

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г.
Шухова»
(БГТУ им. В. Г. Шухова)

КУРСОВАЯ РАБОТА

по дисциплине «Интерфейсы ВС»

Тема: “Создание клиент-серверного RESTful приложения, с использованием
Spring Framework”

Автор работы _____ Бобылев М.С.
(подпись) ВТ-42

Руководитель проекта _____ ст. пр.
(подпись) Торопчин Д.А.

Оценка _____

Белгород 2021 г.

Содержание

Содержание	2
Введение	3
Основная часть	4
2.1 Теоретические сведения	4
2.2 Проектирование базы данных	6
2.3 Разработка серверной части приложения	7
2.4 Разработка клиентской части приложения	8
2.5 Проверка работоспособности	8
Заключение	9
Список литературы	9
Приложение	10

Введение

В данной работе будет описан процесс создания RESTful приложения, серверная часть приложения будет написана на языке Java с использованием Spring Framework, клиентская часть будет представлять собой SPA написанная на языке JavaScript.

Предметная область проекта - сервис для учета и упорядочивания инвентаря. Сервис должен предоставлять возможность управлять хранилищами и их структурой - настраивать секции хранилища и их метки, а также иметь функционал по организации и упорядочиванию хранимых в хранилищах вещей.

Основная часть

2.1 Теоретические сведения

Для разработки серверной части приложения был использован язык Java и фреймворк Spring Framework.

Java - язык программирования со строгой типизацией и объектно-ориентированной парадигмой, исходный код Java программ компилируется не сразу в машинный код как в большинстве компилируемых языков, а в промежуточный байт-код, который может запускаться на специальной виртуальной машине JVM. Благодаря этому программа написанная на языке Java может запускаться на любой платформе для которой реализована виртуальная машина.

Spring Framework - фреймворк для внедрения зависимостей, позволяющий избежать большой связности между сущностями. Проект Spring включает в себя множество различных прикладных библиотек, и для упрощения первоначальной настройки использован Spring Boot. Эта библиотека позволяет сгенерировать начальный шаблон для проект с правильно настроенными конфигурациями и зависимости, а так же проект использующий Spring Boot по умолчанию включает в себя встроенный web-сервер, который можно использовать не только для разработки но и для развертывания готового приложения.

Веб приложения будет спроектировано по паттерну MVC (Model View Controller). MVC - это схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер - таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
- Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

В качестве для хранения данных была выбрана Java реализация SQL базы данных H2, позволяющая разворачивать базу данных в оперативной памяти (in memory database) для удобства в разработке и тестировании проекта. Так как проект разрабатывался в учебных целях было решено не использовать более тяжеловесные реализации SQL баз данных (PostgreSQL, MySQL) в виду ограниченных времени и ресурсов.

При использовании Spring Framework, не обязательно описывать таблицы для сущностей проекта напрямую на языке запросов БД. Существует дополнительная абстракция над базой данных - ORM, предоставляющая

удобный интерфейс к содержимому таблиц и автоматическое создание таблиц в БД при инициализации приложения.

Для описания сущностей проекта необходимо описать модели в виде Java классов, с примененной аннотацией `@Entity`. Поля класса при этом будут выступать столбцами таблиц, для установки ограничений столбцов необходимо использовать дополнительные аннотации.

Далее реализуются следующие сущности: репозитории - объекты предоставляющие доступ к моделям, сервисы - объекты хранящие в себе “бизнес логику” и контроллеры - объекты которые “прикрепляют” действие к запросам на указанные URL.

Реализовав все эти компоненты получится API, позволяющее получать доступ к информации и редактировать ее используя HTTP запросы.

В качестве системы сборки будет использоваться Maven - фреймворк для автоматизации сборки проектов. Он позволяет декларативно описать спецификацию проекта и его зависимости, которые автоматически загрузятся при первой сборке.

Так же для упрощения написания шаблонных конструкций языка Java была использована библиотека Lombok, предоставляющая аннотации, которые автоматически генерируют код методов `get` и `set` у всех полей класса, или же создают конструкторы различных типов.

