (Рисунок 18).

После того как пользователь выставил интересующие его настройки, он может нажать на пункт меню «Расписание», расположенный в верхнем тулбаре. Далее, в зависимости от выбранных настроек, откроется экран с расписанием на текущий день. Проводя пальцем слева-направо или справа-налево, можно перелистывать дни текущей недели (Рисунки 19-20).

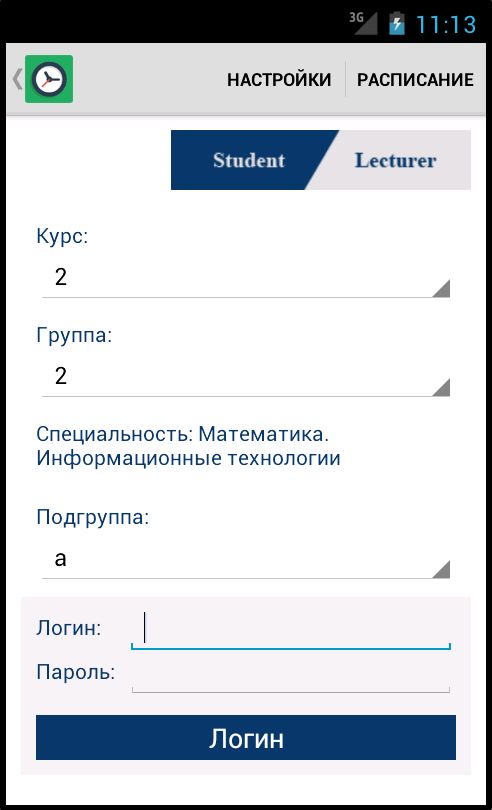


Рисунок 17. Экран настроек для просмотра расписания для студента.

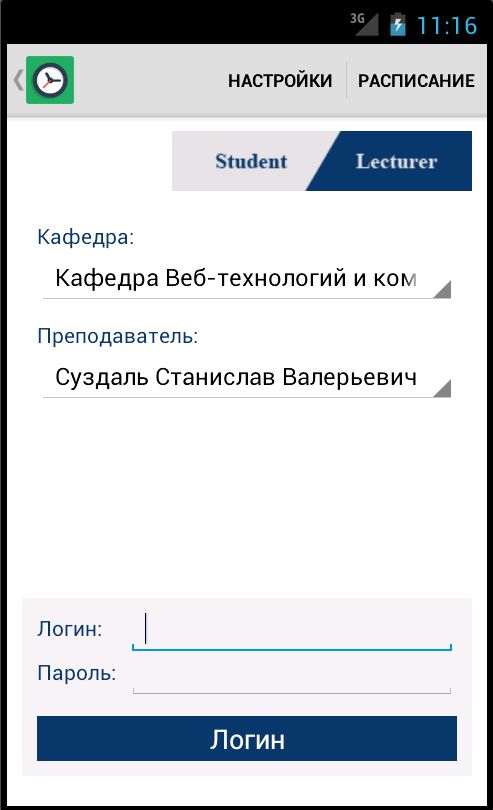


Рисунок 18. Экран настроек для просмотра расписания для преподавателя.

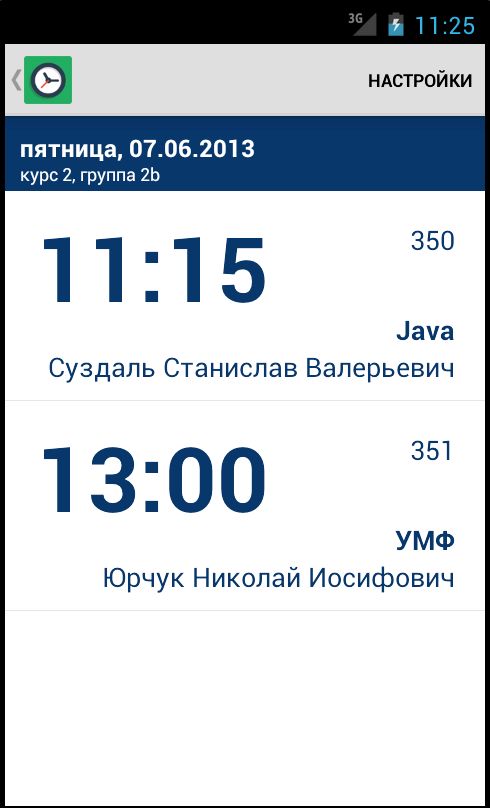


Рисунок 19. Экран с расписанием на день для студента.

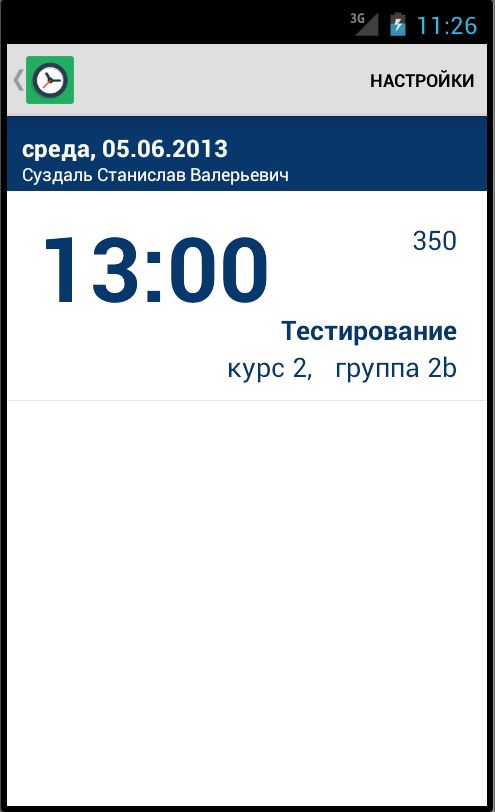


Рисунок 20. Экран с расписанием на день для преподавателя.

Чтобы вернуться к экрану настроек пользователю необходимо нажать на пункт меню «Настройки», расположенный в верхнем тулбаре. Последние выбранные настройки сохраняются в телефоне. Поэтому при следующем открытии приложения будет сразу открываться экран с расписанием на текущий день для студента или преподавателя, в зависимости от последних настроек.

Также пользователь обладает возможностью оставлять пометки на конкретных предметах в расписании. Для этого ему сначала необходимо залогинется. Форма логина расположена на экране настроек (Рисунки 17-18). Удачно залогиневшись, пользователь увидит немного изменившийся экран настроек (Рисунок 21). Пользователь будет оставаться залогиненым до того момента пока не нажмет на кнопку «Выход» или пока не будут стерты данные приложения.

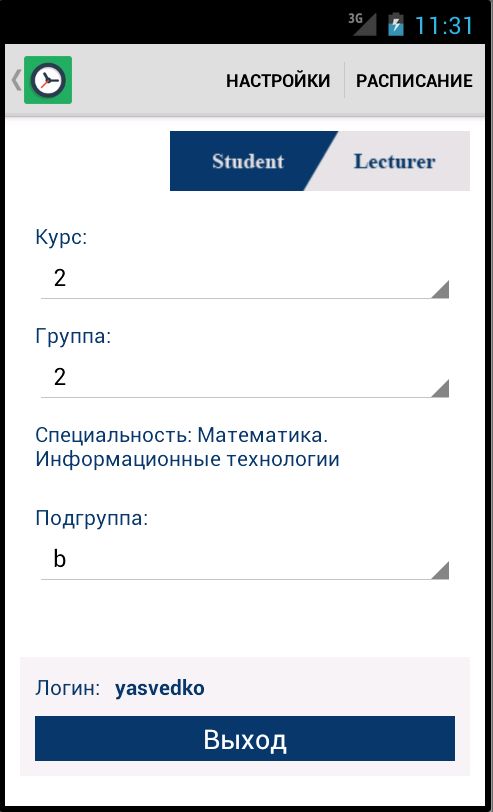


Рисунок 21. Экран настроек для залогиневшегося пользователя.

Для залогиневшегося пользователя на экране с расписанием в правом верхнем углу будет отображаться его логин (Рисунок 22).

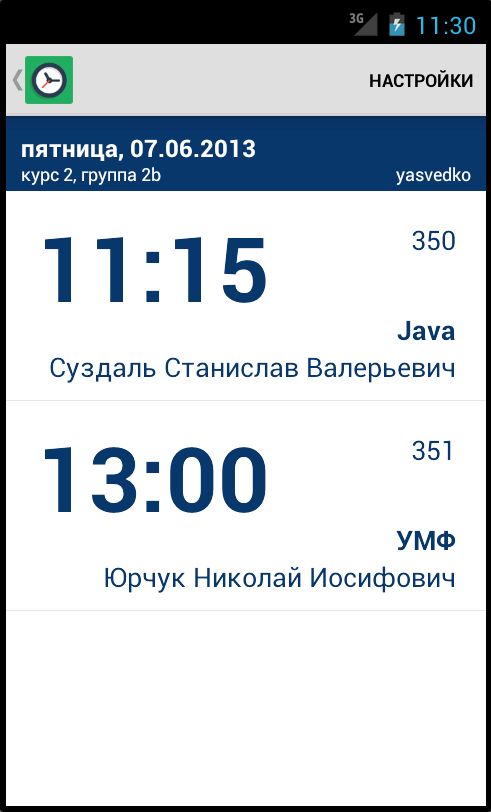


Рисунок 22. Экран с расписанием на день для студента

для залогиневшегося пользователя.

После того как пользователь залогинется, для того, чтобы оставить пометку, ему необходимо выбрать интересующий его предмет. В результате откроется экран для добавления пометок (Рисунок 23). Пользователю необходимо ввести текст пометки и нажать на кнопку «Добавить». Пользователь может видеть свои пометки, а также публичные пометки оставленные старостами групп или преподавателями.

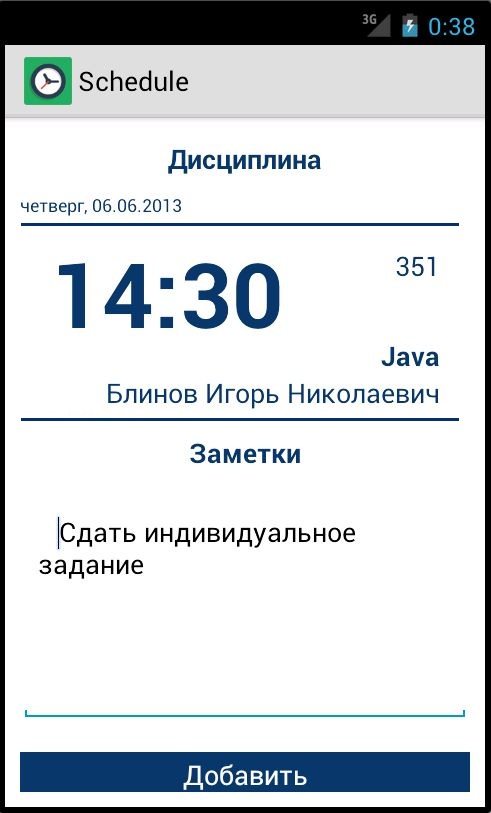


Рисунок 23. Экран для добавления пометок для учебной дисциплины.

### 2.2. Методология тестирования.

В дипломной работе рассмотрены два способа тестирования:

* Ручное тестирование;
* Автоматизированное тестирование.

Ручное тестирование было применено для проверки работоспособности веб-сервисов. Для этого использовалась ПО Fiddler (Рисунок 24).  Fiddler – прокси, который работает с трафиком между вашим компьютером и удаленным сервером. Он позволяет просматривать и менять запросы и их заголовки, cookie, параметры, передаваемые на сервер.



