Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 6

Поддержка архитектуры REST в Spring
Тема

Преподаватель _____ А.С. Черниговский
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цель

Познакомиться с механизмами поддержки архитектуры REST в Spring.

2 Задачи

Взять практическое задание №5 (или №4, на усмотрение студента, при работе с защищенным приложением могут возникнуть трудности) и добавить следующий функционал:

1) преобразовать веб-приложение таким образом, чтобы оно поддерживало архитектуру REST. Должны поддерживаться следующие типы запросов: GET (показ (html) и извлечение (json) всех/одной записей/сущностей), POST (добавление), PUT (редактирование), DELETE (удаление).

2) разработать REST-клиент для вашего приложения, который, используя RestTemplate позволяет выполнять базовые операции по извлечению (GET), добавлению (POST), редактированию (PUT), удалению (DELETE) ресурсов. REST-клиент не обязан иметь пользовательский интерфейс, необходим тестовый пример, который можно запускать из консоли.

3) обязательным условием является сохранение всего предшествующего функционала приложения. Для удовлетворения всем характеристикам RESTархитектуры приложение может быть реорганизовано (убраны GET-запросы с параметрами) или добавлен новый функционал.

4) PUT и DELETE запросы не обязательно делать через запросы из браузера. Достаточно реализации для клиентов-приложений.

3 Описание варианта

Класс «Канцтовары».

4 Листинги программ

4.1 Серверная часть

На листингах 1-3 представлен программный код классов-конфигураторов.

Листинг 1 – Конфигурация БД

```
package ru.nikitin.configs;

import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.jpa.JpaTransactionManager;
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;
import org.springframework.transaction.PlatformTransactionManager;

import javax.persistence.EntityManagerFactory;
import javax.sql.DataSource;
import java.util.Objects;
import java.util.Properties;

@Configuration
@ComponentScan("ru.nikitin")
@PropertySource("classpath:application.properties")
@EnableJpaRepositories("ru.nikitin")
@RequiredArgsConstructor
public class AppConfig {
    private final Environment env;

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(Objects.requireNonNull(env.getProperty("spring.dat
asource.driverClassName")));
```

Окончание листинга 1

```
        dataSource.setUrl(env.getProperty("spring.dataSource.url"));
        dataSource.setUsername(env.getProperty("spring.dataSource.username"));
        dataSource.setPassword(env.getProperty("spring.dataSource.password"));

        return dataSource;
    }

    @Bean
    public Properties jpaProperties() {
        return new Properties();
    }

    @Bean
    @Autowired
    public EntityManagerFactory entityManagerFactory(DataSource dataSource,
Properties jpaProperties) {
        HibernateJpaVendorAdapter vendorAdapter = new
HibernateJpaVendorAdapter();
        LocalContainerEntityManagerFactoryBean factory = new
LocalContainerEntityManagerFactoryBean();

        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("ru.nikitin");
        factory.setDataSource(dataSource);
        factory.setJpaProperties(jpaProperties);
        factory.afterPropertiesSet();
        return factory.getObject();
    }

    @Bean
    @Autowired
    public PlatformTransactionManager transactionManager(EntityManagerFactory
entityManagerFactory) {
        JpaTransactionManager txManager = new JpaTransactionManager();
        txManager.setEntityManagerFactory(entityManagerFactory);
        return txManager;
    }

}
```

Листинг 2 – Конфигурация сервера

```
package ru.nikitin.configs;

import lombok.Data;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.thymeleaf.spring5.SpringTemplateEngine;
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;
import org.thymeleaf.spring5.view.ThymeleafViewResolver;

@Configuration
@EnableWebMvc
@ComponentScan("ru.nikitin.controllers")
public class WebConfig implements WebMvcConfigurer {
    private final ApplicationContext applicationContext;
    @Autowired
    public WebConfig(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver templateResolver = new
                SpringResourceTemplateResolver();
        templateResolver.setApplicationContext(applicationContext);
        templateResolver.setPrefix("/WEB-INF/views/");
        templateResolver.setSuffix(".html");
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine() {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        SpringResourceTemplateResolver resolver = templateResolver();
```

