МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Интернет-программирование»

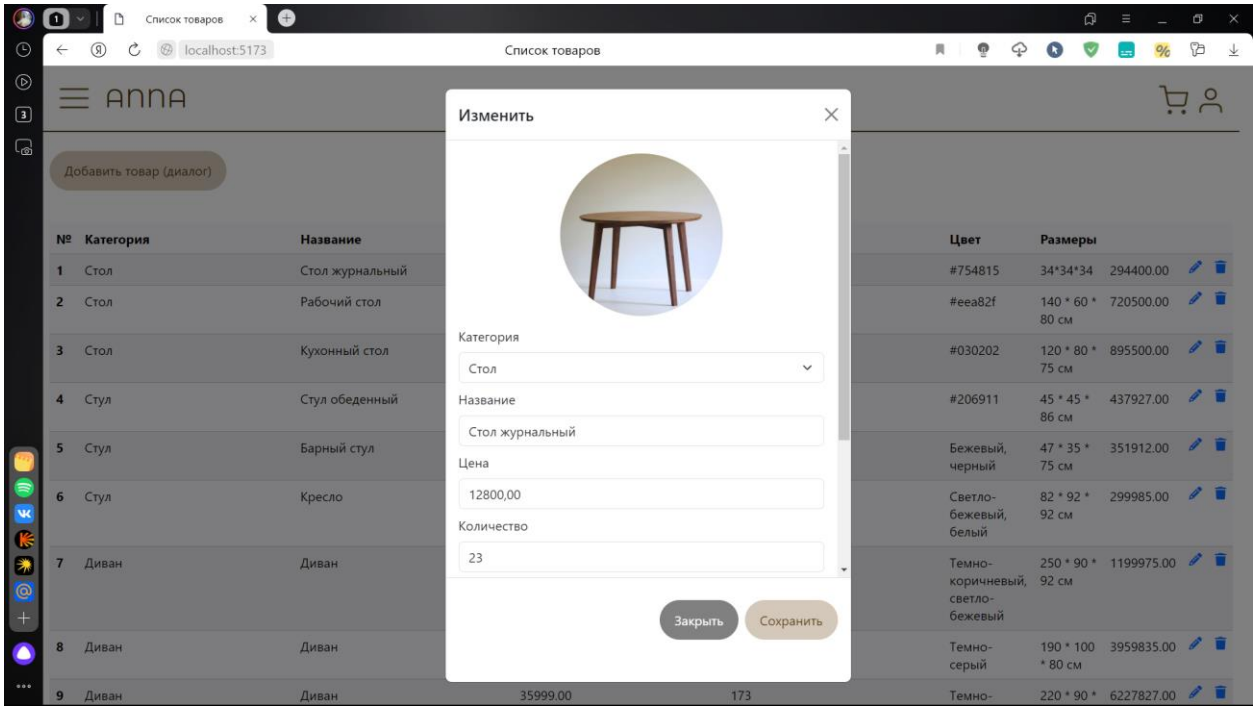
Лабораторная работа №3
AJAX, REST API

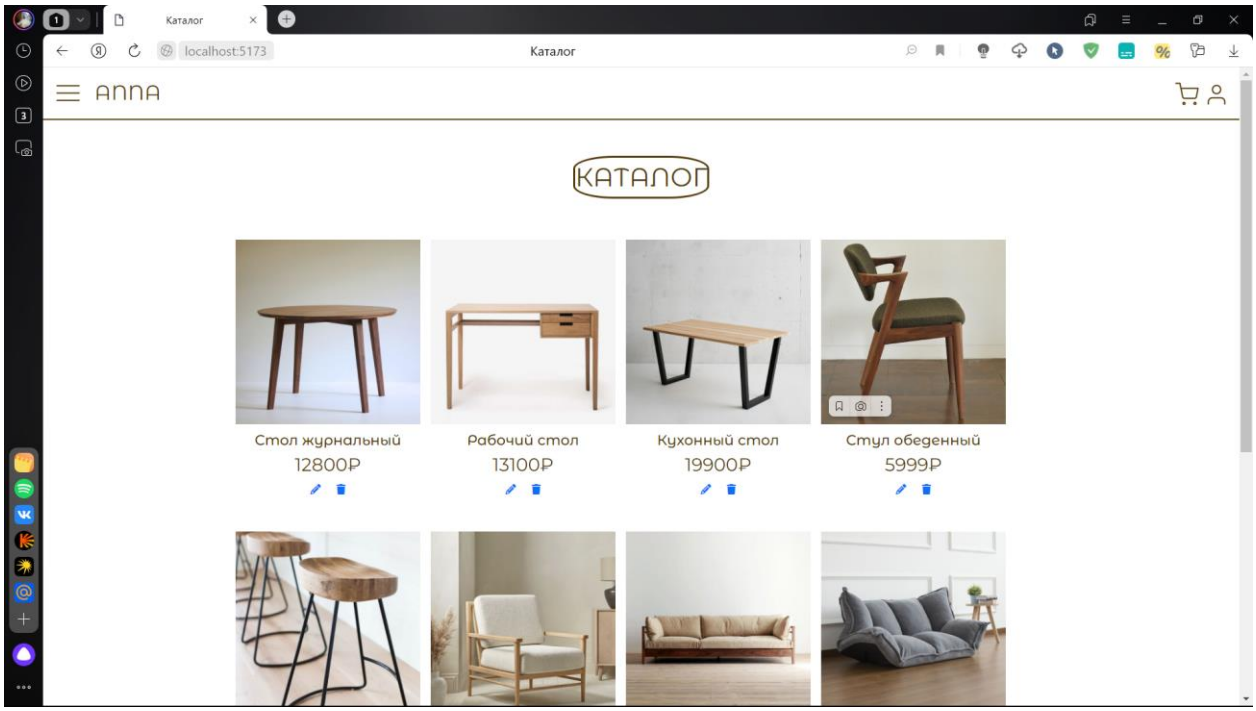
Выполнила:
студентка гр. ПИбд-22
Пучкина А.А.

