ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ
ВЫСШАЯ ШКОЛА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ И СУПЕРКОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Отчет о прохождении стационарной производственной практики (научно-исследовательской работы на тему:

«Создание платформы для сбора сведений для системы составления предварительного расписания сессий»**)**

Суховой Дарьи Викторовны, гр. 3530903/70302

Направление подготовки: 09.03.03 Прикладная информатика

Место прохождения практики: СПбПУ, ИКНТ, ВШИСиСТ

(указывается наименование профильной организации или наименование структурного подразделения

ФГАОУ ВО «СПбПУ», фактический адрес)

Сроки практики: с 01.09.20 по 15.12.20

Руководитель практики от ФГАОУ ВО «СПбПУ»:
Щукин А.В., доцент ВШИСиСТ, к.т.н.
(Ф.И.О., уч. степень, должность)

Консультант практики от профильной организации:
Пархоменко Владимир Андреевич, ассистент
(Ф.И.О., должность)

Оценка:

Руководитель практики
от ФГАОУ ВО «СПбПУ» Щукин А.В.

Консультант практики
от ФГАОУ ВО «СПбПУ» Пархоменко В.А.

Обучающийся Сухова Д.В.

Дата:

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Составление расписания экзаменационных сессий сейчас является сложным и кропотливым процессом, который требует автоматизации. Создание программы, которая поможет сотрудникам университета с распределением экзаменов по датам, времени и аудиториям с учётом пожеланий преподавателей и университетского расписания, сможет значительно ускорить процесс составления расписания, а также устранит противоречия связанные с неустановленными формами сбора сведений. Таким образом, целью моей ВКР будет являться написание системы составления предварительного расписания СПбПу. Для достижения этой цели, в необходимо будет решить следующие задачи:

- Разработать сервис по сбору сведений, необходимых для составления расписания сессии.
- Разработать алгоритм, находящий возможные варианты расписания сессиии на основе полученных сведений.
- Написать сервис для получения данных о расписании с использованием API ruz.spbstu.ru

Целью данной работы является написание системы по сбору сведений, необходимых для составления расписания сессии. Эта система будет использована в качестве одного из сервисов системы составления предварительного расписания.

На данный момент все эти данные собираются составителями расписания вручную, из-за этого пожелания преподавателей заполняются в свободной форме, что, во-первых, усложняет их систематизацию, а во-вторых, влечёт противоречия и недпонимание из-за неопределённых формулировок.

Для создания системы составления предварительного расписания необходимо уметь собирать следующие сведения:

- период сессии
- даты, в которые преподаватель может принимать экзамен
- желаемое время проведения экзаменов для преподавателя
- необходимость компьютеров в аудитории
- необходимость проектора в аудитории
- минимальное количество мест в аудитории
- список доступных аудиторий с количеством мест в них, а также с указанием, есть ли там компьютеры и проектор

- список, отображающий экзамены и зачёты, которые преподаватели должны провести у конкретных групп

Таким образом, для достижения цели данной работы необходимо выполнить несколько задач:

- Создать базу данных для хранения сведений о сессии и пожеланиях преподавателей.
- Написать модуль по сбору пожеланий преподавателей.
- Написать модуль по сбору данных о сессии, аудиториях и запланированных экзаменах.
- Обеспечить разделение ролей пользователей.

ГЛАВА 1. ТЕОРЕТИЕСКАЯ ЧАСТЬ

1.1. Роли пользователей системы

Одним из требований к системе сбора сведений является выделение двух ролей пользователей:

- Администратор - пользователь, который сообщает системе сведения о сессии, аудиториях и о плане экзаменов.
- Преподаватель - пользователь, который сообщает системе о собственных пожеланиях к проведению своих экзаменов.

Рассмотрим возможности этих видов пользователей подробнее. На диаграмме на рисунке 1.1 показаны возможности преподавателя.

Преподаватель может поучаствовать в составлении расписания, указав даты, дни недели и время, когда он доступен для проведения экзаменов иди заётов, написав и какого типа аудитории нужны.

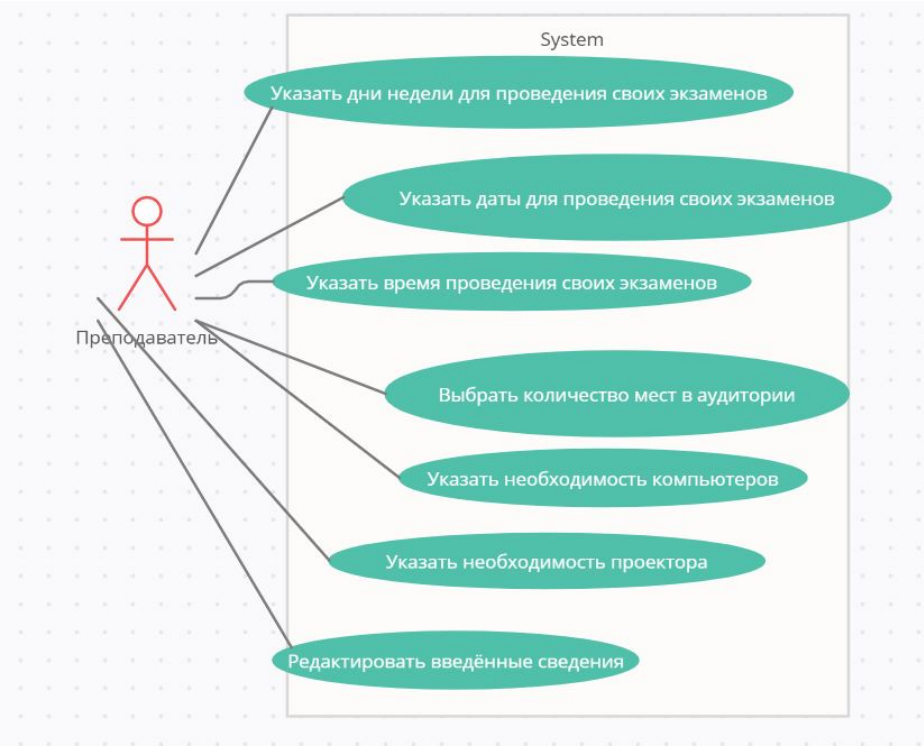

Рис.1.1. Use case диаграмма ППС

Пользователь считается преподавателем, если он был добавлен администратором в список преподавателей.

Возможности администратора показаны на диаграмме на рисунке 1.2.

Администратор заполняет общие сведения о сессии, такие как даты, время, доступные аудитории с указанием, сколько в них мест и есть ли там проектор и компьютеры и предстоящие экзамены по группам и преподавателям.

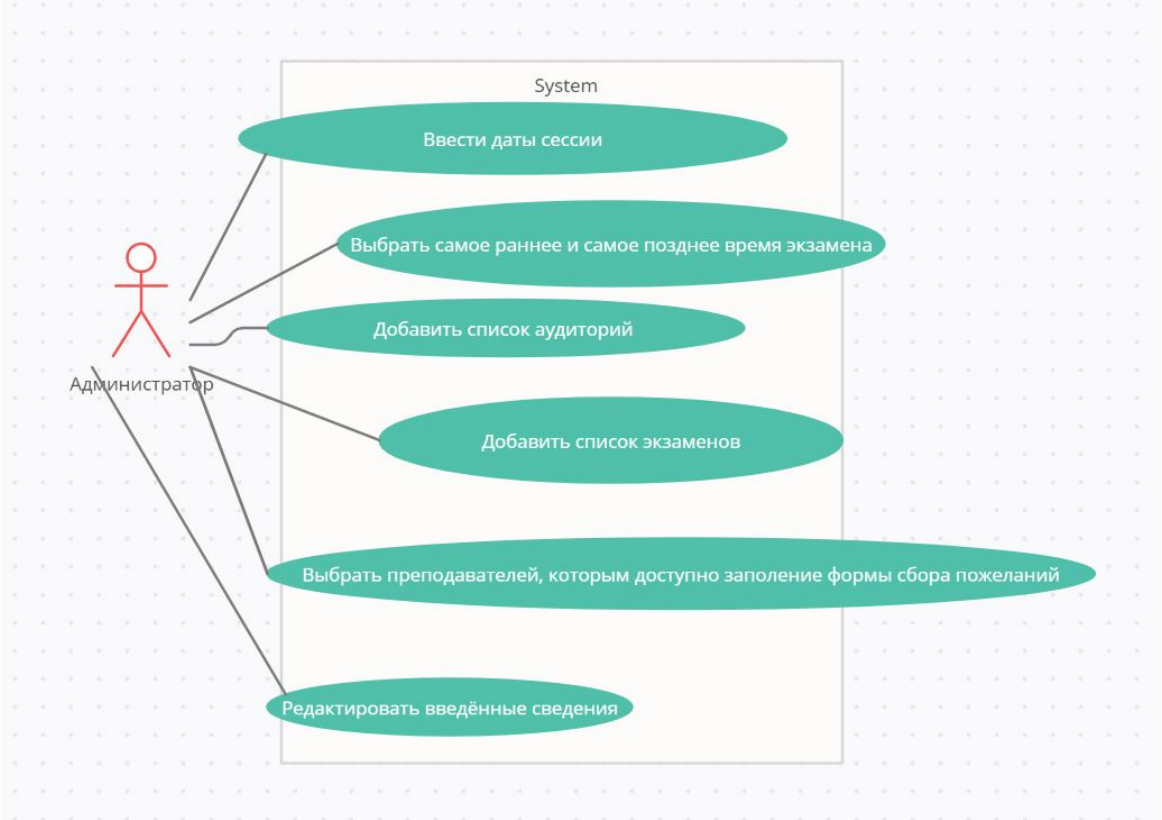

Рис.1.2. Use case диаграмма администратора

Таким образом, когда администратор заполнит все вышеперечисленные поля формы, в системе уже будет минимальный набор данных, необходимый для составления расписания. Далее, преподаватели могут заполнять свои формы с пожеланиями к своему расписанию. Незаполненная преподавателем форма не будет являться проблемой для системы. Пустая форма по умолчанию приравнивается к готовности преподавателя проводить экзамены и зачёты в любой день, в любое время.

