

СОДЕРЖАНИЕ

Введение.....	3
1. Анализ и моделирование предметной области программного средства.....	4
1.1. Описание предметной области	4
1.2. Разработка функциональной модели предметной области	4
1.3. Анализ требований к разрабатываемому программному средству	6
1.4. Спецификация функциональных требований	7
1.4. Разработка информационной модели предметной области	8
2. Проектирование и конструирование программного средства	11
2.1. Постановка задачи.....	11
2.2. Архитектурные решения	11
2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства.....	14
2.4. Проектирование пользовательского интерфейса.....	18
2.5. Обоснование выбора компонентов и технологий для реализации программного средства.....	22
3. Тестирование и проверка работоспособности программного средства.....	25
4. Руководство по развертыванию и использованию программного средства	27
Заключение	37
Список использованных источников	38

ВВЕДЕНИЕ

Важнейшая задача компьютерных систем – хранение и обработка данных. Для её решения были предприняты усилия, которые привели к появлению в конце 60-х – начале 70-х годов специализированного программного обеспечения – систем управления базами данных (database management systems). СУБД позволяют структурировать, систематизировать и организовать данные для их компьютерного хранения и обработки. Невозможно представить себе деятельность современного предприятия или учреждения без использования профессиональных СУБД. Несомненно, они составляют фундамент информационной деятельности во всех сферах – начиная с производства и заканчивая финансами и телекоммуникациями [1].

Курсовой проект посвящен изучению теории и практики разработки и проектирования информационных систем с использованием баз данных. В данном курсовом проекте объектом исследования является среда WEB, как платформа приложений информационных систем.

Предметом исследования является методы разработки автоматизированных систем. Целью курсового проекта является разработка автоматизированной системы учета и регистрации успеваемости студентов.

Курсовой проект предполагает выполнение следующих задач:

- проанализировать и описать предметную область;
- спроектировать и сконструировать программное средство учета и регистрации успеваемости студентов;
- проверить работоспособность полученного приложения;
- описать руководство по развертыванию и использованию программного средства.

В системе проверки на уникальность «Антиплагиат» работа показала результат 78.05%.

1. АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

1.1. Описание предметной области

Автоматизированная система будет использоваться для учета и регистрации успеваемости студентов, что является указанием на использование системы в университетах.

Как известно, при поступлении в университет, студент определяется в систему по нескольким параметрам: типу обучения, виду обучения и группе направления.

Вид обучения – это платное или бесплатное образование.

Тип обучения – это очное или заочное.

При поступлении набираются группы студентов, так что каждая группа имеет свое уникальное название, информацию о годе создания и ожидаемом годе выпуска группы и списка студентов этой группы.

Оценки студентам ставятся на сессии, которая проходит 2 раза в год, или один раз в семестр. У одной группы за все время есть множество сессий, в зависимости от количества лет обучения. Так, если группа обучается 4 года, то у нее будет 8 сессий.

Сессия длится некоторое время, на которое определяется список предметов, по которым будут ставить оценки. На самой сессии каждый студент получает по каждому предмету оценку.

1.2. Разработка функциональной модели предметной области

Отобразим функциональную модель предметной области с использованием методологии SADT. SADT (Structured Analysis and Design Technique) представляет собой методологию анализа и проектирования систем.

Данную методологию целесообразно применять на ранних этапах жизненного цикла, для того чтобы можно было рассмотреть систему до ее воплощения. Методология SADT дает возможность сократить дорогостоящие ошибки на ранних этапах разработки системы, улучшить контакт между пользователями и разработчиками, сгладить переход от анализа к проектированию. Методология SADT – нотация IDEF0 [2].

Разработанная модель представлена на рисунке 1.1.

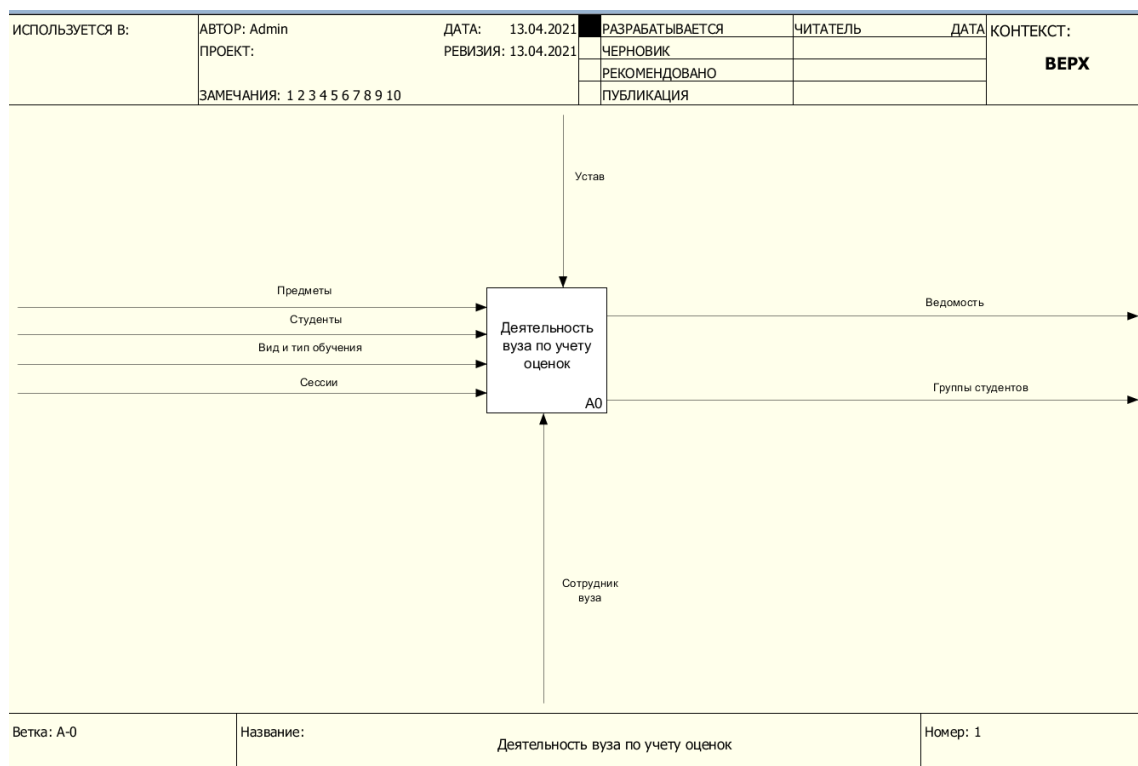


Рисунок 1.1 – Функциональная модель предметной области

Декомпозиция модели состоит из четырех процессов: определение новых групп, занесение информации о студенте, определение предметов на сессию каждой группы и выставление оценок по предметам сессии.

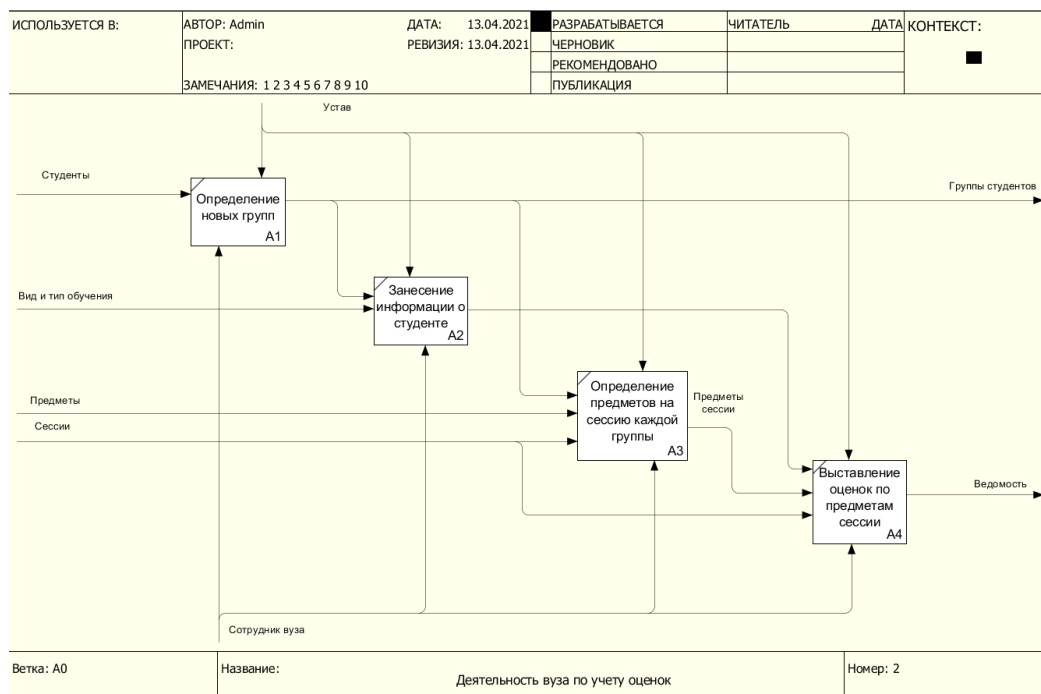


Рисунок 1.2 – Декомпозиция функциональной модели

1.3. Анализ требований к разрабатываемому программному средству

Анализ требований подразумевает их очистку, гарантирующую, что требования понимают все заинтересованные лица, а также тщательное исследование требований на предмет ошибок, пробелов и других недостатков. Кроме того, анализ включает разбиение высокоуровневых требований на более детальные, создание прототипов, анализ осуществимости и согласование приоритетов. Цель анализа — достаточно качественно и подробно описать требования, позволяющие менеджерам реалистично оценить все затраты на проект, а техническому персоналу — начать проектирование, разработку и тестирование [3].

Аналитик совместно с разработчиками должен определить, насколько возможно реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Это позволяет заинтересованным лицам понять риски, связанные с реализацией каждого требования, включая конфликты с другими требованиями, зависимость от внешних факторов и препятствия технического характера. Технически нереализуемые или очень дорогие в реализации требования можно попытаться упростить и в таком виде включить в проект для достижения бизнес-целей.

Полученные общие требования на курсовой проект. Программное средство следует разработать в архитектуре web-приложение с базой данных с использованием объектно-ориентированного языка программирования Java, современных технологий и фреймворков. В рамках работы должны быть представлены: разработка и использование собственной иерархии классов, реализация не менее 2-х паттернов проектирования, сокрытие данных (инкапсуляция), перегрузка методов, переопределение методов, параметризованные классы (шаблоны), абстрактные типы данных (интерфейсы, абстрактные классы), передача параметров по ссылке и по значению, статические методы, обработка исключительных ситуаций.

Уровни архитектуры: приложение должно быть распределено по 2-м отдельным серверам:

Сервер СУБД – СУБД для размещения *базы данных* курсового проекта выбирается студентом самостоятельно.

Сервер Приложений – используется для размещения “серверной” части приложения, представленной *Моделью* на основе ORM технологии (Hibernate/JPA), *Бизнес-логикой* приложения на основе технологий jsp+servlet или иного MVC фреймворка для разработки веб-приложений. По согласованию с руководителем разрешается использовать фреймворки для других платформ: Ruby, .Net, Python, JavaScript.

В качестве языка программирования будет использован язык java в связке с SpringFramework.

1.4. Спецификация функциональных требований

Хорошим вариантом для предоставления пользователю визуального представления функциональных требований будет предоставление диаграммы вариантов использования.



Рисунок 1.3 – Диаграмма вариантов использования

Как видно из диаграммы вариантов использования, пользоваться системой будут пользователи с одной ролью пользователь. Он может работать со всеми таблицами напрямую, что отображено в диаграмме.

Кроме того, он может работать с таблицами с сортированными данными. Т.е., к примеру, при просмотре информации о группах, он может перейти не просто на страницу сессий, а перейти на страницу сессий, относящихся только к выбранной группе. Или пример с сессиями. При переходе от них на страницу оценок можно просматривать и изменять данные как всех оценок, так и оценок выбранной сессии.

Кроме таких вариантов, пользователю будет доступен подсчет общего количества групп и студентов системы.