Окончание листинга 2

```
        resolver.setCharacterEncoding("UTF-8");
        templateEngine.setTemplateResolver(resolver);
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }
    @Override
    public void configureViewResolvers(ViewResolverRegistry registry) {
        ThymeleafViewResolver resolver = new ThymeleafViewResolver();
        resolver.setTemplateEngine(templateEngine());
        resolver.setCharacterEncoding("UTF-8");
        resolver.setContentType("text/html;charset=UTF-8");
        registry.viewResolver(resolver);
    }
}
```

Листинг 3 – Конфигурация сервлета

```
package ru.nikitin.configs;

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServle
tInitializer;

public class DispatcherConfig extends
AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class<?>[] { AppConfig.class };
    }
    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class<?>[] { WebConfig.class };
    }
    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }
}
```

На листингах 4-7 представлен код с контроллерами сервера, реализующие логику запросов.

Листинг 4 – Класс-контроллер регистрации и авторизации

```
package ru.nikitin.server.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import ru.nikitin.server.services.UserService;

@Controller
@RequestMapping("/account")
class RegisterController {
    private final PasswordEncoder passwordEncoder;
    private final UserService userService;

    @Autowired
    public RegisterController(@Qualifier("passwordEncoder") PasswordEncoder
passwordEncoder,
                              @Qualifier("userServiceImpl") UserService
userService) {
        this.passwordEncoder = passwordEncoder;
        this.userService = userService;
    }

    @GetMapping("/register")
    public String register() {
        return "registration";
    }

    @PostMapping("/register")
    public String registerHandler(@RequestParam String username, @RequestParam
String password) {
        try {
            boolean registerSuccess = userService.register(username,
passwordEncoder.encode(password));
            if (registerSuccess)
                System.out.println("Создан пользователь " + username);
            else {
```

Окончание листинга 4

```
                System.out.println("Существует пользователь " + username);
                return "registration";
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return "redirect:/account/auth";
    }

    @GetMapping("/auth")
    public String auth() {
        return "auth";
    }
}
```

Листинг 5 – Класс-контроллер запросов с браузера

```
package ru.nikitin.server.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import ru.nikitin.server.entities.Stationery;
import ru.nikitin.server.services.impl.StationeryServiceImpl;

import javax.validation.Valid;

@Controller
@RequestMapping("/stationery")
class StationeryController {
    private final StationeryServiceImpl stationeryService;

    @Autowired
    public              StationeryController(@Qualifier("stationeryServiceImpl")
StationeryServiceImpl stationeryService) {
        this.stationeryService = stationeryService;
    }
```

Продолжение листинга 5

```
    @GetMapping("/home")
    public String mainPage(Model model) {
        model.addAttribute("stationery", stationeryService.getAll());
        return "home";
    }

    @GetMapping("/add")
    public String addPage(Model model) {
        model.addAttribute("stationery", new Stationery());
        return "add";
    }

    @PostMapping("/add")
    public String addFormHandler(@ModelAttribute("stationery") Stationery
stationery) {
        stationeryService.add(stationery);
        return "redirect:/stationery/home";
    }

    @GetMapping("/edit")
    public String editPage(Model model) {
        model.addAttribute("stationery", new Stationery());
        return "edit";
    }

    @PutMapping("/edit")
    public String editFormHandler(
            @ModelAttribute @Valid Stationery stationery,
            BindingResult bindingResult,
            @RequestParam("id") Integer id) {
        System.out.println(bindingResult);
        if (bindingResult.hasErrors())
            return "redirect:/stationery/edit";

        stationeryService.update(id, stationery);
        return "redirect:/stationery/home";
    }

    @GetMapping("/delete")
    public String deletePage() {
        return "delete";
```

Окончание листинга 5

```
    }

    @DeleteMapping("/delete")
    public String deleteFormHandler( @RequestParam("id") Integer id ) {
        stationeryService.delete(id);
        return "redirect:/stationery/home";
    }

    @GetMapping("/show_criteria")
    public String criteriaPage() {
        return "criteria";
    }

    @PostMapping("/show_criteria")
    public String criteriaFormHandler( @RequestParam String manufacturer,
ModelMap model ) {
        model.addAttribute("stationery",
stationeryService.getByManufacturer(manufacturer));
        return "criteria_result";
    }
}
```