1.2. Схема хранения данных

Схема базы данных системы сбора сведений для составления предварительного расписания сессии представлена на рисунке 1.3

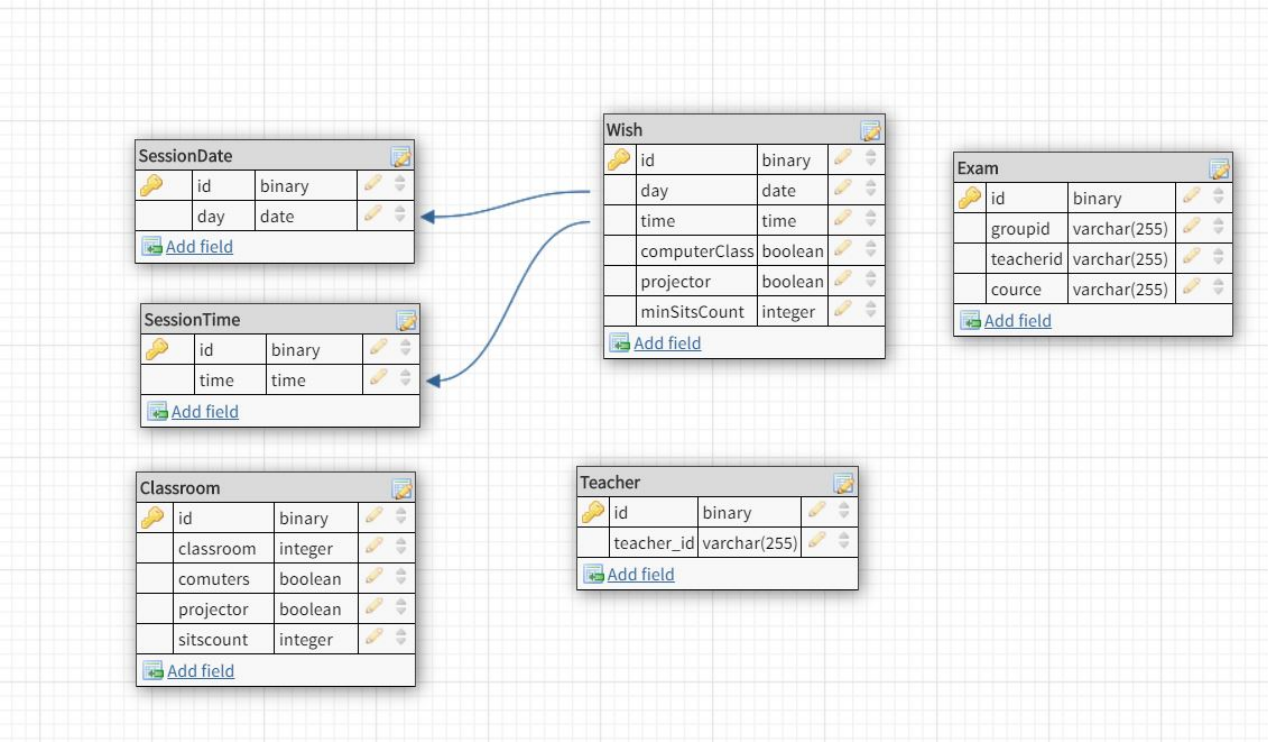

Рис.1.3. Схема базы данных

Сущности схемы:

- Classroom - таблица аудиторий с указанием есть ли там компьютеры или проектор и количеством мест.
    - id - уникальный идетификатор
    - classroom - номер кабинета
    - computers - наличие компьютеров
    - projector - наличие проектора
    - sitscount - кол-во мест в аудитории
- Exam - таблица запланированных на сессию экзаменов.
    - id - уникальный идетификатор
    - groupid - номер учебной группы
    - teacher - логин преподавателя
    - course - код дисциплины
- SessionDates - таблица со всеми доступными для экзамена датами. Такой способ хранения данных о датах сессии был выбран вместо хранения даты начала и окончания, потому что зимняя сессия может состоять из нескольких интервалов, прерванных новогодними праздниками.
    - id - уникальный идетификатор
    - day - дата

- SessionTime - таблица доступного для экзаменов времени.
    - id - уникальный идетификатор
    - time - время начала экзамена
- Wish - таблица пожеланий преподавателей.
    - id - уникальный идетификатор
    - teacherid - логин преподавателя
    - time - время
    - day - дата
    - computerClass - наличие компьютеров
    - projector - наличие проектора
    - minSitsCount - кол-во мест в аудитории
- Teacher - таблица преподавателей, которым доступно редактирование формы пожеланий.
    - id - уникальный идетификатор
    - teacherid - логин преподавателя

ГЛАВА 2. ИНСТРУМЕНТЫ РАЗРАБОТКИ

Система сбора данных для составления расписания сессии была выполнена в форме веб-приложения. В качестве основнго фреймворка был выбран Spring для языка программирования Java.

Java является одним из самых популярных языков программирования, что упрощает дальнейшую работу над приложением в том случае, если за неё возьмётся другой программист. Также Java является кроссплатформенным языком, что избавляет от лишних затрат на портирование кода с одной ОС на другую. Помимо этого, Java имеет ряд полезных свойств:

- Наличие большого количества поддерживаемых библиотек кода и фреймворков
- Поддержка принципов ООП
- Наличие сборщика мусора, который избавляет разработчика от заботы об очистке памяти

Основное преимущество фреймворка Spring - это возможность разработки веб-приложения как набора слабосвязанных компонентов. Чем меньше компоненты приложения связаны, тем проще добавлять новые модули и редактировать по ходу работы существующие.

Благодаря использованию этого фреймворка, можно не прописывать обработку входящих пакетов, собственную систему авторизации и разрешения URL-адресов, потому что всё это уже содержится в библиоткеках Spring.

Также, Spring обеспечивает архитектуру шаблона проектирования MVC, то есть разделяет логику приложения на три модуля:

- Model — модуль, отвечающий за предоставление данных и состоящий из базы данных и бинов.
- View - модуль, отвечающий за визуализацию данных, полученных от контроллера, и передающий контроллеру пользовательские запросы
- Controller — обрабатывает поступающие запросы от View и, используя данные из Model, возвращает ответ обратно View.

Для определения этих блоков в Spring существуют аннотации: @Controller - для класса контроллера, @Repository - для части модели, отвечающей за связь с базой данных и @Service для классов, реализующих логику приложения.

Фронтентд был написан с помощью HTML, CSS, JavaScrypt с использованием библиотеки Thymeleaf из Spring. Это стандартный набор инструментов для написания визуальной части приложения.

В качестве базы данных приложения была выбрана PostgreSQL. Этот выбор был обусловлен тем, что PostgreSQL - объектно-реляционная база, что позволяет хранить в ней объекты разных типов, в том числе даты и даже массивы. Немаловажным стал фактор стоимости. Большинство продакшн баз данных требуют покупку дорогостоящих лицензий. В отличие от них, PostgreSQL - полностью бесплатный продукт, который легко развернуть на любой машине. Ещё одним преимуществом является простота диалекта PostgreSQL и близость его к стандарту.

Для взаимодействия базы данных с приложением используется популярный фреймворк Hibernate. Он позволяет использовать готовые ddl запросы, а также конвертирует результаты селектов в стандартные java-коллекции.

ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Приложение делит аутентифицированных пользователей на две роли - преподавателей и администраторов. Для разделения страниц, к которым каждая из этих групп имеет доступ существет класс WebSecurityConfig, представленный в приложении 11.

Администратор имеет право настроить даты сессии, время экзаменов, отправить список аудиторий, список экзаменов и список преподавателей. На рисунке 3.1 изображена страница навигации по настройкам сессии для администратора.

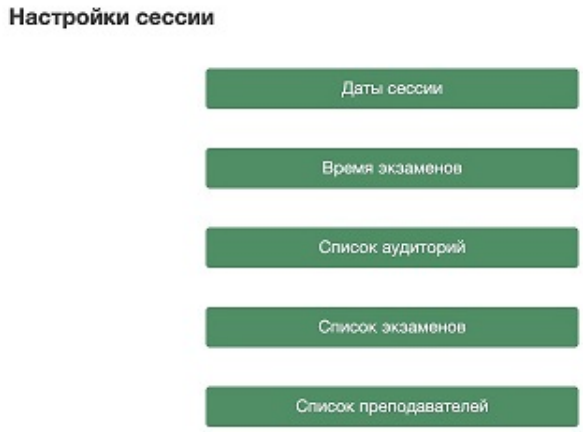

Рис.3.1. Страница навигации по настройкам сессии

Переходя по страницам настроек сессии на рисунках 3.2, 3.3, 3.4, 3.5 и 3.6 и нажимая кнонпку отправки, администратор вызывает функцию создания POST-запроса с сохраняемыми данными (Приложение 9). Для получения запросов с данными о настройках сессии существует класс-контроллер SessionService (Приложение 12)

Этот класс содержит следующие методы:

- String updateDate(@RequestParam(name = "days") String[] days) - настройка дат
- String updatetime(@RequestParam(name = "time1") String time1, @RequestParam(name = "time2") String time2) - настройка времени
- String updateClass(@RequestParam(name = "classroomFile") String classroomFile) - получение списка аудиторий
- String updateexam(@RequestParam(name = "examFile") String examFile) - получение списка экзаменов

– String updateteacher(@RequestParam(name = "teacherFile") String examFile) - получение списка преподавателей

Для получаения дат сессии SessionService метод updateDate вызывает метод createDate из класса SessionDao (Приложение 14). Это класс, в котором содержатся методы для взаимодействия репозиториями данных - настроек сессии.

Чтобы сохранить данные о датах сесии необходимо у репозитория SessionDatesRepository (Приложение 22) вызвать метод Create, который принимает объект класса-модели SessionDates (Приложение 18), который имеет структуру таблицы дат базы данных.