Вариантами использования для пользователя также является возможность просматривать средние балы студентов и предметов сессии, которые высчитываются на стороне сервера.

1.4. Разработка информационной модели предметной области

Построение информационной модели предметной области предполагает выделение сущностей, их атрибутов и первичных ключей, идентификацию связей между сущностями. Общепринятым видом графического изображения реляционной модели данных является ER-диаграмма, на которой сущности изображаются прямоугольниками, соединенные между собой связями. Такое графическое представление облегчает восприятие структуры базы данных по сравнению с текстовым описанием [4].

Ниже показаны все сущности системы с их полями.

Предмет

Код.

Название.

Сессия

Код

Название.

Семестр.

Группа.

Предметы.

Группа

Код.

Название.

Год создания.

Год выпуска.

Форма обучения

Код.

Вид.

Тип обучения.

Код.

Тип.
 Студент
 Код.
 ФИО.
 Дата рождения.
 Телефон.
 Адрес.
 Форма обучения.
 Тип обучения.
 Номер зачетки.
 Группа.
 Оценка
 Код.
 Сессия.
 Предмет.
 Оценка.
 Студент.

При создании базы данных по текущим сущностям, будем иметь следующие отношения таблиц с указанием размера полей

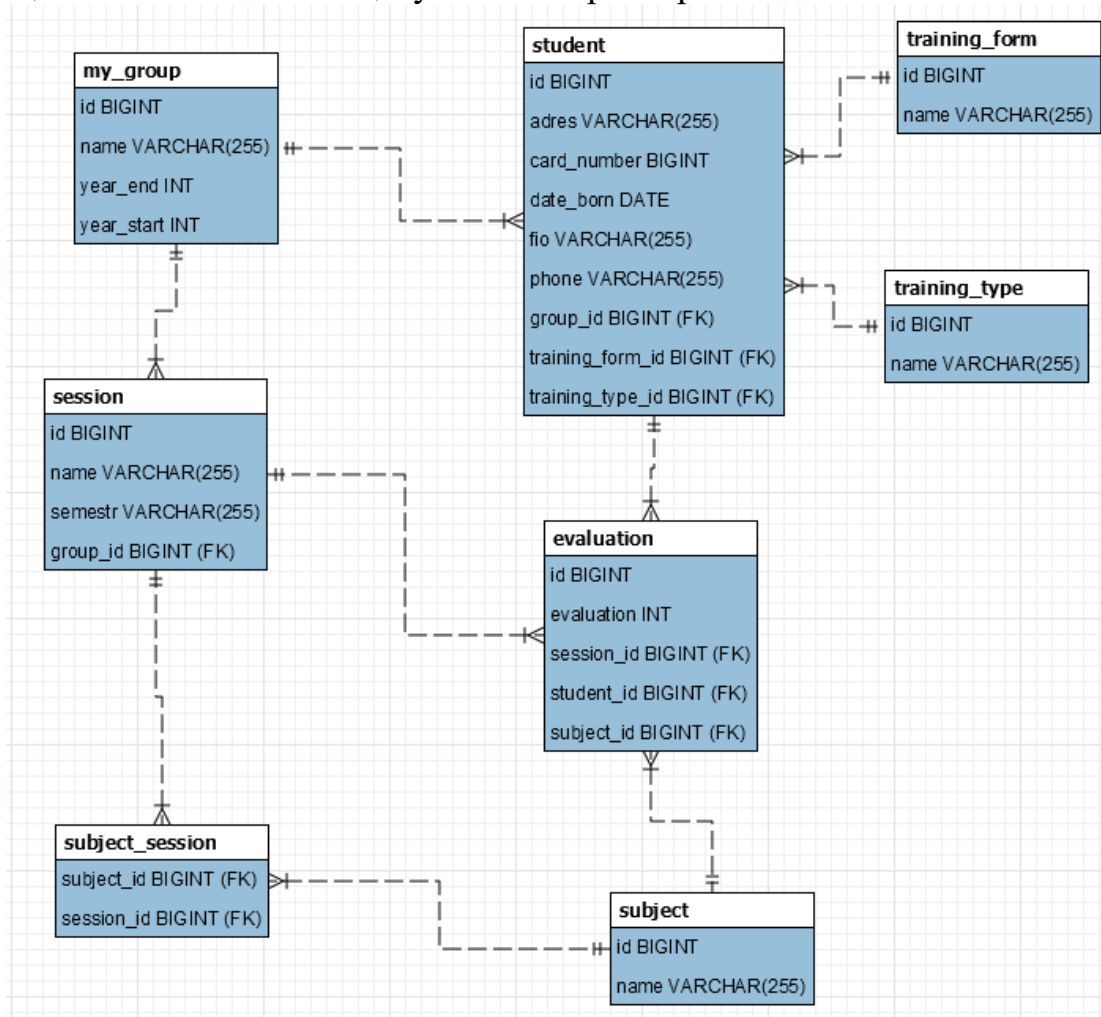


Рисунок 1.4 – Диаграмма базы данных

2. ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1. Постановка задачи

Для достижения цели данного курсового проекта, следует выделить следующие задачи, которые необходимо решить:

- Выбрать архитектуру будущего приложения
- Создать базу данных по информационной модели предыдущего раздела
- Описать алгоритмы, реализующие бизнес-логику приложения
- Спроектировать пользовательский интерфейс.
- Провести тестирование полученного приложения.
- Описать руководство по развертыванию и использованию программного средства.

Кроме того, разрабатываемое приложение должно следующим требованиям

- проектируемая база данных должна отвечать требованиям надёжности, минимальной избыточности, целостности данных
- содержать не менее 5 сущностей и должна быть приведена к третьей нормальной форме
- реализовать приложение необходимо при помощи языка программирования Java и Базы данных SQL.

2.2. Архитектурные решения

Поскольку разработка ведется на основе фреймворка спринг, разумным будет использовать архитектуру Spring MVC.

Фреймворк **Spring MVC** обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними [5].

Model (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).

View (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

Controller (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Вся логика работы Spring MVC построена вокруг **DispatcherServlet**, который принимает и обрабатывает все HTTP-запросы (из UI) и ответы на них. Рабочий процесс обработки запроса DispatcherServlet'ом проиллюстрирован на следующем изображении:

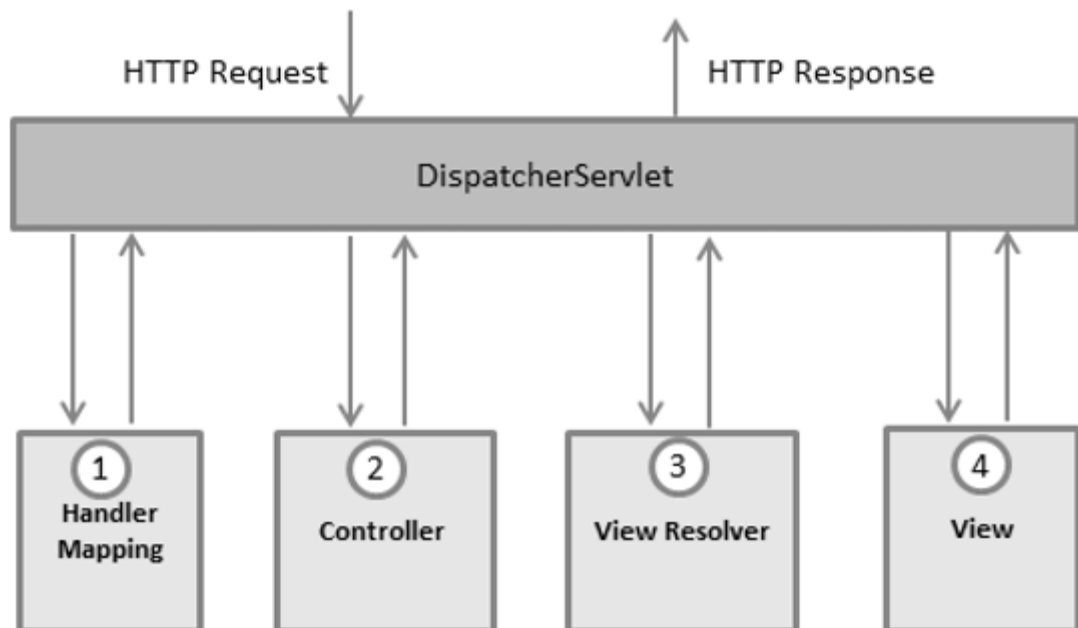


Рисунок 2.1 – DispatcherServlet

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

- После получения HTTP-запроса DispatcherServlet обращается к интерфейсу **HandlerMapping**, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.
- Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в DispatcherServlet имя Вида (View).
- При помощи интерфейса **ViewResolver** DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени.
- После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

Все вышеупомянутые компоненты, а именно, **HandlerMapping**, **Controller** и **ViewResolver**, являются частями интерфейса **WebApplicationContext** extends **ApplicationContext**, с некоторыми дополнительными особенностями, необходимыми для создания web-приложений.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций. Аннотация **@Controller** указывает, что конкретный класс является контроллером. Аннотация **@RequestMapping** используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика.

@Controller

```

@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}

```

Аннотация `Controller` определяет класс как Контроллер Spring MVC. В первом случае, `@RequestMapping` указывает, что все методы в данном Контроллере относятся к URL-адресу `"/hello"`. Следующая аннотация `@RequestMapping(method = RequestMethod.GET)` используется для объявления метода **`printHello()`** как дефолтного метода для обработки HTTP-запросов GET (в данном Контроллере). Вы можете определить любой другой метод как обработчик всех POST-запросов по данному URL-адресу.

Вы можете написать вышеуказанный Контроллер по-другому, указав дополнительные атрибуты для аннотации `@RequestMapping` следующим образом:

```

@Controller
public class HelloController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}

```

Атрибут «`value`» указывает URL, с которым мы связываем данный метод (`value = "/hello"`), далее указывается, что этот метод будет обрабатывать GET-запросы (`method = RequestMethod.GET`). Также, нужно отметить важные моменты в отношении приведённого выше контроллера:

- Вы определяете бизнес-логику внутри связанного таким образом служебного метода. Из него Вы можете вызывать любые другие методы.
- Основываясь на заданной бизнес-логике, в рамках этого метода Вы создаёте Модель (Model). Вы можете добавлять атрибуты Модели, которые будут добавлены в Вид (View). В примере выше мы создаём Модель с атрибутом «`message`».

Данный служебный метод возвращает имя Вода в виде строки `String`. В данном случае, запрашиваемый Вид имеет имя «`hello`».

2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Опишем некоторые алгоритмы работы, использующихся в реализации бизнес-логики.

Взаимодействие с БД.

Для взаимодействия с базой данных, необходимо следовать следующему алгоритму:

- 1) Создать сущность. Сущность представляет собой класс с названием, которому будет соответствовать название таблицы в базе данных и полями, соответствующими полями из базы данных
- 2) Для созданной сущности создать репозиторий. Репозиторий – это интерфейс для спринга, в котором мы описываем свои методы (если это необходимо) для работы с данными в базе данных текущей сущности.
- 3) Используя репозиторий создать класс сервиса сущности. Класс сервиса – это связующий класс, который будет получать данные из базы данных и предоставлять его контроллерам и наоборот.

Пример, как это выглядит, отображен в листинге 1.

Листинг 1. Пример классов для взаимодействия с БД.