Клиентская часть приложения представляет собой SPA (Single page application) реализованная на языке JavaScript с использованием библиотеки React. Данная библиотека используется для создания компонентов пользовательского интерфейса, и реализует только отображение данных и компонентов в DOM дерево и диалект для описания компонентов JSX . Для реализации роутинга использовалась библиотека React Router. Сборка клиентской части совершалась интерпретатором Node.js.

Для упрощения процесса разработки была использована система управления версий git.

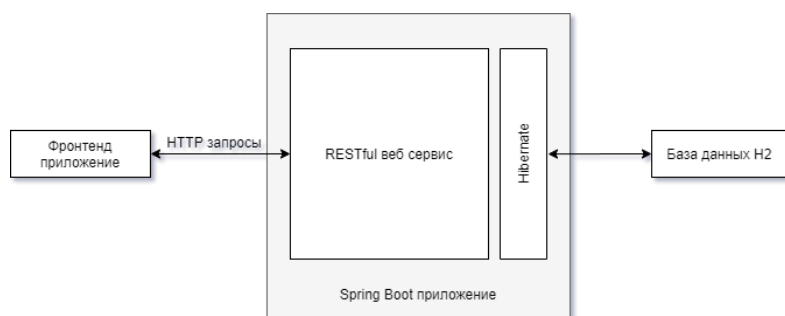


Рис. 1: Архитектура приложения

2.2 Проектирование базы данных

Первым этапом разработки приложения является разработка структуры базы данных. Были спроектированы следующие таблицы:

- thing - информация о типах хранимых вещей
 - id - pk integer autoincrement
 - name - string[128] not_null
- storage - информация о хранилищах
 - id - pk integer autoincrement
 - name - string[128] not_null
- section - информация о секциях хранилища
 - id - pk integer autoincrement
 - name - string[128] not_null
 - storage_id - fk integer storage
- thing_instance - информация о физических хранимых вещах
 - id - pk integer autoincrement
 - name - string[128] not_null
 - section_id - fk integer section
 - thing_id - fk integer section

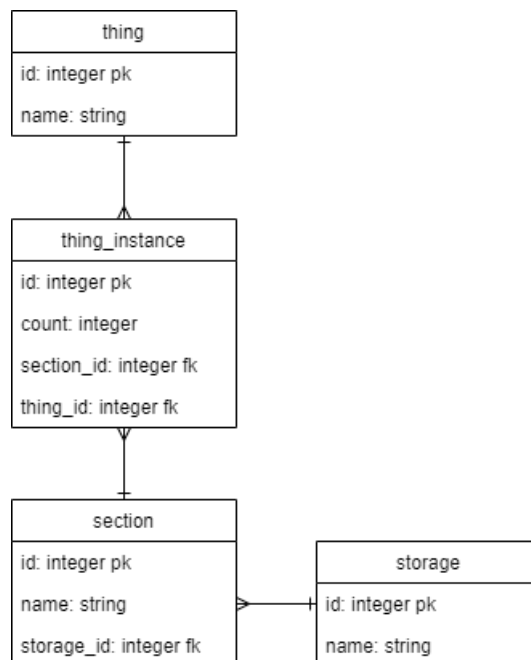


Рис. 2: Структура базы данных проекта

Для инициализации базы данных был написан скрипт на языке DML.

Отрывок из скрипта, заполняющий таблицу *section* (весь скрипт см. Приложение. Содержимое файла data.sql)

```
INSERT INTO section (id,name,storage_id) VALUES
(1,'Секция 1',1),
(2,'Секция 2',1),
(3,'Секция 3',1),
(4,'Секция 4',2);
```

2.3 Разработка серверной части приложения

Далее было получено шаблонное приложение Spring Boot при помощи конфигулятора встроенного в IDE.

Для каждой таблицы базы данных была создана модель (см. Приложение), используя аннотации Spring Framework.

@Entity - аннотация JPA указывающая что данный класс связан с соответствующей таблицей в базе данных.