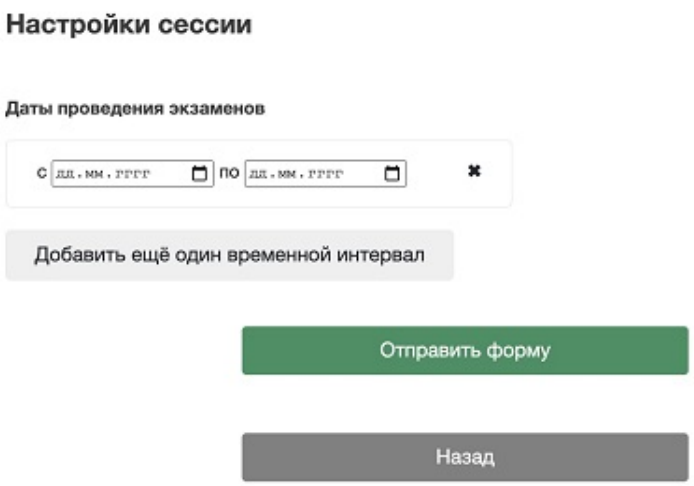

Рис.3.2. Настройка дат сессии

Все остальные параметры обслуживаются по такой же схеме, только изменяются названия классов репозитория и модели. Так, для добавления в базу данных сведений о времени нужно воспользоваться методом create репозитория SessionTimeRepository, принимающего в аргументах модель SessionTime.

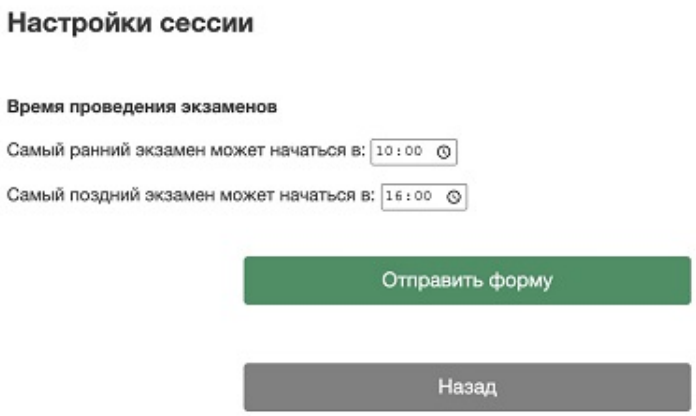

Рис.3.3. Настройка времени экзаменов

При загрузке списка аудиторий вызывается метод createClassroom, который в репозиторий ClassroomRepository создает модель Classroom, представленную в приложении 16.

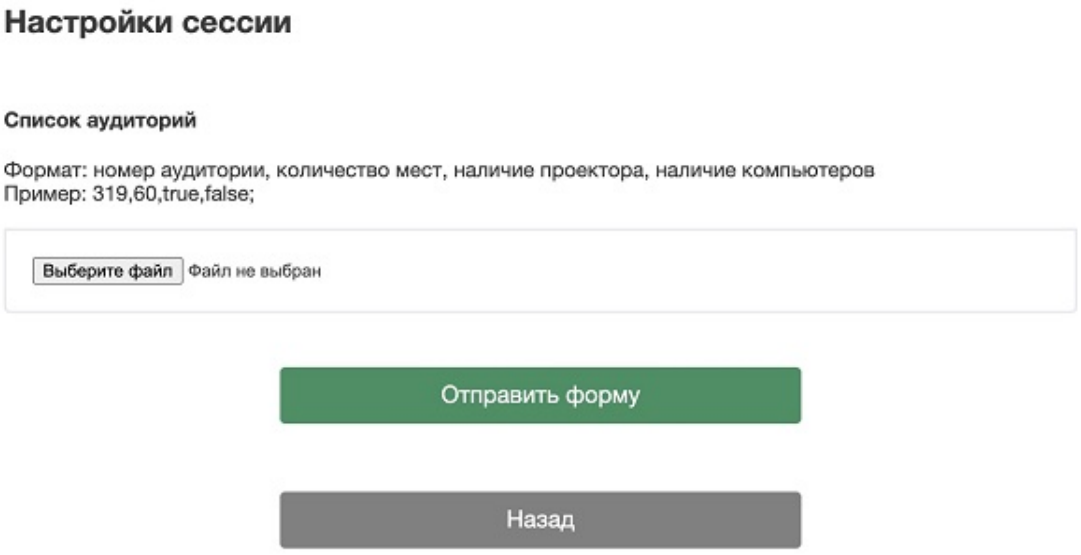

Рис.3.4. Загрузка списка аудиторий

Чтобы загрузить в базу данные об экзаменах, метод createExam добавляет в репозиторий ExamRepository модель Exam, описанную в приложении 17.

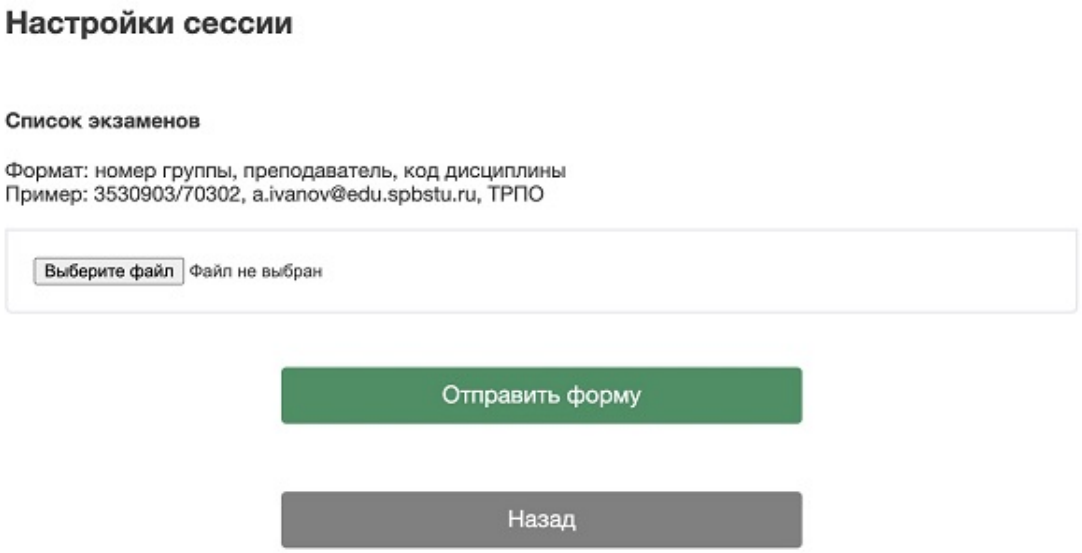

Рис.3.5. Загрузка списка экзаменов

Для добавления данных из списка преподавателей в репозиторий TeacherRepository добавлется модель Teacher из приложени 20.

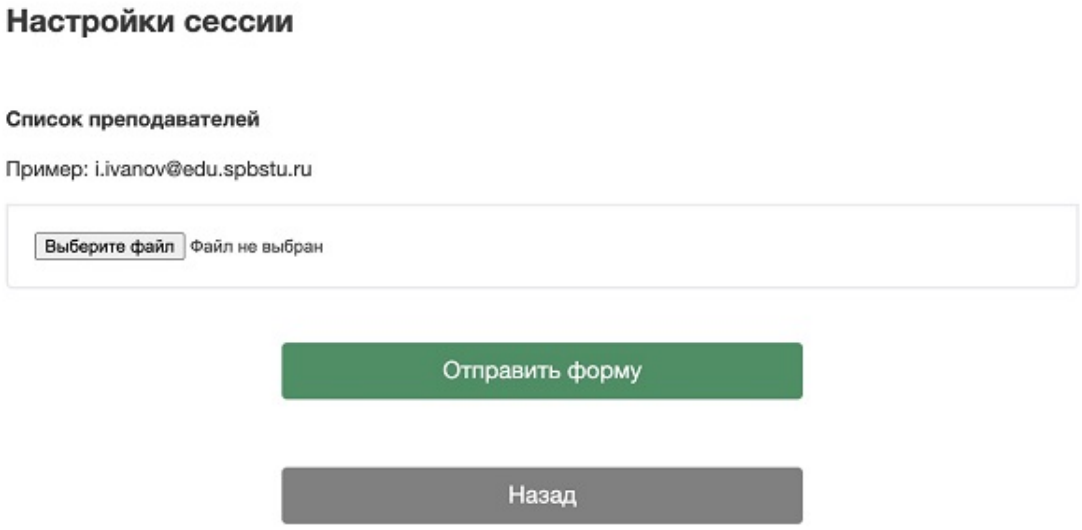

Рис.3.6. Загрузка списка препдавателей

Преподавателям для простоты предлагается заполнить одну форму, представленную на рисунке 3.7. Нажатие кнопки отправки формирует запрос, который будет обработан в классе-контроллере WishService из приложения 13. В нём метод create обрабатывает полученные данные, приводит данные нужным типам и об-

ращается классу WishDAO (прилоение 15), который аккумулирут методы работы с репозиторием с пожеланиями преподавателей. В репозиторий WishRepository добавляются объекты класса-модели Wish из приложения 21.