@Entity

@Data

```
public class TrainingType extends AbstractEntity {
```

```
    private String name;
```

```
    @OneToMany(fetch = FetchType.LAZY, mappedBy =  
"trainingType")
```

```
    private List<Student> students = new ArrayList<Student>();
```

```
    @Override
```

```
    public String toString() {
```

```
        return name;
```

```
    }
```

```
}
```

@Repository

```
public interface TrainingTypeRepository extends  
JpaRepository<TrainingType, Long>{
```

```
}
```

@Service

```
public class TrainingTypeService extends CrudImpl<TrainingType> {
```

```
    public TrainingTypeRepository repository;
```

```

    @Autowired
    public TrainingTypeService(TrainingTypeRepository repository) {
        super(repository);
        this.repository = repository;
    }
}

```

Взаимодействие пользователя с данными

Для реализации взаимодействия используются контроллеры, со следующим алгоритмом

- 1) Ждем действие пользователя. В данном случае, действием называется переход пользователя на любую страницу сервиса или отправка запроса.
- 2) Получаем адрес, на который пользователь посылает запрос. К примеру, это будет адрес «/index» или «/trainingForm/edit»
- 3) В зависимости от адреса, вызываем метод контроллера. Для первого случая, вызывается метод контроллера IndexController, для второго – вызывается метод edit из контроллера TrainingFormController.
- 4) Возвращаем результат метода. Результатом любого метода является страница или пересылка на страницу. Также, в теле метода могут быть прикреплены данные для страницы.

Затем алгоритм повторяется для каждого запроса.

Ниже отобразим пример метода edit из контроллера TrainingFormController. На вход он принимает адрес пользователя и в зависимости от того, передавал ли пользователь id сущности, включает данные из базы данных или создает новую сущность, которую включает в модель ответа. После чего данные с моделью высылаются пользователю и использованием указанного шаблона.

Листинг 2. Пример метода контроллера

```

    @RequestMapping(path = { "/edit", "/edit/{id}" })
    public String edit(Model model, @PathVariable(name = "id", required
= false) Long id) {
        if (id != null) {
            TrainingForm entity = service.read(id);
            model.addAttribute("entity", entity);
        } else {
            model.addAttribute("entity", new TrainingForm());
        }
        return "trainingForm-add-edit";
    }
}

```

Вывод шаблонных данных

Вывод множественных данных осуществляется по шаблонам, которые находятся в папке templates. При отправке пользователем запроса на получения данных, и после завершения метода контроллера происходит следующее

- 1) Из папки берется шаблон с тем именем, результат которого прислал метод контроллера
- 2) Данные контроллера, который он записал в модель передаются выбранному шаблону
- 3) Шаблон через внутренние конструкторы и при необходимости циклы, на стороне сервера создает страницу по данным от конструктора
- 4) Эта созданная страница отправляется пользователю.

Для примера вывода множественных данных можно показать код вывода списка групп. Шаблон использует внутренний цикл для вывода всех элементов списка, получаемого от контроллера.

Листинг 3. Генерация страниц на стороне сервера по шаблону

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">

<head>
    <div th:replace="~{commons :: head}"></div>
    <title>Группы</title>
</head>
<body>
    <div th:replace="~{commons :: nav}"></div>
    <section class="group-list">
        <div class="container my-2">
            <div class="card">
                <div class="row">
                    <div class="card-body col">
                        <h2 class="text-center">Группы</h2>
                        <div th:switch="{list}" class="container my-5">
                            <div class="col-md-12">
                                <h2 th:case="null"> Записей не найдено </h2>
                                <div th:case="*" class="overScroll">
                                    <table class="table table-striped table-responsive-
md">

                                        <thead>
                                            <tr>
                                                <th>Название</th>
                                                <th>Год основания</th>
                                                <th>Год выпуска</th>
                                                <th>Студенты</th>
```