Листинг 6 – Класс-контроллер запросов с java-приложения

```
package ru.nikitin.server.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import ru.nikitin.server.entities.Stationery;
import ru.nikitin.server.services.impl.StationeryServiceImpl;

import java.util.List;

@Controller
@RequestMapping("/json/stationery")
public class StationeryJsonController {
    private final StationeryServiceImpl stationeryService;

    @Autowired
```

Окончание листинга 6

```
    public          StationeryJsonController(@Qualifier("stationeryServiceImpl")
StationeryServiceImpl stationeryService) {
        this.stationeryService = stationeryService;
    }

    @GetMapping(value = "/table", headers = {"Accept=application/json"})
    public @ResponseBody
    List<Stationery> jsonStationeryTable() {
        return stationeryService.getAll();
    }

    @ResponseBody
    @GetMapping(value="/table/{manufacturer}",            headers            =
{"Accept=application/json"})
    public  List<Stationery>  getStationeryObject(@PathVariable("manufacturer")
String manufacturer) {
        return stationeryService.getByManufacturer(manufacturer);
    }


    @PostMapping(value = "/table", headers = "Content-Type=application/json")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void jsonAddPage(@RequestBody Stationery stationery) {
        stationeryService.add(stationery);
    }

    @PutMapping(value = "/table/{id}", headers = {"Accept=application/json"})
    public @ResponseBody
    void  jsonEditPage(@PathVariable("id")  int  id,  @RequestBody  Stationery
stationery) {
        stationeryService.update(id, stationery);
    }

    @DeleteMapping("/table/{id}")
    public @ResponseBody
    void jsonDeletePage(@PathVariable("id") int id) {
        stationeryService.delete(id);
    }
}
```

Листинг 7 – Класс-контроллер корневых запросов

```
package ru.nikitin.server.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class RouteController {

    @GetMapping("/")
    public String main_redirect() {
        return "redirect:/stationery/home";
    }

}
```

На листингах 8-11 представлен код с логикой сервера, реализующий задание.

Листинг 8 – Класс, реализующий обращения к БД

```
package ru.nikitin.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import ru.nikitin.services.StationeryService;
import ru.nikitin.entities.Stationery;
import ru.nikitin.repos.StationeryRepository;

import java.util.List;
import java.util.Optional;

@Service
public class StationeryServiceImpl implements StationeryService {
    @Autowired
    private StationeryRepository stationeryRepository;

    @Override
    public boolean add(Stationery stationery) {
        this.stationeryRepository.save(stationery);
        return true;
    }
```

Окончание листинга 8

```
@Override
    public boolean delete(Integer id) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            return true;
        }
        return false;
    }

    @Override
    public boolean update(Integer id, Stationery stationery) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            this.stationeryRepository.save(stationery);
            return true;
        }
        return false;
    }

    @Override
    public List<Stationery> getByManufacturer(String manufacturer) {
        return
this.stationeryRepository.findStationeryByManufacturer(manufacturer);
    }
    @Override
    public List<Stationery> getAll() {
        return this.stationeryRepository.findAll(Sort.by("type"));
    }

    public boolean checkId(int id) {
        return this.stationeryRepository.findById(id).isPresent();
    }

    public Optional<Stationery> get(Integer id) {
        return this.stationeryRepository.findById(id);
    }
}
```

Листинг 9 – Класс-сущность канцелярского предмета

```
package ru.nikitin.entities;

import lombok.AccessLevel;
import lombok.Setter;

import javax.persistence.*;
import javax.validation.constraints.Min;

@Entity
@Table(name = "stationery")
public class Stationery {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    @Setter(AccessLevel.NONE)
    private Integer id;

    @Column(name = "type")
    private String type;

    @Column(name = "price")
    @Min(0)
    private Double price;

    @Column(name = "amount")
    @Min(0)
    private Integer amount;

    @Column(name = "subtype")
    private String subtype;

    @Column(name = "manufacturer")
    private String manufacturer;

    public Stationery() {
    }

    public Stationery(String type, String subtype, Double price, Integer amount,
String manufacturer) {
        this.type = type;
        this.price = price;
```

