МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения информационных систем и технологий

**О Т Ч Е Т**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

# «Разработка веб-приложений»

**по курсу:**

***"Разработка веб-приложений"***

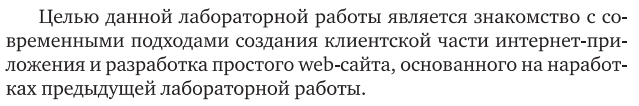
**Вариант 8**

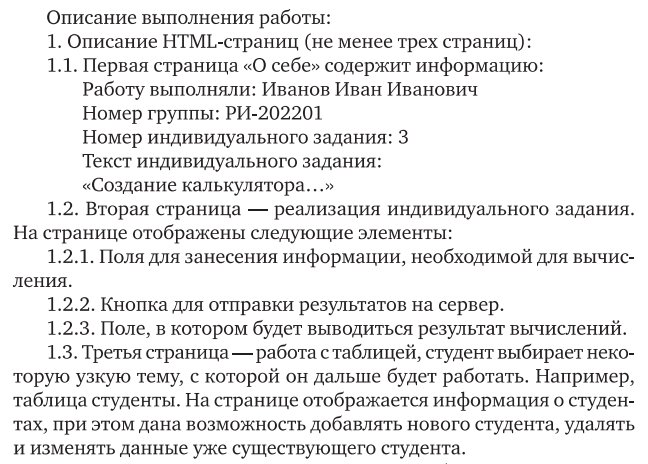
|  |  |
| --- | --- |
| Выполнили: | Студент: Журович К.С.  Гр. 10701121 |
| Проверил: | Пр. Кондратенок Е.В. |