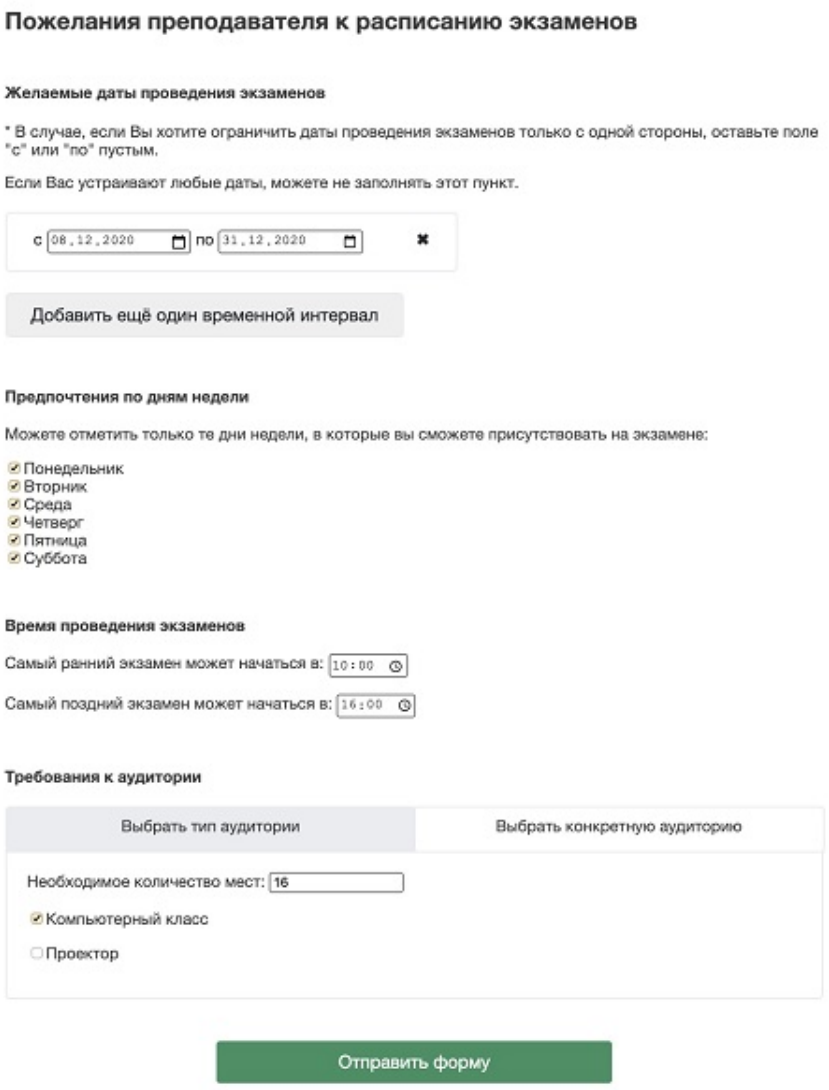

Рис.3.7. Форма сбора пожеланий преподавателей

При этом, если пользователь с ролью Преподаватель даже случайно попытается перейти на страницу редактирования сессии, он не сможет этого сделать и получит ошибку.

ЗАКЛЮЧЕНИЕ

В ходе научно исследовательсной работы были сформулированы цели и задачи будущей ВКР - в частности, написание системы составления предварительного расписания СПбПу. Также, в ходе работы было создано веб-приложение для сбора сведений для составления расписания сессии, которое будет являться одним из модулей системы составления предварительного расписания.

С помощью системы сборы сведений, необходимых для составления расписания, можно аккумулировать пожелания преподавателей о датах и времени экзаменов, а также о типах аудиторий для их проведения. Помимо этого приложение имеет веб-интерфейс ещё и для сбора данных о сесии, например, даты сессии, планируемые экзамены и доступные аудитории. Всё это шаг к автоматизации процесса составления расписания в университете.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Галактика. Расписание учебных занятий. —URL: https://galaktika-it.ru/spb/ruz (дата обращения: 10.07.2020).

2. Документация Hibernate. —URL: https://hibernate.org/orm/documentation/5.4/ (дата обращения: 10.12.2020).

3. Шаблон проекта в Figma. —URL: https://www.figma.com/file/2sp86Lky7QENS60btoH6De/Предварительное-расписание (дата обращения: 10.12.2020).

4. Johnson R. Spring Framework Reference Documentation, 2014

Приложение 1

HTML-код опросника преподавателей

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: пожелания преподавателей</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../js/teacher_wish.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="moodle, ИКНТ: пожелания препода
   вателей"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Пожелания преподавателя к расписанию экзаменов</h2>
<section class="day_section">
<h4>Желаемые даты проведения экзаменов</h4>
* В случае, если Вы хотите ограничить даты проведения экзамено
   в только с одной стороны, оставьте поле "с" или "по" пустым
   .<p>
Если Вас устраивают любые даты, можете не заполнять этот пункт
   .
<div id="calendar">
<div class="item" id="dates0">
<div class="date__items">
с
<input type="date" name="date1" class="date__item date1">
по
<input type="date" name="date2" class="date__item date2">
</div>
<div class="del" id="del_0"> &#10006;</div>
</div>
</div>
```

```
<button id="add_new_date" type="button">
Добавить ещё один временной интервал
</button>
</section>
<section class="day_section">
<h4>Предпочтения по дням недели</h4>
<div class="day">
Можете отметить только те дни недели, в которые вы сможете при
   сутствовать на экзамене:<p>
<input class="dayOfWeekOption" type="checkbox" name="option1"
   checked=true value="1">Понедельник<Br>
<input class="dayOfWeekOption" type="checkbox" name="option2"
   checked=true value="2">Вторник<Br>
<input class="dayOfWeekOption" type="checkbox" name="option3"
   checked=true value="3">Среда<br>
<input class="dayOfWeekOption" type="checkbox" name="option4"
   checked=true value="4">Четверг<br>
<input class="dayOfWeekOption" type="checkbox" name="option5"
   checked=true value="5">Пятница<br>
<input class="dayOfWeekOption" type="checkbox" name="option6"
   checked=true value="6">Суббота</p>
</div>
</section>
<section class="time_section">
<h4>Время проведения экзаменов</h4>
<div class="time">
<div>
Самый ранний экзамен может начаться в:
<input type="time" name="time1" id="time1" class="time__item"
   value="10:00">
</div>
<p>
<div>
Самый поздний экзамен может начаться в:
<input type="time" name="time2" id="time2" class="time__item"
   value="16:00">
</div>
</div>
</section>
<section class="classroom__section">
<h4>Требования к аудитории</h4>
<div class="choose_classroom">
<label name="classroom_selection" class="classroom_selection">
<input type="radio" name="classroom_selection" class="
   hidden_checkbox" value="0" checked>
```

```
<div class="classroom_selection_text"><span>Выбрать тип аудито
   рии</span></div>
</label>
<label name="classroom_selection" class="classroom_selection">
<input type="radio" name="classroom_selection" class="
   hidden_checkbox" value="1">
<div class="classroom_selection_text"><span>Выбрать конкретную
    аудиторию</span></div>
</label>
</div>
<div class="classroom_param">
<div class="count_classroom">
Необходимое количество мест:
<input name="classroom_cnt" type="number" id="classroom_cnt"
   value="16">
</div>
<p>
<input type="checkbox" name="comp" id="comp" value="1">Компьют
   ерный класс
<p>
<input type="checkbox" name="proector" id="proector" value
   ="2">Проектор
<p>
</div>
<div class="concrete_classroom">
<div id="classroom_blocks">
<div class="item">
Введите номер аудитории:
<input type="number" name="classroom_number">
</div>
</div>
<button id="add_classroom" type="button">
Добавить аудиторию
</button>
</div>
</section>
<button type="submit" id="submit" class="submit">Отправить фор
   му</button>
</div>
</body>
</html>
```

Приложение 2

HTML-код навигации для настройки сессии

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../js/session_settings.js"></script
   >
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<button class="submit" onclick="window.location.href = '
   session/date_settings';">Даты сессии</button>
<button class="submit" onclick="window.location.href = '
   session/time_settings';">Время экзаменов</button>
<button class="submit" onclick="window.location.href = '
   session/class_settings';">Список аудиторий</button>
<button class="submit" onclick="window.location.href = '
   session/exam_settings';">Список экзаменов</button>
<button class="submit" onclick="window.location.href = '
   session/teacher_settings';">Список преподавателей</button>
</div>
</body>
</html>
```

Приложение 3

CSS стили приложения

```
body {
font-family: "Open Sans", "Helvetica Neue", Arial, sans-serif;
background-color: #e9ecef;
color: #46c765;
}
.page-wrapper {
box-sizing: border-box;
overflow-wrap: break-word;
text-align: left;
width: 1100px;
background-color: white;
margin: auto;
margin-bottom: 50px;
padding: 20px 0px 20px 0px;
min-height: 600px;
}
h1 {
margin-bottom: 20px;
text-align: center;
}
h2 {
margin-bottom: 18px;
text-align: center;
}
h4 {
margin-top: 30px;
margin-bottom: 20px;
}

nav{
margin-left: 60px;
margin-right: 60px;
width: auto;
display: flex;
justify-content: flex-start;
}
.filter{
color:white;
margin-right: 20px;
```

```
}
.filter_item{
background-color: rgb(233, 236, 239);
height: 25px;
line-height:25px;
}
.border{
display: flex;
margin: 20px 0 20px;
border: 2px solid rgb(233, 236, 239);
border-radius: 5px;
color: rgb(33, 37, 41);
cursor: pointer;
padding: 7px;
text-align: center;
}
.new_meeting{
display: block;
border-color: #46c765;
background-color: #46c765;
color: #ffffff;
margin-top: 10px;
height: 50px;
width: 200px;
text-align: center;
font-size: 16px;
font-weight: bold;
margin-left: 150px;
}
.statuses{
margin-left: 5px;
}
main {
padding: 50px 20px 50px 20px;
display: flex;
flex-wrap: wrap;
justify-content: space-around;
}
.meeting{
display: flex;
justify-content: normal;
width: 400px;
height: 160px;
box-sizing: content-box;
margin: 20px 0 20px;
```

```
border: 2px solid rgb(233, 236, 239);
border-radius: 5px;
padding: 20px;
text-align: center;
color: #7e7e7e;
}
button{
display: block;
margin: 20px 0 20px;
border: 1px solid rgb(233, 236, 239);
border-radius: 5px;
color: rgb(33, 37, 41);
cursor: pointer;
font-family: "Open Sans", "Helvetica Neue", Arial, sans-serif;
font-size: 18px;
height: 45px;
padding: 7px 15px 7px 15px;
text-align: center;
width: 400px;
}

select{
font-size: 14px;
color: #325a46;
font-family: "Open Sans", "Helvetica Neue", Arial, sans-serif;
}
.date_item{
margin: 0 10px;
height: 20px;
}
.date_time{
font-style: italic;
display: block;
width: 100px;
height: 150px;
justify-content: center;
text-align: left;
margin-right:10px;
float: left;
}
.name{
font-size: 18px;
color: #325a46;
}
```

```
.status_item{
border: 2px solid rgb(233, 236, 239);
border-radius: 5px;
width: 200px;
height: 20px;
display: block;
margin: auto;
padding: 5px;
margin-top: 10px;
padding-top: 10px;
}
.info{
width: 300px;
}
.room, .theme, .time_input{
color: #325a46;
margin-top: 10px;
height: 20px;
font-family: "Open Sans", "Helvetica Neue", Arial, sans-serif;
}
.recd{
border-color:#46c765;
color: white;
background-color: #46c765;
}
.sent{
border-color:royalblue;
color: white;
background-color: royalblue;
}
.reject{
border-color:#fa3333;
color: white;
background-color: #fa3333;
}
.incoming{
cursor: pointer;
border: 2px solid #325a46;
border-radius: 5px;
width: 200px;
height: 30px;
display: block;
margin: auto;
padding: 5px;
margin-top: 5px;
```

```
color: #325a46;
background-color: white;
font-size: 14px;
}
.submit{
display: block;
border-color: #46c765;
background-color: #46c765;
color: #ffffff;
height: 50px;
width: 300px;
text-align: center;
font-size: 18px;
font-weight: bold;
margin:auto;
margin-top:40px;
}
.request_creating{
display: flex;
margin:20px;
color:#7e7e7e;
flex-wrap: wrap;
flex-direction: column;
}
.request_creating div{
margin-left: 30px;
margin-bottom: 40px;
}
.page-wrapper_mini{
width:700px;
margin-top: 50px;
padding-bottom: 20px;
}
.theme_input{
width: 400px;
height: 25px;
}
.name_input{
font-size: 14px;
}
.list {
list-style-type: none;
margin: 0;
opacity: 0;
padding: 4px 7px;
```

```
position: absolute;
margin-left: -1000px;
background: white;
box-shadow: 1px 1px 6px rgb(195, 195, 195);
border-radius: 2px;
-webkit-transition: opacity 0.4s;
transition: opacity 0.4s;
}

.list li {
padding: 3px 5px;
cursor: pointer;
}

.list li:hover {
background: lightgray;
}

.show .list {
margin-left: 0;
opacity: 1;
}

.list li .amount {
display: inline-block;
padding: 2px 3px;
background-color: rgb(83, 177, 211);
border-radius: 3px;
margin: 0 0 3px 10px;
font-size: 65%;
color: white;
}
body {
font-family: Arial, sans-serif;
}

input {
padding: 3px 5px;
border: 1px solid gray;
border-radius: 3px;
}
```