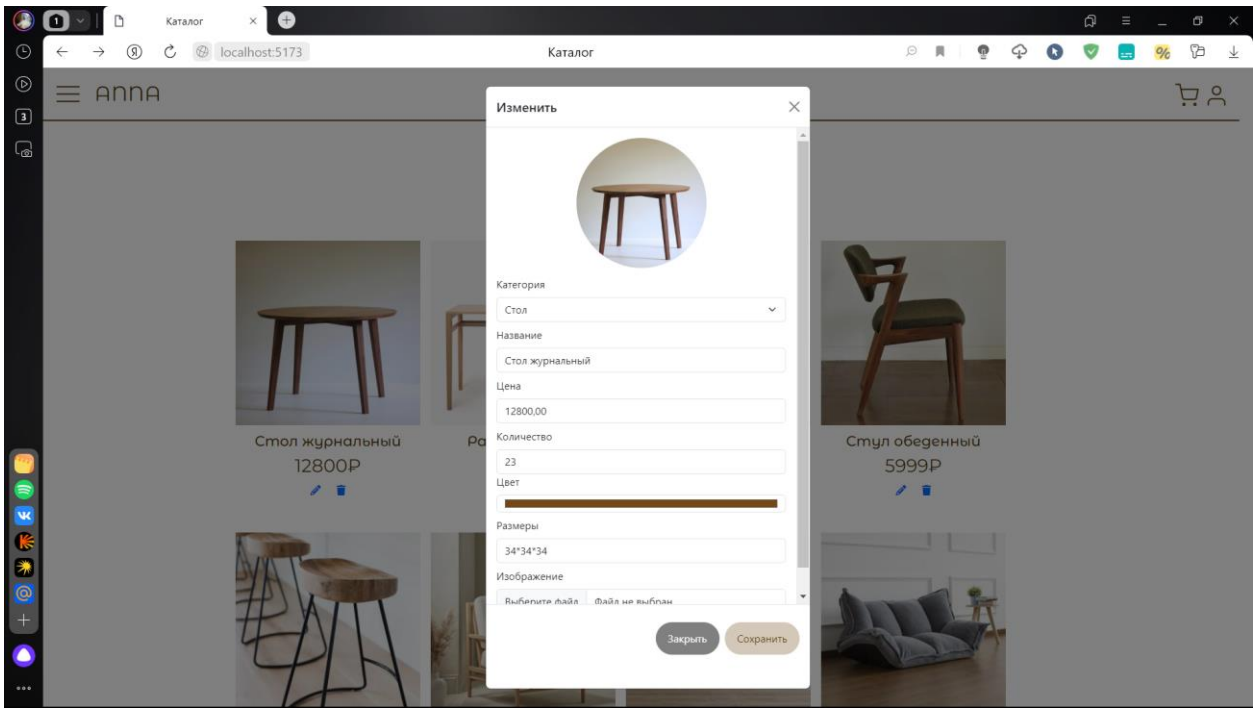
Проверил:
Филиппов А.А.

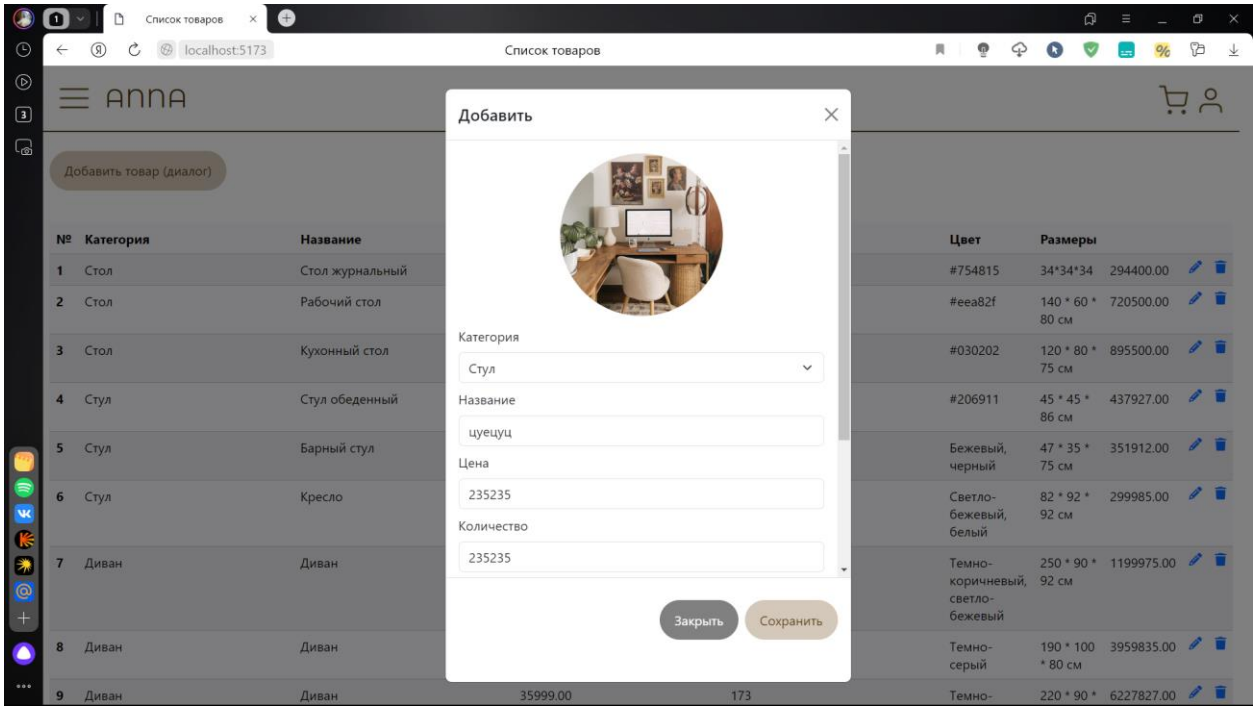
Ульяновск, 2023

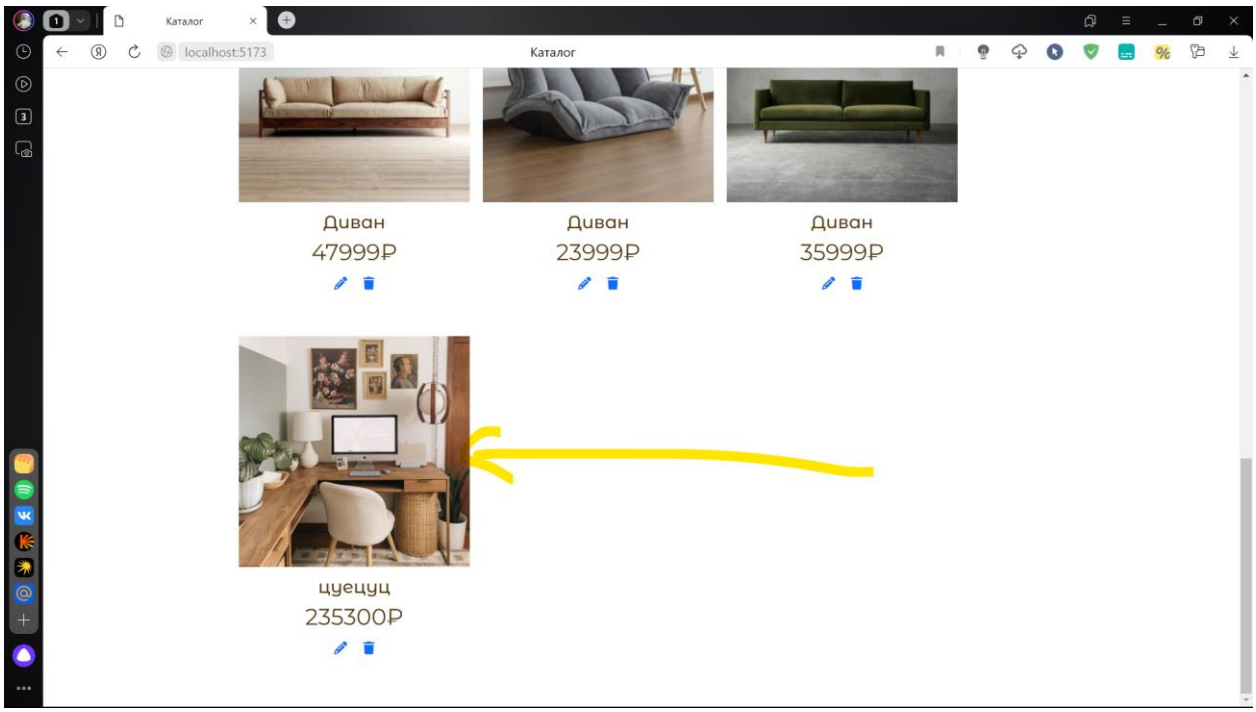
Тема сайта: Магазин дизайнерской мебели






Каталог
localhost:5173
Каталог
ANNA
Изменить
Категория
Стол
Название
Стол журнальный
Цена
12800,00
Количество
23
Цвет
Размеры
34*34*34
Изображение
Закрыть
Сохранить
Стол журнальный
12800Р
Стул обеденный
5999Р


Список товаров
localhost:5173
Список товаров
ANNA
Добавить товар (диалог)
Добавить
Категория
Стул
Название
цуецуц
Цена
235235
Количество
235235
Закрыть
Сохранить
№ Категория Название Цвет Размеры
1 Стол Стол журнальный #754815 34*34*34 294400.00
2 Стол Рабочий стол #eea82f 140 * 60 * 80 см 720500.00
3 Стол Кухонный стол #030202 120 * 80 * 75 см 895500.00
4 Стул Стул обеденный #206911 45 * 45 * 86 см 437927.00
5 Стул Барный стул Бежевый, черный 47 * 35 * 75 см 351912.00
6 Стул Кресло Светло-бежевый, белый 82 * 92 * 92 см 299985.00
7 Диван Диван Темно-коричневый, светло-бежевый 250 * 90 * 92 см 1199975.00
8 Диван Диван Темно-серый 190 * 100 * 80 см 3959835.00
9 Диван Диван 35999.00 173 Темно- 220 * 90 * 6227827.00



Java Script

Lines-modal.js

```
        // Модуль для работы с модальным окном

// импорт компонента Modal из bootstrap
import { Modal } from "bootstrap";
import { cntrls, imagePlaceholder } from "./lines-ui";

// поиск модального окна на странице
const modal = document.getElementById("items-update");
// если он найден, то создается экземпляр компонента Modal
// для программного управления модальным окном
const myModal = modal ? new Modal(modal, {}) : null;

// поиск тега с заголовком модального кона для его смены
const modalTitle = document.getElementById("items-update-title");

// обнуление значений модального окна, т. к.