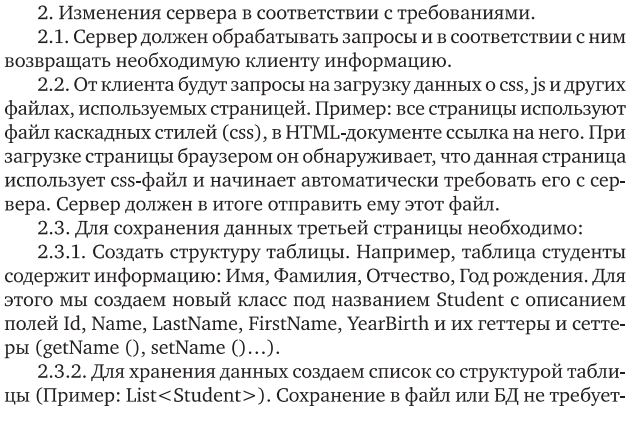
Минск – 2024

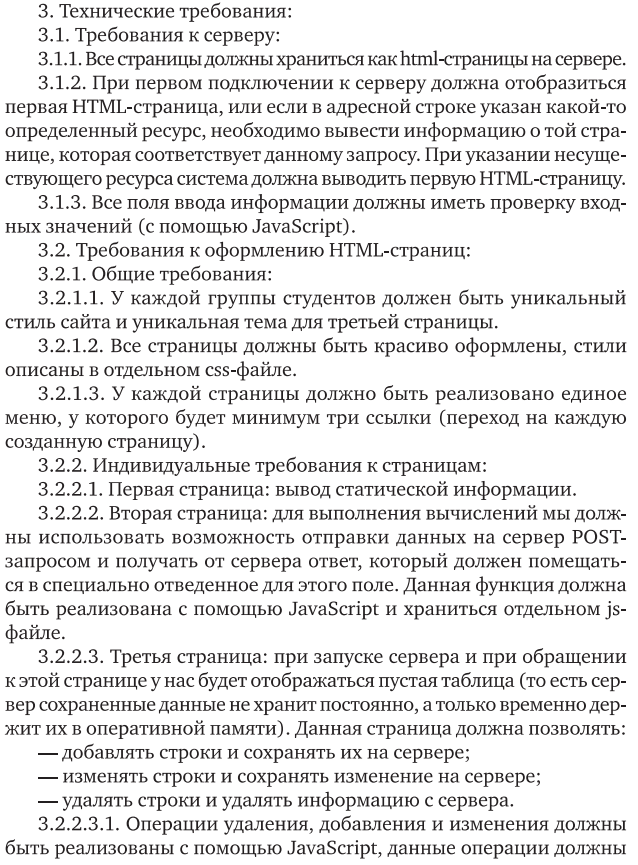
**ЛАБОРАТОРНАЯ РАБОТА 3**

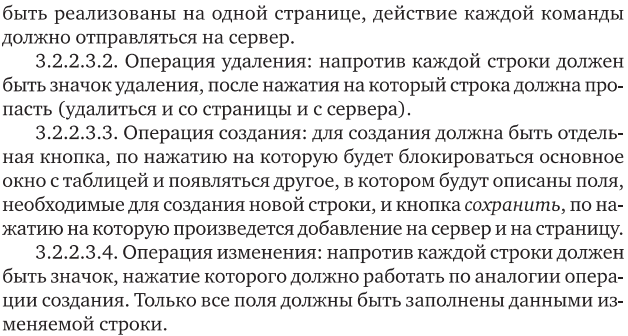
***Цель лабораторной работы:***











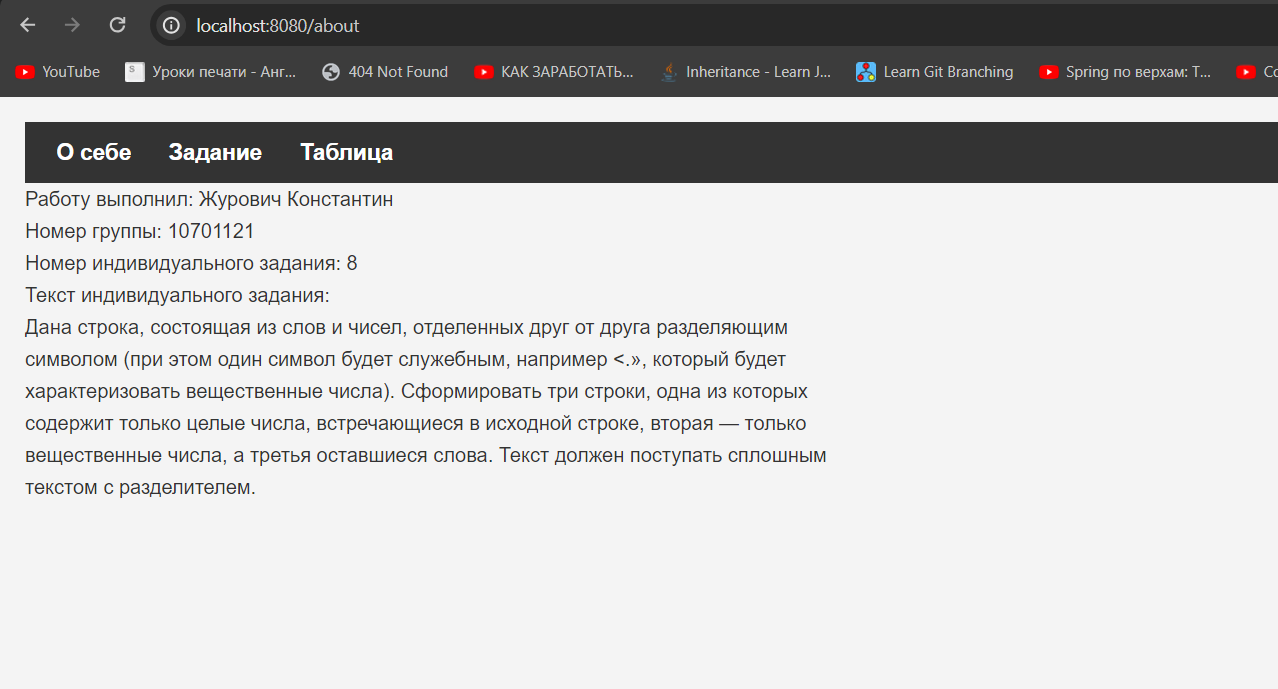


Рисунок 1 – Страница 1

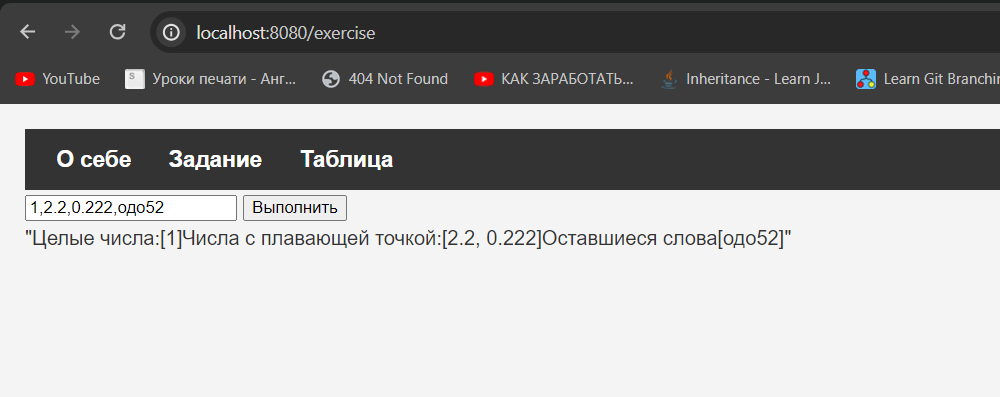


Рисунок 2 – Страница 2

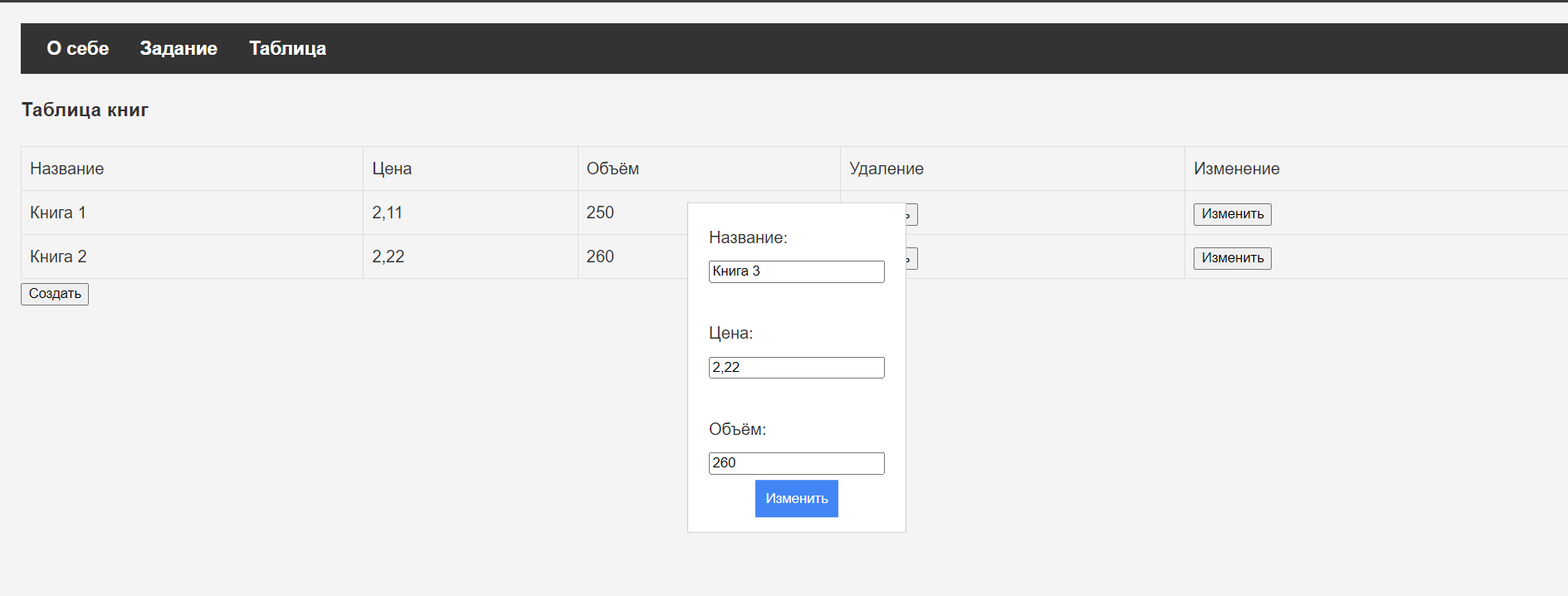


Рисунок 3 – Страница 3

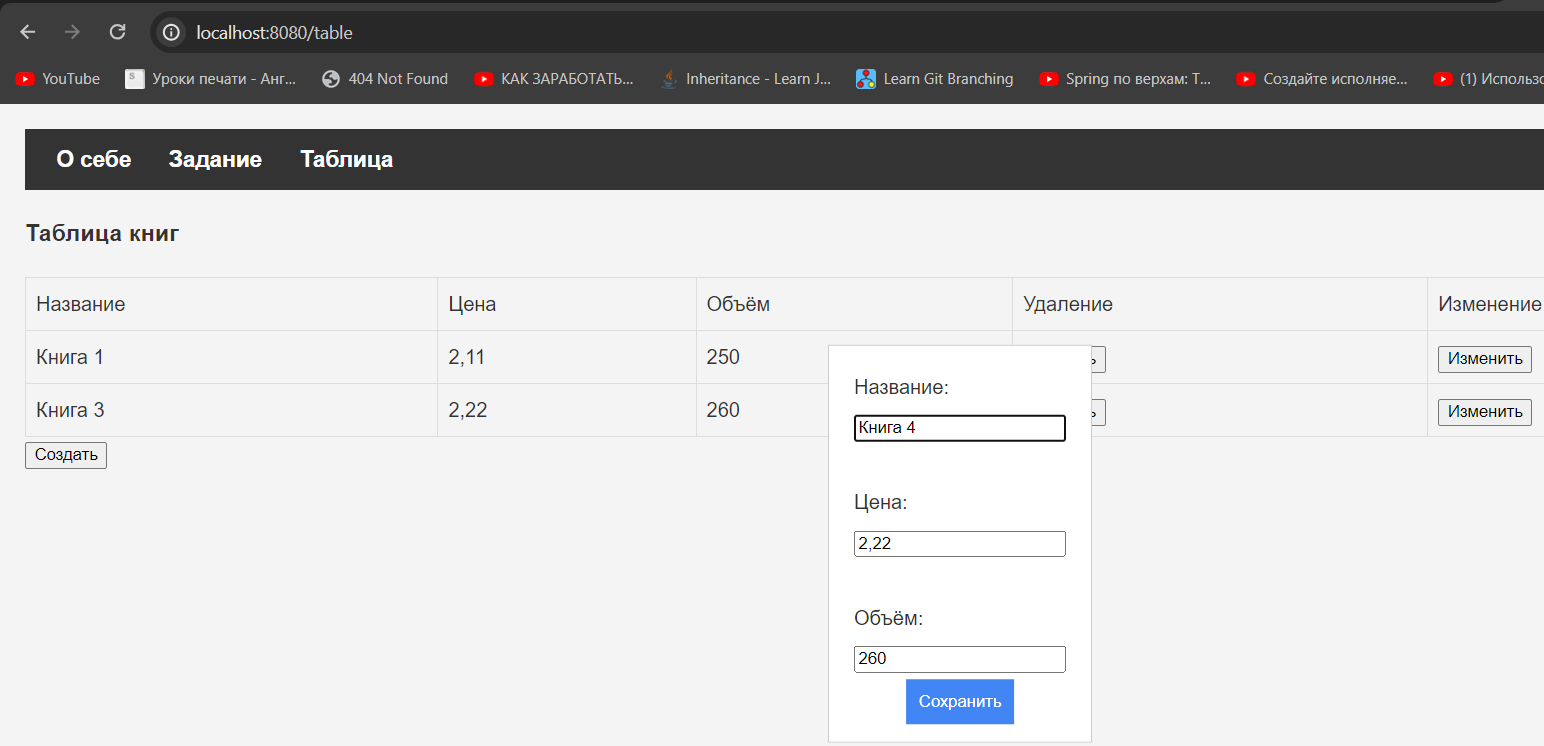


Рисунок 4 – Страница 3

**Листинг**

package org.example;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
  
import java.io.\*;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
import java.util.UUID;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class JavaApplication {  
 public static void main(String[] args) {  
 try (ServerSocket serverSocket = new ServerSocket(8080)){  
 List<Book> books = new ArrayList<>();  
 while (true) {  
 Socket socket = serverSocket.accept();  
 System.*out*.println("Client connected");  
 new Thread(() -> *handleClient*(socket,books)).start();  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 private static void handleClient(Socket clientSocket,List<Book> books) {  
 try (  
 BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));  
 OutputStream outputStream = clientSocket.getOutputStream()  
 ) {  
 // Чтение HTTP-запроса  
 StringBuilder request = new StringBuilder();  
 String line;  
 while ((line = reader.readLine()) != null && !line.isEmpty()) {  
 request.append(line).append("\r\n");  
 }  
  