Приложение 4

HTML-код настройки времени сессии

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../../js/session_settings.js"></
   script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<section class="time_section">
<h4>Время проведения экзаменов</h4>
<div class="time">
<div>
Самый ранний экзамен может начаться в:
<input type="time" name="time1" id="time1" class="time__item"
   value="10:00">
</div>
<p>
<div>
Самый поздний экзамен может начаться в:
<input type="time" name="time2" id="time2" class="time__item"
   value="16:00">
</div>
</div>
</section>
```

```
<button type="submit" id="submit_time" class="submit">Отправит
    ь форму</button>

<button class="back_button" onclick="window.location.href = '/
    session';">Назад</button>
</div>
</body>
</html>
```

Приложение 5

HTML-код настройки списка преподавателей

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../../js/session_settings.js"></
   script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<h4>Список преподавателей</h4>
Пример: i.ivanov@edu.spbstu.ru <br><br>
<div class="classroom_param">
<label for="teacherfile"></label>
<input type="file" id="teacherfile" name="teacherfile"/>
</div>
</section>
<button type="submit" id="submit_teacher" class="submit">Отпра
   вить форму</button>

<button class="back_button" onclick="window.location.href = '/
   session';">Назад</button>
</div>
</body>
</html>
```

Приложение 6

HTML-код настройки списка экзаменов

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../../js/session_settings.js"></
   script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<h4>Список экзаменов</h4>
Формат: номер группы, преподаватель, код дисциплины <br>
Пример: 3530903/70302, a.ivanov@edu.spbstu.ru, ТРПО <br><br>
<div class="classroom_param">
<label for="examfile"></label>
<input type="file" id="examfile" name="examfile"/>
</div>
</section>
<button type="submit" id="submit_exam" class="submit">Отправит
   ь форму</button>

<button class="back_button" onclick="window.location.href = '/
   session';">Назад</button>
</div>
</body>
</html>
```

Приложение 7

HTML-код настройки дат сессии

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../../js/session_settings.js"></
   script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<section class="day_section">
<h4>Даты проведения экзаменов</h4>
<div id="calendar">
<div class="item" id="dates0">
<div class="date__items">
с
<input type="date" name="date1" class="date__item date1">
по
<input type="date" name="date2" class="date__item date2">
</div>
<div class="del" id="del_0"> &#10006;</div>
</div>
</div>
<button id="add_new_date" type="button">
Добавить ещё один временной интервал
</button>
</section>
```

```
<button type="submit" id="submit_date" class="submit">Отправит
    ь форму</button>


<button class="back_button" onclick="window.location.href = '/
    session';">Назад</button>
</div>
</body>
</html>
```

Приложение 8

HTML-код настройки списка аудиторий

```
<!DOCTYPE html>
<html dir="ltr" lang="ru" xml:lang="ru" xmlns:th="http://www.
   w3.org/1999/xhtml">
<head>
<title>ИКНТ: настройки сессии</title>
<meta http-equiv="Content-Type" content="text/html; charset=
   UTF-8"/>
<meta name="_csrf" th:content="${_csrf.token}"/>
<meta name="_csrf_header" th:content="${_csrf.headerName}"/>
<link href="../../teacher_wish.css" rel="stylesheet"/>
<script type="module" src="../../js/session_settings.js"></
   script>
<meta http-equiv="Content-Type" content="text/html; charset=
   utf-8"/>
<meta name="keywords" content="ИКНТ: настройки сессии"/>
<meta name="robots" content="noindex"/>
<meta name="viewport" content="width=device-width, initial-
   scale=1.0">
</head>
<body>
<div class="page-wrapper">
<h2>Настройки сессии</h2>
<h4>Список аудиторий</h4>
Формат: номер аудитории, количество мест, наличие проектора, н
   аличие компьютеров <br>
Пример: 319,60,true,false; <br><br>
<div class="classroom_param">
<label for="classfile"></label>
<input type="file" id="classfile" name="classfile"/>
</div>
</section>
<button type="submit" id="submit_class" class="submit">Отправи
   ть форму</button>

<button class="back_button" onclick="window.location.href = '/
   session';">Назад</button>
</div>
</body>
</html>
```

Приложение 9

Скрипт с запросами для настройки сессии

```
document.addEventListener("DOMContentLoaded", () => {

const classroom = document.getElementsByClassName("
   hidden_checkbox");
const days = document.getElementById("calendar");
const add_days = document.getElementById("add_new_date");
const submit_exam = document.getElementById("submit_exam");
const submit_class = document.getElementById("submit_class");
const submit_time = document.getElementById("submit_time");
const submit_date = document.getElementById("submit_date");
const submit_teacher = document.getElementById("submit_teacher
   ");

let date_count = 0;
let class_count = 1;
let dels = [];
dels[0] = document.getElementById("del_0");
if (dels[0]){
dels[0].onclick = () => {
document.getElementById('dates' + 0).style.display = "none";
};
}
if (add_days){
add_days.onclick = () => {
date_count++;
days.innerHTML += (createNewDays(date_count));
days.onclick = function (e) {
if (e.target.className === "del") {
e.target.parentElement.remove();
}
};
};
}

if (submit_class){
submit_class.onclick = () => {
const classroomFile = document.getElementById("classfile");
submitClass(classroomFile.files[0]);
};
}
```

```
if (submit_time){
submit_time.onclick = () => {
const time1 = document.getElementById("time1");
const time2 = document.getElementById("time2");
submitTime(time1.value, time2.value);
};
}
if (submit_exam){
submit_exam.onclick = () => {
const examFile = document.getElementById("examfile");
submitExam(examFile.files[0]);
};
}
if (submit_teacher){
submit_teacher.onclick = () => {
const teacherFile = document.getElementById("teacherfile");
submitTeacher(teacherFile.files[0]);
};
}
if (submit_date){
submit_date.onclick = () => {
const daysIntervals = document.getElementsByClassName("
   date__item");
let daysarr = [];
if(daysIntervals.length === 0) {
daysarr = null;
} else {
for (let i=0; i<daysIntervals.length; i++){
daysarr.push(daysIntervals.item(i).value);
}
}
submitDate(daysarr);
};
}
}
);

function createNewDays(date_count) {
return '<div class="item" id="dates' + date_count + '">\n' +
'                    <div class="date__items">\n' +
'                        c \n' +
'                        <input type="date" name="date1" class
   ="date__item">\n' +
'                        до \n' +
```

```
'                        <input type="date" name="date2" class
   ="date__item">\n' +
'                    </div>\n' +
'                    <div class="del" id="del_' + date_count +
    '"> &#10006;</div>\n' +
'                </div>'
};
const token = getMeta("_csrf");
const header = getMeta("_csrf_header");
function getMeta(metaName) {
const metas = document.getElementsByTagName('meta');

for (let i = 0; i < metas.length; i++) {
if (metas[i].getAttribute('name') === metaName) {
return metas[i].getAttribute('content');
}
}

return '';
}
function createNewClassroom() {
return '<div class="item">\n' +
'                        Введите номер аудитории:\n' +
'                        <input type="number" name="
   classroom_number">\n' +
'                    </div>'
};
function readFile(input) {
let file = input;

let reader = new FileReader();

reader.readAsText(file);

reader.onload = function() {
let res = reader.result;
console.log(res);
return res;
};

reader.onerror = function() {
console.log(reader.error);
};

}
```

```
function submitSession(classroomFile,  time1, time2, daysarr)
   {
var request = new XMLHttpRequest();
console.log(daysarr);
request.open('POST', '/update', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
//            let fileText = readFile(classroomFile);
//            console.log(fileText);
frm.append('time1', time1);
console.log(time1);
frm.append('time2', time2);
console.log(time2);
frm.append('days', daysarr);
console.log("SEND FRM");
let fileText;
let reader = new FileReader();
reader.readAsText(classroomFile);
reader.onload = function() {
fileText = reader.result;
console.log(fileText);
frm.append('classroomFile', fileText);
request.send(frm)
};
// window.open('/sended','_top');
};
function submitTime(time1, time2) {
var request = new XMLHttpRequest();
request.open('POST', '/update_time', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
frm.append('time1', time1);
console.log(time1);
frm.append('time2', time2);
console.log(time2);
```

```
request.send(frm);
window.open('/session','_top');
};
function submitClass(classroomFile) {
var request = new XMLHttpRequest();
request.open('POST', '/update_class', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
console.log("SEND FRM");
let fileText;
let reader = new FileReader();
reader.readAsText(classroomFile);
reader.onload = function() {
fileText = reader.result;
console.log(fileText);
frm.append('classroomFile', fileText);
request.send(frm)
};
window.open('/session','_top');
};
function submitTeacher(teacherFile) {
var request = new XMLHttpRequest();
request.open('POST', '/update_teacher', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
console.log("SEND FRM");
let fileText;
let reader = new FileReader();
reader.readAsText(teacherFile);
reader.onload = function() {
fileText = reader.result;
console.log(fileText);
frm.append('teacherFile', fileText);
request.send(frm)
};
```

```
window.open('/session','_top');
};
function submitExam(examFile) {
var request = new XMLHttpRequest();
request.open('POST', '/update_exam', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
console.log("SEND FRM");
let fileText;
let reader = new FileReader();
reader.readAsText(examFile);
reader.onload = function() {
fileText = reader.result;
console.log(fileText);
frm.append('examFile', fileText);
request.send(frm)
};
window.open('/session','_top');
};
function submitDate(daysarr) {
var request = new XMLHttpRequest();
console.log(daysarr);
request.open('POST', '/update_date', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
frm.append('days', daysarr);
request.send(frm)
window.open('/session','_top');
}
```

