КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |
|---|---|---|

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

Разработка ресурса REST/JSON сервиса

по дисциплине: Технология разработки серверных информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. _____ _____ _____

подпись, дата | инициалы, фамилия

Санкт-Петербург
2023

**Текст и вариант задания:**

13. Торговля акциями на бирже.

**Описание разрабатываемого продукта:**

В программе создан сервис для покупки, продажи и просмотра портфеля акций.

**Задание на лабораторную работу.**

1. Определите перечень Rest-сервисов, выполняющих те же действия, что и в лабораторной работе 1. Внимательно отнеситесь к вопросу какой HTTP метод использует тот или иной сервис и какие коды HTTP он может возвращать.
2. Опишите Ваш API с помощью OpenAPI v3.
3. Средствами swagger сгенерируйте сервер spring
4. Замените pom файл в сгенерированном сервере на файл из моего примера
5. Поменяйте названия пакетов на уникальные
6. Реализуйте приложение.

**Текст основных фрагментов кода:**

Itemobj.java

```
package com.example.RestJavaProject1;
public class ItemObj {

    private String stock_name;
    private int stockID;
    private String purchase_date;

    // Конструктор
    public ItemObj(String stock_name, int stockID, String purchase_date) {
        this.stock_name = stock_name;
        this.stockID = stockID;
        this.purchase_date = purchase_date;
    }
```

```
// Геттеры и сеттеры
public String getStock_name() {
    return stock_name;
}

public void setStock_name(String stock_name) {
    this.stock_name = stock_name;
}

public int getStockID() {
    return stockID;
}

public void setStockID(int stockID) {
    this.stockID = stockID;
}

public String getPurchase_date() {
    return purchase_date;
}

public void setPurchase_date(String purchase_date) {
    this.purchase_date = purchase_date;
}

@Override
public String toString() {
    return "ItemObj{" +
            "stock_name='" + stock_name + '\'' +
```

```
            ", stockID=" + stockID +
            ", purchase_date='" + purchase_date + '\'' +
            '}';
    }
}
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Торговля акциями</title>
</head>
<body>

<h2>Покупка акции</h2>
<form action="/buyStock" method="post">
  Название акции: <input type="text" name="stock_name"><br>
  ID акции: <input type="number" name="stockID"><br>
  Дата покупки: <input type="date" name="purchase_date"><br>
  <input type="submit" value="Купить">
</form>

<h2>Продажа акции</h2>
<form action="/sellStock" method="post">
  Название акции: <input type="text" name="stock_name"><br>
  ID акции: <input type="number" name="stockID"><br>
  <input type="submit" value="Продать">
</form>
```

```
<h2>Список акций</h2>
<table border="1">
  <tr>
    <th>Название акции</th>
    <th>ID акции</th>
    <th>Дата покупки</th>
  </tr>
  <!-- Данные для таблицы будут добавляться динамически с сервера -->
</table>

</body>
</html>
```

App.js

```
function formatDate(dateString) {
    const options = { year: 'numeric', month: 'long', day: 'numeric' };
    return new Date(dateString).toLocaleDateString('ru-RU', options);
}

function getStocks() {
    fetch('/stocks')
        .then(response => response.json())
        .then(stocks => {
            const stocksTable = document.getElementById('stocks');
            stocksTable.innerHTML = '';
            const headerRow = document.createElement('tr');
            ['№', 'Название акции', 'ID акции', 'Дата покупки'].forEach(header => {
                const headerCell = document.createElement('th');
                headerCell.textContent = header;
```

```
            headerRow.appendChild(headerCell);
        });
        stocksTable.appendChild(headerRow);
        stocks.forEach((stock, index) => {
            const row = document.createElement('tr');
            const indexCell = document.createElement('td');
            indexCell.textContent = index + 1;
            row.appendChild(indexCell);
            ['stock_name', 'stockID', 'purchase_date'].forEach(field => {
                const cell = document.createElement('td');
                cell.textContent = stock[field];
                row.appendChild(cell);
            });
            stocksTable.appendChild(row);
        });
    });
}

function buyStock() {
    const stock_name = document.getElementById('stock_name').value;
    const stockID = document.getElementById('stockID').value;
    const purchase_date = document.getElementById('purchase_date').value;
    const stock = { stock_name, stockID, purchase_date };

    fetch('/buyStock', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
```

```
        body:
`stock_name=${stock_name}&stockID=${stockID}&purchase_date=${purchase_date}`,
    })
    .then(response => response.text())
    .then(result => {
        console.log('Акция куплена:', result);
        getStocks();
    });
}

function sellStock() {
    const stock_name = document.getElementById('stock_name_sell').value;
    const stockID = document.getElementById('stockID_sell').value;

    fetch(`/sellStock?stock_name=${stock_name}&stockID=${stockID}`, {
        method: 'POST',
    })
    .then(response => response.text())
    .then(result => {
        console.log('Акция продана:', result);
        getStocks();
    });
}

getStocks();
```

stocksontroller

```
package com.example.RestJavaProject1;

import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

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

@RestController
public class StockController {

    private final List<ItemObj> stocks = new ArrayList<>();

    @GetMapping("/stocks")
    public List<ItemObj> getAllStocks() {
        return stocks;
    }

    @PostMapping("/buyStock")
    public ResponseEntity<String> buyStock(@RequestParam String stock_name,
@RequestParam int stockID, @RequestParam String purchase_date) {
        stocks.add(new ItemObj(stock_name, stockID, purchase_date));
        return ResponseEntity.ok("Stock bought successfully!");
    }

    @PostMapping("/sellStock")
    public ResponseEntity<String> sellStock(@RequestParam String stock_name,
@RequestParam int stockID) {
        ItemObj stockToRemove = null;
        for (ItemObj stock : stocks) {
            if (stock.getStock_name().equals(stock_name) && stock.getStockID() == stockID) {
                stockToRemove = stock;
```

```
                break;
            }
        }
        if (stockToRemove != null) {
            stocks.remove(stockToRemove);
            return ResponseEntity.ok("Stock sold successfully!");
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Stock not found!");
        }
    }
}
```

Swaggerconfig.java

```
package com.example.RestJavaProject1;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
```

```
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.example.RestJavaProject1"))
            .paths(PathSelectors.any())
            .build();
    }
}
```