 // Поиск соответствия  
 if (request.toString().startsWith("GET")) {  
  
 // Регулярное выражение для поиска строки после "GET /" и перед " HTTP/1.1"  
 Pattern pattern = Pattern.*compile*("GET /(.+?) HTTP/1\\.1");  
 Matcher matcher = pattern.matcher(request.toString());  
 String extractedString = new String("");  
 if (matcher.find()) {  
 // Извлечение найденной строки  
 extractedString = matcher.group(1);  
 }  
 String htmlContent = "";  
 String contentType = "";  
 if(Objects.*equals*(extractedString, "exercise")){  
 htmlContent = *loadFile*("src/main/resources/exercise.html");  
 contentType="text/html";  
 }else if(Objects.*equals*(extractedString,"table")){  
 htmlContent = *loadFile*("src/main/resources/table.html");  
 contentType="text/html";  
 }else if(Objects.*equals*(extractedString,"table/info")){  
 htmlContent = *convertListToJson*(books);  
 contentType="application/json";  
 } else if (Objects.*equals*(extractedString, "js/exercise.js")) {  
 // Загрузка и отправка содержимого JavaScript файла  
 htmlContent = *loadFile*("src/main/resources/js/exercise.js");  
 contentType="application/javascript";  
 } else if (Objects.*equals*(extractedString, "js/table.js")) {  
 htmlContent = *loadFile*("src/main/resources/js/table.js");  
 contentType="application/javascript";  
 } else if (Objects.*equals*(extractedString, "css/main.css")) {  
 // Загрузка и отправка содержимого JavaScript файла  
 htmlContent = *loadFile*("src/main/resources/css/main.css");  
 contentType="text/css";  
 } else if (Objects.*equals*(extractedString, "css/exercise.js")) {  
 htmlContent = *loadFile*("src/main/resources/css/exercise.css");  
 contentType="text/css";  
 } else if (Objects.*equals*(extractedString, "css/table.css")) {  
 htmlContent = *loadFile*("src/main/resources/css/table.css");  
 contentType="text/css";  
 }else {  
 htmlContent = *loadFile*("src/main/resources/about.html");  
 contentType="text/html";  
 }  
 *sendResponseToClient*(htmlContent,outputStream,contentType);  
  
 } else if (request.toString().startsWith("POST")) {  
 StringBuilder stringBuilder = new StringBuilder();  
 while (reader.ready()) {  
 stringBuilder.append((char) reader.read());  
 }  
 String body = stringBuilder.toString();  
 // Регулярное выражение для поиска строки после "GET /" и перед " HTTP/1.1"  
 Pattern pattern = Pattern.*compile*("POST /(.+?) HTTP/1\\.1");  
 Matcher matcher = pattern.matcher(request.toString());  
 String extractedString = new String("");  
 if (matcher.find()) {  
 // Извлечение найденной строки  
 extractedString = matcher.group(1);  
 }  
  
 String responseData = "";  
 if(Objects.*equals*(extractedString, "exercise")){  
 Result result = new Result(body.replace("\"", ""));  
 responseData="\"" +result.getResult()+ "\"";  
 }else if(Objects.*equals*(extractedString,"table/add")){  
 Book book = new ObjectMapper().readValue(body,Book.class);  
 book.setId(UUID.*randomUUID*().toString());  
 books.add(book);  
 responseData = *convertListToJson*(books);  
 }else if(Objects.*equals*(extractedString,"table/delete")){  
 Book book = new ObjectMapper().readValue(body,Book.class);  
 for (Book book1:books){  
 if(Objects.*equals*(book1.getId(), book.getId()))  
 {  
 books.remove(book1);  
 break;  
 }  
 }  
 responseData = *convertListToJson*(books);  
 }else if(Objects.*equals*(extractedString,"table/change")){  
 Book book = new ObjectMapper().readValue(body,Book.class);  
 for (Book book1:books){  
 if(Objects.*equals*(book1.getId(), book.getId()))  
 {  
 book1.setName(book.getName());  
 book1.setPrice(book.getPrice());  
 book1.setVolume(book.getVolume());  
 break;  
 }  
 }  
 responseData = *convertListToJson*(books);  
 }  
  
 *sendResponseToClient*(responseData,outputStream,"application/json");  
  
 } else {  
 *sendResponseToClient*("",outputStream,"application/json");  
 }  
 clientSocket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private static void sendResponseToClient(String htmlContent,OutputStream outputStream,String contentType) {  
 try {  
 int contentLength = 0;  
 contentLength = htmlContent.getBytes("UTF-8").length;  
  