Рисунок 24. Сервис тестирования Fiddler.

Для тестирования андроид приложения была использована библиотека Robotium (Рисунок 25). Robotium – бесплатный общедоступный фреймворк для автоматизированного тестирования андроид приложений методом Blackbox. Он включает в себя большой перечень проверок, а также огромные возможности взаимодействия я приложением. Robotium – это как Selenium, только для Android.



Рисунок 25. Библиотека Robotium для автоматизированного тестирования андроид приложений.

Рассмотрим использование библиотеки Robotium на примере. Ниже приведен код теста просмотра расписания для студента.

**import** android.test.ActivityInstrumentationTestCase2;

**import** com.jayway.android.robotium.solo.Solo;

**import** com.mmf.R;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.Locale;

**public** **class** OptionActivityTest **extends** ActivityInstrumentationTestCase2<BaseActivity> {

**private** **static** **final** String COURSE = "2";

**private** **static** **final** String GROUP = "2";

**private** **static** **final** String SUBGROUP = "b";

**public** OptionActivityTest() {

**super**(BaseActivity.class);

}

**private** Solo solo;

**@Override**

**protected** void setUp() **throws** Exception {

solo = **new** Solo(getInstrumentation(), getActivity());

}

**@Override**

**protected** void tearDown() **throws** Exception {

solo.finishOpenedActivities();

}

**public** void testStudentScheduleView(){

solo.clickOnView(solo.getView(R.id.course\_spinner));

solo.clickOnText(COURSE);

solo.clickOnView(solo.getView(R.id.group\_spinner));

solo.clickOnText(GROUP);

solo.clickOnView(solo.getView(R.id.subgroup\_spinner));

solo.clickOnText(SUBGROUP);

solo.clickOnActionBarItem(R.id.menu\_schedule);

solo.assertCurrentActivity(

"Expected Lesson activity",

LessonActivity.class.getSimpleName());

String date = getDate();

boolean actual = solo.waitForText(date);

assertTrue(actual);

}

**private** String getDate() {

String date = "";

Calendar calendar = Calendar.getInstance();

SimpleDateFormat dateFormat = **new** SimpleDateFormat(

"EEEE, dd.MM.yyyy",

**new** Locale("ru", "RU"));

**if** (calendar.get(Calendar.DAY\_OF\_WEEK) == 1) { // if currentDay == Sunday

calendar.add(Calendar.DATE, 1); // +1 day to Monday

date = dateFormat.format(calendar.getTime());

} **else** {

date = dateFormat.format(new Date());

}

**return** date;

}

}

# ГЛАВА 3 ОРГАНИЗАЦИОННАЯ ЧАСТЬ

### 3.1. Перспективы развития

В ближайшем будущем планируется внедрить систему на факультете. Сервер предположительно будет находиться на территории Механико-Математического Факультета Белорусского Государственного Университета. Тестовое внедрение системы будет проходить на этом же факультете.

Студенты и преподаватели получат возможность быстро и легко узнавать об изменениях в учебном расписании. Для этого им всего лишь необходимо установить андроид-приложение на свои мобильные устройства.

В будущем планируется добавить следующий функционал:

* Возможность просмотра расписания на неделю/месяц;
* Возможность просмотра расписания консультаций и экзаменов во время сессии;
* Возможность прикреплять файлы, например, фотографии конспектов;
* Предоставить пользователям возможность добавлять различного рода информацию о студентах и преподавателях;
* Добавить вкладку «Избранное» для сохранения расписания наиболее часто просматриваемых групп и преподавателей;
* Создать удобный и информативный виджет на мобильные устройства.

### 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 JPA, Spring, JAX-RS.

В итоге была спроектирована архитектура серверной и клиентской частей. Была разработана серверная часть с использованием технологий Java EE. Было разработано андроид-приложение, позволяющее просматривать расписание занятий, как в режиме онлайн, так и офлайн. Также реализована возможность 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>

# ПРИЛОЖЕНИЕ

## Серверная часть

1. **Отображение модели Schedule на таблицу базы данных, используя аннотации JPA**

**import** javax.persistence.\*;

**import** java.util.HashSet;

**import** java.util.Set;

**@Entity**

**@Table**(name = "schedule")

**public** **class** ScheduleEntity **implements** EntityClass<Long>{

**private** **static** **final** long serialVersionUID = 133698310763260308L;

**private** Long id;

**private** ClassroomEntity classroom;

**private** DisciplineTimeEntity disciplineTime;

**private** StudyEntity study;

**private** Integer dayOfWeek;

**private** Integer everyNWeek;

**private** Integer week;

**private** Set<NoteEntity> notes = **new** HashSet<NoteEntity>();

**@Id**

**@GeneratedValue**(strategy = GenerationType.AUTO)

**@Override**

**public** Long getId() {

**return** id;

}

**@Override**

**public** void setId(Long id) {

**this**.id = id;

}

**@ManyToOne**

**@JoinColumn**(name = "IdClassroom")

**public** ClassroomEntity getClassroom() {

**return** classroom;

}

**public** void setClassroom(ClassroomEntity classroom) {

**this**.classroom = classroom;

}

**@ManyToOne**

**@JoinColumn**(name = "IdDisciplineTime")

**public** DisciplineTimeEntity getDisciplineTime() {

**return** disciplineTime;

}

**public** void setDisciplineTime(DisciplineTimeEntity disciplineTime) {

**this**.disciplineTime = disciplineTime;

}

**@ManyToOne**

**@JoinColumn**(name = "IdStudy")

**public** StudyEntity getStudy() {

**return** study;

}

**public** void setStudy(StudyEntity study) {

**this**.study = study;

}

**public** Integer getDayOfWeek() {

**return** dayOfWeek;

}

**public** void setDayOfWeek(Integer dayOfWeek) {

**this**.dayOfWeek = dayOfWeek;

}

**public** Integer getEveryNWeek() {

**return** everyNWeek;

}

**public** void setEveryNWeek(Integer everyNWeek) {

**this**.everyNWeek = everyNWeek;

}

**public** Integer getWeek() {

**return** week;

}

**public** void setWeek(Integer week) {

**this**.week = week;

}

**@OneToMany**(mappedBy = "schedule", fetch = FetchType.LAZY)

**public** Set<NoteEntity> getNotes() {

**return** notes;

}

**public** void setNotes(Set<NoteEntity> notes) {

**this**.notes = notes;

}

}

1. **Бизнес уровень**

**import** com.mmf.business.BusinessServiceException;

**import** com.mmf.business.ScheduleService;

**import** com.mmf.business.domain.Schedule;

**import** com.mmf.business.domain.utils.\*;

**import** com.mmf.db.dao.\*;

**import** com.mmf.db.model.\*;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.transaction.annotation.Transactional;

**import** javax.inject.Named;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Calendar;

**import** java.util.List;

**import** java.util.Locale;

**@Named**

**public** **class** ScheduleServiceImpl **extends** AbstractCrudService<Long, Schedule, ScheduleEntity, ScheduleDao> **implements** ScheduleService {

**@Autowired**

**private** ScheduleDao scheduleDao;

**@Autowired**

**private** ClassroomDao classroomDao;

**@Autowired**

**private** DisciplineTimeDao disciplineTimeDao;

**@Autowired**

**private** StudyDao studyDao;

**@Override**

**protected** ScheduleDao getDao() {

**return** scheduleDao;}

**@Override**

**public** void convertToEntity(Schedule domain, ScheduleEntity entity)

**throws** BusinessServiceException {

**if** (domain != **null**) {

**try** {

ScheduleHelper.convertToEntity(domain, entity);

ClassroomEntity classroomEntity = classroomDao.

getEntityInstance(domain.getClassroomId());

**if** (classroomEntity == **null**){

**throw** **new** BusinessServiceException("Such classroom doesn't exist.");

}

**if** (entity != **null**){

entity.setClassroom(classroomEntity);

}

DisciplineTimeEntity disciplineTimeEntity = disciplineTimeDao.

getEntityInstance(domain.getDisciplineTimeId());

**if** (disciplineTimeEntity == **null**){

**throw** **new** BusinessServiceException("Such disciplineTime doesn't exist.");

}

**if** (entity != **null**){

entity.setDisciplineTime(disciplineTimeEntity);

}

StudyEntity studyEntity = studyDao.getEntityInstance(domain.getStudyId());

**if**(studyEntity == **null**){

**throw** **new** BusinessServiceException("Such study doesn't exist.");

}

**if** (entity != **null**){

entity.setStudy(studyEntity);

}

} **catch** (DataAccessException e) {

**throw** **new** BusinessServiceException("Conversion to group entity error.", e);

}

}

}

**@Override**

**public** Schedule convertToDomain(ScheduleEntity entity)

**throws** BusinessServiceException {

**if** (entity == **null**) {

**return** **null**;

}

Schedule schedule = ScheduleHelper.convertToDomain(entity);

schedule.setClassroom(ClassroomHelper.convertToDomain(entity.getClassroom()));

schedule.setDisciplineTime(DisciplineTimeHelper.

convertToDomain(entity.getDisciplineTime()));

schedule.setStudy(StudyHelper.convertToDomain(entity.getStudy()));

schedule.setGroup(GroupHelper.convertToDomain(entity.getStudy().getGroup()));

schedule.setLecturer(LecturerHelper.

convertToDomain(entity.getStudy().getLecturer()));

schedule.setDiscipline(DisciplineHelper.

convertToDomain(entity.getStudy().getCurriculum().getDiscipline()));

**for** (NoteEntity noteEntity : entity.getNotes()){

schedule.getNotes().add(NoteHelper.convertToDomain(noteEntity));

}

**return** schedule;

}

**@Override**

**@Transactional**(rollbackFor = BusinessServiceException.class)

**public** List<Schedule> getSchedule(int semester, int yearOfEntrance,

String groupName, String subGroupName)

**throws** BusinessServiceException {

List<Schedule> responseList = **new** ArrayList<Schedule>();

List<ScheduleEntity> scheduleList = **new** ArrayList<ScheduleEntity>();

**for** (int i = 2; i <= 7; i++) {

scheduleList.addAll(scheduleDao.getScheduleForDay(

semester, yearOfEntrance, groupName, subGroupName, i));

}

**if** (scheduleList.isEmpty()) {

**return** responseList;

}

**for** (ScheduleEntity entity : scheduleList) {

responseList.add(convertToDomain(entity));

}

**return** responseList;

}

**@Override**

**@Transactional**(rollbackFor = BusinessServiceException.class)

**public** List<Schedule> getSchedule(long lecturerId, int semester)

**throws** BusinessServiceException {

List<Schedule> responseList = **new** ArrayList<Schedule>();

List<ScheduleEntity> scheduleList = **new** ArrayList<ScheduleEntity>();

**for** (int i = 2; i <= 7; i++) {

scheduleList.addAll(scheduleDao.getScheduleForDay(semester, lecturerId, i));

}

**if** (scheduleList.isEmpty()) {

**return** responseList;

}

**for** (ScheduleEntity entity : scheduleList) {

Schedule schedule = convertToDomain(entity);

setDay(schedule);

responseList.add(schedule);

}

**return** responseList;

}

**private** void setDay(Schedule response) {

Calendar calendar = Calendar.getInstance();

calendar.set(Calendar.DAY\_OF\_WEEK, response.getDayOfWeek());

String dayTitle = **new** SimpleDateFormat(

"EEEE", new Locale("ru", "RU")).format(calendar.getTime());

response.setDayTitle(dayTitle.substring(0, 1).toUpperCase() + dayTitle.substring(1));