```

        <th>Сессии</th>
    </tr>
</thead>
<tbody>
    <tr th:each="entity : ${list}">
        <td th:text="${entity.name}"></td>
        <td th:text="${entity.yearStart}"></td>
        <td th:text="${entity.yearEnd}"></td>
        <td class="btnInTable widthLastCol" >
            <a
th:href="@{/group/getStudent/{id}(id=${entity.id})}" class="btn btn-primary ">
                Студенты
            </a>
        </td>
        <td class="btnInTable widthLastCol" >
            <a
th:href="@{/group/getSession/{id}(id=${entity.id})}" class="btn btn-primary ">
                Сессии
            </a>
        </td>
        <td class="btnInTable widthLastCol" >
            <a
th:href="@{/group/edit/{id}(id=${entity.id})}" class="btn btn-primary ">
                <i class="fas fa-
edit">Редактировать</i>
            </a>
        </td>
        <td class="btnInTable widthLastCol">
            <a
th:href="@{/group/delete/{id}(id=${entity.id})}" class="btn btn-primary">
                <i class="fas fa-trash">Удалить</i>
            </a>
        </td>
    </tr>
</tbody>
</table>
</div>
<p class="d-flex justify-content-end">
    <a href="/group/edit" class="btn btn-primary mx-
2">
        <i class="fas fa-plus"> Добавить запись </i></a>
</p>
<p class="d-flex justify-content-end">

```

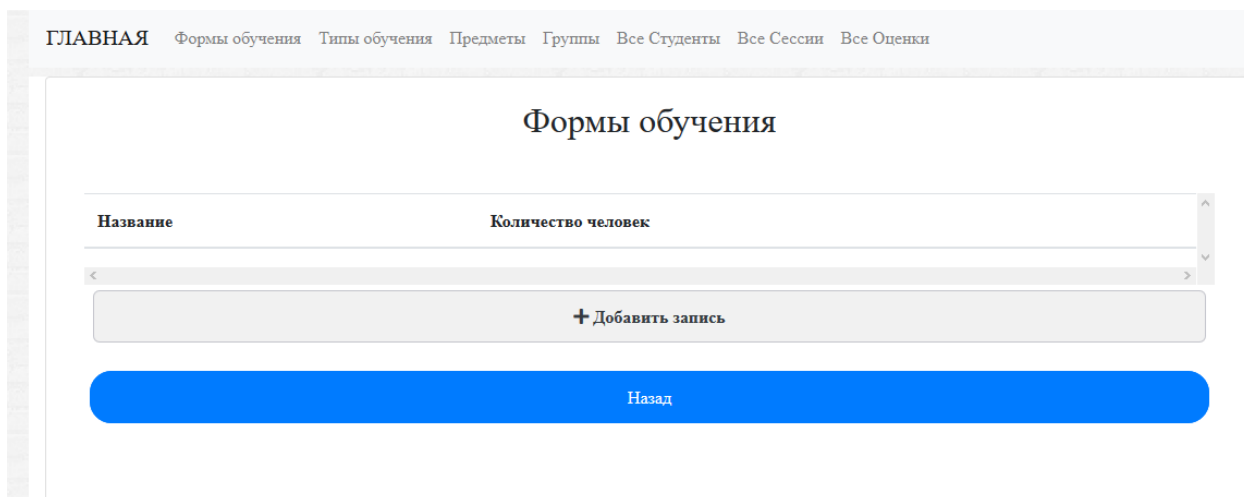



Рисунок 2.3 – Вид страницы формы обучения

На каждой из такой страницы будут отображены все атрибуты данных и будет возможность перейти на страницу добавления/изменения/удаления данных текущей таблицы. На текущей странице также будет отображено количество студентов на этой форме обучения.

При переходе на любой странице по пункту добавления записи, будет открыта страница добавления/изменения, на которой будут все атрибуты записи. Так, для форм обучения страница добавления показана на рисунке 2.4.

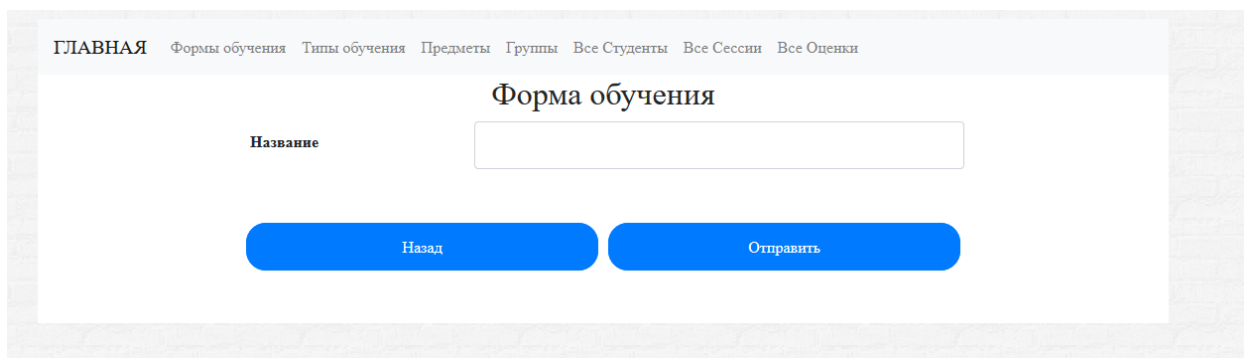


Рисунок 2.4 – Вид страница добавления формы обучения
Вид формы групп показан ниже.

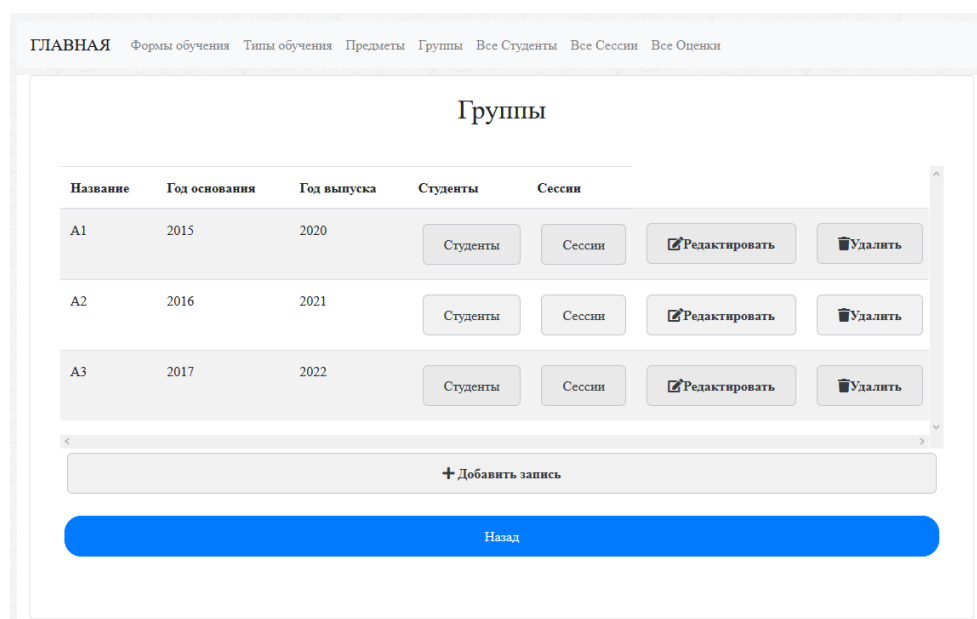


Рисунок 2.5 – Вид формы групп

Что примечательно, с данной формы можно перейти на форму студентов и сессий, чтобы показать данные, касающиеся конкретно выбранной группы.

Вид страницы студентов показан ниже.

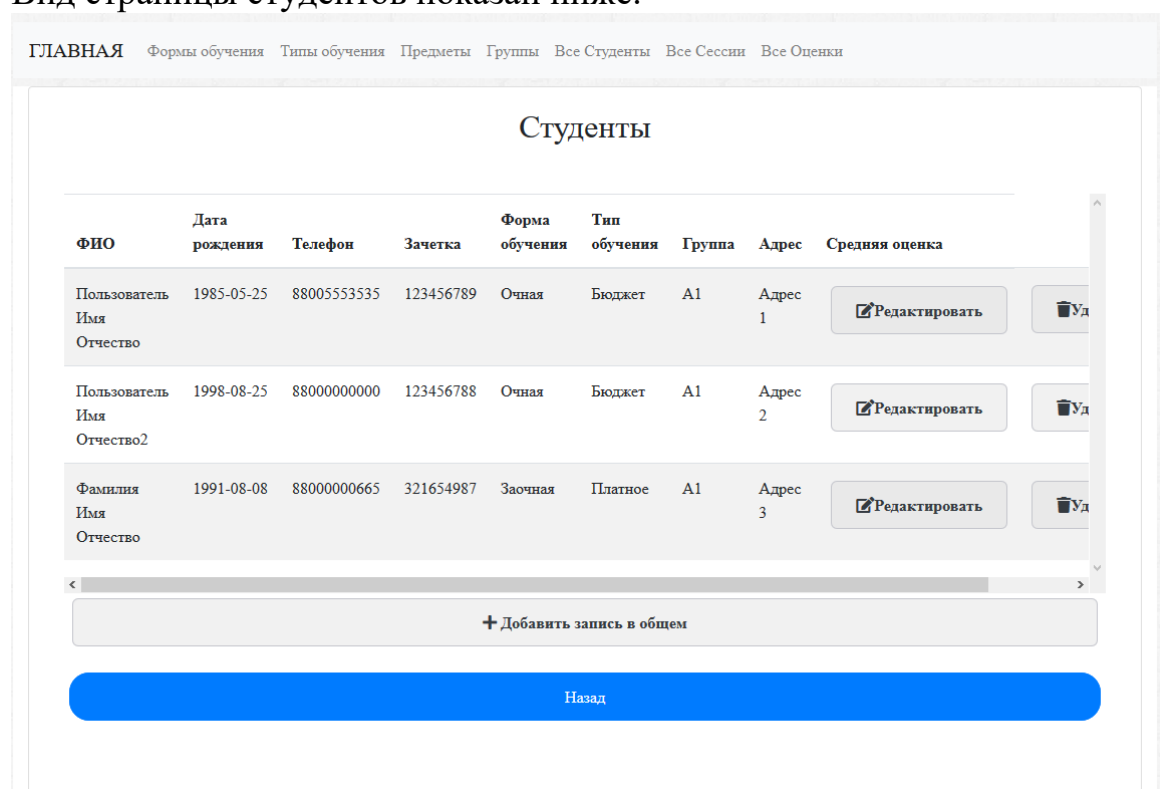


Рисунок 2.6 – Вид страницы студентов

Вид страниц сессий и оценок, а также страниц добавления их отображен на рисунках ниже.

ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Студенты

ФИО студента	<input type="text"/>
Дата рождения	<input type="text" value="дд . мм . гггг"/>
Телефон	<input type="text"/>
Адрес	<input type="text"/>
Зачетка	<input type="text" value="0"/>
Форма обучения	<input type="text" value="Очная"/>
Тип обучения	<input type="text" value="Бюджет"/>
Группа	<input type="text" value="A1 (2015 - 2020)"/>

Назад Отправить

Рисунок 2.7 – Вид страницы добавления студента

ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Сессии

Название	Семестр	Группа	Предметы	Оценки		
Зимняя 2020	1	A1 (2015 - 2020)	<input type="text" value="Предметы"/>	<input type="text" value="Оценки"/>	<input type="button" value="Редактировать"/>	<input type="button" value="Удалить"/>
Весенняя 2020	2	A1 (2015 - 2020)	<input type="text" value="Предметы"/>	<input type="text" value="Оценки"/>	<input type="button" value="Редактировать"/>	<input type="button" value="Удалить"/>
Зимняя 2021	3	A1 (2015 - 2020)	<input type="text" value="Предметы"/>	<input type="text" value="Оценки"/>	<input type="button" value="Редактировать"/>	<input type="button" value="Удалить"/>
Зимняя 2021	1	A2 (2016 - 2021)	<input type="text" value="Предметы"/>	<input type="text" value="Оценки"/>	<input type="button" value="Редактировать"/>	<input type="button" value="Удалить"/>

Назад

Рисунок 2.8 – Вид страницы сессий

ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Сессия

Название	<input type="text"/>
Семестр	<input type="text"/>
Группа	<input type="text" value="A1 (2015 - 2020)"/>

Назад Отправить

Рисунок 2.9 – Вид страницы добавления сессии

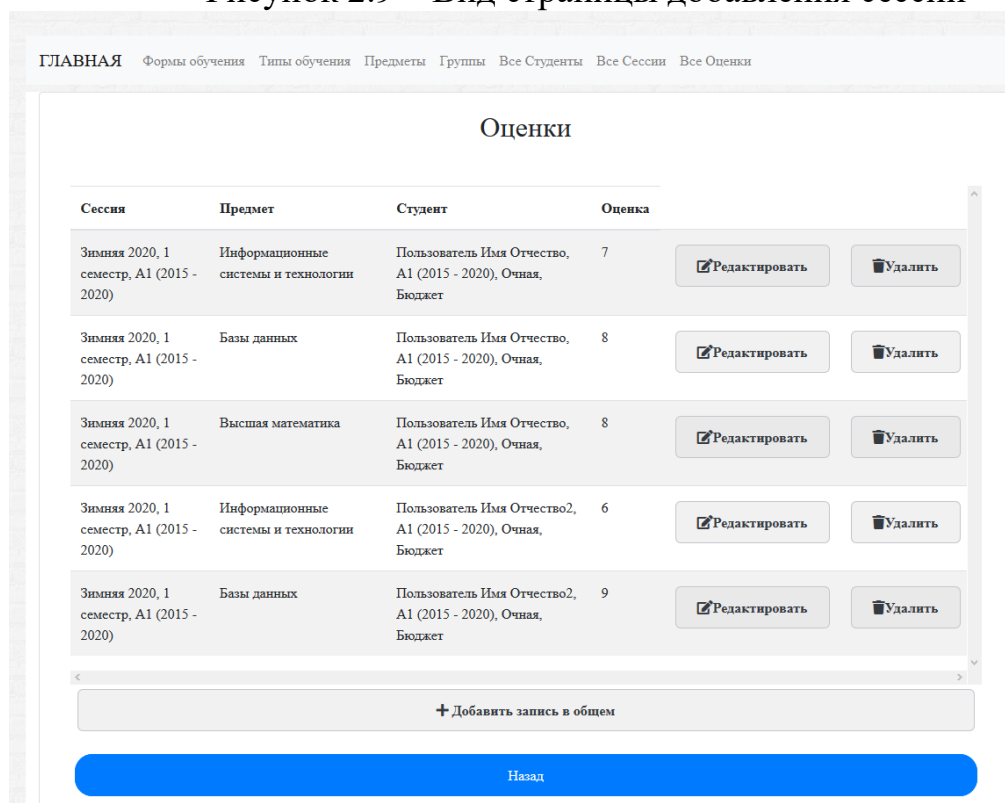


Рисунок 2.10 – Вид страницы оценок

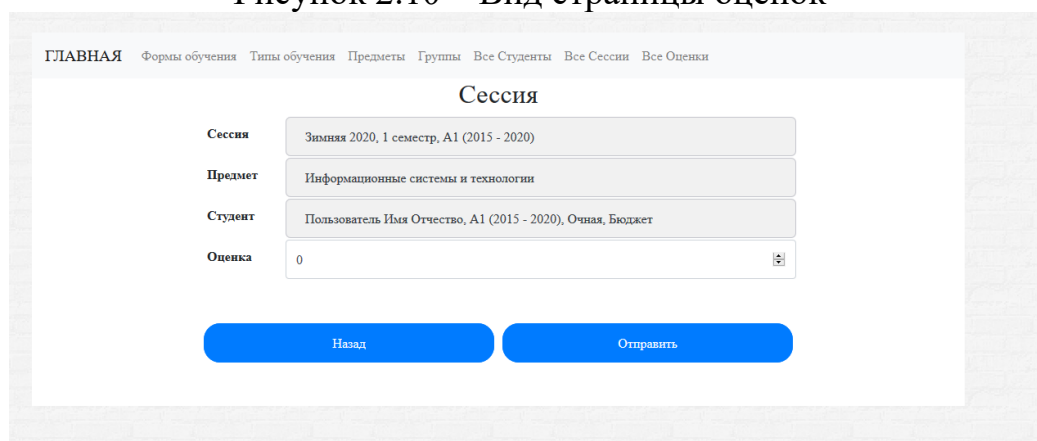


Рисунок 2.1 – Вид страницы добавления оценки

2.5. Обоснование выбора компонентов и технологий для реализации программного средства.

Архитектура данного проекта – это веб-приложение. Архитектура приложения во многом предопределило и технологии, используемые для построения проекта.

Основные технологии, применяемые при создании приложения:

- серверный язык *JAVA*;
- язык запросов *SQL (MySQL)*;
- фреймворк *Spring*;
- язык разметки *HTML*;
- таблица стилей *CSS*;

– клиентский язык *JavaScript*.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Язык Java появился в 1995 году – 90-е годы были вообще урожайными на новые языки и концепции программирования. В таком Эдеме языков важно было не заблудиться, по ошибке приняв за Священный Грааль технологию, которая не пройдет испытания временем. Java прошел испытания, хотя и очень долгие. Очень не рекомендуется путать этот язык с JavaScript – они по виду похожи, но это совсем разные языки [7].

Вероятно, в Java впервые реализовали концепцию того, что язык должен быть максимально изолирован от платформы разработки, чтобы применять его без изменений везде: в компьютерах, часах, сотовых телефонах, бытовой технике. С «железной частью» должна была справляться виртуальная машина (JVM), которая, собственно, и создавалась индивидуально под каждое устройство. Сам же язык был неизменен и в качестве результата выдавал байт-код. С самого начала было известно, что код не может исполняться очень быстро, но многие устройства не требовали высокой скорости исполнения. Кроме того, со временем появились оптимизирующие компиляторы, так что, в среднем, программа на Java работает раза в 2-3 медленнее, чем на C++. Постоянное сравнение с C/C++ здесь не случайно: многие современные языки взяли за основу его конструкции и синтаксис, так что, бывает, узнать сходу язык очень трудно. Вместе с тем, Java с тех пор сильно «размножилась», и даже J#, J и прочие аналоги являются не родными братьями, а лишь подобием.

Сама идея языка, вполне, кстати, достаточного для создания софта любой сложности, была сначала не понята: был ли, мол, смысл создавать между аппаратурой и кодом промежуточные слои исполняющих машин? Со временем сомнения рассеялись: появилась мультязычная платформа.

NET, и даже в Windows появились слои – аппаратно-зависимые, платформо-независимые. Самое же простое объяснение – софт стал очень сложным, а программисты очень ленивыми, чтобы переписывать программы под каждый отдельный аппарат.

Но вернемся к языку. Как уже говорилось, чем-то он похож на C++, чем-то на старый добрый Бейсик. Нет сейчас ни одного языка, который бы не хвалился своими возможностями ООП, и Java здесь не отличается от канонов: классы и объекты здесь используются везде, даже в самых примитивных задачах вроде вывода строки на экран. Из особенностей можно отметить, что все объекты в языке создаются только динамически, а все функции являются методами классов. Множественное наследование не поддерживается, как в C++, как и «опасные» указатели. ООП дает много преимуществ, но и требует слишком многого – в случае Java памяти устройства никогда не будет слишком много. В остальном же, имеются библиотеки классов для практически всех задач; преимущественно – под написание клиентских и серверных приложений. Хозяин Java – Oracle –

успешно использует язык для использования в разработках своей одноименной СУБД. На сегодняшний день язык считается наиболее востребованным на рынке.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET [8].

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.2.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бóльшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Следующие две технологии — это *HTML* и *CSS*, которые предопределены архитектурой проекта. Имеется единственное обоснование их выбора — это веб-архитектура разрабатываемого программного средства.

HTML — язык гипертекстовой разметки, который используется для структурирования и отображения веб-страницы и ее контента. Для разработки данного программного средства будет применяться версия этого языка *HTML5*.

CSS — это формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки *HTML*. Применяемая версия данной технологии — *CSS3*.

3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Для проверки работы спроектированной автоматизированной системы, разработаны тесты, с помощью которых можно оценить корректную работу веб-сервиса. В таблицах 3.1-3.2 приведены результаты тестирования.

Таблица 1.1 – Набор тестов для автоматизированной системы

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Главная страница	Загрузка главной страницы	Вывод количества групп и студентов	Да
Страница добавления формы обучения	Перейти на страницу. Ввести новую форму. Нажать кнопку отправить	Переход на страницу форм обучения с выводом только что созданного значения	Да
Страница Сессий. Страница групп.	Перейти на страницу. Добавить две сессии для разных групп. Перейти на страницу групп. В строке группы выбрать пункт сессии	Переход на страницу сессий, на которой отображены только те сессии, в которых участвует выбранная ранее группа	Да
Страница оценок. Страница сессий.	Перейти на страницу сессий. В первой сессии выбрать отображения оценок. Выбрать пункт добавления оценки	На странице добавления оценки в выпадающем меню предметов будут отображены только предметы текущей сессии, а в выпадающем меню студентов только студенты группы которой участвует в сессии	Да

Таблица 4.2 – Набор тестов отображения информации

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Все страницы	Перейти на любую из	Корректное	Да

	вкладок произвольном порядке.	в отображение информации, отображение того, чего ожидает пользователь.	
Все вкладки	Увеличение масштаба отображения средствами браузера.	Появление скроллов справа и снизу, корректное отображение информации.	Да
Все вкладки	Обновление страницы используя средства браузера.	Состояние веб- сервиса до обновления и после не изменилось.	Да
Меню браузера	Нажатие функциональных кнопок браузера «Назад» и «Вперед» в произвольный момент работы веб- приложения.	Корректное поведение веб-сервиса. Отображение информации.	Да

4. РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для установки веб-сервиса необходимым требованием является наличие Java Virtual Machine, Apache Tomcat 8, а также MySQL Server 8.0.1 в качестве целевой СУБД, maven 3.6.

База данных для веб-сервиса генерируется сама после первого запуска приложения.

Для запуска приложения на клиентском компьютере, достаточно в консоли перейти в папку проекта и написать код