 String response = "HTTP/1.1 200 OK\r\nContent-Type: "+contentType+"; charset=UTF-8\r\nContent-Length: " +  
 contentLength + "\r\n\r\n" +  
 htmlContent;  
 // Отправка ответа клиенту  
 outputStream.write(response.getBytes());  
 outputStream.flush();  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 private static String loadFile(String filePath) {  
 try {  
 // Чтение содержимого HTML-файла  
 Path path = Paths.*get*(filePath);  
 byte[] fileBytes = Files.*readAllBytes*(path);  
 return new String(fileBytes, StandardCharsets.*UTF\_8*);  
 } catch (IOException e) {  
 e.printStackTrace();  
 return "<html><body><h1>Error loading HTML file</h1></body></html>";  
 }  
 }  
 private static String convertListToJson(List<Book> books) {  
 try {  
 ObjectMapper objectMapper = new ObjectMapper();  
 return objectMapper.writeValueAsString(books);  
 } catch (Exception e) {  
 e.printStackTrace();  
 return "";  
 }  
 }  
}

public class Result implements ResultIMPL {  
 private List<Integer> intNumbers;  
 private List<Double> doubleNumbers;  
 private List<String> words;  
  
 public Result(String inputString) {  
  
 // Разделение строки по ","  
 String[] items = inputString.split(",");  
  
 this.intNumbers = new ArrayList<>();  
 this.doubleNumbers = new ArrayList<>();  
 this.words = new ArrayList<>();  
  
 for (String item : items) {  
 try {  
 intNumbers.add(Integer.*parseInt*(item));  
 } catch (NumberFormatException eInt) {  
 try {  
 doubleNumbers.add(Double.*parseDouble*(item));  
 } catch (NumberFormatException eDouble) {  
 words.add(item);  
 }  
 }  
 }  
 }  
 @Override  
 public String getResult() {  
 return "Целые числа:" + intNumbers.toString() +  
 "Числа с плавающей точкой:" + doubleNumbers +  
 "Оставшиеся слова" + words.toString();  
 }  
}

public interface ResultIMPL {  
 String getResult();  
}

package org.example;  
  
import java.util.UUID;  
  
public class Book {  
 private String id;  
 private String name;  
 private String price;  
 private String volume;  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPrice() {  
 return price;  
 }  
  
 public void setPrice(String price) {  
 this.price = price;  
 }  
  
 public String getVolume() {  
 return volume;  
 }  
  
 public void setVolume(String volume) {  
 this.volume = volume;  
 }  
  
 public String getId() {  
 return id;  
 }  
}

**table.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Таблица</title>  
 <link rel="stylesheet" href="/css/main.css">  
 <link rel="stylesheet" href="/css/table.css">  
  
</head>  
 <body>  
 <header>  
 <ul class="nav-links">  
 <li><a href="/about">О себе</a></li>  
 <li><a href="/exercise">Задание</a></li>  
 <li><a href="/table">Таблица</a></li>  
 </ul>  
 </header>  
 <h3>Таблица книг</h3>  
 <table id="table-id">  
 <tr>  
 <td>Название</td>  
 <td>Цена</td>  
 <td>Объём</td>  
 <td>Удаление</td>  
 <td>Изменение</td>  
 </tr>  
 </table>  
 <button onclick="openModal()">Создать</button>  
  
 <!-- Модальное окно -->  
 <div id="modal" class="modal">  
 <label for="name">Название:</label>  
 <input type="text" id="name" required><br>  
  
 <label for="price">Цена:</label>  
 <input type="text" id="price" required><br>  
  
 <label for="volume">Объём:</label>  
 <input type="text" id="volume" required><br>  
  
 <button id="button-modal-save">Сохранить</button>  
 <button id="button-modal-edit">Изменить</button>  
  
 </div>  
 <script src="/js/table.js"></script>  
 </body>  
</html>

**Exercise.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Задание</title>  
 <link rel="stylesheet" href="/css/main.css">  
 <link rel="stylesheet" href="/css/exercise.css">  
  
</head>  
<body>  
<header>  
 <ul class="nav-links">  
 <li><a href="/about">О себе</a></li>  
 <li><a href="/exercise">Задание</a></li>  
 <li><a href="/table">Таблица</a></li>  
 </ul>  
</header>  
<div>  
 <input id="input-field" type="text" required/>  
 <button type="submit" onclick="sendToServer()">Выполнить</button>  
</div>  
<div id="result">  
</div>  
<script src="/js/exercise.js"></script>  
</body>  
</html>