// используется одно окно для всех операций
function resetValues() {
  cntrls.lineId.value = "";
  cntrls.itemsType.value = "";
  cntrls.name.value = "";
  cntrls.price.value = parseFloat(0).toFixed(2);
  cntrls.count.value = 0;
  cntrls.color.value = "";
  cntrls.size.value = "";
```

```
  cntrls.image.value = "";
  cntrls.imagePreview.src = imagePlaceholder;
}

// функция для показа модального окна
// перед показом происходит заполнение формы для редактирования
// если объект item не пуст
export function showUpdateModal(item) {
  modalTitle.innerHTML = item === null ? "Добавить" : "Изменить";
  console.info(item);

  if (item) {
    cntrls.lineId.value = item.id;
    cntrls.itemsType.value = item.itemsId;
    cntrls.name.value = item.name;
    cntrls.price.value = item.price;
    cntrls.count.value = item.count;
    cntrls.color.value = item.color;
    cntrls.size.value = item.size;
    // заполнение превью
    // Если пользователь выбрал изображение, то оно загружается
    // в тэг image с id image - preview
    // иначе устанавливается заглушка, адрес которой указан в imagePlaceholder
    cntrls.imagePreview.src = item.image ? item.image : imagePlaceholder;
  } else {
    resetValues();
  }
  myModal.show();
}

// функция для скрытия модального окна
export function hideUpdateModal() {
  resetValues();

  // удаление класса was-validated для скрытия результатов валидации
  cntrls.form.classList.remove("was-validated");

  myModal.hide();
}
```

Lines-rest-api.js

```
// модуль для работы с REST API сервера

// адрес сервера
const serverUrl = "http://localhost:8081";
```

```
// функция возвращает объект нужной структуры для отправки на сервер
function createLineObject(item, name, price, count, color, size, image) {
  return {
    itemsId: item,
    name,
    price: parseFloat(price).toFixed(2),
    count,
    color,
    size,
    sum: parseFloat(price * count).toFixed(2),
    image,
  };
}

// обращение к серверу для получения всех типов товара (get)
export async function getAllItemTypes() {
  const response = await fetch(`${serverUrl}/items`);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

// обращение к серверу для получения всех записей (get)
export async function getAllLines() {
  const response = await fetch(`${serverUrl}/lines?_expand=items`);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

// обращение к серверу для получения записи по первичному ключу (id) (get)
// id передается в качестве части пути URL get-запроса
export async function getLine(id) {
  const response = await fetch(`${serverUrl}/lines/${id}?_expand=items`);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

// обращение к серверу для создания записи (post)
// объект отправляется в теле запроса (body)
export async function createLine(item, name, price, count, color, size, image) {
  const itemObject = createLineObject(
    item,
    name,
```

```
    price,
    count,
    color,
    size,
    image
  );

  const options = {
    method: "POST",
    body: JSON.stringify(itemObject),
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
  };

  const response = await fetch(`${serverUrl}/lines`, options);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

// обращение к серверу для обновления записи по id (put)
// объект отправляется в теле запроса (body)
// id передается в качестве части пути URL get-запроса
export async function updateLine(
  id,
  item,
  name,
  price,
  count,
  color,
  size,
  image
) {
  const itemObject = createLineObject(
    item,
    name,
    price,
    count,
    color,
    size,
    image
  );

  const options = {
    method: "PUT",
```

```
    body: JSON.stringify(itemObject),
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
  };

  const response = await fetch(`${serverUrl}/lines/${id}`, options);
  if (!response.ok) {
    throw response.statusText;
  }
  await response.json();
  location.reload();
}

// обращение к серверу для удаления записи по id (delete)
// id передается в качестве части пути URL get-запроса
export async function deleteLine(id) {
  const options = {
    method: "DELETE",
  };

  const response = await fetch(`${serverUrl}/lines/${id}`, options);
  if (!response.ok) {
    throw response.statusText;
  }
  await response.json();
  location.reload();
}
```

Lines-ui.js

```
// модуль для работы с элементами управления

// объект для удобного получения элементов
// при обращении к атрибуту объекта вызывается
// нужная функция для поиска элемента
export const cntrls = {
  container: document.getElementById("my-id-for-text"),
  button: document.getElementById("items-add"),
  table: document.querySelector("#items-table tbody"),
  form: document.getElementById("items-form"),
  lineId: document.getElementById("items-line-id"),
  itemsType: document.getElementById("item"),
  name: document.getElementById("name"),
  price: document.getElementById("price"),
  count: document.getElementById("count"),
  color: document.getElementById("color"),
```

```
  size: document.getElementById("size"),
  image: document.getElementById("image"),
  imagePreview: document.getElementById("image-preview"),
};

// Дефолтное превью
export const imagePlaceholder = "https://via.placeholder.com/200";

// функция создает тег option для select
// <option value="" selected>name</option>
export function createItemsOption(name, value = "", isSelected = false) {