```
mvn clean package spring-boot:run
```

Если запуск решено производить посредством war-архива, то его нужно переместить или скопировать в папку, где установлен Apache Tomcat 8. Далее необходимо запустить Apache Tomcat 8.

Для запуска веб-сервиса необходимо открыть браузер на вашем компьютере и написать в адресную строку следующий URL-адрес: <http://localhost:8080/>.

Далее опишем руководство по использованию программного средства.

При заходе на главную страницу, пользователь будет видеть общее количество групп и студентов в системе и пункт меню для перехода на другие страницы.

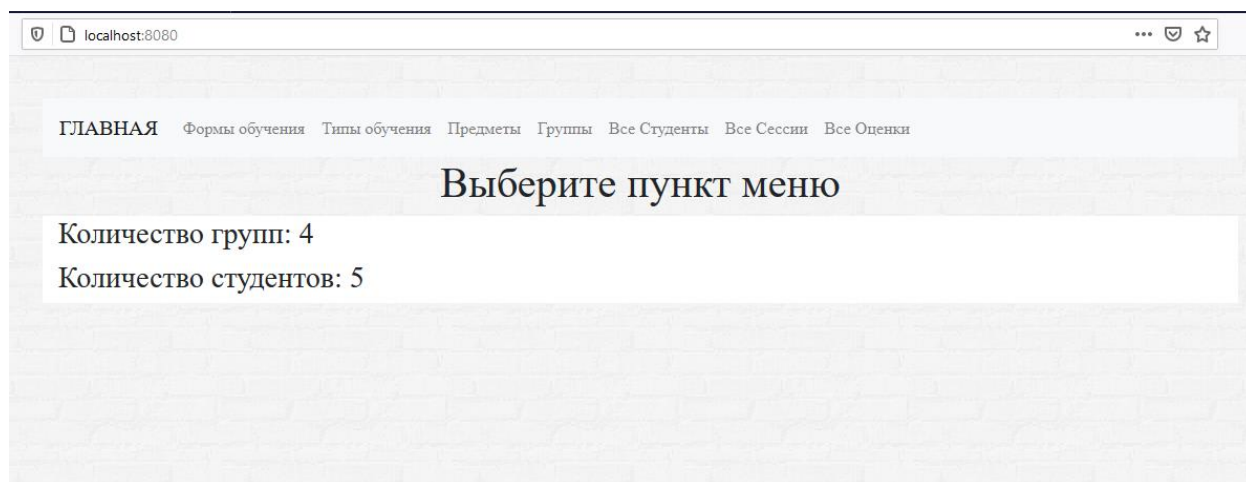


Рисунок 4.1 – Главная страница

При первом запуске пользователю нужно будет заполнить страницы форм обучения и типов обучения. Напротив названия формы и типа отображается количество человек, которые учатся по данным атрибутам.

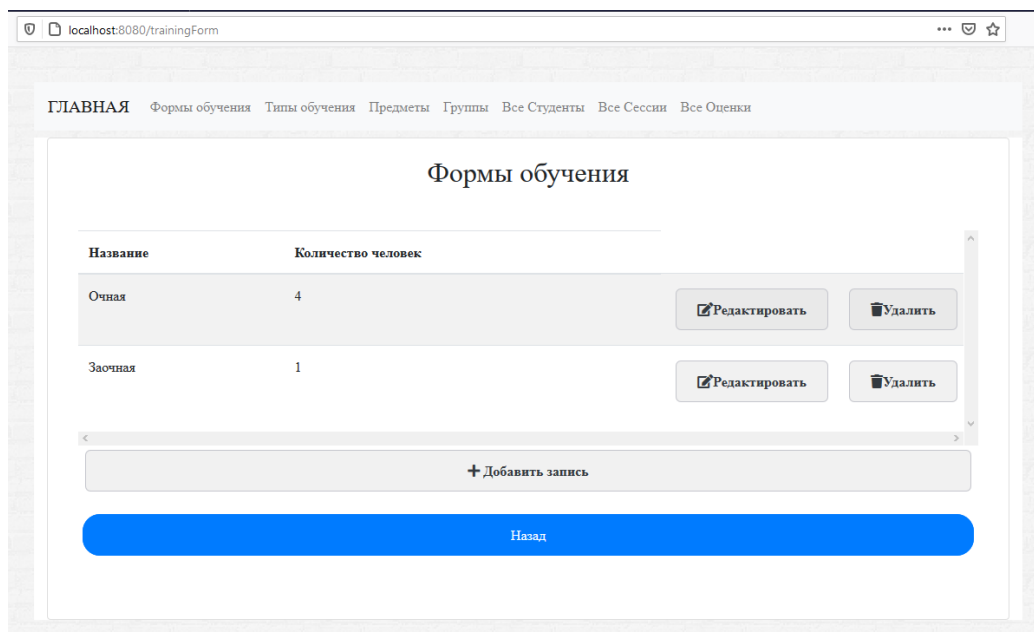


Рисунок 4.2 –Страница форм обучения

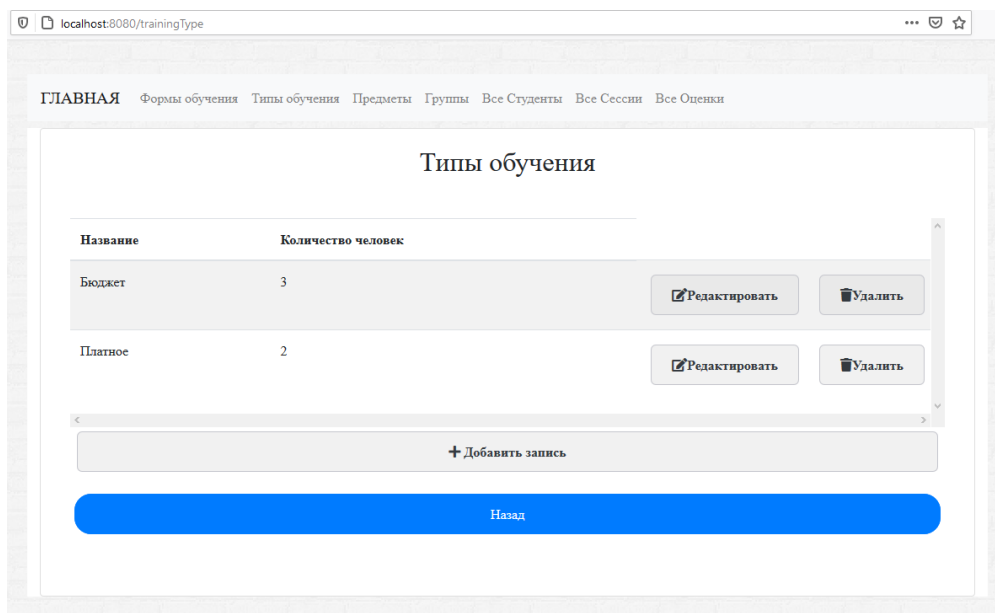


Рисунок 4.3 –Страница типов обучения

При переходе на страницу предметов, будет отображен список всех предметов системы. Далее эти предметы будут использоваться в сессиях и оценках.

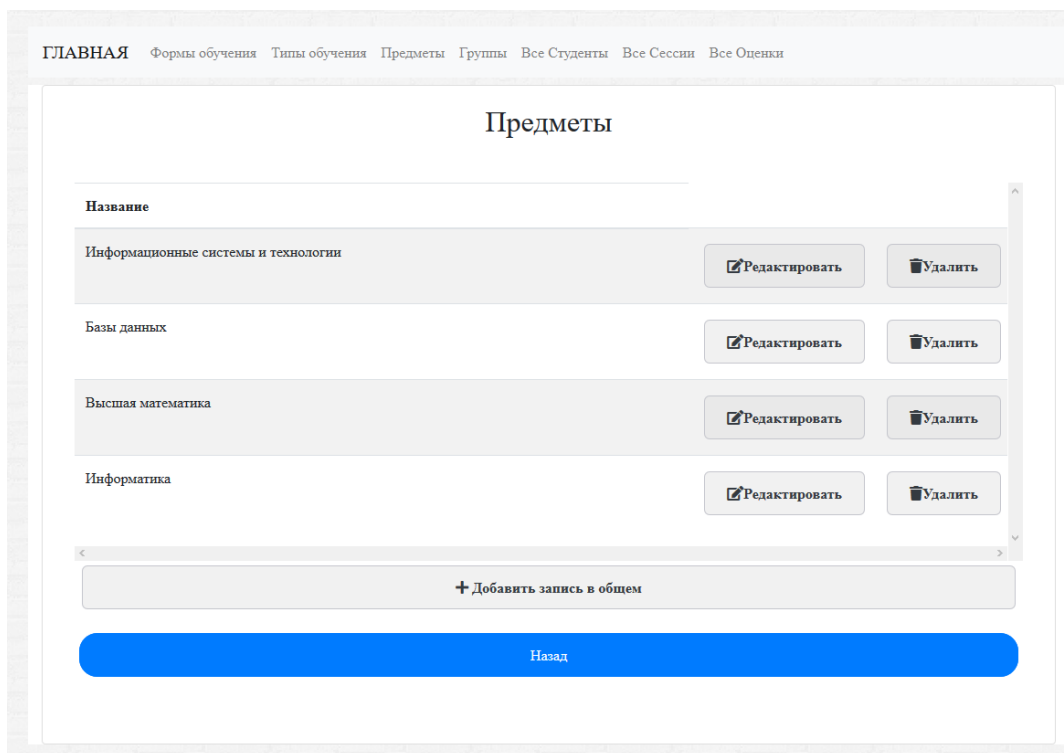


Рисунок 4.4 – Страница предметов

Для добавления нового предмета стоит нажать соответствующую кнопку.

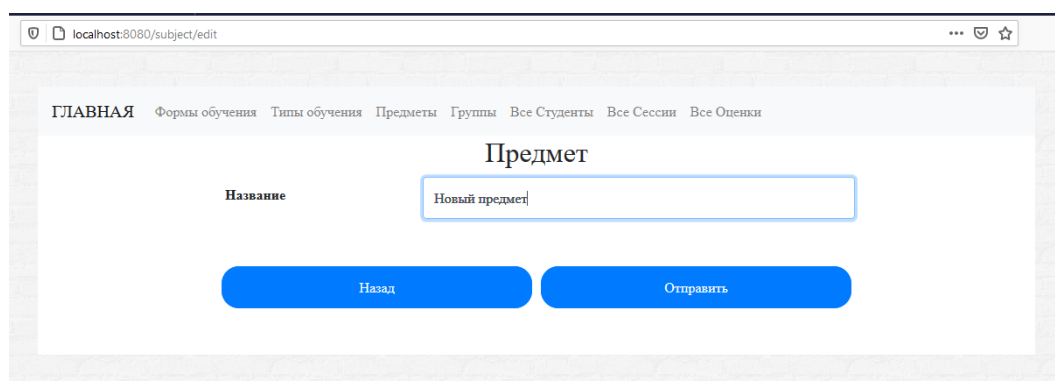


Рисунок 4.5 – Страница добавления предметов

После добавления нового предмета, он отображается в конце списка.

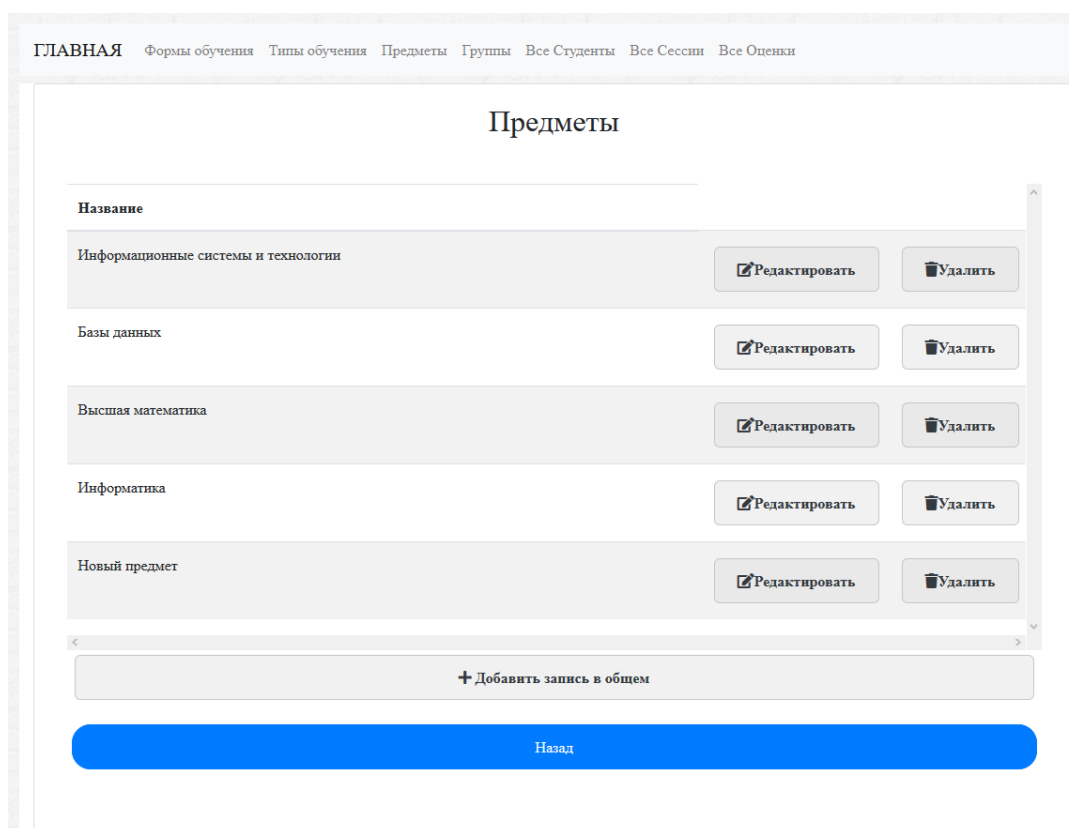


Рисунок 4.6 – Страница предметов после добавления нового

При переходе на страницу групп, будет доступен список их всех.

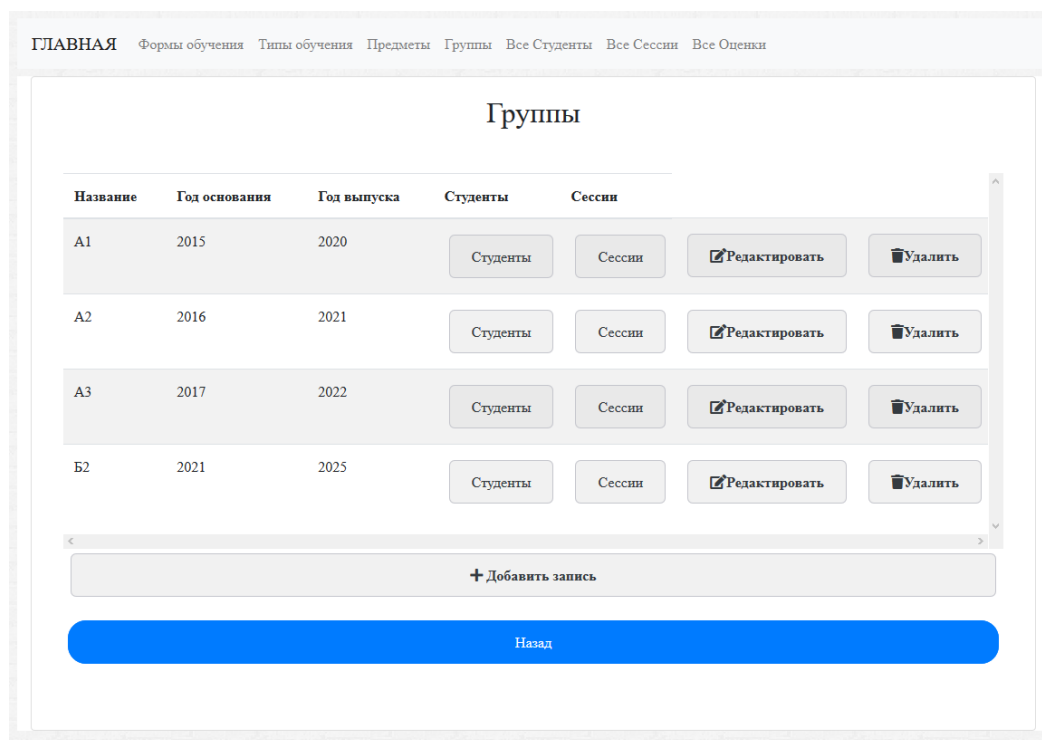


Рисунок 4.7 – Страница групп

Для добавления новой группы, нужно нажать соответствующую кнопку и заполнить все поля.