Продолжение листинга 9

```
        this.amount = amount;
        this.subtype = subtype;
        this.manufacturer = manufacturer;
    }

    public Integer getId() {
        return id;
    }

    public String getType() {
        return type;
    }

    public Double getPrice() {
        return price;
    }

    public Integer getAmount() {
        return amount;
    }

    public String getSubtype() {
        return subtype;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setType(String type) {
        this.type = type;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public void setAmount(Integer amount) {
        this.amount = amount;
    }
```

Окончание листинга 9

```
    public void setSubtype(String subtype) {
        this.subtype = subtype;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    @Override
    public String toString() {
        return "id: " + this.id + " | " + "type: " + this.type + " | " + "subtype:
" + this.subtype + " | "
                + "price: " + this.price + " | " + "manufacturer: " +
this.manufacturer
                + " | " + "amount: " + this.amount;
    }
}
```

Листинг 10 – Класс-репозиторий БД

```
package ru.nikitin.repos;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import ru.nikitin.entities.Stationery;

import java.util.List;

@Repository
public interface StationeryRepository extends JpaRepository<Stationery, Integer>
{
    List<Stationery> findStationeryByManufacturer(String manufacturer);
}
```

Листинг 11 – Класс, реализующий обращения к БД

```
package ru.nikitin.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import ru.nikitin.services.StationeryService;
import ru.nikitin.entities.Stationery;
```

Продолжение листинга 11

```
import ru.nikitin.repos.StationeryRepository;

import java.util.List;
import java.util.Optional;

@Service
public class StationeryServiceImpl implements StationeryService {
    @Autowired
    private StationeryRepository stationeryRepository;

    @Override
    public boolean add(Stationery stationery) {
        this.stationeryRepository.save(stationery);
        return true;
    }

    @Override
    public boolean delete(Integer id) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            return true;
        }
        return false;
    }

    @Override
    public boolean update(Integer id, Stationery stationery) {
        if (checkId(id)) {
            this.stationeryRepository.deleteById(id);
            this.stationeryRepository.save(stationery);
            return true;
        }
        return false;
    }

    @Override
    public List<Stationery> getByManufacturer(String manufacturer) {
        return
this.stationeryRepository.findStationeryByManufacturer(manufacturer);
    }
    @Override
```

Окончание листинга 11

```
    public List<Stationery> getAll() {
        return this.stationeryRepository.findAll(Sort.by("type"));
    }

    public boolean checkId(int id) {
        return this.stationeryRepository.findById(id).isPresent();
    }

    public Optional<Stationery> get(Integer id) {
        return this.stationeryRepository.findById(id);
    }
}
```

4.2 Клиентская часть

На листингах 12-14 представлены листинги с логикой клиентской части программы.

Листинг 12 – Обработчик пользовательского ввода

```
package ru.nikitin.client;

import ru.nikitin.server.entities.Stationery;

import java.util.Scanner;

public class InputHandler {
    public static String inputString(boolean lower) {
        Scanner in = new Scanner(System.in);
        String text = in.nextLine();
        while (text.equals("")) {
            System.out.println("Поле не должно быть пустым!");
            text = in.nextLine();
        }
        if (lower)
            text = text.toLowerCase();
        return text;
    }

    public static int inputNat(boolean strict) {
        Scanner in = new Scanner(System.in);
```

Продолжение листинга 12

```
        int x;
        while (true) {
            try {
                x = in.nextInt();
                in.nextLine();

                if (x < 0)
                    throw new java.util.InputMismatchException("Value is below
zero!");

                if (strict && x <= 0)
                    throw new java.util.InputMismatchException("Value is below
zero!");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value!");
                in.nextLine();
            }
        }
        return x;
    }

    public static Double inputDouble(double bottomBound, double upperBound) {
        Scanner in = new Scanner(System.in);

        double x;
        while (true) {
            try {
                x = in.nextDouble();
                in.nextLine();

                if (x < bottomBound || x > upperBound)
                    throw new java.util.InputMismatchException("Incorrect
value!");
                break;

            } catch (java.util.InputMismatchException e) {
                System.out.println("Input the correct value!");
                in.nextLine();
            }
```