}

}

1. **REST сервис**

**import** com.mmf.business.BusinessServiceException;

**import** com.mmf.business.LecturerService;

**import** com.mmf.business.ScheduleService;

**import** com.mmf.business.domain.Schedule;

**import** com.mmf.rest.response.schedule.ScheduleGroupResponse;

**import** com.mmf.rest.response.schedule.ScheduleResponse;

**import** com.mmf.rest.util.DomainUtil;

**import** com.mmf.rest.util.NullPropertyException;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** javax.ws.rs.\*;

**import** javax.ws.rs.core.MediaType;

**import** javax.ws.rs.core.Response;

**import** java.util.Calendar;

**import** java.util.LinkedList;

**import** java.util.List;

**@Service**

**@Path**("schedule")

**public** **class** ScheduleResource **extends** CrudResource<Schedule, ScheduleService>{

**@Autowired**

**private** ScheduleService scheduleService;

**@Autowired**

**private** LecturerService lecturerService;

**@Override**

**protected** ScheduleService getService() {

**return** scheduleService;

}

**@Override**

**protected** void validate(Schedule domain) {

**try** {

DomainUtil.checkingForNotNull(domain.getWeek());

DomainUtil.checkingForNotNull(domain.getDayOfWeek());

DomainUtil.checkingForNotNull(domain.getStudyId());

DomainUtil.checkingForNotNull(domain.getClassroomId());

DomainUtil.checkingForNotNull(domain.getDisciplineTimeId());

} **catch** (NullPropertyException e) {

**throw** **new** RestServiceException(

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

}

}

**@Override**

**protected** void updateFields(Schedule domain, Schedule newDomain) {

domain.setDayOfWeek(newDomain.getDayOfWeek());

domain.setWeek(newDomain.getWeek());

domain.setClassroomId(newDomain.getClassroomId());

domain.setDisciplineTimeId(newDomain.getDisciplineTimeId());

domain.setStudyId(newDomain.getStudyId());

}

**@Override**

**@GET**

**@Path**("/{id}")

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

**public** Response get(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(**new** ScheduleResponse(domain)).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@Override**

**@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());

}

}

**@GET**

**@Path**("/{id}/classroom")

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

**public** Response getClassroom(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getClassroom()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/study")

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

**public** Response getStudy(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getStudy()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/disciplineTime")

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

**public** Response getDisciplineTime(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getDisciplineTime()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/notes")

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

**public** Response getNotes(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getNotes()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/group")

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

**public** Response getGroup(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getGroup()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/lecturer")

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

**public** Response getLecturer(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getLecturer()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

}

}

**@GET**

**@Path**("/{id}/discipline")

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

**public** Response getDiscipline(**@PathParam**("id") long id){

**try** {

Schedule domain = getService().get(id);

DomainUtil.checkingForNotNull(domain);

**return** Response.ok(domain.getDiscipline()).

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

} **catch** (NullPropertyException e) {

**return** Response.noContent().build();

} }

**@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);

}

}

**private** Response getScheduleForStudent(int course, int group, String subGroup) {

int currentYear = Calendar.getInstance().get(Calendar.YEAR);

int currentMonth = Calendar.getInstance().get(Calendar.MONTH);

int semester;

int yearOfEntrance;

**if** (currentMonth < Calendar.JULY) {

semester = course \* 2;

yearOfEntrance = currentYear - course;

} **else** {

semester = course \* 2 - 1;

yearOfEntrance = currentYear - course + 1;

}

String subGroupName = "".equals(subGroup) ?

subGroup : String.valueOf(group) + subGroup;

**try** {

List<Schedule> scheduleList = scheduleService.getSchedule(

semester, yearOfEntrance, String.valueOf(group), subGroupName);

List<ScheduleResponse> scheduleResponseList =

**new** LinkedList<ScheduleResponse>();

ScheduleGroupResponse groupResponse = **new** ScheduleGroupResponse();

int currentWeek = Calendar.getInstance().get(Calendar.WEEK\_OF\_YEAR)%2;

groupResponse.setCurrentWeek(currentWeek == 0 ? 2 : currentWeek);

int day = 2;

**for** (Schedule response : scheduleList) {

**if** (day != response.getDayOfWeek()){

groupResponse.setSchedule(scheduleResponseList, day);

scheduleResponseList.clear();

day = response.getDayOfWeek();

}

response.setLecturer(lecturerService.get(response.getLecturerId()));

scheduleResponseList.add(**new** ScheduleResponse(response));

}

groupResponse.setSchedule(scheduleResponseList, day);

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

}

}

**private** Response getScheduleForLecturer(long lecturerId) {

**try** {

int currentMonth = Calendar.getInstance().get(Calendar.MONTH);

int semester;

**if** (currentMonth < Calendar.JULY) {

semester = 0;

} **else** {

semester = 1;

}

List<Schedule> scheduleList = scheduleService.getSchedule(lecturerId, semester);

List<ScheduleResponse> scheduleResponseList =

**new** LinkedList<ScheduleResponse>();

ScheduleGroupResponse groupResponse = **new** ScheduleGroupResponse();

int currentWeek = Calendar.getInstance().get(Calendar.WEEK\_OF\_YEAR)%2;

groupResponse.setCurrentWeek(currentWeek == 0 ? 2 : currentWeek);

int day = 2;

**for** (Schedule response : scheduleList) {

**if** (day != response.getDayOfWeek()){

groupResponse.setSchedule(scheduleResponseList, day);

scheduleResponseList.clear();

day = response.getDayOfWeek();

}

response.setLecturer(lecturerService.get(response.getLecturerId()));

scheduleResponseList.add(**new** ScheduleResponse(response));

}

groupResponse.setSchedule(scheduleResponseList, day);

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

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(e.getErrorCode());

}

}

}

1. **Настройка Application Spring Context**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:security="http://www.springframework.org/schema/security"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/security http://www.springframework.org/schema/security/spring-security.xsd">

<context:annotation-config/>

<context:component-scan base-package="

com.mmf.business,

com.mmf.db,

com.mmf.rest"/>

<tx:annotation-driven transaction-manager="transactionManager"/>

<bean name="transactionManager" class=

"org.springframework.orm.jpa.JpaTransactionManager">

<property name="entityManagerFactory" ref="entityManagerFactory"/>

</bean>

<bean name="entityManagerFactory" class=

"org.springframework.orm.jpa.LocalEntityManagerFactoryBean">

<property name="persistenceUnitName" value="persistence"/>

</bean>

<bean class=

"org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor"/>

<!-- Spring security settings-->

<bean id="customUserDetailsService" class=

"com.mmf.rest.service.CustomUserDetailsService">

</bean>

<security:authentication-manager alias="auth-manager">

<security:authentication-provider user-service-ref="customUserDetailsService">

<security:password-encoder hash="sha">

<security:salt-source user-property="passwordSalt" />

</security:password-encoder>

</security:authentication-provider>

</security:authentication-manager>

<bean id="passwordGenerator" class="com.mmf.rest.util.PasswordGenerator">

<property name="passwordEncoder" ref="passwordEncoder"/>

<property name="saltSource" ref="saltSource"/>

</bean>

<bean id="saltSource" class="org.springframework.security.authentication.dao.ReflectionSaltSource">

<property name="userPropertyToUse" value="passwordSalt" />

</bean>

<bean id="passwordEncoder" class="org.springframework.security.authentication.encoding.ShaPasswordEncoder" />

<bean id="daoAuthenticationProvider" class="org.springframework.security.authentication.dao.DaoAuthenticationProvider">

<property name="userDetailsService" ref="customUserDetailsService"/>

<property name="saltSource" ref="saltSource"/>

<property name="passwordEncoder" ref="passwordEncoder"/>

</bean>

<bean id="authenticationManager" class="org.springframework.security.authentication.ProviderManager">

<property name="providers">

<list>

<ref local="daoAuthenticationProvider"/>

</list>

</property>

</bean>

<security:http pattern="/rest/bsu/mmf/\*\*" create-session="never" auto-config="true" use-expressions="true">

<security:intercept-url pattern="/rest/bsu/mmf/login" access="hasAnyRole('ROLE\_USER')" />

<security:intercept-url pattern="/rest/bsu/mmf/group/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/group/\*/specialty" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/group/\*/students" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/group/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/specialty/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/specialty/\*/groups" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/specialty/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/department/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/department/\*/lecturers" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/department/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/discipline/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/discipline/\*/disciplineType" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/discipline/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/disciplineType/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/disciplineType/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/disciplineTime/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/disciplineTime/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/curriculum/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/curriculum/\*/discipline" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/curriculum/\*/specialty" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/curriculum/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/classroom/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/classroom/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/study/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/study/\*/lecturer" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/study/\*/group" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/study/\*/curriculum" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/study/list" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule/\*" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule/\*/classroom" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule/\*/study" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule/\*/disciplineTime" access="permitAll" />

<security:intercept-url pattern="/rest/bsu/mmf/schedule/list" access="permitAll" />

<!--<security:intercept-url pattern="/rest/bsu/mmf/\*\*" access="permitAll" />-->

<security:intercept-url pattern="/rest/bsu/mmf/\*\*" access="hasAnyRole('ROLE\_ADMIN')" />

<security:http-basic />

</security:http>

<security:global-method-security secured-annotations="enabled" />

</beans>

1. **Spring Security сервис**

**import** com.mmf.business.BusinessServiceException;

**import** com.mmf.business.LecturerService;

**import** com.mmf.business.StudentService;

**import** com.mmf.business.UserService;

**import** com.mmf.business.domain.User;

**import** com.mmf.rest.RestServiceException;

**import** org.apache.commons.httpclient.HttpStatus;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.core.GrantedAuthority;

**import** org.springframework.security.core.authority.SimpleGrantedAuthority;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.core.userdetails.UsernameNotFoundException;

**import** java.util.ArrayList;

**import** java.util.Collection;

**import** java.util.LinkedList;

**public** **class** CustomUserDetailsService **implements** UserDetailsService {

**@Autowired**

**private** UserService userService;

**@Autowired**

**private** StudentService studentService;

**@Autowired**

**private** LecturerService lecturerService;

**@Override**

**public** UserDetails loadUserByUsername(String username)

**throws** UsernameNotFoundException {

User user;

**try** {

user = userService.getUser(username);

**if** (user == **null**) {

**throw** **new** UsernameNotFoundException(null);

}

**return** buildUserFromUserEntity(user);

} **catch** (BusinessServiceException e) {

**throw** **new** RestServiceException(HttpStatus.SC\_UNAUTHORIZED);

}

}

**private** UserDetails buildUserFromUserEntity(User user)

**throws** BusinessServiceException {

// Add user role access rights

Collection<GrantedAuthority> authorities = **new** ArrayList<GrantedAuthority>();

**if** (studentService.get(user.getId()) != **null**) {

((LinkedList<SimpleGrantedAuthority>) user.getAuthorities()).

add(**new** SimpleGrantedAuthority("ROLE\_STUDENT"));

}

**if** (lecturerService.get(user.getId()) != **null**) {

((LinkedList<SimpleGrantedAuthority>) user.getAuthorities()).

add(**new** SimpleGrantedAuthority("ROLE\_LECTURER"));

}

**if** (user.getAdmin()) {

((LinkedList<SimpleGrantedAuthority>) user.getAuthorities()).

add(**new** SimpleGrantedAuthority("ROLE\_ADMIN"));

}

((LinkedList<SimpleGrantedAuthority>) user.getAuthorities()).

add(**new** SimpleGrantedAuthority("ROLE\_USER"));