localhost:8080/group/edit

ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Группа

Название Пожалуйста, заполните это поле.

Год основания

Год выпуска

[Назад](#) [Отправить](#)

Рисунок 4.8 – Страница добавления новой группы

После добавления группы, новая будет отображена внизу.

ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Группы

Название	Год основания	Год выпуска	Студенты	Сессии		
A1	2015	2020	Студенты	Сессии	Редактировать	Удалить
A2	2016	2021	Студенты	Сессии	Редактировать	Удалить
A3	2017	2022	Студенты	Сессии	Редактировать	Удалить
B2	2021	2025	Студенты	Сессии	Редактировать	Удалить
Новая группа	2021	2022	Студенты	Сессии	Редактировать	Удалить

[+ Добавить запись](#)

[Назад](#)

Рисунок 4.9 – Страница групп после добавления новой группы

Что интересно, с текущей страницы можно перейти на страницу студентов через верхнее меню, чтобы видеть список всех студентов, или же сразу перейти к студентам группы. Перейдем вторым вариантом.

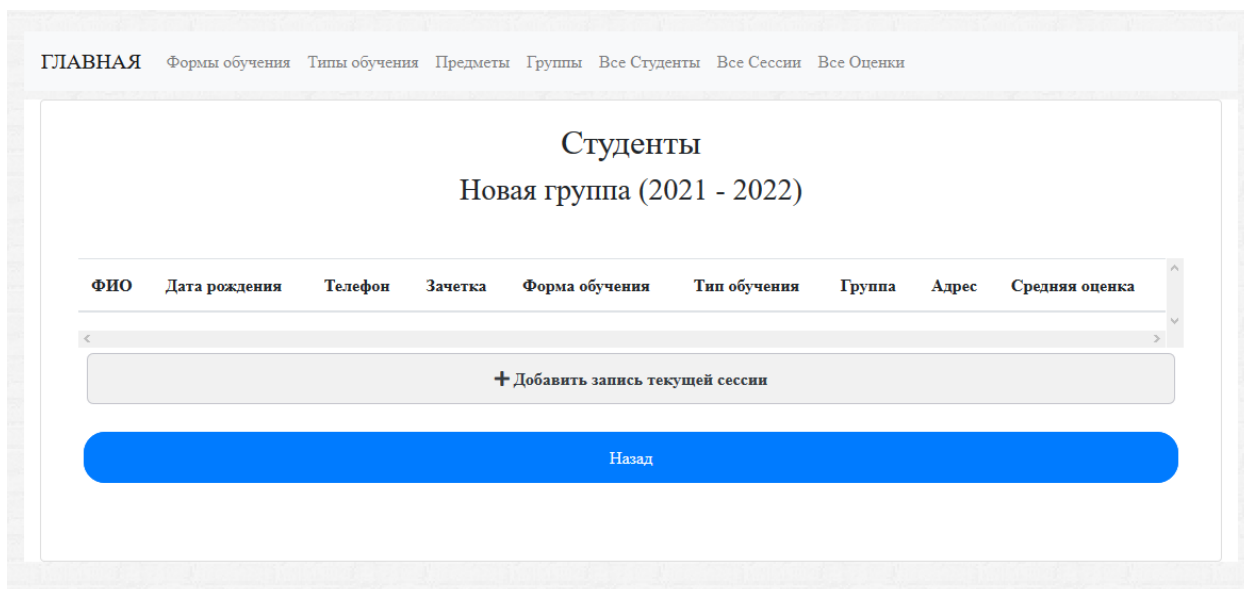


Рисунок 4.10 – Страница студентов новой группы

Как видно из рисунка, у нас не отображаются студенты группы, поскольку эту группу мы создали только что а студентов в нее еще не добавили. Добавим студента к группе.

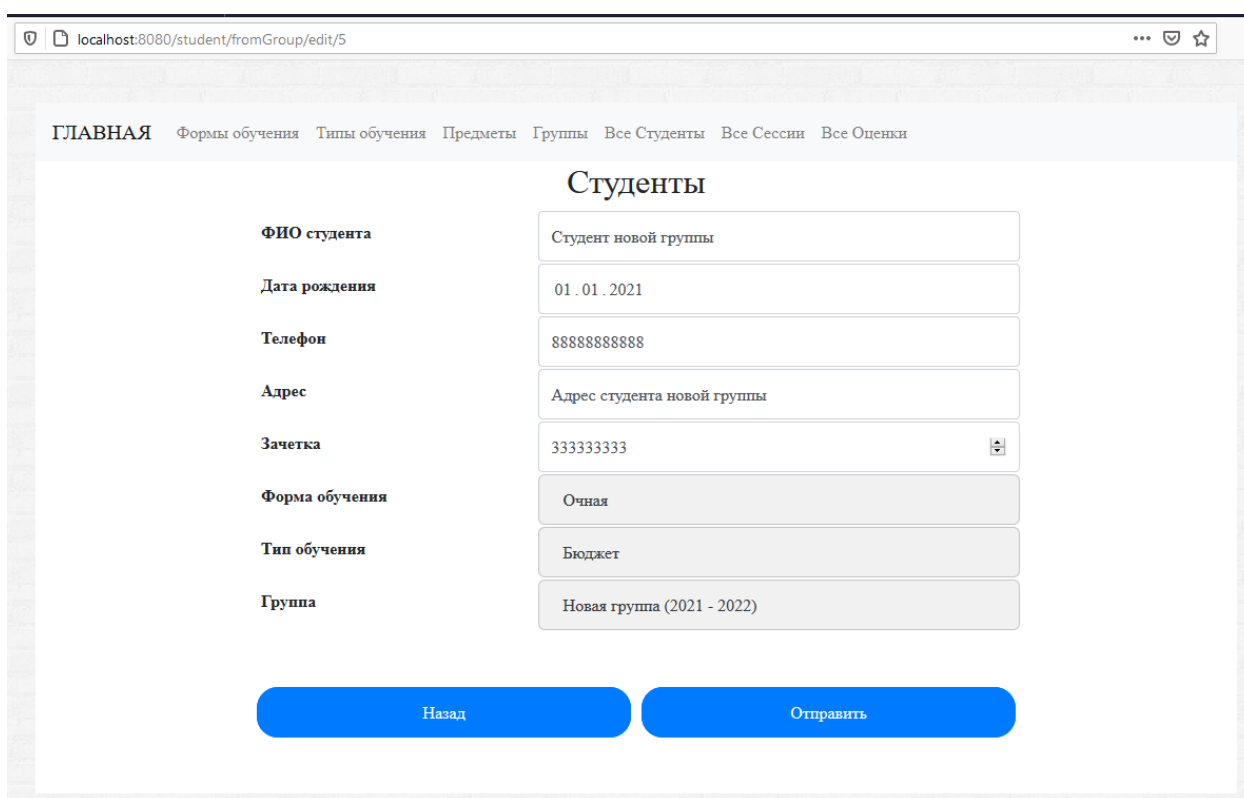


Рисунок 4.11 – Страница добавления студентов к текущей группе

После добавления студента происходит переадресация на страницу всех студентов, новый студент показан внизу страницы.

Можно заметить, что у студентов есть средняя оценка, но не у всех. Она отображается только тем, кто числится в сдававших сессию.

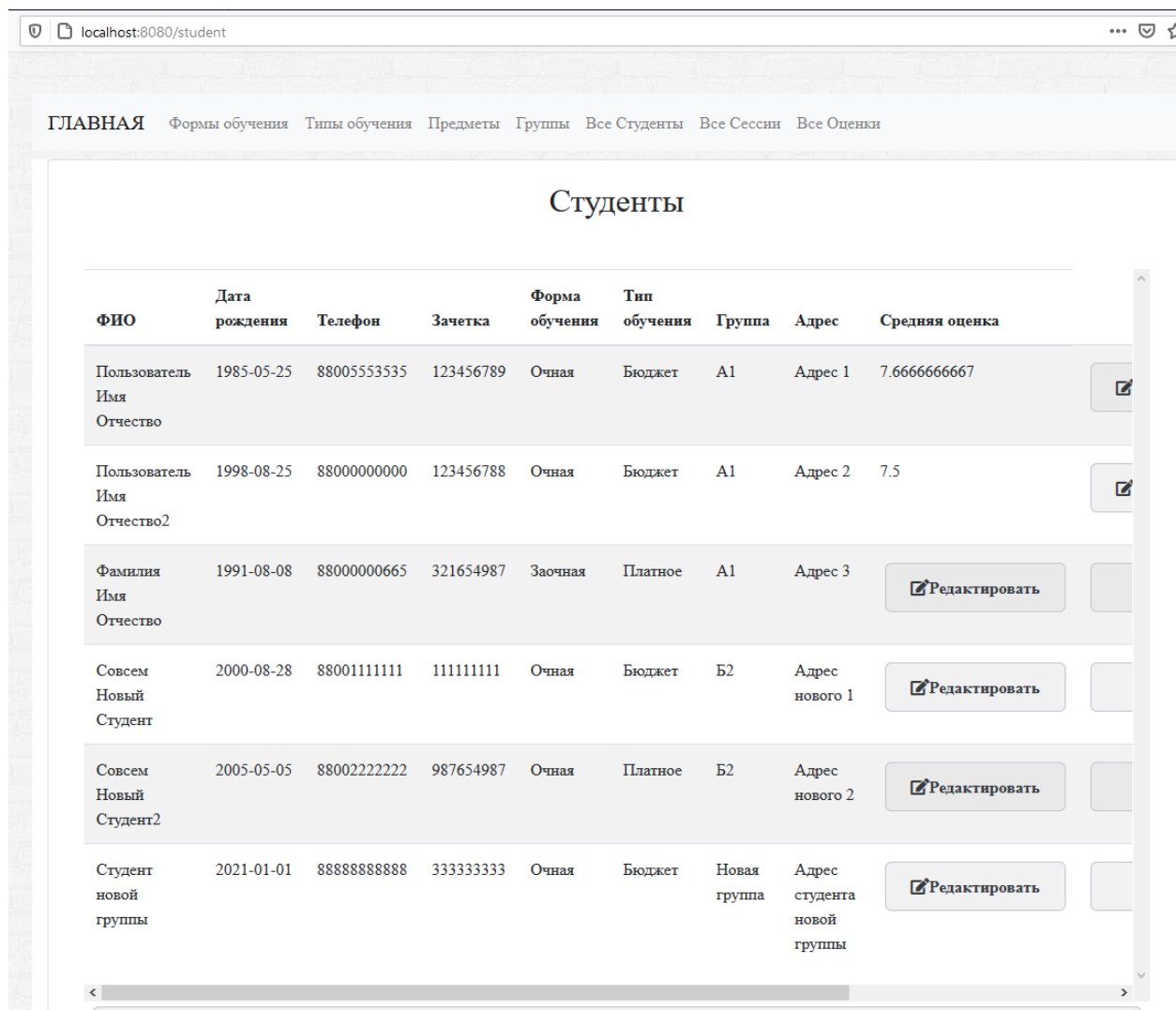


Рисунок 4.12 – Страница студентов

А теперь, если вернуться на страницу групп и перейти к студентам новой группы, будет отображен только наш добавленный студент этой группы.