Приложение 10

Скрипт с запросами для записи пожеланий преподавателей

```
document.addEventListener("DOMContentLoaded", () => {

const classroom = document.getElementsByClassName("
   hidden_checkbox");
const classroom_param = document.getElementsByClassName("
   classroom_param");
const concrete_classroom = document.getElementsByClassName("
   concrete_classroom");
const days = document.getElementById("calendar");
const rooms = document.getElementById("classroom_blocks");
const add_days = document.getElementById("add_new_date");
const add_classroom = document.getElementById("add_classroom")
   ;
const submit = document.getElementById("submit");
const submit_class = document.getElementById("submit_class");
const submit_date = document.getElementById("submit_date");
const submit_time = document.getElementById("submit_time");
const submit_dof = document.getElementById("submit_dof");

let date_count = 0;
let class_count = 1;
let dels = [];
dels[0] = document.getElementById("del_0");
if(dels[0]){
dels[0].onclick = () => {
document.getElementById('dates' + 0).style.display = "none";
};
}
if (classroom){
classroom.item(0).onclick = () => {
classroom_param.item(0).style.position = "initial";
concrete_classroom.item(0).style.position = "absolute";
};


classroom.item(1).onclick = () => {
classroom_param.item(0).style.position = "absolute";
concrete_classroom.item(0).style.position = "initial";
};
}
```

```
if (add_days){
add_days.onclick = () => {
date_count++;
days.innerHTML += (createNewDays(date_count));
days.onclick = function (e) {
if (e.target.className === "del") {
e.target.parentElement.remove();
}
};
};
}
if (add_classroom){
add_classroom.onclick = () => {
if (class_count < 3) {
class_count++;
rooms.innerHTML += (createNewClassroom());
if (class_count === 3) {
add_classroom.style.position = "absolute";
add_classroom.style.left = -9999 + "px";
}
}
};
}

if (submit_class){
submit_class.onclick = () => {
const computer = document.getElementById("comp");
const projector = document.getElementById("proector");
const classroom_cnt = document.getElementById("classroom_cnt")
   ;

submitWishClass(computer.value, projector.value, classroom_cnt
   .value);

};
}
if (submit_date){
submit_date.onclick = () => {
const daysIntervals = document.getElementsByClassName("
   date__item");
let daysarr = [];
if(daysIntervals.length === 0) {
daysarr = null;
} else {
for (let i=0; i<daysIntervals.length; i++){
```

```
daysarr.push(daysIntervals.item(i).value);
}
}
submitWishDate(daysarr);

};
}
if (submit_dof){
submit_dof.onclick = () => {
const computer = document.getElementById("comp");
const projector = document.getElementById("proector");
const classroom_cnt = document.getElementById("classroom_cnt")
   ;
const dayOfWeek = document.getElementsByClassName("
   dayOfWeekOption");
const daysIntervals = document.getElementsByClassName("
   date__item");
const time1 = document.getElementById("time1");
const time2 = document.getElementById("time2");
let daysarr = [];
if(daysIntervals.length === 0) {
daysarr = null;
} else {
for (let i=0; i<daysIntervals.length; i++){
daysarr.push(daysIntervals.item(i).value);
}
}
submitWishDOF(computer.value, projector.value, classroom_cnt.
   value, dayOfWeek, time1.value, time2.value, daysarr);

};
}
if (submit_time){
submit_time.onclick = () => {
const time1 = document.getElementById("time1");
const time2 = document.getElementById("time2");
submitWishTime(time1.value, time2.value);

};
}
if (submit){
submit.onclick = () => {
const computer = document.getElementById("comp");
const projector = document.getElementById("proector");
```

```
const classroom_cnt = document.getElementById("classroom_cnt")
   ;
const dayOfWeek = document.getElementsByClassName("
   dayOfWeekOption");
const daysIntervals = document.getElementsByClassName("
   date__item");
const time1 = document.getElementById("time1");
const time2 = document.getElementById("time2");
let daysarr = [];
if(daysIntervals.length === 0) {
daysarr = null;
} else {
for (let i=0; i<daysIntervals.length; i++){
daysarr.push(daysIntervals.item(i).value);
}
}
submitWish(computer.value, projector.value, classroom_cnt.
   value, dayOfWeek, time1.value, time2.value, daysarr);

};
}



}

);

function createNewDays(date_count) {
return '<div class="item" id="dates' + date_count + '">\n' +
'                    <div class="date__items">\n' +
'                        c \n' +
'                        <input type="date" name="date1" class
   ="date__item">\n' +
'                        до \n' +
'                        <input type="date" name="date2" class
   ="date__item">\n' +
'                    </div>\n' +
'                    <div class="del" id="del_' + date_count +
    '"> &#10006;</div>\n' +
'                </div>'
};
const token = getMeta("_csrf");
const header = getMeta("_csrf_header");
function getMeta(metaName) {
const metas = document.getElementsByTagName('meta');
```

```
for (let i = 0; i < metas.length; i++) {
if (metas[i].getAttribute('name') === metaName) {
return metas[i].getAttribute('content');
}
}

return '';
}
function createNewClassroom() {
return '<div class="item">\n' +
'                        Введите номер аудитории:\n' +
'                        <input type="number" name="
   classroom_number">\n' +
'                    </div>'
};
function submitWish(computer, projector, minSitsCount,
   daysOfWeek, time1, time2, daysarr) {
var request = new XMLHttpRequest();
console.log(daysarr);
request.open('POST', '/create', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
let i = 0;
let dayOfWeek = [];
while (i < 6) {
if (daysOfWeek[i].checked){
dayOfWeek.push(daysOfWeek[i].value);

}
i++;
}
frm.append('computer', computer);
console.log(computer);
frm.append('projector', projector);
console.log(projector);
frm.append('minSitsCount', minSitsCount);
console.log(minSitsCount);
frm.append('daysOfWeek', dayOfWeek);
console.log(dayOfWeek);
```

```
frm.append('time1', time1);
frm.append('time2', time2);
if(daysarr.length === 0){
frm.append('days', ["",""]);
} else {
frm.append('days', daysarr);
}
console.log(daysarr);
console.log(computer);
request.send(frm)
window.open('/sended','_top');
};
function submitWishClass(computer, projector, minSitsCount) {
var request = new XMLHttpRequest();
request.open('POST', '/create_class', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
frm.append('computer', computer);
console.log(computer);
frm.append('projector', projector);
console.log(projector);
frm.append('minSitsCount', minSitsCount);
console.log(minSitsCount);
request.send(frm)
window.open('/sended','_top');
};
function submitWishDOF(codaysOfWeek) {
var request = new XMLHttpRequest();
request.open('POST', '/create', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
let i = 0;
let dayOfWeek = [];
while (i < 6) {
if (daysOfWeek[i].checked){
```

```
dayOfWeek.push(daysOfWeek[i].value);

}
i++;
}
frm.append('daysOfWeek', dayOfWeek);
console.log(dayOfWeek);
request.send(frm)
window.open('/sended','_top');
};
function submitWishTime(time1, time2) {
var request = new XMLHttpRequest();
request.open('POST', '/create_time', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
frm.append('time1', time1);
frm.append('time2', time2);
request.send(frm)
window.open('/sended','_top');
};
function submitWishDate(daysarr) {
var request = new XMLHttpRequest();
console.log(daysarr);
request.open('POST', '/create_date', true);
request.setRequestHeader(header, token);
request.addEventListener('readystatechange', function () {
if ((request.readyState === 4) && (request.status === 200)) {
console.log("checked");
}
});
var frm = new FormData();
if(daysarr.length === 0){
frm.append('days', ["",""]);
} else {
frm.append('days', daysarr);
}
console.log(daysarr);
request.send(frm)
window.open('/sended','_top');
};
```

Приложение 11

WebSecurityConfig класс

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.security.config.annotation.
    authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.
    builders.HttpSecurity;
import org.springframework.security.config.annotation.web.
    configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.
    configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.
    BCryptPasswordEncoder;
import org.springframework.security.crypto.password.
    PasswordEncoder;

@EnableWebSecurity
public class WebSecurityConfig extends
    WebSecurityConfigurerAdapter {
@Override
protected void configure(HttpSecurity http) throws Exception {
http
.headers().frameOptions().sameOrigin().and()
.cors().and()
.authorizeRequests()
.antMatchers("/session/**","/session/**/**").hasAnyRole("ADMIN
    ")
.antMatchers("/teacher_wish/**", "/sended/**").hasAnyRole("
    TEACHER")
.anyRequest().authenticated()
.and().formLogin().permitAll()
.and().logout().permitAll();
}