Так же для описания связей или ограничений БД используются аннотации @Id, @ManyToOne, @OneToOne. Для автогенерации уникального идентификатора был применена аннотация @GeneratedValue, она принимает в качестве аргумента стратегию подбора нового идентификатора при его создании.

Для каждой модели был создан интерфейс-репозиторий реализующий интерфейс CrudRepository, в котором описаны типичные операции над объектами:

- Получение объекта по идентификатору.
- Удаление объекта.
- Частичное объекта.
- Создание объекта.

Используя функционал предоставленный репозиториями были реализованы классы-сервисы, в которых описаны методы-обертки над репозиториями в которых содержится вся “бизнес логика”.

Аннотация @RestController позволяет создать класс преобразующие все входящие HTTP запросы по указанным URL в вызовы методов класса к которому применена эта аннотация.

Аннотации @GetMapping, @PostMapping, @DeleteMapping принимают аргумент строку с URL, отправив на него соответствующий запрос, вызовется метод к которому применена аннотация. Так же в строке с URL могут быть указаны дополнительные переменные.

API принимает и отдает данные в формате сериализации JSON.

Пример объекта Thing сериализованного в формате JSON

```
"thing": {
  "id": 1,
  "name": "Штука 1" }
```

Полная схема API проекта выглядит следующим образом:

URL	Доступные методы запроса
/things	GET POST
/things/{id}	GET DELETE
/instances	GET POST
/instances/{id}	GET DELETE
/instances_of_section/{id}	GET
/instances_of_thing/{id}	GET
/storages	GET POST
/storages/{id}	GET DELETE
/sections	GET POST
/sections/{id}	GET DELETE
/sections_in_storage/{id}	GET

2.4 Разработка клиентской части приложения

Стартовый шаблон клиентского приложения был создан при помощи утилиты create-react-app, которая предоставляет полностью сконфигурированное окружение для быстрой разработки React приложения: настроенный сборщик, линтер и каркас проекта.

В качестве http клиента была использована библиотека axios.

Дополнительные библиотеки и фреймворки для построения интерфейса не использовались.

Пример react компонента

```
import React, {useEffect,useState} from 'react' import {Link}
from "react-router-dom"; import './NavBar.css'

export default function NavBar(){ return(
  <nav className='navbar'>
    <Link to="/things" className="navbar-item">Штучки</Link>
    <Link to="/storages" className="navbar-item">Хранилища</Link>
  </nav>
);
}
```

2.5 Проверка работоспособности

Работоспособность API проверялась при помощи утилиты Postman.

При GET запросе по URL /storages, приходил ответ содержащий JSON файл следующего содержания


```
[
  {
    "id": 1,
    "name": "Ящик 1"
  },
  {
    "id": 2,
    "name": "Ящик 2"
  },
  {
    "id": 3,
    "name": "Ящик 3"
  }
]
```

Работоспособность клиентской части проверялась в браузере в отладочном режиме

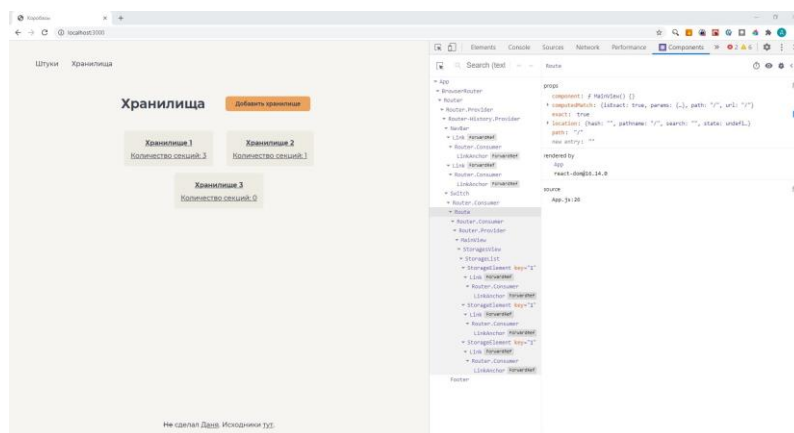


Рис. 3: Проверка работоспособности клиентского приложения

Заключение

В ходе данной работы были изучены архитектурные паттерны проектирования такие как REST, MVC, освоены навыки работы с библиотеками Spring и реляционной базой данных H2. Так же в процессе освоения данных технологий было реализовано RESTful приложение для учета и упорядочивания инвентаря.