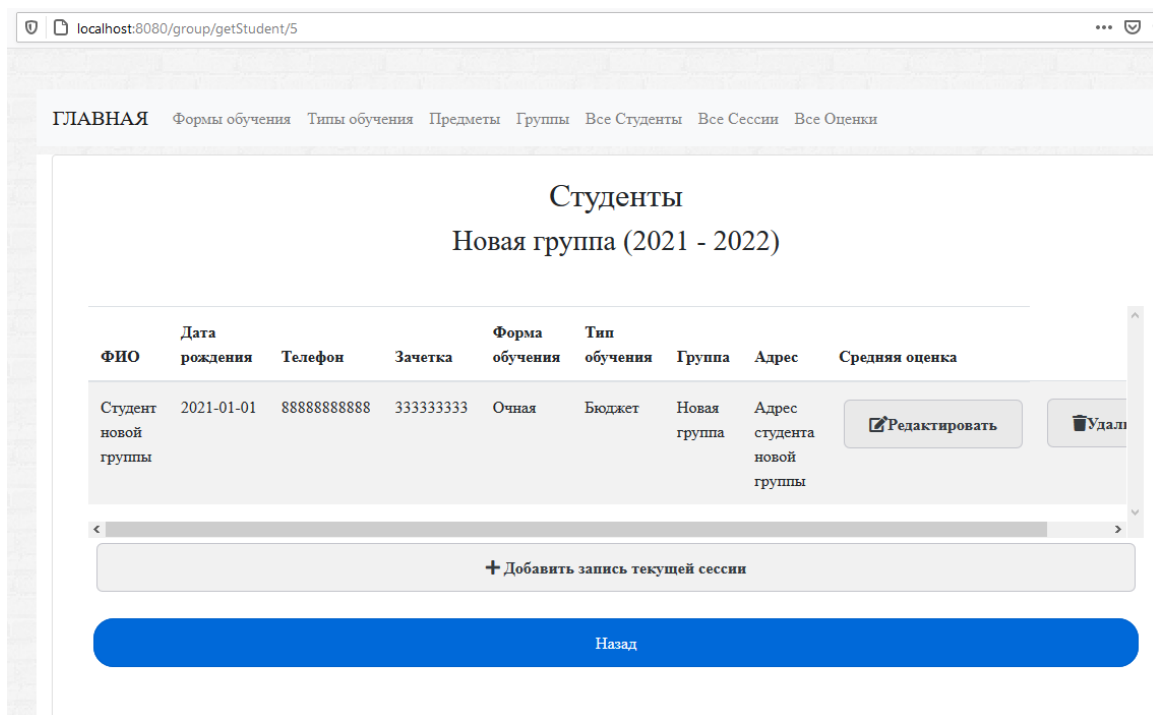


Рисунок 4.13 – Страница студентов выбранной группы

С сессиями точно так же работает, как и с группами. Можно перейти на страницу предметов сессии или оценок.

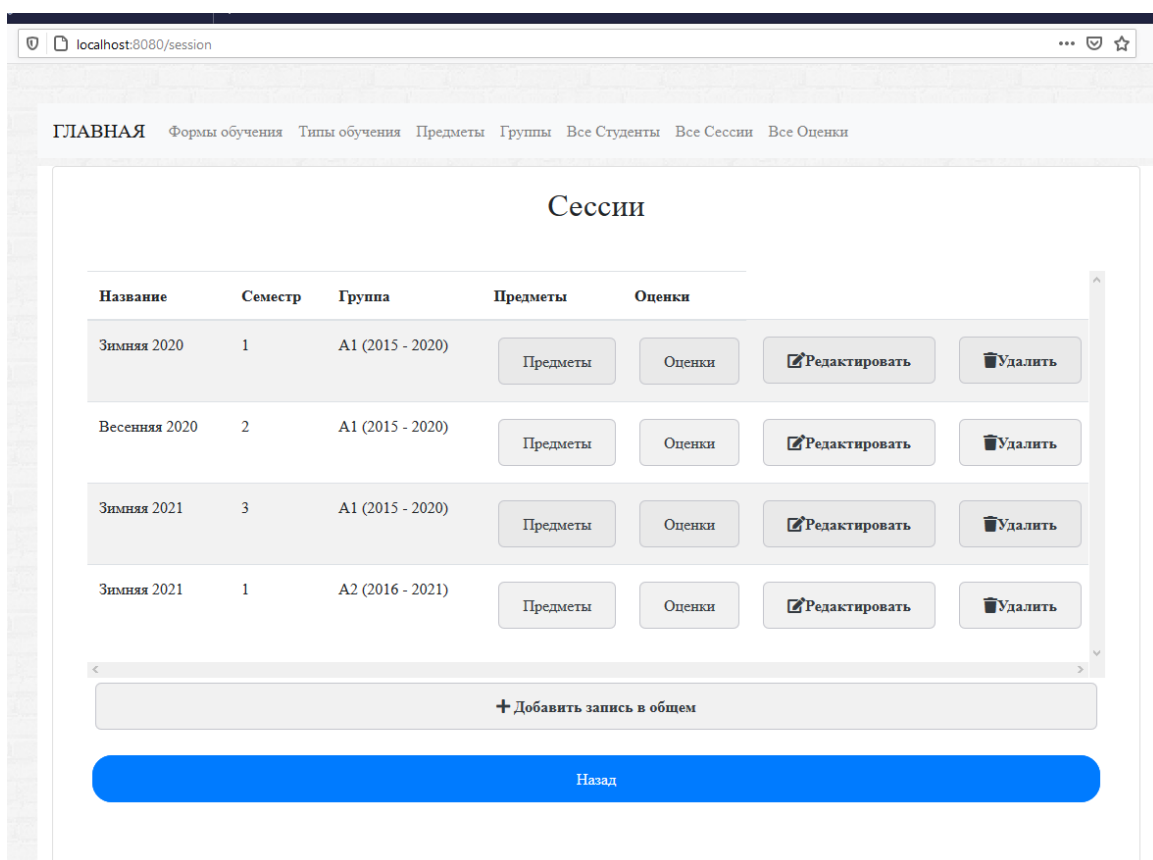


Рисунок 4.14 – Страница сессий

Стоит сказать, что напротив каждого предмета в любой сессии сразу же стоит средняя оценка по всем сдавшим этот предмет студентам.

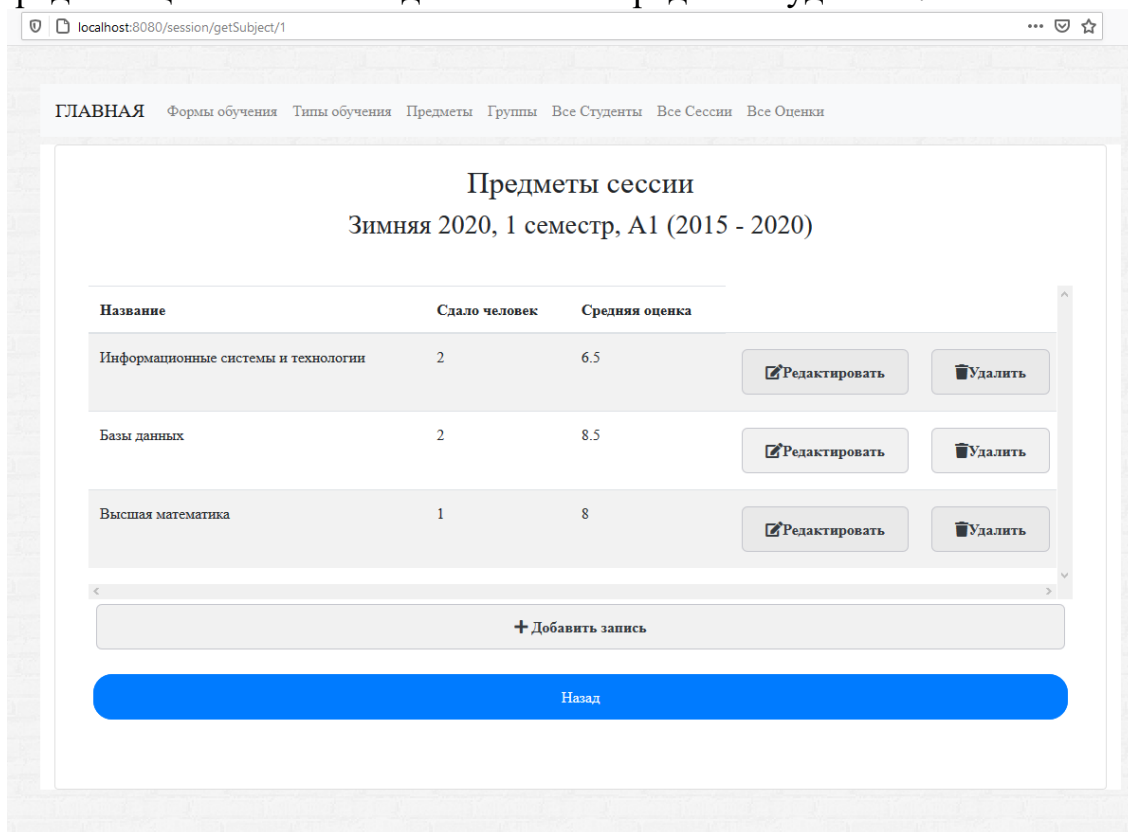


Рисунок 4.15 – Страница предметов сессий

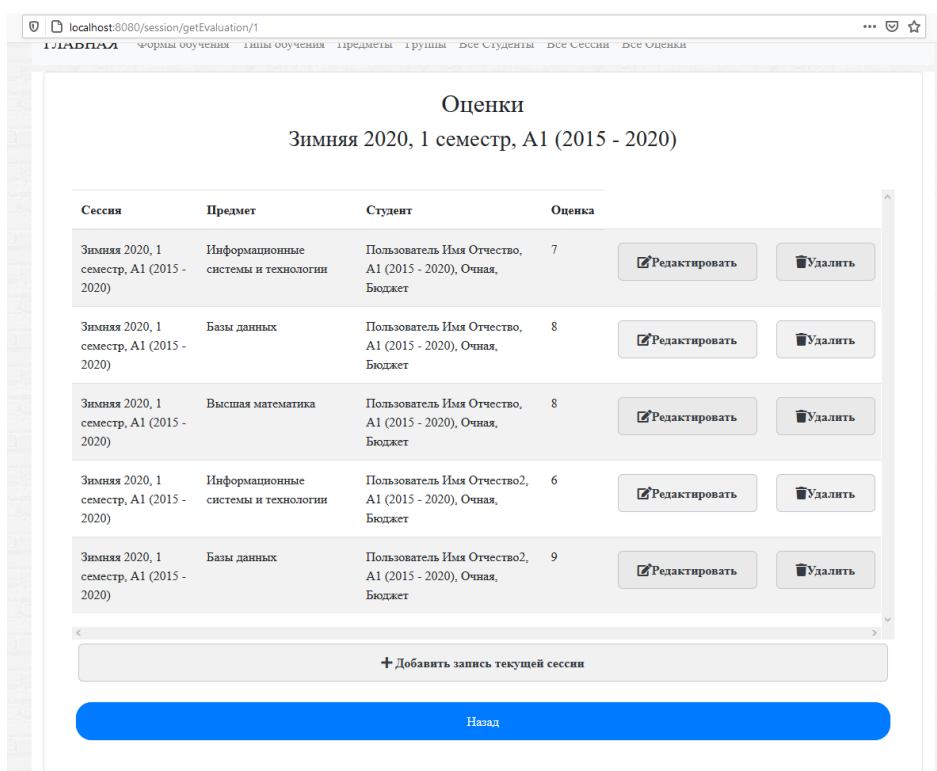
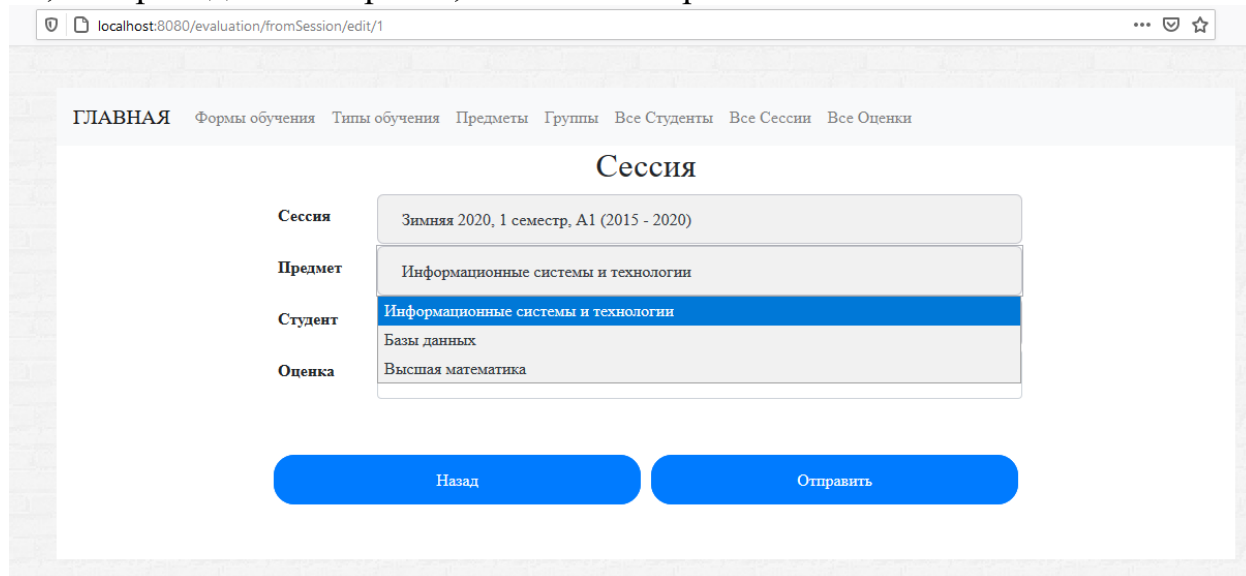


Рисунок 4.16 – Страница оценок

Стоит заметить, что при добавлении оценки сессии, в списках предметов доступны только те, что указаны в самой сессии. Так, наш предмет, который добавлен ранее, в ней не отображается.



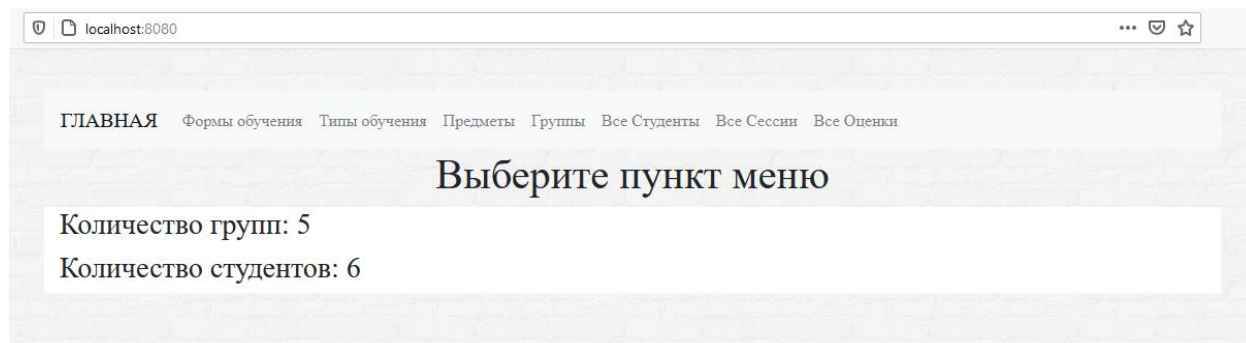
Сессия

Сессия	Зимняя 2020, 1 семестр, А1 (2015 - 2020)
Предмет	Информационные системы и технологии
Студент	Информационные системы и технологии
Оценка	Высшая математика

Назад Отправить

Рисунок 4.17 – Страница добавления новой оценки сессии

После добавления нового студента и группы, при переходе на главную видим обновленную информацию.



ГЛАВНАЯ Формы обучения Типы обучения Предметы Группы Все Студенты Все Сессии Все Оценки

Выберите пункт меню

Количество групп: 5

Количество студентов: 6

Рисунок 4.18 – Главная страница после всех манипуляций

ЗАКЛЮЧЕНИЕ

Итогом написанной курсовой работы является программное приложение, которое предоставляет возможность учета и регистрации успеваемости студента. Данное приложение является доступным и понятным любому пользователю, располагает удобным и минималистичным интерфейсом, а также соответствует всем требованиям, предъявленным к курсовому проекту.