  const option = document.createElement("option");
  option.value = value || "";
  option.selected = isSelected;
  option.text = name;
  return option;
}

// функция создает ссылку (a) для таблицы
// содержимое тега a заполняется необходимой иконкой (icon)
// при нажатии вызывается callback
// ссылка "оборачивается" тегом td
// <td><a href="#" onclick="callback()"><i class="fa-solid icon"></i></a></td>
function createTableAnchor(icon, callback) {
  const i = document.createElement("i");
  i.classList.add("fa-solid", icon);

  const a = document.createElement("a");
  a.href = "#";
  a.appendChild(i);
  a.onclick = (event) => {
    // чтобы в URL не добавлялась решетка
    event.preventDefault();
    event.stopPropagation();
    callback();
  };

  const td = document.createElement("td");
  td.appendChild(a);
  return td;
}

// функция создает колонку таблицы с текстом value
// <td>value</td>
function createTableColumn(value) {
  const td = document.createElement("td");
  td.textContent = value;
  return td;
```

```
}

// функция создает строку таблицы
// <tr>
// <th scope="row">index + 1</th>
// <td>item.items.name</td>
// <td>parseFloat(item.price).toFixed(2))</td>
// <td>item.count</td>
// <td>parseFloat(item.sum).toFixed(2))</td>
// <td><a href="#" onclick="editCallback()"><i class="fa-solid fa-pencil"></i></a></td>
// <td><a href="#" onclick="editPageCallback()"><i class="fa-solid fa-pen-to-
square"></i></a></td>
// <td><a href="#" onclick="deleteCallback()"><i class="fa-solid fa-trash"></i></a></td>
// </tr>
export function createTableRow(
  item,
  index,
  editCallback,
  editPageCallback,
  deleteCallback
) {
  const rowNumber = document.createElement("th");
  rowNumber.scope = "row";
  rowNumber.textContent = index + 1;

  const row = document.createElement("tr");
  row.id = `line-${item.id}`;

  row.appendChild(rowNumber);
  row.appendChild(createTableColumn(item.items.name));
  row.appendChild(createTableColumn(item.name));
  row.appendChild(createTableColumn(parseFloat(item.price).toFixed(2)));
  row.appendChild(createTableColumn(item.count));
  row.appendChild(createTableColumn(item.color));
  row.appendChild(createTableColumn(item.size));
  row.appendChild(createTableColumn(parseFloat(item.sum).toFixed(2)));
  // редактировать в модальном окне
  row.appendChild(createTableAnchor("fa-pencil", editCallback));
  // удаление
  row.appendChild(createTableAnchor("fa-trash", deleteCallback));
  return row;
}

export function createTableRowOnIndex(
  item,
  index,
  editCallback,
  editPageCallback,
```

```
  deleteCallback
) {
  console.log(item);

  const img = document.createElement("img");
  img.classList.add("objects");
  img.src = item.image;
  img.alt = "object";
  img.style.height = "300px";
  img.style.display = "block";
  img.style.marginLeft = "auto";
  img.style.marginRight = "auto";

  const name = document.createElement("div");
  name.classList.add("caption");
  name.innerText = item.name;
  name.style.padding = "0px";
  name.style.fontSize = "22px";
  name.style.color = "#533908";
  name.style.fontFamily = "Montserrat Alternates";
  name.style.fontWeight = 500;

  const price = document.createElement("div");
  price.classList.add("caption_2");
  price.innerText = parseInt(item.price) + "₽";
  price.style.padding = "0";
  price.style.marginBottom = "40px";
  price.style.fontSize = "28px";
  price.style.color = "#533908";
  price.style.fontFamily = "Montserrat Alternates";
  price.style.fontWeight = 400;

  const a = document.createElement("a");
  a.href = "products/productTable1.html";
  a.style.textDecoration = "none";

  a.appendChild(img);
  a.appendChild(name);
  a.appendChild(price);

  const buttonsContainer = document.createElement("div");
  buttonsContainer.style.display = "flex";
  buttonsContainer.style.justifyContent = "center";

  const editButton = createTableAnchor("fa-pencil", editCallback);
  const deleteButton = createTableAnchor("fa-trash", deleteCallback);
  deleteButton.style.marginLeft = "1.2vw";
  buttonsContainer.appendChild(editButton);
```

```
  buttonsContainer.appendChild(deleteButton);
  buttonsContainer.style.marginTop = "-40px";
  buttonsContainer.style.marginBottom = "40px";
  a.appendChild(buttonsContainer);

  const div = document.createElement("div");
  div.classList.add("col");
  div.classList.add("clear-float");
  div.appendChild(a);

  const uniqueId = `div-${index + 1}`;
  div.id = uniqueId;

  div.style.float = "left";
  div.style.boxSizing = "border-box";
  return div;
}
```

Lines.js

```
// модуль с логикой

import { hideUpdateModal, showUpdateModal } from "./lines-modal";
import {
  createLine,
  deleteLine,
  getAllItemTypes,
  getAllLines,
  getLine,
  updateLine,
} from "./lines-rest-api";
import {
  cntrls,
  createItemsOption,
  createTableRow,
  createTableRowOnIndex,
  imagePlaceholder,
} from "./lines-ui";

async function drawItemsSelect() {
  // вызов метода REST API для получения списка типов товаров
  const data = await getAllItemTypes();
  // очистка содержимого select
  // удаляется все, что находится между тегами <select></select>
  // но не атрибуты
  if (cntrls.itemsType) {
    cntrls.itemsType.innerHTML = "";
    cntrls.itemsType.appendChild(
```

```
      createItemsOption("Выберите значение", "", true)
    );