**return** user;

}

}

## Клиентская часть

* 1. **Создание и обновление базы данных**

**import** android.content.Context;

**import** android.database.sqlite.SQLiteDatabase;

**import** android.database.sqlite.SQLiteOpenHelper;

**import** android.util.Log;

**import** com.mmf.util.EntityRegistry;

**import** com.mmf.db.dao.AbstractEntityDao;

**import** com.mmf.db.dao.utils.Column;

**import** com.mmf.db.dao.utils.Table;

**import** java.util.List;

**public** **class** DatabaseConnector **extends** SQLiteOpenHelper **implements** DBVersions {

**public** **static** **final** String TAG = "DatabaseConnector";

**public** **static** **final** String DATABASE\_NAME = "scheduleDB";

**public** DatabaseConnector(Context context) {

**super**(context, DATABASE\_NAME, **null**, CURRENT\_VERSION);

}

**@Override**

**public** void onCreate(SQLiteDatabase db) {

initDb(db, CURRENT\_VERSION, CURRENT\_VERSION);

}

**@Override**

**public** void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

**if** (oldVersion == DATABASE\_VERSION\_1) {

List<AbstractEntityDao<?>> entityDaoList = EntityRegistry.get().getEntityDaoList();

**for** (AbstractEntityDao<?> dao : entityDaoList) {

db.execSQL("DROP TABLE IF EXISTS " + dao.getTable());

}

onCreate(db);

} **else** {

initDb(db, oldVersion, newVersion);

}

}

void initDb(SQLiteDatabase db, int oldVersion, int newVersion) {

**if** (oldVersion == CURRENT\_VERSION && newVersion == CURRENT\_VERSION) {

Log.d(TAG, String.format("Create new database v.%d", newVersion));

List<AbstractEntityDao<?>> entityDaoList = EntityRegistry.get().getEntityDaoList();

for (AbstractEntityDao<?> dao : entityDaoList) {

createTable(db, dao.getTable());

}

} **else** **if** (oldVersion < CURRENT\_VERSION) {

Log.d(TAG, String.format("Create update database from v.%d to v.%d", oldVersion, newVersion));

List<AbstractEntityDao<?>> entityDaoList = EntityRegistry.get().getEntityDaoList();

for (AbstractEntityDao<?> dao : entityDaoList) {

Table table = dao.getTable();

**if** (table.getInDbSinceVersion() <= oldVersion) {

updateTable(db, dao.getTable(), oldVersion, newVersion);

} **else** {

createTable(db, table);

}

}

}

}

**public** void createTable(SQLiteDatabase db, Table table) {

String script = table.getCreationScript();

Log.d(TAG, script);

db.execSQL(script);

}

**public** void updateTable(SQLiteDatabase db, Table table, int fromVersion, int toVersion) {

StringBuilder script = **new** StringBuilder();

String tableName = table.getName();

List<Column> columns = table.getColumns(fromVersion + 1, toVersion);

**for** (Column column : columns) {

script.setLength(0);

script.append("ALTER TABLE ").append(tableName).append(

" ADD COLUMN ").append(column.name()).append(" ")

.append(column.definition()).append(';');

Log.d(TAG, script.toString());

db.execSQL(script.toString());

}

}

}

* 1. **Вызов веб-сервисов**

**import** android.util.Log;

**import** com.google.gson.Gson;

**import** com.google.gson.GsonBuilder;

**import** com.google.gson.reflect.TypeToken;

**import** com.mmf.db.model.Schedule;

**import** com.mmf.rest.deserializer.InitialDataDeserializer;

**import** com.mmf.rest.deserializer.ScheduleDeserializer;

**import** com.mmf.rest.domain.InitialData;

**import** com.mmf.rest.exceptions.RestException;

**import** com.mmf.rest.exceptions.UnexpectedResponseCodeException;

**import** com.mmf.prefs.CredentialsPrefs;

**import** com.mmf.util.Logger;

**import** org.apache.http.HttpEntity;

**import** org.apache.http.HttpResponse;

**import** org.apache.http.HttpStatus;

**import** org.apache.http.auth.AuthenticationException;

**import** org.apache.http.auth.InvalidCredentialsException;

**import** org.apache.http.auth.UsernamePasswordCredentials;

**import** org.apache.http.client.ClientProtocolException;

**import** org.apache.http.client.methods.HttpGet;

**import** org.apache.http.impl.auth.BasicScheme;

**import** org.apache.http.impl.client.DefaultHttpClient;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.lang.reflect.Type;

**import** java.net.MalformedURLException;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** RestRequester {

**private** **static** **final** String REST\_API = "rest/bsu/mmf/";

**private** **static** **final** String TAG = "RestRequester";

**public** **static** **final** String SERVER\_HTTP\_DEV = "http://192.168.0.2:8080/";

**private** **static** InputStreamReader getReader(String apiUrl)

**throws** InvalidCredentialsException, RestException {

InputStreamReader inputStreamReader = **null**;

**try** {

DefaultHttpClient httpClient = **new** DefaultHttpClient();

HttpGet request = **new** HttpGet(SERVER\_HTTP\_DEV + apiUrl);

request.setHeader(new BasicScheme().authenticate

(**new** UsernamePasswordCredentials(

CredentialsPrefs.LoginDefault.get(), CredentialsPrefs.PasswordDefault.get()),

request));

request.setHeader("Content-type", "application/json");

request.setHeader("Accept-Encoding", "utf-8");

HttpResponse response = httpClient.execute(request);

int responseCode = response.getStatusLine().getStatusCode();

**if** (responseCode == HttpStatus.SC\_UNAUTHORIZED) {

**throw** **new** InvalidCredentialsException("You unauthorized to use service");

}

**if** (responseCode != HttpStatus.SC\_OK) {

**throw** **new** UnexpectedResponseCodeException(responseCode);

}

HttpEntity entity = response.getEntity();

inputStreamReader = new InputStreamReader(entity.getContent());

**return** inputStreamReader;

} **catch** (MalformedURLException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} catch (ClientProtocolException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} catch (IOException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} catch (AuthenticationException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

}

}

**public** **static** boolean login(String login, String password)

**throws** InvalidCredentialsException, RestException {

**try** {

DefaultHttpClient httpClient = **new** DefaultHttpClient();

HttpGet request = **new** HttpGet(SERVER\_HTTP\_DEV + REST\_API + "login");

request.setHeader(**new** BasicScheme().authenticate(

**new** UsernamePasswordCredentials(login, password), request));

request.setHeader("Content-type", "application/json");

request.setHeader("Accept-Encoding", "utf-8");

HttpResponse response = httpClient.execute(request);

int responseCode = response.getStatusLine().getStatusCode();

**if** (responseCode == HttpStatus.SC\_UNAUTHORIZED) {

**throw** **new** InvalidCredentialsException("You unauthorized to use service");

}

**if** (responseCode != HttpStatus.SC\_OK) {

**throw** **new** UnexpectedResponseCodeException(responseCode);

}

**return** true;

} **catch** (MalformedURLException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} **catch** (ClientProtocolException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} **catch** (IOException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

} **catch** (AuthenticationException e) {

Log.e(TAG, e.getMessage());

**throw** **new** RestException(e);

}

}

**public** **static** List<Schedule> gesSchedule(int course, int group, String subGroup) **throws** RestException, InvalidCredentialsException {

List<Schedule> scheduleList = new ArrayList<Schedule>();

InputStreamReader inputStreamReader = **null**;

**try**{

StringBuilder params = **new** StringBuilder("?course=");

params.append(course);

params.append("&group=");

params.append(group);

params.append("&subGroup=");

params.append(subGroup);

inputStreamReader = getReader(REST\_API + "schedule" + params.toString());

**if**(inputStreamReader != **null**){

GsonBuilder gsonBuilder = **new** GsonBuilder();

gsonBuilder.registerTypeAdapter(List.class, **new** ScheduleDeserializer());

Gson gson = gsonBuilder.create();

Type listType = **new** TypeToken<List<Schedule>>() {}.getType();

scheduleList = gson.fromJson(inputStreamReader, listType);

}

**return** scheduleList;

} **finally** {

**if** (inputStreamReader != **null**) {

**try** {

inputStreamReader.close();

} **catch** (IOException e) {

Logger.getInstance().error(e);

}

}

}

}

**public** **static** List<Schedule> gesSchedule(long lecturerId)

**throws** RestException, InvalidCredentialsException {

List<Schedule> scheduleList = **new** ArrayList<Schedule>();

InputStreamReader inputStreamReader = **null**;

**try**{

StringBuilder params = **new** StringBuilder("?lecturerId=");

params.append(lecturerId);

inputStreamReader = getReader(REST\_API + "schedule" + params.toString());

**if**(inputStreamReader != **null**){

GsonBuilder gsonBuilder = **new** GsonBuilder();

gsonBuilder.registerTypeAdapter(List.class, **new** ScheduleDeserializer());

Gson gson = gsonBuilder.create();

Type listType = **new** TypeToken<List<Schedule>>() {}.getType();

scheduleList = gson.fromJson(inputStreamReader, listType);

}

**return** scheduleList;

} **finally** {

**if** (inputStreamReader != **null**) {

**try** {

inputStreamReader.close();

} catch (IOException e) {

Logger.getInstance().error(e);

}

}

}

}

**public** **static** InitialData getInitialData()

**throws** InvalidCredentialsException, RestException {

InitialData data = **new** InitialData();

InputStreamReader inputStreamReader = **null**;

try {

inputStreamReader = getReader(REST\_API + "initialData");

**if** (inputStreamReader != **null**) {

GsonBuilder gsonBuilder = **new** GsonBuilder();

gsonBuilder.registerTypeAdapter(List.class, **new** InitialDataDeserializer());

Gson gson = gsonBuilder.create();

Type listType = **new** TypeToken<List<InitialData>>() {}.getType();

data = gson.fromJson(inputStreamReader, listType);

}

**return** data;

} **finally** {

**if** (inputStreamReader != **null**) {

**try** {

inputStreamReader.close();

} **catch** (IOException e) {

Logger.getInstance().error(e);

}

}

}

}

}

* 1. **Парсинг JSON**

**import** com.google.gson.\*;

**import** com.mmf.db.model.Lecturer;

**import** com.mmf.db.model.Schedule;

**import** java.lang.reflect.Type;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** ScheduleDeserializer **implements** JsonDeserializer<List<Schedule>> {

**private** List<String> weekDays = Arrays.asList("monday", "tuesday", "wednesday", "thursday", "friday", "saturday");

**@Override**

**public** List<Schedule> deserialize(JsonElement jsonElement, Type type, JsonDeserializationContext jsonDeserializationContext) **throws** JsonParseException {

JsonObject jsonObject = jsonElement.getAsJsonObject();

List<Schedule> scheduleList = **new** ArrayList<Schedule>();

**for**(String day : weekDays){

JsonArray daySchedule = jsonObject.getAsJsonArray(day);

**for**(JsonElement element : daySchedule){

JsonObject object = element.getAsJsonObject();

Schedule schedule = new Schedule();

schedule.setDay(object.get("dayOfWeek").getAsInt());

schedule.setWeek(object.get("week").getAsInt());

schedule.setClassroom(object.

getAsJsonObject("classroom").get("number").getAsInt());

schedule.setDiscipline(object.

getAsJsonObject("discipline").get("name").getAsString());

JsonObject group = object.getAsJsonObject("group");

schedule.setCourse(group.get("course").getAsInt());

schedule.setGroupNumber(group.get("number").getAsInt());

String subgroup = group.get("subgroup").isJsonNull() ?