Список литературы

- [1] Java Platform, Standard Edition 8 API Specification [Электронный ресурс]
URL: <https://docs.oracle.com/javase/8/docs/api/>
- [2] Java™ Platform, Standard Edition 7 API Specification [Электронный ресурс]
URL: <https://docs.oracle.com/javase/7/docs/api/>
- [3] Representational state transfer - Wikipedia [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Representational_state_transfer

[4] Lombok features	[Электронный ресурс]	URL:
https://projectlombok.org/features/all		
[5] Spring Boot Docs	[Электронный ресурс]	URL:
https://spring.io/projects/spring-boot		
[6] Spring Framework Docs	[Электронный ресурс]	URL:
https://spring.io/projects/spring-framework		
[7] H2 Documentation	[Электронный ресурс]	URL:
https://www.h2database.com/javadoc/index.html		
[8] Maven Documentation	[Электронный ресурс]	URL:
https://maven.apache.org/guides/index.html		
[9] React Docs	[Электронный ресурс]	URL: https://reactjs.org/docs
[10] facebook/create-react-app - Github	[Электронный ресурс]	URL: https://github.com/facebook/create-react-app

Приложение

Содержимое файла App.css

```
html{ font-size: 16px; }

body{
  margin: 0; height: 100%; font-family:
  'Montserrat', sans-serif; font-weight: 500; color:
  #3B434E; background-color: #F6F5F1;
}

a{
  color: #3B434E; text-decoration:
  underline; transition-duration: 0.1s;
}

a:hover{
  color: #EEA35D; }

input{
  width: calc(300px + 20%); text-
  align: center;

  padding-top: 8px; padding-
  bottom: 8px; outline: none;

  border-color: #EEA35D; border-
  radius: 7px; border-style: solid;

  font-family: 'Montserrat', sans-serif; font-size:
  1em; }

button{
```

```

    height: 36px; padding:
    10px 24px; margin: 8px
    16px;

    color: #3B434E; background-
    color: #EEA35D;

    font-weight: 700; font-family:
    inherit; outline: none;

    border-style: none; border-
    radius: 7px; transition-duration:
    0.1s;
}

button:focus{
    border-style: none;
}

button:hover{
    background-color: #ffc98d;
}

.root{ height: 100%; display: flex; min-
    height: 100vh; flex-direction:
    column; align-content: space-
    between;
}

table{
    background-color: #eeeebe3;
    margin: 0px; padding: 16px;
    border-radius: 7px; border-
    collapse: separate; border-
    spacing: 8px;
}

tr,td{
    text-align: justify;
}

.container{ min-height: calc(100vh - 80px);
}

.form-input{ margin-bottom:
    24px;
}

.form-input p{ font-weight: 500;
}

.form-input .form-error{ color:
    #9b0000; font-size: 0.8em; font-
    weight: 700;
}

```

```
.form-error{ color:
#9b0000; font-size: 0.8em;
font-weight: 700; }
```

Содержимое файла App.js

```
import React from "react"; import {
  BrowserRouter,
  Switch,
  Route,
  Redirect
} from "react-router-dom";

import NavBar from "../components/NavBar/NavBar"; import Footer
from "../components/Footer/Footer";

import MainView from "../views/MainView/MainView"; import ThingView
from "../views/ThingView/ThingView"; import StoragesView from
"./views/StoragesView/StoragesView"; import "../css/App.css"

export default function App(){ return(
  <BrowserRouter>
    <div className="container">
      <NavBar/>
      <Switch>
        <Route exact path="/" component={MainView}/>
        <Route path="/thing" component={ThingView}/>
        <Route path="/storages" component={StoragesView}/> </Switch>
      </div>
      <Footer/>
    </BrowserRouter>
  )
}
```