    // цикл по результату ответа от сервера
    // используется лямбда-выражение
    // (item) => {} аналогично function(item) {}
    data.forEach((item) => {
      cntrls.itemsType.appendChild(createItemsOption(item.name, item.id));
    });
  }
}

async function drawLinesTable() {
  console.info("Try to load data");
  if (!cntrls.table) {
    return;
  }
  // вызов метода REST API для получения всех записей
  const data = await getAllLines();
  // очистка содержимого table
  // удаляется все, что находится между тегами <table></table>
  // но не атрибуты
  cntrls.table.innerHTML = "";
  // цикл по результату ответа от сервера
  // используется лямбда-выражение
  // (item, index) => {} аналогично function(item, index) {}
  data.forEach((item, index) => {
    cntrls.table.appendChild(
      createTableRow(
        item,
        index,
        // функции передаются в качестве параметра
        // это очень удобно, так как аргументы функций доступны только
        // в данном месте кода и не передаются в сервисные модули
        () => showUpdateModal(item),
        () => location.assign(`page-edit.html?id=${item.id}`),
        () => removeLine(item.id)
      )
    );
  });
}
async function drawLinesTableOnIndex() {
  console.info("Try to load data On Index");
  if (!cntrls.container) {
    return;
  }
  // вызов метода REST API для получения всех записей
  const data = await getAllLines();
  // очистка содержимого table
```

```
  // удаляется все, что находится между тегами <table></table>
  // но не атрибуты
  cntrls.container.innerHTML = "";
  // цикл по результату ответа от сервера
  // используется лямбда-выражение
  // (item, index) => {} аналогично function(item, index) {}
  data.forEach((item, index) => {
    cntrls.container.appendChild(
      createTableRowOnIndex(
        item,
        index,
        // функции передаются в качестве параметра
        // это очень удобно, так как аргументы функций доступны только
        // в данном месте кода и не передаются в сервисные модули
        () => showUpdateModal(item),
        () => location.assign(`catalog.html?id=${item.id}`),
        () => removeLine(item.id)
      )
    );
  });
}
async function addLine(item, name, price, count, color, size, image) {
  console.info("Try to add item");
  // вызов метода REST API для добавления записи
  const data = await createLine(item, name, price, count, color, size, image);
  console.info("Added");
  console.info(data);
  // загрузка и заполнение table
  drawLinesTable();
}

async function editLine(id, item, name, price, count, color, size, image) {
  console.info("Try to update item");
  // вызов метода REST API для обновления записи
  const data = await updateLine(
    id,
    item,
    name,
    price,
    count,
    color,
    size,
    image
  );
  console.info("Updated");
  console.info(data);
  // загрузка и заполнение table
  drawLinesTable();
```

```
}

async function removeLine(id) {
  if (!confirm("Do you really want to remove this item?")) {
    console.info("Canceled");
    return;
  }
  console.info("Try to remove item");
  // вызов метода REST API для удаления записи
  const data = await deleteLine(id);
  console.info(data);
  // загрузка и заполнение table
  drawLinesTable();
}

// функция для получения содержимого файла в виде base64 строки
// https://ru.wikipedia.org/wiki/Base64
async function readFile(file) {
  const reader = new FileReader();

  // создание Promise-объекта для использования функции
  // с помощью await (асинхронно) без коллбэков (callback)
  // https://learn.javascript.ru/promise
  return new Promise((resolve, reject) => {
    // 2. "Возвращаем" содержимое когда файл прочитан
    // через вызов resolve
    // Если не использовать Promise, то всю работу по взаимодействию
    // с REST API пришлось бы делать в обработчике (callback) функции
    // onloadend
    reader.onloadend = () => {
      const fileContent = reader.result;
      // Здесь могла бы быть работа с REST API
      // Чтение заканчивает выполняться здесь
      resolve(fileContent);
    };
    // 3. Возвращаем ошибку
    reader.onerror = () => {
      // Или здесь в случае ошибки
      reject(new Error("oops, something went wrong with the file reader."));
    };
    // Шаг 1. Сначала читаем файл
    // Чтение начинает выполняться здесь
    reader.readAsDataURL(file);
  });
}

// функция для обновления блока с превью выбранного изображения
async function updateImagePreview() {
```

```
  // получение выбранного файла
  // возможен выбор нескольких файлов, поэтому необходимо получить только первый
  const file = cntrls.image.files[0];
  // чтение содержимого файла в виде base64 строки
  const fileContent = await readFile(file);
  console.info("base64 ", fileContent);
  // обновление атрибута src для тега img с id image-preview
  cntrls.imagePreview.src = fileContent;
}

// Функция для обработки создания и редактирования элементов таблицы через модальное окно
// Если хотите делать через страницу, то удалите эту функцию
export function linesForm() {
  console.info("linesForm");

  // загрузка и заполнение select со списком товаров
  drawItemsSelect();
  // загрузка и заполнение table
  drawLinesTable();

  // Вызов функции обновления превью изображения при возникновении
  // события oncahnge в тэге input с id image
  cntrls.image.addEventListener("change", () => updateImagePreview());