@Autowired
public void configure(AuthenticationManagerBuilder auth)
    throws Exception {
auth.inMemoryAuthentication()
.passwordEncoder(passwordEncoder())
```

```
.withUser("user").password(passwordEncoder().encode("pass")).
   roles("TEACHER")
.and().withUser("user2").password(passwordEncoder().encode("
   pass2")).roles("ADMIN");
}

@Bean
public PasswordEncoder passwordEncoder() {
return new BCryptPasswordEncoder();
}
}
```

Приложение 12

SessionService класс

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import ru.technopolis.dao.SessionDao;
import ru.technopolis.model.Classroom;
import ru.technopolis.model.Exam;
import ru.technopolis.model.Teacher;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

@Controller
public class SessionService {

public SessionDao dao;

@Autowired
public SessionService(SessionDao dao) {
this.dao = dao;
}

@RequestMapping(value = "/session")
public String index(Model model) {
return "session/index";
}

@RequestMapping(value = "/session/date_settings")
public String dateSettings(Model model) {
return "session/date_settings/index";
}
@RequestMapping(value = "/session/class_settings")
```

```
public String classSettings(Model model) {
return "session/class_settings/index";
}
@RequestMapping(value = "/session/time_settings")
public String timeSettings(Model model) {
return "session/time_settings/index";
}
@RequestMapping(value = "/session/exam_settings")
public String examSettings(Model model) {
return "session/exam_settings/index";
}
@RequestMapping(value = "/session/teacher_settings")
public String teacherSettings(Model model) {
return "session/teacher_settings/index";
}

@RequestMapping(value = "/update", method = RequestMethod.POST
   )
public @ResponseBody
String update(
@RequestParam(name = "classroomFile") String classroomFile,
@RequestParam(name = "time1") String time1,
@RequestParam(name = "time2") String time2,
@RequestParam(name = "days") String[] days
) {
System.out.println("HEY...");
List<Integer> times = new ArrayList<>();
for (int i = Integer.parseInt(time1.substring(0, 2)); i <
   Integer.parseInt(time2.substring(0, 2)); i += 2) {
times.add(i);
}
List<Date> dates = new ArrayList<>();
DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
Calendar c = Calendar.getInstance();
System.out.println(days.length);
for (int i = 0; i < days.length / 2; i++) {
Date date1 = null;
Date date2 = null;
try {
if (!(days.length < 2 * i || days[2 * i] == null)) {
date1 = format.parse(days[2 * i]);
}
} catch (ParseException e) {
date1 = null;
e.printStackTrace();
```

```
}
try {
if (!(days.length < 2 * i + 1 || days[2 * i + 1] == null)) {
date2 = format.parse(days[2 * i + 1]);
}
} catch (ParseException e) {
date2 = null;
e.printStackTrace();
}

while (date1.compareTo(date2) <= 0) {
if (date1.getDay() != 0) {
dates.add(date1);
}
c.setTime(date1);
c.add(Calendar.DATE, 1);
date1 = c.getTime();
}
}
List<Classroom> classrooms = new ArrayList<>();
String[] lines = classroomFile.split("\\n");
for (String line : lines) {
//Формат: номер аудитории, количество мест, наличие проектора,
    наличие компьютеров
String[] str = line.split(",");
Integer num = Integer.parseInt(str[0]);
Integer sits = Integer.parseInt(str[1]);
boolean projector = Boolean.valueOf(str[2]);
boolean comp = Boolean.valueOf(str[3]);
Classroom cl = new Classroom(num, sits, projector, comp);
System.out.println("Class: " + cl.toString());
classrooms.add(cl);
}
dao.createDates(dates);
dao.createTimes(times);
dao.createClassrooms(classrooms);
System.out.println("SLEEP");
try {
Thread.sleep(10000);
} catch (InterruptedException e) {
e.printStackTrace();
}
Date[] x = dao.getSessionDates();
for (Date d : x) {
System.out.println(d);
```

```
}
List<Classroom> cl = dao.getClassrooms();
for (Classroom cls : cl) {
System.out.println(cls.toString());
}
return "session/test";
// return "index";
}
@RequestMapping(value = "/update_exam", method = RequestMethod
   .POST)
public @ResponseBody
String update_exam(
@RequestParam(name = "examFile") String examFile
) {
String[] lines = examFile.split("\\n");
for (String line : lines) {
//Формат: номер группы, преподаватель, дисциплина
String[] str = line.split(",");
String group = str[0];
String teacher = str[1];
String course = str[2];
Exam ex = new Exam(group,teacher,course);
System.out.println("Class: " + ex.toString());
dao.createExam(ex);
}
return "session";
}
@RequestMapping(value = "/update_teacher", method =
   RequestMethod.POST)
public @ResponseBody
String update_teacher(
@RequestParam(name = "teacherFile") String examFile
) {
String[] lines = examFile.split("\\n");
for (String line : lines) {
dao.createTeacher(new Teacher(line));
}
return "session";
}
@RequestMapping(value = "/update_time", method = RequestMethod
   .POST)
public @ResponseBody
String update_time(
@RequestParam(name = "time1") String time1,
@RequestParam(name = "time2") String time2
```

```
) {
for (int i = Integer.parseInt(time1.substring(0, 2)); i <
   Integer.parseInt(time2.substring(0, 2)); i += 2) {
dao.createTime(i);
}
return "session";
}
@RequestMapping(value = "/update_class", method =
   RequestMethod.POST)
public @ResponseBody
String updateClass(
@RequestParam(name = "classroomFile") String classroomFile
) {
String[] lines = classroomFile.split("\\n");
for (String line : lines) {
//Формат: номер аудитории, количество мест, наличие проектора,
    наличие компьютеров
String[] str = line.split(",");
Integer num = Integer.parseInt(str[0]);
Integer sits = Integer.parseInt(str[1]);
boolean projector = Boolean.valueOf(str[2]);
boolean comp = Boolean.valueOf(str[3]);
Classroom cl = new Classroom(num, sits, projector, comp);
System.out.println("Class: " + cl.toString());
dao.createClassroom(cl);
}
return "session";
}

@RequestMapping(value = "/update_date", method = RequestMethod
   .POST)
public @ResponseBody
String updateDate(
@RequestParam(name = "days") String[] days
) {
DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
Calendar c = Calendar.getInstance();
System.out.println(days.length);
for (int i = 0; i < days.length / 2; i++) {
Date date1 = null;
Date date2 = null;
try {
if (!(days.length < 2 * i || days[2 * i] == null)) {
date1 = format.parse(days[2 * i]);
}
```

```
} catch (ParseException e) {
date1 = null;
e.printStackTrace();
}
try {
if (!(days.length < 2 * i + 1 || days[2 * i + 1] == null)) {
date2 = format.parse(days[2 * i + 1]);
}
} catch (ParseException e) {
date2 = null;
e.printStackTrace();
}

while (date1.compareTo(date2) <= 0) {
if (date1.getDay() != 0) {
dao.createDate(date1);
}
c.setTime(date1);
c.add(Calendar.DATE, 1);
date1 = c.getTime();
}
}

return "session";
}
}
```

Приложение 13

WishService класс

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import ru.technopolis.dao.SessionDao;
import ru.technopolis.dao.WishDAO;
import ru.technopolis.model.Wish;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

@Controller
public class WishService {
private int minTime;
private int maxTime;
private WishDAO dao;
private SessionDao sessionDao;

@Autowired
public WishService(WishDAO dao
) {
this.dao = dao;
}

@RequestMapping(value = "/teacher_wish")
public String index(Model model) {
model.addAttribute("wishes", dao.getWishes());
return "teacher_wish/index";
}



@RequestMapping(value = "/create", method = RequestMethod.POST
   )
public @ResponseBody
```

```
String create(
@RequestParam(name = "computer") String computerClass,
@RequestParam(name = "projector") String projector,
@RequestParam(name = "minSitsCount") Integer minSitsCount,
@RequestParam(name = "daysOfWeek") Integer[] dof,
@RequestParam(name = "time1") String time1,
@RequestParam(name = "time2") String time2,
@RequestParam(name = "days") String[] days
) {
System.out.println("CREATE!");
System.out.println("dates " + Arrays.toString(sessionDao.
   getSessionDates()));
System.out.println("days " + Arrays.toString(sessionDao.
   getSessionDates()));
List<Integer> times = new ArrayList<>();
for (int i = minTime; i < maxTime; i += 2) {
if (i >= Integer.parseInt(time1.substring(0, 2)) &&
i <= Integer.parseInt(time2.substring(0, 2))) {
times.add(i);
}
}
List<Integer> daysOfWeek = List.of(dof);
DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
Date endOfSession = sessionDao.getSessionDates()[1];
Date date = sessionDao.getSessionDates()[0];
Calendar c = Calendar.getInstance();
while (date.compareTo(endOfSession) <= 0) {
c.setTime(date);
c.add(Calendar.DATE, 1);
date = c.getTime();
for (int i = 0; i < days.length / 2; i++) {
Date date1 = null;
Date date2 = null;
try {
if (!(days.length < 2 * i || days[2 * i] == null)) {
date1 = format.parse(days[2 * i]);
}
} catch (ParseException e) {
date1 = null;
e.printStackTrace();
}
try {
if (!(days.length < 2 * i + 1 || days[2 * i + 1] == null)) {
date2 = format.parse(days[2 * i + 1]);
}
```

```
} catch (ParseException e) {
date2 = null;
e.printStackTrace();
}

if (((days[2 * i] == null || date.after(date1) || date.equals(
   date1)))
&& ((days[2 * i + 1] == null) || date.before(date2) || date.
   equals(date2))
) {
Date finalDate = date;
if (daysOfWeek.contains(finalDate.getDay())) {
for (Integer time : times) {
System.out.println("CREATE " + date);
dao.create(date, time, Boolean.valueOf(computerClass), Boolean
   .valueOf(projector), minSitsCount);
}
}
}
}
}
return "sended/sended_teacher";
}