Содержимое файла CourseworkApplication.java

```
package ru.bstu.vt41.mds.coursework;

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

@SpringBootApplication public class
CourseworkApplication {

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

}
```

Содержимое файла data.sql

```
INSERT INTO storage (id,name) VALUES
(1,'Хранилище 1'),
(2,'Хранилище 2'),
(3,'Хранилище 3');
```

```

INSERT INTO thing (id,name) VALUES
(1,'Штука 1'),
(2,'Штука 2'),
(3,'Штука 3');
INSERT INTO section (id,name,storage_id) VALUES
(1,'Секция 1',1),
(2,'Секция 2',1),
(3,'Секция 3',1),
(4,'Секция 4',2);

INSERT INTO thing_instance (id,count,thing_id,section_id) VALUES
(1,10,1,1),
(2,11,2,2),
(3,12,2,3),
(4,13,1,1),
(5,14,2,2),
(6,15,1,3),
(7,16,1,1),
(8,17,2,2),
(9,18,2,3),
(10,19,1,1),
(11,20,2,2),
(12,0,1,3);

```

Содержимое файла index.js

```

import React from 'react'; import
ReactDOM from 'react-dom'; import App
from './App';

ReactDOM.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>, document.getElementById('root')
);

```

Содержимое файла NavBar.css

```

.navbar{ display: flex; align-
  items: center;

  height: 80px; padding-top:
  16px; padding-bottom: 16px;
  padding-left: 48px; padding-
  right: 48px;
}

.navbar-item{ color: #3B434E;
  text-decoration: none;
  padding-left: 16px; padding-
  right: 16px;
}

.navbar-logo{ color:
  #EEA35D;

```

```

    font-size: 56px; font-family: 'Amatic SC',
    cursive; margin-right: 16px;
}

@media screen and (max-width: 900px) {
    .navbar-logo{ display:
        none;
    }

    .navbar{ padding-left: 24px;
        padding-right: 24px;
    }
}

.navbar-logo:hover{ color:
    #ffc98d;
}

.push-right{ margin-right:
0px; margin-left: auto; }

```

Содержимое файла NavBar.js

```

import React, {useEffect,useState} from 'react' import {Link}
from "react-router-dom"; import './NavBar.css'

export default function NavBar(){ return(
    <nav className='navbar'>
        <Link to="/things" className="navbar-item">Штуки<
        <Link to="/storages" className="navbar-item">Хранилища
    </nav>
);
}

```

```

</Link>
</Link>

```

Содержимое файла Section.java

```

package ru.bstu.vt41.mds.coursework.models;

import lombok.AllArgsConstructor; import
lombok.Data; import javax.persistence.*;

@Data
@Entity public class Section
{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY) private
    Integer id; private String name;

    @ManyToOne(cascade = CascadeType.ALL) private
    Storage storage;
}

```

Содержимое файла SectionController.java

```

package ru.bstu.vt41.mds.coursework.controllers;

```

```

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.web.bind.annotation.*; import
ru.bstu.vt41.mds.coursework.models.Section; import
ru.bstu.vt41.mds.coursework.services.SectionService; import java.util.List;

@RestController
public class SectionController {
    @Autowired
    SectionService service;

    @GetMapping("/sections")
    private List<Section> getAllSections() { return
        service.getAllSections();
    }

    @GetMapping("/sections_in_storage/{storageId}")
    private List<Section> getSectionsInStorage(@PathVariable("storageId") int storageId) { return
        service.getSectionsInStorage(storageId);
    }

    @GetMapping("/sections/{id}")
    private Section getSection(@PathVariable("id") int id) { return
        service.getSectionById(id);
    }

    @DeleteMapping("/sections/{id}")
    private void deleteSection(@PathVariable("id") int id) { service.delete(id);
    }

    @PostMapping("/sections")
    private int saveStorage(@RequestBody Section section) {
        service.saveOrUpdate(section); return section.getId();
    }
}