  //  обработчик события нажатия на кнопку для показа модального окна
  cntrls.button.addEventListener("click", () => showUpdateModal(null));

  // обработчик события отправки формы
  // возникает при нажатии на кнопку (button) с типом submit
  // кнопка должна находится внутри тега form
  cntrls.form.addEventListener("submit", async (event) => {
    console.info("Form onSubmit");
    // отключение стандартного поведения формы при отправке
    // при отправке страница обновляется и JS перестает работать
    event.preventDefault();
    event.stopPropagation();
    // если форма не прошла валидацию, то ничего делать не нужно
    if (!cntrls.form.checkValidity()) {
      return;
    }

    let imageBase64 = "";
    // Получение выбранного пользователем изображения в виде base64 строки
    // Если пользователь ничего не выбрал, то не нужно сохранять в БД
    // дефолтное изображение
    if (cntrls.imagePreview.src !== imagePlaceholder) {
      // Загрузка содержимого атрибута src тэга img с id image-preview
```

```
    // Здесь выполняется HTTP запрос с типом GET
    const result = await fetch(cntrls.imagePreview.src);
    // Получение из HTTP-ответа бинарного содержимого
    const blob = await result.blob();
    // Получение base64 строки для файла
    // Здесь выполняется Promise из функции readFile
    // Promise позволяет писать линейный код для работы с асинхронными методами
    // без использования обработчиков (callback) с помощью await
    imageBase64 = await readFile(blob);
  }

  // получение id строки для редактирования
  // это значение содержится в скрытом input
  const currentId = cntrls.lineId.value;
  // если значение id не задано,
  // то необходимо выполнить добавление записи
  // иначе обновление записи
  if (!currentId) {
    await addLine(
      cntrls.itemsType.value,
      cntrls.name.value,
      cntrls.price.value,
      cntrls.count.value,
      cntrls.color.value,
      cntrls.size.value,
      imageBase64
    );
  } else {
    await editLine(
      currentId,
      cntrls.itemsType.value,
      cntrls.name.value,
      cntrls.price.value,
      cntrls.count.value,
      cntrls.color.value,
      cntrls.size.value,
      imageBase64
    );
  }

  // после выполнения добавления/обновления модальное окно скрывается
  hideUpdateModal();
 });
}
export async function linesFormOnIndex() {
 console.info("linesFormOnIndex");

 await drawItemsSelect();
```

```
await drawLinesTableOnIndex();

// обработчик события отправки формы
// возникает при нажатии на кнопку (button) с типом submit
// кнопка должна находится внутри тега form
cntrls.form.addEventListener("submit", async (event) => {
  console.info("Form onSubmit");
  // отключение стандартного поведения формы при отправке
  // при отправке страница обновляется и JS перестает работать
  event.preventDefault();
  event.stopPropagation();
  // если форма не прошла валидацию, то ничего делать не нужно
  if (!cntrls.form.checkValidity()) {
    return;
  }

  let imageBase64 = "";
  // Получение выбранного пользователем изображения в виде base64 строки
  // Если пользователь ничего не выбрал, то не нужно сохранять в БД
  // дефолтное изображение
  if (cntrls.imagePreview.src !== imagePlaceholder) {
    // Загрузка содержимого атрибута src тэга img с id image-preview
    // Здесь выполняется HTTP запрос с типом GET
    const result = await fetch(cntrls.imagePreview.src);
    // Получение из HTTP-ответа бинарного содержимого
    const blob = await result.blob();
    // Получение base64 строки для файла
    // Здесь выполняется Promise из функции readFile
    // Promise позволяет писать линейный код для работы с асинхронными методами
    // без использования обработчиков (callback) с помощью await
    imageBase64 = await readFile(blob);
  }

  // получение id строки для редактирования
  // это значение содержится в скрытом input
  const currentId = cntrls.lineId.value;
  // если значение id не задано,
  // то необходимо выполнить добавление записи
  // иначе обновление записи
  if (!currentId) {
    await addLine(
      cntrls.itemsType.value,
      cntrls.name.value,
      cntrls.price.value,
      cntrls.count.value,
      cntrls.color.value,
      cntrls.size.value,
      imageBase64
```

```
      );
    } else {
      await editLine(
        currentId,
        cntrls.itemsType.value,
        cntrls.name.value,
        cntrls.price.value,
        cntrls.count.value,
        cntrls.color.value,
        cntrls.size.value,
        imageBase64
      );
    }

    // после выполнения добавления/обновления модальное окно скрывается
    hideUpdateModal();
  });
}
// Функция для обработки создания и редактирования элементов таблицы через страницу page-edit.html
// Если хотите делать через модальное окно, то удалите эту функцию
export async function linesPageForm() {
  console.info("linesPageForm");

  // загрузка и заполнение select со списком товаров
  drawItemsSelect();

  // func1 = (id) => {} аналогично function func1(id) {}
  const goBack = () => location.assign("/page4.html");

  // Вызов функции обновления превью изображения при возникновении
  // события onchange в тэге input с id image
  cntrls.image.addEventListener("change", () => updateImagePreview());

  // получение параметров GET-запроса из URL
  // параметры перечислены после символа ? (?id=1&color=black&...)