Окончание листинга 12

```
        }
        return x;
    }

    public static int inputId() {
        System.out.println("Введите id элемента:");
        return inputNat(false);
    }

    public static Stationery inputStationery() {

        System.out.println("Введите тип канцтовара (обязательное поле):");
        String type = inputString(true);

        System.out.println("Введите подтип канцтовара:");
        String subtype = inputString(true).toLowerCase();

        System.out.println("Введите цену товара (обязательное поле):");
        Double price = inputDouble(0, Double.POSITIVE_INFINITY);

        System.out.println("Введите количество канцтовара на складе:");
        int amount = inputNat(false);

        System.out.println("Введите производителя канцтовара:");
        String manufacturer = inputString(true).toLowerCase();

        return new Stationery(type, subtype, price, amount, manufacturer);
    }
}
```

Листинг 13 – Обработчик http-запросов

```
package ru.nikitin.client;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.springframework.web.client.RestTemplate;
```

Окончание листинга 13

```
import ru.nikitin.server.entities.Stationery;

import java.net.http.HttpClient;
import java.util.Optional;

public class Requests {
    private static final RestTemplate restTemplate = new RestTemplate();

    public static Stationery[] getStationeryTable(String targetURL) {
        return restTemplate.getForObject(targetURL, Stationery[].class);
    }

    public static void postStationery(String targetURL, Stationery stationery) {
        restTemplate.postForObject(targetURL, stationery, Stationery.class);
    }

    public static void updateStationery(String targetURL, Stationery stationery)
{
        restTemplate.put(targetURL, stationery, stationery.getId());
    }

    public static void deleteStationery(String targetURL, int id) {
        restTemplate.delete(targetURL, id);
    }
}
```

Листинг 14 – Класс Main

```
package ru.nikitin.client;

import ru.nikitin.server.entities.Stationery;

import java.net.MalformedURLException;
import java.net.URL;

public class Main {

    public static void main(String[] args) {

        try {
            URL url = new URL("http://localhost:8080/");
        } catch (MalformedURLException e) {
```

Продолжение листинга 14

```
            System.out.println("Failing to establish a connection!");
            return;
        }

        while (true) {
            System.out.println("Введите команду:");
            System.out.println("1. Добавить новый элемент;");
            System.out.println("2. Вывести содержимое БД;");
            System.out.println("3. Редактировать элемент;");
            System.out.println("4. Удалить элемент;");
            System.out.println("5.      Поиск      продукции      определенного
производителя;");
            System.out.println("6. Завершение работы приложения.");

            int choice = InputHandler.inputNat(true);
            switch (UserMenu.values()[choice - 1]) {
                case ADD -> {
                    Stationery stationery = InputHandler.inputStationery();

Requests.postStationery("http://localhost:8080/json/stationery/table",
stationery);
                    System.out.println("Element has been successfully added!");
                }

                case OUTPUT -> {
                    Stationery[]                    response                    =
Requests.getStationeryTable("http://localhost:8080/json/stationery/table");

                    for (Stationery stationery: response) {
                        System.out.println(stationery);
                    }

                }

                case UPDATE -> {
                    int id = InputHandler.inputId();
                    Stationery stationery = InputHandler.inputStationery();
                    Requests.updateStationery(
                            "http://localhost:8080/json/stationery/table/" + id,
                            stationery
                    );
```

Окончание листинга 14

```
                    System.out.println("Element has been successfully updated!");
                }

                case REMOVE -> {
                    int id = InputHandler.inputId();

                    Requests.deleteStationery(
                            "http://localhost:8080/json/stationery/table/" + id,
                            id
                    );

                    System.out.println("Element has been successfully deleted!");
                }

                case SEARCH -> {
                    System.out.println("Введите производителя:");
                    String manufacturer = InputHandler.inputString(true);

                    Stationery[] response = Requests.getStationeryTable(
                            "http://localhost:8080/json/stationery/table/"     +
manufacturer
                    );

                    for (Stationery stationery: response) {
                        System.out.println(stationery);
                    }
                }

                case EXIT -> {
                    System.exit(1);
                }

                default -> {
                    System.out.println("Введите значение от 1 до 6!");
                }
            }
        }
    }
}
```

5 Вывод

В результате работы были закреплены на практике знания сетевых запросов и практики REST и их применения в фреймворке Spring, а также реализована программа по заданию.