@RequestMapping(value = "/read", method = RequestMethod.POST)
public @ResponseBody
ResponseEntity<Wish[]> read() {
System.out.println("READ");
return ResponseEntity.ok(dao.getWishes());
}
}
```

Приложение 14

SessionDao класс

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.
   Transactional;
import ru.technopolis.model.*;
import ru.technopolis.repository.*;

import java.util.Date;
import java.util.List;

@Component
public class SessionDao {
private SessionDatesRepository sessionDatesRepository;
private final SessionTimeRepository sessionTimeRepository;
private final ExamRepository examRepository;
private final TeacherRepository teacherRepository;
private final ClassroomRepository classroomRepository;

@Autowired
public SessionDao(SessionDatesRepository
   sessionDatesRepository,
SessionTimeRepository sessionTimeRepository,
ExamRepository examRepository,
TeacherRepository teacherRepository,
ClassroomRepository classroomRepository) {
this.sessionDatesRepository = sessionDatesRepository;
this.sessionTimeRepository = sessionTimeRepository;
this.classroomRepository = classroomRepository;
this.teacherRepository = teacherRepository;
this.examRepository = examRepository;
}

@Transactional
public SessionDao createTimes(List<Integer> time) {
for (Integer t : time) {
if (sessionTimeRepository.countTimes(t) == 0) {
sessionTimeRepository.save(new SessionTime(t));
}
}
return this;
```

```
}

@Transactional
public SessionDao createTime(Integer t) {
if (sessionTimeRepository.countTimes(t) == 0) {
sessionTimeRepository.save(new SessionTime(t));
}
return this;
}

@Transactional
public SessionDao createClassrooms(List<Classroom> classrooms)
    {
for (Classroom classroom : classrooms) {
classroomRepository.save(classroom);
}
return this;
}

@Transactional
public SessionDao createClassroom(Classroom classroom) {
classroomRepository.save(classroom);
return this;
}

@Transactional
public SessionDao createDates(List<Date> days) {
for (Date d : days) {
if (sessionDatesRepository.countDates(d) == 0) {
sessionDatesRepository.save(new SessionDates(d));
}
}
return this;
}

@Transactional
public SessionDao createDate(Date d) {
if (sessionDatesRepository.countDates(d) == 0) {
sessionDatesRepository.save(new SessionDates(d));
}
return this;
}

@Transactional
public SessionDao createExams(List<Exam> exams) {
```

```
for (Exam d : exams) {
examRepository.save(d);
}
return this;
}
@Transactional
public SessionDao createExam(Exam exam) {
examRepository.save(exam);
return this;
}

@Transactional
public SessionDao createTeacher(Teacher t) {
teacherRepository.save(t);
return this;
}

public Date[] getSessionDates() {
return sessionDatesRepository.sessionDates().toArray(new Date
   [0]);
}

public Integer getMinTime() {
return sessionTimeRepository.getMinTime();
}

public Integer getMaxTime() {
return sessionTimeRepository.getMaxTime();
}

public List<Classroom> getClassrooms() {
return classroomRepository.allClassroom();
}
}
```

Приложение 15

WishDAO класс

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.context.
   SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.
   Transactional;
import ru.technopolis.model.Wish;
import ru.technopolis.repository.WishRepository;

import java.util.Date;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

@Component
public class WishDAO {

private final WishRepository repo;

@Autowired
public WishDAO(WishRepository repo) {
this.repo = repo;
}

@Transactional
public Wish create(Date day,
Integer time,
Boolean computerClass,
Boolean projector,
Integer minSitsCount) {
Wish wish = new Wish(getUserName(),
time,
day,
computerClass,
projector,
minSitsCount);
repo.save(wish);
return wish;
}

public ConcurrentMap<String, Wish> getWishList() {
```

```
return wishList;
}

private static ConcurrentMap<String, Wish> wishList = new
   ConcurrentHashMap<>();

private Wish getWish() {
return wishList.get(getUserName());
}

public static String getUserName() {
return
SecurityContextHolder.getContext().getAuthentication().getName
   ();
}

public Wish update(Date day,
Integer time,
Boolean computerClass,
Boolean projector,
Integer minSitsCount) {
wishList.put(getUserName(),
new Wish(
getUserName(),
time,
day,
computerClass,
projector,
minSitsCount
));
return getWish();
}

public Wish[] getWishes() {
// checkDB();
Wish[] wishes = new Wish[wishList.size()];

return wishList.values().toArray(wishes);
}
}
```

Приложение 16

Classroom класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;

import javax.persistence.*;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class Classroom {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long id;

@Column(nullable = false)
public  Integer classroom;
@Column(nullable = false)
public  boolean computerClass;
@Column(nullable = false)
public  boolean projector;
@Column(nullable = false)
public  int sits_count;


public Classroom(){}

public Classroom(Integer classroom,  int sits_count, boolean
   projector, boolean computerClass) {
this.classroom = classroom;
this.computerClass = computerClass;
this.projector = projector;
this.sits_count = sits_count;
}

public String toString(){
return "Num: " + classroom + " computers: " + computerClass+
" projector: " + projector + " count of sits: " + sits_count;
}
}
```

Приложение 17

Exam класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;

import javax.persistence.*;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class Exam {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long id;

@Column(nullable = false)
public String group_id;
@Column(nullable = false)
public String teacher;
@Column(nullable = false)
public String course;

public Exam(String group_id, String teacher, String course) {
this.group_id = group_id;
this.teacher = teacher;
this.course = course;
}

public Exam() {
}

public String toString(){
return "group: " + group_id + " teacher: " + teacher+
" course: " + course;
}
}
```

Приложение 18

SessionDates класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;

import javax.persistence.*;
import java.util.Date;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class SessionDates {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long id;

@Column(nullable = false)
public Date day;

public SessionDates(Date day) {
this.day = day;
}

public SessionDates() {
}
}
```

Приложение 19

SessionTime класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;
import org.hibernate.validator.constraints.UniqueElements;

import javax.persistence.*;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class SessionTime {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long id;

public SessionTime() {
}

@Column(nullable = false)
public  Integer time;

public SessionTime(Integer time) {
this.time = time;
}
}
```

Приложение 20

Teacher класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;

import javax.persistence.*;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class Teacher {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
public Long id;
@Column(nullable = false)
public String teacher_id;

public Teacher(String teacher) {
this.teacher_id = teacher;
}

public Teacher() {
}

public String toString(){
return "Teacher: " + teacher_id;
}
}
```

Приложение 21

Wish класс

```
import lombok.Data;
import lombok.experimental.Accessors;
import org.hibernate.envers.Audited;

import javax.persistence.*;
import java.util.Date;

@Data
@Accessors(chain = true)
@Entity
@Audited
public class Wish {
@Id @GeneratedValue(strategy = GenerationType.AUTO)
public Long id;
@Column(nullable = false)
public  String teacher_id;
@Column(nullable = false)
public  Integer time;
@Column(nullable = false)
public  Date day;
@Column(nullable = false)
public  boolean computerClass;
@Column(nullable = false)
public  boolean projector;
@Column(nullable = false)
public  int minSitsCount;
public Wish(){

}
public Wish(
String teacher_id,
Integer time,
Date day,
boolean computerClass,
boolean projector,
int minSitsCount
) {
this.teacher_id = teacher_id;
this.time = time;
this.day = day;
```

```
this.computerClass = computerClass;
this.projector = projector;
this.minSitsCount = minSitsCount;
}

public String getTeacherId() {
return teacher_id;
}

public String toString() {
String res = "id: " + teacher_id +
" time: " + time +
" day: " + day +
" computerClass: " + computerClass +
" projector: " + projector +
" minSitsCount: " + minSitsCount;
return res;
}
}
```

Приложение 22

Классы репозиториев

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.technopolis.model.Classroom;
import ru.technopolis.model.SessionDates;

import java.util.Date;
import java.util.List;

@Repository
public interface ClassroomRepository extends CrudRepository<
   Classroom, Long> {
@Query("SELECT r FROM Classroom r ")
public List<Classroom> allClassroom();
}

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.technopolis.model.Exam;

import java.util.List;

@Repository
public interface ExamRepository extends CrudRepository<Exam,
   Long> {
@Query("SELECT r FROM Exam r ")
public List<Exam> allExams();
}

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.technopolis.model.Classroom;
import ru.technopolis.model.SessionDates;

import java.util.Date;
import java.util.List;

@Repository
```

```
public interface SessionDatesRepository extends CrudRepository
   <SessionDates, Long>{
@Query("SELECT day FROM SessionDates r ")
public List<Date> sessionDates();

@Query("SELECT count(day) FROM SessionDates r where day = ?1")
public Integer countDates(Date date);
}

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.technopolis.model.SessionDates;
import ru.technopolis.model.SessionTime;

import java.util.Date;
import java.util.List;

@Repository
public interface SessionTimeRepository extends CrudRepository<
   SessionTime, Long> {
@Query("SELECT time FROM SessionTime r ")
public List<Integer> sessionTime();
@Query("SELECT max(time) FROM SessionTime r ")
public Integer getMaxTime();
@Query("SELECT min(time) FROM SessionTime r ")
public Integer getMinTime();
@Query("SELECT count(time) FROM SessionTime r where time =
   ?1")
public Integer countTimes(Integer time);
}

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import ru.technopolis.model.Exam;
import ru.technopolis.model.Teacher;

import java.util.List;



@Repository
public interface TeacherRepository extends CrudRepository<
   Teacher, Long> {
@Query("SELECT r FROM Teacher r ")
```

```
public List<Exam> allTeachers();
}

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Repository;

import javax.transaction.Transactional;
import java.util.List;
import org.springframework.data.jpa.repository.Query;
import ru.technopolis.model.Wish;

@Repository
public interface WishRepository extends CrudRepository<Wish,
   Long> {

@Query("SELECT r FROM Wish r "
+ "WHERE teacher_id = ?1 ")
public List<Wish> wishesByTeacher(String teacherId);

@Query("SELECT r FROM Wish r ")
public List<Wish> selectAll();
}
```

Приложение 23

Application класс

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
   SpringBootApplication;

@SpringBootApplication
public class Application {

public static void main(String[] args) {
SpringApplication.run(Application.class, args);
}

}
```