**About.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>О себе</title>  
 <link rel="stylesheet" href="/css/main.css">  
</head>  
<body>  
<header>  
 <ul class="nav-links">  
 <li><a href="/about">О себе</a></li>  
 <li><a href="/exercise">Задание</a></li>  
 <li><a href="/table">Таблица</a></li>  
 </ul>  
</header>  
<div>  
 Работу выполнил: Журович Константин  
</div>  
<div>  
 Номер группы: 10701121  
</div>  
<div>  
 Номер индивидуального задания: 8  
</div>  
<div>  
 Текст индивидуального задания:  
 <br>  
 Дана строка, состоящая из слов и чисел, отделенных друг от друга разделяющим <br>  
 символом (при этом один символ будет служебным, например <.», который будет <br>  
 характеризовать вещественные числа). Сформировать три строки, одна из которых <br>  
 содержит только целые числа, встречающиеся в исходной строке, вторая — только<br>  
 вещественные числа, а третья оставшиеся слова. Текст должен поступать сплошным <br>  
 текстом с разделителем.  
</div>  
  
</body>  
</html>

**Exercise.css**

/\* Стили для ввода текста и кнопки \*/  
body {  
 font-family: Arial, sans-serif;  
 background-color: #f4f4f4;  
 margin: 20px;  
}  
  
div {  
 margin-bottom: 20px;  
}  
  
input {  
 padding: 10px;  
 font-size: 14px;  
}  
  
button {  
 padding: 10px;  
 font-size: 14px;  
 cursor: pointer;  
 background-color: #4285f4;  
 color: #fff;  
 border: none;  
}  
  
/\* Стили для результата \*/  
#result {  
 font-size: 16px;  
 margin-top: 20px;  
 padding: 10px;  
 border: 1px solid #ddd;  
 background-color: #fff;  
}

**Main.css**

/\* Reset стилей для обнуления стандартных браузерных стилей \*/  
body, h1, h2, h3, p, ul, li {  
 margin: 0;  
 padding: 0;  
}  
  
/\* Базовные стили \*/  
body {  
 font-family: Arial, sans-serif;  
 line-height: 1.6;  
 background-color: #f4f4f4;  
 color: #333;  
 margin: 20px;  
}  
  
header {  
 background-color: #333;  
 color: #fff;  
 padding: 10px;  
}  
  
nav {  
 display: flex;  
 justify-content: center;  
}  
  
.nav-links {  
 list-style: none;  
 display: flex;  
}  
  
.nav-links li {  
 margin: 0 15px;  
}  
  
.nav-links a {  
 text-decoration: none;  
 color: #fff;  
 font-weight: bold;  
 font-size: 18px;  
}  
  
/\* Область с информацией о студенте и задании \*/  
.info-section {  
 margin-top: 20px;  
 padding: 20px;  
 background-color: #fff;  
 border: 1px solid #ddd;  
 border-radius: 5px;  
}  
  
/\* Стили для списков и текста внутри области с информацией \*/  
.info-section div {  
 margin-bottom: 10px;  
}  
  
/\* Стили для задания \*/  
.task-section {  
 margin-top: 20px;  
 padding: 20px;  
 background-color: #fff;  
 border: 1px solid #ddd;  
 border-radius: 5px;  
}  
  
/\* Дополнительные стили по желанию \*/

**Table.css**

.modal {  
  
}  
  
h3 {  
 margin-top: 20px;  
}  
  
/\* Стили для таблицы \*/  
#table-id {  
 border-collapse: collapse;  
 width: 100%;  
 margin-top: 20px;  
}  
  
#table-id th, #table-id td {  
 border: 1px solid #ddd;  
 padding: 8px;  
 text-align: left;  
}  
  
#table-id th {  
 background-color: #333;  
 color: #fff;  
}  
  
/\* Стили для модального окна \*/  
.modal {  
 display: none;  
 position: fixed;  
 top: 50%;  
 left: 50%;  
 transform: translate(-50%, -50%);  
 padding: 20px;  
 background-color: #fff;  
 border: 1px solid #ccc;  
 z-index: 1000;  
}  
  
.modal label, .modal input, .modal button {  
 display: block;  
 margin-bottom: 10px;  
}  
  
.modal button {  
 display: none;  
 padding: 10px;  
 cursor: pointer;  
 background-color: #4285f4;  
 color: #fff;  
 border: none;  
 position: fixed;  
 top: 90%;  
 left: 50%;  
 transform: translate(-50%, -50%);  
 z-index: 1000;  
}