"" : group.get("subgroup").getAsString();

schedule.setSubGroup(subgroup);

schedule.setLecturer(**new** Lecturer(

object.getAsJsonObject("lecturer").get("id").getAsLong()));

JsonObject time = object.getAsJsonObject("disciplineTime");

schedule.setNumber(time.get("number").getAsInt());

schedule.setTime(time.get("startTime").getAsString());

scheduleList.add(schedule);

}

}

**return** scheduleList;

}

}

* 1. **Экран «Настройки»**

**import** android.os.AsyncTask;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.\*;

**import** com.mmf.R;

**import** com.mmf.db.model.Department;

**import** com.mmf.db.model.Lecturer;

**import** com.mmf.db.model.Specialty;

**import** com.mmf.prefs.CredentialsPrefs;

**import** com.mmf.prefs.OptionPrefs;

**import** com.mmf.rest.DataLoader;

**import** com.mmf.rest.exceptions.RestException;

**import** com.mmf.service.BusinessLayerException;

**import** com.mmf.service.DepartmentService;

**import** com.mmf.service.LecturerService;

**import** com.mmf.service.SpecialtyService;

**import** com.mmf.util.Logger;

**import** com.mmf.util.SpinnerUtils;

**import** com.mmf.view.ToggleButton;

**import** org.apache.http.auth.InvalidCredentialsException;

**public** **class** OptionActivity **extends** BaseActivity {

**private** Spinner courseSpinner;

**private** Spinner groupSpinner;

**private** Spinner subgroupSpinner;

**private** Spinner lecturerSpinner;

**private** Spinner departmentSpinner;

**private** ArrayAdapter<Integer> courseAdapter;

**private** ArrayAdapter<Integer> groupAdapter;

**private** ArrayAdapter<String> subgroupAdapter;

**private** ArrayAdapter<Lecturer> lecturerAdapter;

**private** ArrayAdapter<Department> departmentAdapter;

**private** DepartmentService departmentService;

**private** LecturerService lecturerService;

**@Override**

**protected** void onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.option);

departmentService = **new** DepartmentService();

lecturerService = **new** LecturerService();

**try** {

initSpinners();

**if**(CredentialsPrefs.IsLogined.get()){

findViewById(R.id.layout\_login).setVisibility(View.INVISIBLE);

findViewById(R.id.layout\_logout).setVisibility(View.VISIBLE);

((TextView)findViewById(R.id.text\_loginName)).

setText(CredentialsPrefs.Login.get());

} **else** {

findViewById(R.id.layout\_login).setVisibility(View.VISIBLE);

findViewById(R.id.layout\_logout).setVisibility(View.INVISIBLE);

}

Button buttonLogin = (Button) findViewById(R.id.button\_login);

buttonLogin.setOnClickListener(**new** View.OnClickListener() {

**@Override**

public void onClick(View v) {

final String login = ((EditText)findViewById(R.id.login)).getText().toString();

final String password = ((EditText)findViewById(R.id.password)).

getText().toString();

**new** AsyncTask<Object, Object, Boolean>(){

**@Override**

**protected** Boolean doInBackground(Object... param) {

**try** {

**return** DataLoader.getInstance().login(login, password);

} **catch** (InvalidCredentialsException e) {

Logger.getInstance().error(e);

} **catch** (RestException e) {

Logger.getInstance().error(e);

}

**return** false;

}

**@Override**

protected void onPostExecute(Boolean isLogin) {

**if** (isLogin){

findViewById(R.id.layout\_login).setVisibility(View.INVISIBLE);

findViewById(R.id.layout\_logout).setVisibility(View.VISIBLE);

CredentialsPrefs.IsLogined.put(true);

CredentialsPrefs.Login.put(login);

CredentialsPrefs.Password.put(password);

((TextView)findViewById(R.id.text\_loginName)).

setText(CredentialsPrefs.Login.get());

} **else** {

Toast.makeText(OptionActivity.this, getString(

R.string.validate\_messages\_incorrect\_login\_or\_password), Toast.LENGTH\_LONG).show();

}

}

}.execute();

}

});

Button buttonLogout = (Button) findViewById(R.id.button\_logout);

buttonLogout.setOnClickListener(**new** View.OnClickListener() {

**@Override**

**public** void onClick(View v) {

CredentialsPrefs.IsLogined.put(false);

CredentialsPrefs.Login.put("");

CredentialsPrefs.Password.put("");

findViewById(R.id.layout\_login).setVisibility(View.VISIBLE);

findViewById(R.id.layout\_logout).setVisibility(View.INVISIBLE);

}

});

toggleButton = (ToggleButton) findViewById(R.id.toggle\_button);

toggleButton.setViews(

findViewById(R.id.layout\_student), findViewById(R.id.layout\_lecturer));

} **catch** (BusinessLayerException ble) {

Logger.getInstance().error(ble);

}

}

**private** void initSpinners() **throws** BusinessLayerException {

courseSpinner = (Spinner) findViewById(R.id.course\_spinner);

courseSpinner.setOnItemSelectedListener(

**new** AdapterView.OnItemSelectedListener() {

**@Override**

**public** void onItemSelected(AdapterView<?> adapterView,

View view, int position, long id) {

OptionPrefs.Course.put((Integer) adapterView.getSelectedItem());

}

**@Override**

**public** void onNothingSelected(AdapterView<?> parent) {

}

});

courseAdapter = SpinnerUtils.getCourseAdapter(this);

courseSpinner.setAdapter(courseAdapter);

courseSpinner.setSelection(courseAdapter.getPosition(OptionPrefs.Course.get()));

groupSpinner = (Spinner) findViewById(R.id.group\_spinner);

groupSpinner.setOnItemSelectedListener(

**new** AdapterView.OnItemSelectedListener() {

**@Override**

**public** void onItemSelected(AdapterView<?> adapterView,

View view, int i, long l) {

OptionPrefs.Group.put((Integer) adapterView.getSelectedItem());

Integer selected = (Integer) adapterView.getSelectedItem();

**new** AsyncTask<Integer, Object, Specialty>(){

**@Override**

**protected** Specialty doInBackground(Integer... param) {

SpecialtyService service = **new** SpecialtyService();

**return** service.getSpecialtyByGroupNumber(param[0]);

}

**@Override**

**protected** void onPostExecute(Specialty specialty) {

TextView specialtyView = (TextView)findViewById(R.id.specialty);

**if** (specialty != **null**){

specialtyView.setVisibility(View.VISIBLE);

specialtyView.setText(getString(R.string.specialty) + " "

+ specialty.getName());

} **else** {

specialtyView.setText("");

specialtyView.setVisibility(View.GONE);

}

}

}.execute(selected);

}

**@Override**

**public** void onNothingSelected(AdapterView<?> adapterView) {

}

});

groupAdapter = SpinnerUtils.getGroupAdapter(this);

groupSpinner.setAdapter(groupAdapter);

groupSpinner.setSelection(groupAdapter.getPosition(OptionPrefs.Group.get()));

subgroupSpinner = (Spinner) findViewById(R.id.subgroup\_spinner);

subgroupSpinner.setOnItemSelectedListener(

**new** AdapterView.OnItemSelectedListener() {

**@Override**

**public** void onItemSelected(AdapterView<?> adapterView,

View view, int position, long id) {

OptionPrefs.Subgroup.put(adapterView.getSelectedItem().toString());

}

**@Override**

**public** void onNothingSelected(AdapterView<?> parent) {

}

});

subgroupAdapter = SpinnerUtils.getSubGroupAdapter(this);

subgroupSpinner.setAdapter(subgroupAdapter);

subgroupSpinner.setSelection(subgroupAdapter.getPosition(OptionPrefs.Subgroup.get()));

departmentSpinner = (Spinner) findViewById(R.id.department\_spinner);

departmentSpinner.setOnItemSelectedListener(

**new** AdapterView.OnItemSelectedListener() {

**@Override**

**public** void onItemSelected(AdapterView<?> adapterView,

View view, int i, long l) {

Department selected = (Department) adapterView.getSelectedItem();

**if** (selected != **null**){

OptionPrefs.Department.put(selected.getId());

}

**new** AsyncTask<Long, Object, ArrayAdapter<Lecturer>>(){

**@Override**

**protected** ArrayAdapter<Lecturer> doInBackground(Long... param) {

**return** SpinnerUtils.getLecturerAdapter(OptionActivity.this, param[0]);

}

**@Override**

**protected** void onPostExecute(ArrayAdapter<Lecturer> adapter) {

lecturerAdapter = adapter;

lecturerSpinner.setAdapter(lecturerAdapter);

}

}.execute(selected.getId());

}

**@Override**

**public** void onNothingSelected(AdapterView<?> adapterView) {

}

});

departmentAdapter = SpinnerUtils.getDepartmentAdapter(this);

departmentSpinner.setAdapter(departmentAdapter);

Department department = departmentService.

getDepartment(OptionPrefs.Department.get());

**if** (department != **null**){

departmentSpinner.setSelection(departmentAdapter.getPosition(department));

}

lecturerSpinner = (Spinner) findViewById(R.id.lecturer\_spinner);

lecturerSpinner.setOnItemSelectedListener(

**new** AdapterView.OnItemSelectedListener() {

**@Override**

**public** void onItemSelected(AdapterView<?> adapterView,

View view, int position, long id) {

Lecturer lecturer = (Lecturer)adapterView.getSelectedItem();

**if** (lecturer != **null**){

OptionPrefs.Lecturer.put(lecturer.getId());

}

}

**@Override**

**public** void onNothingSelected(AdapterView<?> parent) {

}

});

lecturerAdapter = SpinnerUtils.getLecturerAdapter(this);

lecturerSpinner.setAdapter(lecturerAdapter);

Lecturer lecturer = lecturerService.getLecturer(OptionPrefs.Lecturer.get());

**if** (lecturer != **null**){

lecturerSpinner.setSelection(lecturerAdapter.getPosition(lecturer));

}

}

}

* 1. **Разметка экрана «Настройки»**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:padding="10dip"

android:background="#fff">

<com.mmf.view.ToggleButton

android:id="@+id/toggle\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

/>

<LinearLayout android:orientation="vertical"

android:id="@+id/layout\_student"

android:padding="10dip"

android:layout\_below="@id/toggle\_button"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:text="@string/course\_prompt"

android:textColor="#0b386a"

/>

<Spinner

android:id="@+id/course\_spinner"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/course\_prompt"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:text="@string/group\_prompt"

android:textColor="#0b386a"

/>

<Spinner

android:id="@+id/group\_spinner"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/group\_prompt"

/>

<TextView

android:id="@+id/specialty"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:textColor="#0b386a"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dip"

android:text="@string/subgroup\_prompt"

android:textColor="#0b386a"

/>

<Spinner

android:id="@+id/subgroup\_spinner"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/subgroup\_prompt"

/>

</LinearLayout>

<LinearLayout android:orientation="vertical"

android:id="@+id/layout\_lecturer"

android:padding="10dip"

android:layout\_below="@id/toggle\_button"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:visibility="invisible">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:text="@string/department\_prompt"

android:textColor="#0b386a"

/>

<Spinner

android:id="@+id/department\_spinner"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/department\_prompt"

/>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:text="@string/lecturer\_prompt"

android:textColor="#0b386a"

/>

<Spinner

android:id="@+id/lecturer\_spinner"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:prompt="@string/lecturer\_prompt" />

</LinearLayout>

<RelativeLayout android:id="@+id/layout\_login"

android:padding="10dip"

android:layout\_marginBottom="10dip"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:background="#f0f0f0">