```

Содержимое файла SectionRepo.java

```

package ru.bstu.vt41.mds.coursework.repos;

import org.springframework.data.repository.CrudRepository; import
ru.bstu.vt41.mds.coursework.models.Section; public interface SectionRepo extends
CrudRepository<Section,Integer> {}

```

Содержимое файла SectionService.java

```

package ru.bstu.vt41.mds.coursework.services;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Service; import
ru.bstu.vt41.mds.coursework.models.Section; import
ru.bstu.vt41.mds.coursework.repos.SectionRepo;

import java.util.ArrayList; import
java.util.List;

@Service
public class SectionService {

```

```

@Autowired SectionRepo
repo;
public List<Section> getAllSections() { List<Section> sections = new
    ArrayList<>(); repo.findAll().forEach(sections::add);
    return sections;
}

public List<Section> getSectionsInStorage(int storageId){ List<Section>
    sections = new ArrayList<>(); repo.findAll().forEach(section -> { if
    (section.getStorage().getId() == storageId){ sections.add(section);
        }
    }); return sections;
}

public Section getSectionById(int id) { return
    repo.findById(id).get();
}

public void saveOrUpdate(Section section) { repo.save(section);
}

public void delete(int id) { repo.deleteById(id);
}
}

```

Содержимое файла Storage.java

```

package ru.bstu.vt41.mds.coursework.models;

import lombok.AllArgsConstructor; import
lombok.Data; import javax.persistence.*;

@Data
@Entity public class Storage
{
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY) private
Integer id; private String name; }

```

Содержимое файла StorageController.java

```

package ru.bstu.vt41.mds.coursework.controllers; import

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

org.springframework.web.bind.annotation.*; import

ru.bstu.vt41.mds.coursework.models.Storage; import

ru.bstu.vt41.mds.coursework.services.StorageService; import java.util.List;

@CrossOrigin
@RestController
public class StorageController {
    @Autowired
    StorageService service;
}

```



```

@GetMapping("/storages") private List<Storage>
getAllStorages() { return service.getAllStorages();
}

@GetMapping("/storages/{id}")
private Storage getStorage(@PathVariable("id") int id) { return
    service.getStorageById(id);
}

@DeleteMapping("/storages/{id}")
private void deleteStorage(@PathVariable("id") int id) {
    service.delete(id);
}

@PostMapping("/storages")
private int saveStorage(@RequestBody Storage storage) {
    service.saveOrUpdate(storage); return storage.getId();
}
}

```

Содержимое файла StorageElement.css

```

.storage-element{ margin: 0.5em 1em;
padding: 1em; list-style:none;
border-radius: 7px; background-
color: #eeebe3; transition-
duration: 0.1s;
}

.storage-element:hover{ color:
    #3B434E; background-color:
    #e5e2d7;
}

a{ text-decoration: none;
}

.storage-element-header{ font-
weight: bold; margin: 4px 0px 12px
0px; }

```

Содержимое файла StorageElement.js

```

import React from "react"; import {Link} from "react-
router-dom"; import './StorageElement.css' import
{Spinner} from "../Spinner/Spinner";

export default function StorageElement(props){ const data =
    props.storageInfo

    if (data == null){ return <Spinner
        spinnerSize='small' />
    }else{ return(
        <Link to={"/storage/" + data.id}>
            <div className = "storage-element">
                <div className= "storage-element-header">
                    {data.name}
                </div>
            </div>
        </div>
    )}
}

```

```

        <div className= "storage-element-section-counter">
            {"Количество секций: " + data.sections.length}
        </div>
    </div>
</Link>
)
}
}

```

Содержимое файла StorageList.css

```

.storage-list{ display: flex; justify-
content: center; flex-wrap: wrap;
padding: 0; }