Данная программа является web- приложением, что говорит о том, что множество клиентов, расположенных на определенном расстоянии друг от друга могут одновременно общаться с сервером.

Вся используемая информация хранится в базе данных, разработанной для данного проекта. Доступ к базе данных был реализован только со стороны веб-сервиса.

Проанализировав все моменты можно сказать, что данная программа соответствует всем требованиям курсового проекта и все задачи, поставленные перед разработкой программы, соблюдены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Атрибутивная информация [Электронный ресурс]. – Режим доступа: <http://geum.ru/next/art-199278.php>. – Дата доступа: 13.04.2021.
2. Диаграмма структуры программного приложения (SSD) [Электронный ресурс]. – Режим доступа: https://studopedia.su/3_24465_SADT-diagrammi.html. – Дата доступа: 13.04.2021.
3. Цель анализа требований в проектах [Электронный ресурс]. – Режим доступа: <https://analytics.infozone.pro/requirements-analysis/analysis-of-requirements-wiegers-2004/>. – Дата доступа: 13.04.2021.
4. Разработка информационной модели данных [Электронный ресурс]. – Режим доступа: https://studbooks.net/2239768/informatika/razrabotka_informatsionnoy_modeli_dannyh. – Дата доступа: 13.04.2021.
5. Фреймворк Spring MVC [Электронный ресурс]. – Режим доступа: <https://itnan.ru/post.php?c=1&p=336816>. – Дата доступа: 13.04.2021.
6. Проектирование пользовательского интерфейса [Электронный ресурс]. – Режим доступа: <https://www.visualpharm.ru/prototyping.html>. – Дата доступа: 13.04.2021.
7. Серверные языки: Java (обзор) [Электронный ресурс]. – Режим доступа: <https://codomaza.com/article/servernye-jazyki-java-obzor>. – Дата доступа: 13.04.2021.
8. Spring Framework [Электронный ресурс]. – Режим доступа: <https://google-info.org/325966/1/spring-framework.html>. – Дата доступа: 13.04.2021.