<TextView android:id="@+id/text\_login"

android:layout\_width="60dip"

android:layout\_height="wrap\_content"

android:text="@string/login"

android:textColor="#0b386a"

/>

<EditText

android:id="@+id/login"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/text\_login"

android:layout\_toRightOf="@id/text\_login"

android:layout\_alignBaseline="@id/text\_login"

/>

<TextView android:id="@+id/text\_password"

android:layout\_below="@id/text\_login"

android:layout\_width="60dip"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dip"

android:text="@string/password"

android:textColor="#0b386a"

/>

<EditText

android:id="@+id/password"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:password="true"

android:layout\_below="@id/text\_password"

android:layout\_toRightOf="@id/text\_password"

android:layout\_alignBaseline="@id/text\_password"

/>

<Button

android:id="@+id/button\_login"

android:layout\_below="@id/text\_password"

android:layout\_width="fill\_parent"

android:layout\_height="30dip"

android:layout\_marginTop="20dip"

android:background="#0b386a"

android:textColor="#fff"

android:text="@string/button\_login" />

</RelativeLayout>

<RelativeLayout android:id="@+id/layout\_logout"

android:padding="10dip"

android:layout\_marginBottom="10dip"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:background="#f0f0f0"

android:visibility="invisible">

<TextView android:id="@+id/text\_view\_login"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/login"

android:textColor="#0b386a"

/>

<TextView android:id="@+id/text\_loginName"

android:layout\_below="@id/text\_view\_login"

android:layout\_toRightOf="@id/text\_view\_login"

android:layout\_marginLeft="10dip"

android:layout\_alignBaseline="@id/text\_view\_login"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textStyle="bold"

android:textColor="#0b386a" />

<Button

android:id="@+id/button\_logout"

android:layout\_below="@id/text\_view\_login"

android:layout\_width="fill\_parent"

android:layout\_height="30dip"

android:layout\_marginTop="10dip"

android:background="#0b386a"

android:textColor="#fff"

android:text="@string/button\_logout" />

</RelativeLayout>

</RelativeLayout>

## REST API

**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 |

**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 |

**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 |

**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 |

**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 |

**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 |

**Учебная дисциплина** (Discipline)

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

**Тип учебной дисциплины** (DisciplineType)

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

**Аудитория** (Classroom)

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

**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 |

**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 |

**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) |

**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 |

**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 |

**Логин**

**/rest/bsu/mmf/login**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_USER  ROLE\_STUDENT  ROLE\_LECTURER  ROLE\_ADMIN |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пользователе (ресурс **student** или **lecturer**)  **JSON student:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"1e3960302be7c4f7e1b360fadf320afbb20bdf96",  "groupId":2,  "yearOfEntrance":2011,  "praepostor":false,  "authorities":[  {"authority":"ROLE\_STUDENT"},  {"authority":"ROLE\_USER"},  {"authority":"ROLE\_ADMIN"}  ],  "admin":true  }  **JSON lecturer:**  {  "id":4,  "name":"Станислав",  "surname":"Суздаль",  "patronymic":"Валерьевич",  "fullName":"Суздаль Станислав Валерьевич",  "login":"suzdal",  "password":"1H8M7ArYttfoBrQipfJDkvXpwrI=",  "departmentId":4,  "authorities":[  {"authority":"ROLE\_LECTURER"},  {"authority":"ROLE\_USER"}  ],  "admin":false  } |
| 401 |  | Вернется, если введены неверные логин или пароль |

**Пользователь** (User)

**/rest/bsu/mmf/user/{userId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пользователе (ресурс **user**)  **JSON:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"70d0b1fbf6e5cb5fd4486bc295b99ac4149d23f6",  "authorities":[  {"authority":"ROLE\_USER"},  {"authority":"ROLE\_ADMIN"}  ],  "admin":true  } |
| 204 |  | Вернется, если такого пользователя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Actions**

**/rest/bsu/mmf/user/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "login":"yasvedko",  "password":"12345",  "admin":true  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет создан |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/{userId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id": 20,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "login":"yasvedko",  "password":"12345",  "admin":true  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет изменен |
| 204 | Вернется, если такого пользователя не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/{userId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| userId | Уникальный идентификатор пользователя |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пользователь будет удален |
| 204 | Вернется, если такого пользователя не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пользователей (ресурс **Array[]** **users**) |
| 401 |  | Вернется, если пользователь не авторизован |

**Студент** (Student)

**/rest/bsu/mmf/user/student/{studentId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о студенте (ресурс **student**)  **JSON:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"1e3960302be7c4f7e1b360fadf320afbb20bdf96",  "group":{  "id":2,  "number":1,  "course":2,  "subgroup":"a",  "year":2011,  "specialtyId":1  },  "yearOfEntrance":2011,  "praepostor":false,  "authorities":[  {"authority":"ROLE\_STUDENT"},  {"authority":"ROLE\_USER"},  {"authority":"ROLE\_ADMIN"}  ],  "admin":true  } |
| 204 |  | Вернется, если такого студента не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Aspects**

**/rest/bsu/mmf/user/student/{studentId}/notes**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пометок оставленных студентом (ресурс **Array[]** **notes**) |
| 204 |  | Вернется, если такого студента не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/student/{studentId}/group**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о группе, в которой учится студент (ресурс **group**)  **JSON:**  {  "id":2,  "number":1,  "course":2,  "subgroup":"a",  "year":2011,  "specialtyId":1  } |
| 204 |  | Вернется, если такого студента не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Actions**

**/rest/bsu/mmf/user/student/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Елена",  "surname":"Анатольевна",  "patronymic":"Киреева",  "login":"ekireeva",  "password":"ekireeva",  "groupId":2,  "yearOfEntrance":2012,  "admin":true,  "praepostor":false  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если студент будет создан |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/student/{studentId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":20,  "name":"Елена",  "surname":"Анатольевна",  "patronymic":"Киреева",  "login":"ekireeva",  "password":"ekireeva",  "groupId":2,  "yearOfEntrance":2011,  "admin":true,  "praepostor":false  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если студент будет изменен |
| 204 | Вернется, если такого студента не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/student/{studentId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если студент будет удален |
| 204 | Вернется, если такого студента не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/student/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список студентов (ресурс **Array[]** **students**) |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/student/{studentId}/schedule**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется расписание для студента (ресурс **Array[]** **schedule**) |
| 204 |  | Вернется, если такого студента не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/student/{studentId}/scheduleForDay**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| studentId | Уникальный идентификатор студента |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется расписание для студента на текущий день (ресурс **schedule**) |
| 204 |  | Вернется, если такого студента не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Преподаватель** (Lecturer)

**/rest/bsu/mmf/user/lecturer/{lecturerId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о преподавателе (ресурс **lecturer**)  **JSON:**  {  "id":4,  "name":"Станислав",  "surname":"Суздаль",  "patronymic":"Валерьевич",  "fullName":"Суздаль Станислав Валерьевич",  "login":"suzdal",  "password":"1H8M7ArYttfoBrQipfJDkvXpwrI=",  "department":{  "id":4,"  name":"Кафедра Веб-технологий и компьютерного моделирования ",  "description":"Описание кафедры читайте на сайте БГУ"  },  "authorities":[  {"authority":"ROLE\_LECTURER"},  {"authority":"ROLE\_USER"}  ],  "admin":false  } |
| 204 |  | Вернется, если такого преподавателя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Aspects**

**/rest/bsu/mmf/user/lecturer/{lecturerId}/notes**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пометок оставленных преподавателем (ресурс **Array[]** **notes**) |
| 204 |  | Вернется, если такого преподавателя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/lecturer/{lecturerId}/department**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о кафедре, которой принадлежит преподаватель (ресурс **department**)  **JSON:**  {  "id":4,"  name":"Кафедра Веб-технологий и компьютерного моделирования ",  "description":"Описание кафедры читайте на сайте БГУ"  } |
| 204 |  | Вернется, если такого преподавателя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Actions**

**/rest/bsu/mmf/user/lecturer/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Станислав",  "surname":"Суздаль",  "patronymic":"Валерьевич",  "login":"suzdal",  "password":"suzdal",  "departmentId":1,  "admin":false  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если преподаватель будет создан |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/lecturer/{lecturerId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":26,  "name":"Станислав",  "surname":"Суздаль",  "patronymic":"Валерьевич",  "login":"suzdal",  "password":"suzdal",  "departmentId":2,  "admin":false  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если преподаватель будет изменен |
| 204 | Вернется, если такого преподавателя не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/user/lecturer/{lecturerId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если преподаватель будет удален |
| 204 | Вернется, если такого преподавателя не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/lecturer/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список преподавателей (ресурс **Array[]** **lecturers**) |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/lecturer/{lecturerId}/schedule**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется расписание для преподавателя (ресурс **Array[]** **schedules**) |
| 204 |  | Вернется, если такого преподавателя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/user/lecturer/{lecturerId}/scheduleForDay**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| lecturerId | Уникальный идентификатор преподавателя |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется расписание для преподавателя на текущий день (ресурс **schedule**) |
| 204 |  | Вернется, если такого преподавателя не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Группа** (Group)

**/rest/bsu/mmf/group/{groupId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| groupId | Уникальный идентификатор группы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о группе (ресурс **group**)  **JSON:**  {  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialty":{  "id":1,  "name":"Математика. Научно-производственная деятельность",  "description":"Квалификация – Математик"},  "students":[  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"1e3960302be7c4f7e1b360fadf320afbb20bdf96",  "groupId":2,  "yearOfEntrance":2011,  "praepostor":null,  "admin":true  }  ]  } |
| 204 |  | Вернется, если такой группы не существует |

**Aspects**

**/rest/bsu/mmf/group/{groupId}/specialty**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| groupId | Уникальный идентификатор группы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о специальности группы  (ресурс **specialty**)  **JSON:**  {  "id":1,  "name":"Математика. Научно-производственная деятельность",  "description":"Квалификация – Математик"  } |
| 204 |  | Вернется, если такой группы не существует |

**/rest/bsu/mmf/group/{groupId}/students**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| groupId | Уникальный идентификатор группы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список студентов учащихся в этой группе (ресурс **Array[] students**)  **JSON:**  [  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"1e3960302be7c4f7e1b360fadf320afbb20bdf96",  "groupId":2,  "yearOfEntrance":2011,  "praepostor":null,  "admin":true  }  ] |
| 204 |  | Вернется, если такой группы не существует |

**Actions**

**/rest/bsu/mmf/group/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |  |
| --- | --- | --- |
| application/json | **Создание подгруппы:**  {  "number":12,  "course":2,  "subgroup":"a",  "year":2011,  "specialtyId":2  } | **Создание группы:**  {  "number":12,  "course":2,  "year":2011,  "specialtyId":2  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если группа будет создана |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/group/{groupId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| groupId | Уникальный идентификатор группы |

**Тело запроса**

|  |  |  |
| --- | --- | --- |
| application/json | **Обновление подгруппы:**  {  "id":36,  "number":12,  "course":2,  "subgroup":"b",  "year":2012,  "specialtyId":2  } | **Обновление группы:**  {  "id":36,  "number":12,  "course":2,  "year":2012,  "specialtyId":2  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если группа будет изменена |
| 204 | Вернется, если такой группы не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/group/{groupId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| groupId | Уникальный идентификатор группы |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если группа будет удалена |
| 204 | Вернется, если такой группы не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/group/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список групп (ресурс **Array[]** **groups**) |