```

Содержимое файла StorageList.js

```

import React, {useEffect, useState} from "react"; import {Link}
from "react-router-dom"; import './StorageList.css'
import StorageElement from "../StorageElement/StorageElement"; import
{Spinner} from "../Spinner/Spinner"; import axios from 'axios'; export default
function StorageList(props){ const [components,setComponents] =
useState(null);

const getStorages = async () =>{ try{ const
response = await axios({ method: 'get',
url: 'localhost:8080/storages',
});
console.log(response)
return Promise.resolve(response.data);
} catch (e){ return
null
}
}

useEffect(
    ()=> {
        getStorages().then( data => {setComponents( data.map( data => <StorageElement key={data.id}
storageInfo={data}/>)
        })
    },[])

return(
    <ul className="storage-list">
        {components}
    </ul>
)
}
}

```

Содержимое файла StorageRepo.java

```

package ru.bstu.vt41.mds.coursework.repos;

```

```
import org.springframework.data.repository.CrudRepository; import
ru.bstu.vt41.mds.coursework.models.Storage; public interface StorageRepo extends
CrudRepository<Storage,Integer> {}
```

Содержимое файла StorageService.java

```
package ru.bstu.vt41.mds.coursework.services;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Service; import
ru.bstu.vt41.mds.coursework.models.Storage; import
ru.bstu.vt41.mds.coursework.repos.StorageRepo;

import java.util.ArrayList; import
java.util.List;

@Service
public class StorageService {
    @Autowired StorageRepo
    repo;

    public List<Storage> getAllStorages() { List<Storage> storages =
        new ArrayList<>(); repo.findAll().forEach(storages::add);
        return storages;
    }

    public Storage getStorageById(int id) { return
        repo.findById(id).get();
    }

    public void saveOrUpdate(Storage storage) { repo.save(storage);
    } public void delete(int id) {

        repo.deleteById(id);

    }
}
```

Содержимое файла Thing.java

```
package ru.bstu.vt41.mds.coursework.models;

import lombok.AllArgsConstructor; import
lombok.Data; import javax.persistence.*;

@Data
@Entity public class
Thing {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY) private
Integer id; private String name; }
```

Содержимое файла ThingController.java

```
package ru.bstu.vt41.mds.coursework.controllers;
```

```

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.web.bind.annotation.*; import
ru.bstu.vt41.mds.coursework.models.Thing; import
ru.bstu.vt41.mds.coursework.services.ThingService; import java.util.List;

@RestController public class
ThingController {
    @Autowired
    ThingService service;

    @GetMapping("/things") private List<Thing>
getAllThings() { return service.getAllThings();
}

    @GetMapping("/things/{id}")
private Thing getThing(@PathVariable("id") int id) { return
    service.getThingById(id);
}

    @DeleteMapping("/things/{id}")
private void deleteThing(@PathVariable("id") int id) { service.delete(id);
}

    @PostMapping("/things")
private int saveThing(@RequestBody Thing person) {
    service.saveOrUpdate(person); return person.getId();
}
}

```

Содержимое файла ThingInstance.java

```

package ru.bstu.vt41.mds.coursework.models;

import lombok.AllArgsConstructor; import
lombok.Data; import javax.persistence.*;

@Data
@Entity
public class ThingInstance {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY) private
Integer id; private Long count;

    @ManyToOne(cascade = CascadeType.ALL) private
Thing thing;

    @ManyToOne(cascade = CascadeType.ALL) private
Section section;
}

```

Содержимое файла ThingInstanceController.java

```

package ru.bstu.vt41.mds.coursework.controllers;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.web.bind.annotation.*; import
ru.bstu.vt41.mds.coursework.models.Thing; import