**Exercise.js**

function sendToServer(){  
 let xhr = new XMLHttpRequest();  
 xhr.open("POST", "http://localhost:8080/exercise", true);  
 xhr.setRequestHeader("Content-Type", "application/json");  
  
 xhr.onreadystatechange = function () {  
 if (xhr.readyState === 4 && xhr.status === 200) {  
 // Выполняем колбэк с полученными данными  
 ***document***.getElementById("result").innerText = xhr.responseText;  
 }  
 };  
 let data = ***document***.getElementById("input-field").value;  
 xhr.send(***JSON***.stringify(data));  
}

**Table.js**

sendGetToServer();  
function sendGetToServer(){  
 let xhr = new XMLHttpRequest();  
 xhr.open("GET", "http://localhost:8080/table/info", true);  
 xhr.onreadystatechange = function () {  
 if (xhr.readyState === 4 && xhr.status === 200) {  
 displayTable(***JSON***.parse(xhr.responseText));  
 }  
 };  
 xhr.send();  
}  
function sendPostToServer(url,data){  
 let xhr = new XMLHttpRequest();  
 xhr.open("POST", "http://localhost:8080"+url, true);  
 xhr.setRequestHeader("Content-Type", "application/json");  
  
 xhr.onreadystatechange = function () {  
 if (xhr.readyState === 4 && xhr.status === 200) {  
 displayTable(***JSON***.parse(xhr.responseText));  
 }  
 };  
 xhr.send(***JSON***.stringify(data));  
}  
function openModal() {  
 // Отобразить модальное окно  
 ***document***.getElementById('modal').style.display = 'block';  
 ***document***.getElementById('button-modal-save').style.display = 'block';  
 ***document***.getElementById('button-modal-save').onclick = saveData;  
}  
  
function saveData() {  
 // Создание объекта с данными  
 let data = { name:***document***.getElementById('name').value,  
 price: ***document***.getElementById('price').value,  
 volume:***document***.getElementById('volume').value };  
  
 // Отправка данных на сервер  
 sendPostToServer("/table/add", data);  
 // Скрыть модальное окно после сохранения  
 ***document***.getElementById('button-modal-save').style.display = 'none';  
  
 ***document***.getElementById('modal').style.display = 'none';  
}  
function displayTable(table) {  
 let tableElement = ***document***.getElementById('table-id');  
  
 // Очистим текущую таблицу перед добавлением новых данных  
 clearTable(tableElement);  
  
 // Перебираем каждую запись в списке и добавляем ее в таблицу  
 table.forEach(item => {  
 let newRow = tableElement.insertRow(-1);  
 let cell1 = newRow.insertCell(0);  
 let cell2 = newRow.insertCell(1);  
 let cell3 = newRow.insertCell(2);  
 let cell4 = newRow.insertCell(3);  
 let cell5 = newRow.insertCell(4);  
  
 cell1.innerHTML = item.name;  
 cell2.innerHTML = item.price;  
 cell3.innerHTML = item.volume;  
 let buttonDelete = ***document***.createElement('button');  
 buttonDelete.textContent = 'Удалить';  
 buttonDelete.addEventListener('click', function() {  
 deleteItem(item);  
 });  
 cell4.appendChild(buttonDelete);  
  
 let buttonChange = ***document***.createElement('button');  
 buttonChange.textContent = 'Изменить';  
 buttonChange.addEventListener('click', function() {  
 // Отобразить модальное окно  
 ***document***.getElementById('modal').style.display = 'block';  
 ***document***.getElementById('button-modal-edit').style.display = 'block';  
 ***document***.getElementById('button-modal-edit').onclick = function() {  
 editItem(item);  
 };  
  
 ***document***.getElementById('name').value=item.name;  
 ***document***.getElementById('price').value=item.price;  
 ***document***.getElementById('volume').value = item.volume;  
  
 });  
 cell5.appendChild(buttonChange);  
 });  
}  
  
function deleteItem(item) {  
 sendPostToServer("/table/delete",item);  
}  
  
function editItem(item) {  
  
 let data = {  
 id:item.id,  
 name:***document***.getElementById('name').value,  
 price: ***document***.getElementById('price').value,  
 volume:***document***.getElementById('volume').value };  
 // Отправка данных на сервер  
 sendPostToServer("/table/change", data);  
  
 ***document***.getElementById('button-modal-edit').style.display = 'none';  
 ***document***.getElementById('modal').style.display = 'none';  
}  
// Функция для очистки таблицы  
function clearTable(tableElement) {  
 while (tableElement.rows.length > 1) {  
 tableElement.deleteRow(1);  
 }  
}