**Специальность** (Specialty)

**/rest/bsu/mmf/specialty/{specialtyId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| specialtyId | Уникальный идентификатор специальности |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о специальности (ресурс **specialty**)  **JSON:**  {  «id»:2,  «name»:»Математика. Научно-педагогическая деятельность»,  «description»:»Квалификация – Математик. Преподаватель математики и информатики»,  “groups”:[  {“id”:7,”number”:3,”course”:5,”subgroup”:null,”year”:2008},  {“id”:9,”number”:3,”course”:5,”subgroup”:”a”,”year”:2008},  {“id”:8,”number”:3,”course”:5,”subgroup”:”b”,”year”:2008}  ]  } |
| 204 |  | Вернется, если такой специальности не существует |

**Aspects**

**/rest/bsu/mmf/specialty/{specialtyId}/groups**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| specialtyId | Уникальный идентификатор специальности |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список групп данной специальности  (ресурс **Array[] groups**)  **JSON:**  [  {“id”:7,”number”:3,”course”:5,”subgroup”:null,”year”:2008},  {“id”:9,”number”:3,”course”:5,”subgroup”:”a”,”year”:2008},  {“id”:8,”number”:3,”course”:5,”subgroup”:”b”,”year”:2008}  ] |
| 204 |  | Вернется, если такой специальности не существует |

**Actions**

**/rest/bsu/mmf/specialty/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Математика. Научно-педагогическая деятельность ",  "description":"Квалификация – Математик. Преподаватель математики и информатики "  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если специальность будет создана |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/specialty/{specialtyId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| specialtyId | Уникальный идентификатор специальности |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":8,  "name":" Математика. Научно-педагогическая деятельность ",  "description":" Квалификация – Математик. Преподаватель математики и информатики "  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если специальность будет изменена |
| 204 | Вернется, если такой специальности не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/specialty/{specialtyId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| specialtyId | Уникальный идентификатор специальности |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если специальность будет удалена |
| 204 | Вернется, если такой специальности не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/specialty/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список специальностей (ресурс **Array[]** **specialties**) |

**Кафедра** (Department)

**/rest/bsu/mmf/department/{departmentId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| departmentId | Уникальный идентификатор кафедры |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о кафедре (ресурс **department**)  **JSON:**  {  "id":1,  "name":"Кафедра высшей алгебры и защиты информации ",  "description":"Описание кафедры читайте на сайте БГУ",  "lecturers":[  {  "id":11,  "name":"Денис",  "surname":"Васильев",  "patronymic":"Владимирович",  "fullName":"Васильев Денис Владимирович",  "login":"vasil",  "password":"WxqEsIyybmULxlII8V/IyDYK9Gc=",  "departmentId":1,  "admin":false  }  ]  } |
| 204 |  | Вернется, если такой кафедры не существует |

**Aspects**

**/rest/bsu/mmf/department/{departmentId}/lecturers**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| departmentId | Уникальный идентификатор кафедры |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список преподавателей данной кафедры  (ресурс **Array[] lecturers**)  **JSON:**  [  {  “id”:8,  “name”:”Кирилл”,  “surname”:”Атрохов”,  “patronymic”:”Георгиевич”,  “fullName”:”Атрохов Кирилл Георгиевич”,  “login”:”atrohov”,  “password”:”waF9UJ4g4UtGy1XMDsaSjXMqxfk=”,”  departmentId”:2,  “admin”:false  },  {  “id”:13,  “name”:”Валерий”,  “surname”:”Липницкий”,  “patronymic”:”Антонович”,  “fullName”:”Липницкий Валерий Антонович”,  “login”:”lipnic”,  “password”:”qxJ308IdXO8tfRecF4xE8d9AJig=”,  “departmentId”:2,  “admin”:false  }  ] |
| 204 |  | Вернется, если такой кафедры не существует |

**Actions**

**/rest/bsu/mmf/department/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Кафедра",  "description":"Описание"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если кафедра будет создана |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/department/{departmentId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| departmentId | Уникальный идентификатор кафедры |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":13,  "name":"Кафедра",  "description":"Описание кафедры"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если кафедра будет изменена |
| 204 | Вернется, если такой кафедры не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/department/{departmentId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| departmentId | Уникальный идентификатор кафедры |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если кафедра будет удалена |
| 204 | Вернется, если такой кафедры не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/department/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список кафедр (ресурс **Array[]** **departments**) |

**Пометка** (Note)

**/rest/bsu/mmf/note/{noteId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| noteId | Уникальный идентификатор пометки |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пометке (ресурс **note**)  **JSON:**  {  "id":1,  "date":"30.05.2013",  "user":{  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"70d0b1fbf6e5cb5fd4486bc295b99ac4149d23f6",  "authorities":[],  "admin":true},  "schedule":{  "id":1,  "classroomId":1,  "disciplineTimeId":2,  "studyId":1,  "groupId":1,  "lecturerId":5,  "disciplineId":1,  "dayOfWeek":2,  "dayTitle":null,  "week":0},  "color":"#ababab",  "text":"note1"  } |
| 204 |  | Вернется, если такой пометки не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Aspects**

**/rest/bsu/mmf/note/{noteId}/schedule**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| noteId | Уникальный идентификатор пометки |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется элемент расписания, к которому была добавлена пометка (ресурс **schedule**)  **JSON:**  {  "id":1,  "classroom":{  "id":1,  "number":"606",  "capacity":90},  "disciplineTime":{  "id":2,  "startTime":"09:45",  "endTime":"11:05",  "number":2,  "breakTime":0},  "study":{  "id":1,  "groupId":1,  "curriculumId":1,  "lecturerId":5},  "group":{  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialtyId":1},  "lecturer":{  "id":5,  "name":"Василий",  "surname":"Волков",  "patronymic":"Михайлович",  "fullName":"Волков Василий Михайлович",  "login":"volkov",  "password":"lbCWkw2ia3XaIjMrkIpSvE6UQC4=",  "authorities":[],  "departmentId":4,  "admin":false},  "discipline":{  "id":1,  "name":"Численные методы",  "disciplineTypeId":1},  "dayOfWeek":2,  "dayTitle":"Понедельник",  "week":0,  "notes":[]  } |
| 204 |  | Вернется, если такой пометки не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/note/{noteId}/user**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| noteId | Уникальный идентификатор пометки |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о пользователе, который оставил пометку (ресурс **user**)  **JSON:**  {  "id":1,  "name":"Светлана",  "surname":"Войтех",  "patronymic":"Геннадьевна",  "fullName":"Войтех Светлана Геннадьевна",  "login":"yasvedko",  "password":"70d0b1fbf6e5cb5fd4486bc295b99ac4149d23f6",  "authorities":[],  "admin":true  } |
| 204 |  | Вернется, если такой пометки не существует |
| 401 |  | Вернется, если пользователь не авторизован |

**Actions**

**/rest/bsu/mmf/note/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "text": "note",  "date": "30.05.2013",  "userId": 1,  "scheduleId": 1,  "color": "#ababab"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пометка будет создана |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/note/{noteId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| noteId | Уникальный идентификатор пометки |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":1,  "text": "text ",  "date": "30.05.2013",  "userId": 1,  "scheduleId": 1,  "color": "#ababab"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пометка будет изменена |
| 204 | Вернется, если такой пометки не существует |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/note/{noteId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Параметры**

|  |  |
| --- | --- |
| noteId | Уникальный идентификатор пометки |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если пометка будет удалена |
| 204 | Вернется, если такой пометки не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/note/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN  ROLE\_STUDENT  ROLE\_LECTURER |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пометок (ресурс **Array[]** **notes**) |
| 401 |  | Вернется, если пользователь не авторизован |

**Учебная дисциплина** (Discipline)

**/rest/bsu/mmf/discipline/{disciplineId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| disciplineId | Уникальный идентификатор учебной дисциплины |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об учебной дисциплине (ресурс **discipline**)  **JSON:**  {  "id":1,  "name":"Численные методы",  "disciplineType":  {  "id":1,  "type":"lecture"  }  } |
| 204 |  | Вернется, если такого предмета не существует |

**Aspects**

**/rest/bsu/mmf/discipline/{disciplineId}/disciplineType**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| disciplineId | Уникальный идентификатор учебной дисциплины |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о типе учебной дисциплины  (ресурс **disciplineType**)  **JSON:**  {  “id”:1,  “type”:”lecture”  } |
| 204 |  | Вернется, если такого предмета не существует |

**Actions**

**/rest/bsu/mmf/discipline/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "name":"Численные методы",  "disciplineTypeId":1  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная дисциплина будет создана |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/discipline/{disciplineId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineId | Уникальный идентификатор учебной дисциплины |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":15,  "name":"Численные методы",  "disciplineTypeId":1  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная дисциплина будет изменена |
| 204 | Вернется, если такого предмета не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/discipline/{disciplineId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineId | Уникальный идентификатор учебной дисциплины |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная дисциплина будет удалена |
| 204 | Вернется, если такого предмета не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/discipline/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список учебных дисциплин (ресурс **Array[]** **disciplines**) |

**Тип учебной дисциплины** (DisciplineType)

**/rest/bsu/mmf/disciplineType/{disciplineTypeId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| disciplineTypeId | Уникальный идентификатор типа учебной дисциплины |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о типе учебной дисциплине (ресурс **disciplineType**)  **JSON:**  {  "id":1,  "type":"lecture"  } |
| 204 |  | Вернется, если такого типа учебной дисциплины не существует |

**Actions**

**/rest/bsu/mmf/disciplineType/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "type":"exam"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если тип учебной дисциплины будет создан |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/disciplineType/{disciplineTypeId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineTypeId | Уникальный идентификатор типа учебной дисциплины |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":3,  "type":"exam"  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если тип учебной дисциплины будет изменен |
| 204 | Вернется, если такого типа учебной дисциплины не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/disciplineType/{disciplineTypeId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineTypeId | Уникальный идентификатор типа учебной дисциплины |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если тип учебной дисциплины будет удален |
| 204 | Вернется, если такого типа учебной дисциплины не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/disciplineType/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список типов учебных дисциплин (ресурс **Array[]** **disciplineTypes**) |

**Время занятий** (DisciplineTime)

**/rest/bsu/mmf/disciplineTime/{disciplineTimeId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| disciplineTimeId | Уникальный идентификатор времени занятий |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о времени занятий (ресурс **disciplineTime**)  **JSON:**  {  "id":1,  "startTime":"08:15",  "endTime":"09:35",  "number":1,  "breakTime":0  } |
| 204 |  | Вернется, если такого времени занятий не существует |

**Actions**

**/rest/bsu/mmf/disciplineTime/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "startTime":"08:15",  "endTime":"09:35",  "number":1,  "breakTime":0  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если время занятий будет создано |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/disciplineTime/{disciplineTimeId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineTimeId | Уникальный идентификатор времени занятий |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":9,  "startTime":"08:15",  "endTime":"09:35",  "number":1,  "breakTime":15  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если время занятий будет изменено |
| 204 | Вернется, если такого времени занятий не существует |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/disciplineTime/{disciplineTimeId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| disciplineTimeId | Уникальный идентификатор времени занятий |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если время занятий будет удалено |
| 204 | Вернется, если такого времени занятий не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/disciplineTime/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список, содержащий время занятий (ресурс **Array[]** **disciplineTimes**) |

**Учебная программа** (Curriculum)

**/rest/bsu/mmf/curriculum/{curriculumId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| curriculumId | Уникальный идентификатор учебной программы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об учебной программе (ресурс **curriculum**)  **JSON:**  {  "id":1,  "hours":30,  "semester":4,  "exam":true,  "setoff":false,  "discipline":{  "id":1,  "name":"Численные методы",  "disciplineTypeId":1},  "specialty":{  "id":1,  "name":"Математика. Научно-производственная деятельность",  "description":"Квалификация – Математик"}  } |
| 204 |  | Вернется, если такой учебной программы не существует |