```

```

ru.bstu.vt41.mds.coursework.models.ThingInstance; import
ru.bstu.vt41.mds.coursework.services.ThingInstanceService; import
ru.bstu.vt41.mds.coursework.services.ThingService; import java.util.List;

@RestController
public class ThingInstanceController {
    @Autowired
    ThingInstanceService service;

    @GetMapping("/instances")
    private List<ThingInstance> getAllThingInstance() { return
        service.getAllThingInstances();
    }

    @GetMapping("/instances_in_section/{sectionId}") private List<ThingInstance>
    getThingInstancesInSection(@PathVariable("sectionId") int
    , sectionId) { return service.getThingInstancesInSection(sectionId);
    }

    @GetMapping("/instances_of_thing/{thingId}")
    private List<ThingInstance> getInstancesOfThing(@PathVariable("thingId") int thingId) { return
        service.getThingInstancesInSection(thingId);
    }

    @GetMapping("/instances/{id}")
    private ThingInstance getThingInstance(@PathVariable("id") int id) { return
        service.getThingInstancesById(id);
    }

    @DeleteMapping("/instances/{id}")
    private void deleteThingInstance(@PathVariable("id") int id) { service.delete(id);
    }

    @PostMapping("/instances")
    private int saveThingInstance(@RequestBody ThingInstance instance) {
        service.saveOrUpdate(instance); return instance.getId();
    }
}

```

Содержимое файла ThingInstanceRepo.java

```

package ru.bstu.vt41.mds.coursework.repos;

import org.springframework.data.repository.CrudRepository; import
ru.bstu.vt41.mds.coursework.models.ThingInstance; public interface ThingInstanceRepo extends
CrudRepository<ThingInstance,Integer> {}

```

Содержимое файла ThingInstanceService.java

```

package ru.bstu.vt41.mds.coursework.services;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Service; import
ru.bstu.vt41.mds.coursework.models.Thing; import
ru.bstu.vt41.mds.coursework.models.ThingInstance; import
ru.bstu.vt41.mds.coursework.repos.ThingInstanceRepo; import
ru.bstu.vt41.mds.coursework.repos.ThingRepo;

```

```

import java.util.ArrayList; import
java.util.List;

@Service
public class ThingInstanceService {
    @Autowired
    ThingInstanceRepo repo;
    public List<ThingInstance> getAllThingInstances() {
        List<ThingInstance> instances = new ArrayList<>();
        repo.findAll().forEach(instances::add); return instances;
    }

    public List<ThingInstance> getThingInstancesInSection(int sectionId){
        List<ThingInstance> instances = new ArrayList<>();
        repo.findAll().forEach(instance -> { if (instance.getSection().getId()
        == sectionId){ instances.add(instance);
        }
        });
        return instances;
    } public List<ThingInstance> getInstancesForThing(int thingId){

        List<ThingInstance> instances = new ArrayList<>();

        repo.findAll().forEach(instance -> { if (instance.getThing().getId() == thingId){

            instances.add(instance);

        }
        });
        return instances;
    }

    public ThingInstance getThingInstancesById(int id) { return
        repo.findById(id).get();
    }

    public void saveOrUpdate(ThingInstance instance) { repo.save(instance);
    }

    public void delete(int id) { repo.deleteById(id);
    }
}

```

Содержимое файла ThingRepo.java

```

package ru.bstu.vt41.mds.coursework.repos;

import org.springframework.data.repository.CrudRepository; import
ru.bstu.vt41.mds.coursework.models.Thing; public interface ThingRepo extends
CrudRepository<Thing,Integer> {}

```

Содержимое файла ThingService.java

```

package ru.bstu.vt41.mds.coursework.services;

import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.stereotype.Service; import

```

```

ru.bstu.vt41.mds.coursework.models.Thing; import
ru.bstu.vt41.mds.coursework.repos.ThingRepo;

import java.util.ArrayList; import
java.util.List;

@Service
public class ThingService {
    @Autowired ThingRepo
    repo;

    public List<Thing> getAllThings() { List<Thing> things = new
        ArrayList<>(); repo.findAll().forEach(things::add);
        return things;
    }

    public Thing getThingById(int id) { return
        repo.findById(id).get();
    }

    public void saveOrUpdate(Thing thing) { repo.save(thing);
    }

    public void delete(int id) {
        repo.deleteById(id);
    }
}

```