**Aspects**

**/rest/bsu/mmf/curriculum/{curriculumId}/discipline**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| curriculumId | Уникальный идентификатор учебной программы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется учебная дисциплина, для которой была составлена учебная программа (ресурс **discipline**)  **JSON:**  {  "id":1,  "name":"Численные методы",  "disciplineTypeId":1  } |
| 204 |  | Вернется, если такой учебной программы не существует |

**/rest/bsu/mmf/curriculum/{curriculumId}/specialty**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| curriculumId | Уникальный идентификатор учебной программы |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о специальности, для которой была создана учебная программа (ресурс **specialty**)  **JSON:**  {  "id":1,  "name":"Математика. Научно-производственная деятельность",  "description":"Квалификация – Математик"  } |
| 204 |  | Вернется, если такой учебной программы не существует |

**Actions**

**/rest/bsu/mmf/curriculum/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "hours":30,  "semester":5,  "exam":true,  "setoff":false,  "disciplineId":1,  "specialtyId":1  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная программа будет создана |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/curriculum/{curriculumId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| curriculumId | Уникальный идентификатор учебной программы |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":15,  "hours":35,  "semester":5,  "exam":true,  "setoff":false,  "disciplineId":1,  "specialtyId":1  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная программа будет изменена |
| 204 | Вернется, если такой учебной программы не существует |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/curriculum/{curriculumId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| curriculumId | Уникальный идентификатор учебной программы |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если учебная программа будет удалена |
| 204 | Вернется, если такой учебной программы не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/curriculum/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список учебных программ (ресурс **Array[]** **curriculums**) |

**Аудитория**(Classroom)

**/rest/bsu/mmf/classroom/{classroomId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| classroomId | Уникальный идентификатор аудитории |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об аудитории (ресурс **classroom**)  **JSON:**  {  "id":1,  "number":"606",  "capacity":90  } |
| 204 |  | Вернется, если такой аудитории не существует |

**Actions**

**/rest/bsu/mmf/classroom/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "number":"352",  "capacity":30  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если аудитория будет создана |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/classroom/{classroomId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| classroomId | Уникальный идентификатор аудитории |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":15,  "number":"352",  "capacity":35  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если аудитория будет изменена |
| 204 | Вернется, если такой аудитории не существует |
| 401 | Вернется, если пользователь не авторизован |
| 400 | Если обязательное поле будет иметь неверное значение (null или неверный тип) |
| 500 | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/classroom/{classroomId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| classroomId | Уникальный идентификатор аудитории |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если аудитория будет удалена |
| 204 | Вернется, если такой аудитории не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/classroom/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список аудиторий (ресурс **Array[]** **classrooms**) |

**Занятие** (Study)

**/rest/bsu/mmf/study/{studyId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о занятии (ресурс **study**)  **JSON:**  {  "id":1,  "group":{  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialtyId":1},  "curriculum":{  "id":1,  "hours":30,  "semester":4,  "exam":true,  "setoff":false,  "disciplineId":1,  "specialtyId":1},  "lecturer":{  "id":5,  "name":"Василий",  "surname":"Волков",  "patronymic":"Михайлович",  "fullName":"Волков Василий Михайлович",  "login":"volkov",  "password":"lbCWkw2ia3XaIjMrkIpSvE6UQC4=",  "authorities":[],  "departmentId":4,  "admin":false}  } |
| 204 |  | Вернется, если такого занятия не существует |

**Aspects**

**/rest/bsu/mmf/study/{studyId}/lecturer**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о преподавателе, который ведет это занятие (ресурс **lecturer**)  **JSON:**  {  "id":5,  "name":"Василий",  "surname":"Волков",  "patronymic":"Михайлович",  "fullName":"Волков Василий Михайлович",  "login":"volkov",  "password":"lbCWkw2ia3XaIjMrkIpSvE6UQC4=",  "authorities":[],  "departmentId":4,  "admin":false  } |
| 204 |  | Вернется, если такого занятия не существует |

**/rest/bsu/mmf/study/{studyId}/group**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о группе, которой читается это занятие (ресурс **group**)  **JSON:**  {  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialtyId":1  } |
| 204 |  | Вернется, если такого занятия не существует |

**/rest/bsu/mmf/study/{studyId}/curriculum**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об учебной программе этого занятия (ресурс **group**)  **JSON:**  {  "id":1,  "hours":30,  "semester":4,  "exam":true,  "setoff":false,  "disciplineId":1,  "specialtyId":1  } |
| 204 |  | Вернется, если такого занятия не существует |

**Actions**

**/rest/bsu/mmf/study/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "groupId":1,  "curriculumId":1,  "lecturerId":6  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если занятие будет создано |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/study/{studyId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":15,  "groupId":1,  "curriculumId":2,  "lecturerId":6  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если занятие будет изменено |
| 204 | Вернется, если такого занятия не существует |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/study/{studyId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| studyId | Уникальный идентификатор занятия |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если занятие будет удалено |
| 204 | Вернется, если такого занятия не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/study/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список занятий (ресурс **Array[]** **studies**) |

**Элемент Расписания** (Schedule)

**/rest/bsu/mmf/schedule/{scheduleId}**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об элементе расписания (ресурс **schedule**)  **JSON:**  {  "id":1,  "classroom":{  "id":1,  "number":"606",  "capacity":90},  "disciplineTime":{  "id":2,  "startTime":"09:45",  "endTime":"11:05",  "number":2,  "breakTime":0},  "study":{  "id":1,  "groupId":1,  "curriculumId":1,  "lecturerId":5},  "group":{  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialtyId":1},  "lecturer":{  "id":5,  "name":"Василий",  "surname":"Волков",  "patronymic":"Михайлович",  "fullName":"Волков Василий Михайлович",  "login":"volkov",  "password":"lbCWkw2ia3XaIjMrkIpSvE6UQC4=",  "authorities":[],  "departmentId":4,  "admin":false},  "discipline":{  "id":1,  "name":"Численные методы",  "disciplineTypeId":1},  "dayOfWeek":2,  "dayTitle":"Понедельник",  "week":0,  "notes":[]  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**Aspects**

**/rest/bsu/mmf/schedule/{scheduleId}/classroom**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об аудитории, в которой будет проводиться занятие (ресурс **classroom**)  **JSON:**  {  "id":1,  "number":"606",  "capacity":90  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/study**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о занятии (ресурс **study**)  **JSON:**  {  "id":1,  "groupId":1,  "curriculumId":1,  "lecturerId":5  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/disciplineTime**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о времени проведения занятия (ресурс **disciplineTime**)  **JSON:**  {  "id":2,  "startTime":"09:45",  "endTime":"11:05",  "number":2,  "breakTime":0  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/group**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация об учебной группе (ресурс **group**)  **JSON:**  {  "id":1,  "number":1,  "course":2,  "subgroup":null,  "year":2011,  "specialtyId":1  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/lecturer**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о преподавателе (ресурс **lecturer**)  **JSON:**  {  "id":5,  "name":"Василий",  "surname":"Волков",  "patronymic":"Михайлович",  "fullName":"Волков Василий Михайлович",  "login":"volkov",  "password":"lbCWkw2ia3XaIjMrkIpSvE6UQC4=",  "authorities":[],  "departmentId":4,  "admin":false  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/discipline**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется информация о предмете (ресурс **discipline**)  **JSON:**  {  "id":1,  "name":"Численные методы",  "disciplineTypeId":1  } |
| 204 |  | Вернется, если такого элемента расписания не существует |

**/rest/bsu/mmf/schedule/{scheduleId}/notes**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список пометок, оставленных на этот элемент расписания (ресурс **Array[] notes**) |
| 204 |  | Вернется, если такого элемента расписания не существует |

**Actions**

**/rest/bsu/mmf/schedule/add**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "classroomId":1,  "disciplineTimeId":4,  "studyId":1,  "dayOfWeek":2,  "week":0  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если элемент расписания будет создан |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/schedule/{scheduleId}/edit**

|  |  |
| --- | --- |
| HTTP Method | POST |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Тело запроса**

|  |  |
| --- | --- |
| application/json | {  "id":15,  "classroomId":5,  "disciplineTimeId":4,  "studyId":1,  "dayOfWeek":2,  "week":0  } |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если элемент расписания будет изменен |
| 204 | Вернется, если такого элемента расписания не существует |
| 401 | Вернется, если пользователь не авторизован |
|  | Вернется, если тело запроса будет иметь неправильный формат |

**/rest/bsu/mmf/schedule/{scheduleId}/delete**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Required |
| Roles | ROLE\_ADMIN |

**Параметры**

|  |  |
| --- | --- |
| scheduleId | Уникальный идентификатор элемента расписания |

**Ответ**

|  |  |
| --- | --- |
| 200 | Вернется, если элемент расписания будет удален |
| 204 | Вернется, если такого элемента расписания не существует |
| 401 | Вернется, если пользователь не авторизован |

**/rest/bsu/mmf/schedule/list**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список элементов расписания (ресурс **Array[]** **schedules**) |

**/rest/bsu/mmf/schedule?course=?&group=?&subgroup=?&lecturerId=?**

|  |  |
| --- | --- |
| HTTP Method | GET |
| Authentication | Not Required |

**Параметры**

|  |  |
| --- | --- |
| course | Номер курса |
| group | Номер группы |
| subgroup | Подгруппа |
| lecturerId | Уникальный идентификатор преподавателя |

Чтобы получить расписание для студента необходимо передавать параметры: **course, group, subgroup.**

Чтобы получить расписание для преподавателя необходимо передавать параметры: **lecturerId.**

**Ответ**

|  |  |  |
| --- | --- | --- |
| 200 | application/json | Вернется список элементов расписания, сгруппированных по дням и отсортированных по времени проведения пары (ресурс **Array[]** **schedules**)  **JSON:**  {  "currentWeek":1,  "monday":[ Array[] schedules],  "tuesday":[ Array[] schedules],  "wednesday":[ Array[] schedules],  "thursday":[ Array[] schedules],  "friday":[ Array[] schedules],  "saturday":[ Array[] schedules],  } |
| 400 |  | Вернется, если будет передан неверный набор параметров |

# АННОТАЦИЯ

## Abstract.

This research paper describes the process of developing RESTful web-services and Android application.

In the first chapter of the paper the analogues of the application are discussed. Also there is a brief overview of existing technologies for development in this chapter. This overview includes the description of the JPA 2.0, Spring framework, MySQL, REST-service, JAX-RS and Android technologies. Further there is a detailed definition of the model and functionality of the created application. Also the description and requirements for the application are provided.

The second chapter contains a detailed review and description of the development process of the application. There is a description of the data model for server and client sides. Also the architecture of REST services is provided. After that the REST API Endpoints are considered. Then the architecture of the information model is established. The following describes the Balsamiq Mockups program for creation an Android application prototype. The chapter provides a description of the functionality and screenshots of the Android application. The methods of the testing are also considered here.

The third chapter contains the organization part. Some options of the project extension prospects are supposed here.

## Keywords.

Java EE, Android, REST API, Apache Tomcat, Java Persistence API 2.0 (JPA 2.0), Spring, MySQL, [Spring Security](http://ru.wikipedia.org/wiki/Spring_Security), JAX-RS, schedule, API Endpoints, Balsamiq Mockups, Fiddler, Robotium, Blackbox testing.