  const urlParams = new URLSearchParams(location.search);

  // получение значения конкретного параметра (id)
  // указан только при редактировании
  const currentId = urlParams.get("id");
  // если id задан
  if (currentId) {
    try {
      // вызов метода REST API для получения записи по первичному ключу(id)
      const line = await getLine(currentId);
      // заполнение формы для редактирования
      cntrls.itemsType.value = line.itemsId;
```

```
    cntrls.name.value = line.name;
    cntrls.price.value = line.price;
    cntrls.count.value = line.count;
    cntrls.color.value = line.color;
    cntrls.size.value = line.size;
    // заполнение превью
    // Если пользователь выбрал изображение, то оно загружается
    // в тэг image с id image - preview
    // иначе устанавливается заглушка, адрес которой указан в imagePlaceholder
    cntrls.imagePreview.src = line.image ? line.image : imagePlaceholder;
  } catch {
    // в случае ошибки происходит возврат к page4
    goBack();
  }
}

// обработчик события отправки формы
// возникает при нажатии на кнопку (button) с типом submit
// кнопка должна находится внутри тега form
cntrls.form.addEventListener("submit", async (event) => {
  console.info("Form onSubmit");
  // отключение стандартного поведения формы при отправке
  // при отправке страница обновляется и JS перестает работать
  event.preventDefault();
  event.stopPropagation();
  // если форма не прошла валидацию, то ничего делать не нужно
  if (!cntrls.form.checkValidity()) {
    return;
  }

  let imageBase64 = "";
  // Получение выбранного пользователем изображения в виде base64 строки
  // Если пользователь ничего не выбрал, то не нужно сохранять в БД
  // дефолтное изображение
  if (cntrls.imagePreview.src !== imagePlaceholder) {
    // Загрузка содержимого атрибута src тэга img с id image-preview
    // Здесь выполняется HTTP запрос с типом GET
    const result = await fetch(cntrls.imagePreview.src);
    // Получение из HTTP-ответа бинарного содержимого
    const blob = await result.blob();
    // Получение base64 строки для файла
    // Здесь выполняется Promise из функции readFile
    // Promise позволяет писать линейный код для работы с асинхронными методами
    // без использования обработчиков (callback) с помощью await
    imageBase64 = await readFile(blob);
  }

  // если значение параметра запроса не задано,
```

```
    // то необходимо выполнить добавление записи
    // иначе обновление записи
    if (!currentId) {
      await addLine(
        cntrls.itemsType.value,
        cntrls.name.value,
        cntrls.price.value,
        cntrls.count.value,
        cntrls.color.value,
        cntrls.size.value,
        imageBase64
      );
    } else {
      await editLine(
        currentId,
        cntrls.itemsType.value,
        cntrls.name.value,
        cntrls.price.value,
        cntrls.count.value,
        cntrls.color.value,
        cntrls.size.value,
        imageBase64
      );
    }
    // возврат к странице page4
    goBack();
  });
}
```

Добавление диалогового окна на странице каталога для редактирования, script для вывода товаров на страницу каталога

```
<script type="module">
    import validation from "./js/validation";
    import { linesFormOnIndex } from "./js/lines";
    import { linesForm } from "./js/lines";
    document.addEventListener("DOMContentLoaded", () => {
      validation();
      linesFormOnIndex();
      linesForm();
    });
  </script>

  <catalog>
    <p class="title">КАТАЛОГ</p>
    <div class="container-catalog">
      <div id="my-id-for-text" class="container table"></div>
```

```
  </div>
</catalog>
<div
  id="items-update"
  class="modal fade"
  tabindex="-1"
  data-bs-backdrop="static"
  data-bs-keyboard="false"
>
  <div class="modal-dialog modal-dialog-scrollable">
    <form id="items-form" class="needs-validation" novalidate>
      <div class="modal-content">
        <div class="modal-header fixed-header">
          <h1 class="modal-title fs-5" id="items-update-title"></h1>
          <button
            type="button"
            class="btn-close"
            data-bs-dismiss="modal"
            aria-label="Close"
          ></button>
        </div>
        <div class="modal-body">
          <div class="text-center">
            <img
              id="image-preview"
              src="https://via.placeholder.com/200"
              class="rounded rounded-circle"
              alt="placeholder"
            />
          </div>
          <input id="items-line-id" type="number" hidden />
          <div class="mb-2">
            <label for="item" class="form-label">Категория</label>
            <select
              id="item"
              class="form-select"
              name="selected"
              required
            ></select>
          </div>
          <div class="mb-2">
            <label class="form-label" for="description">Название</label>
            <input
              id="name"
              name="name"
              class="form-control"
              type="text"
              maxlength="100"
```

```
    required
   />
  </div>
  <div class="mb-2">
   <label class="form-label" for="price">Цена</label>
   <input
    id="price"
    name="price"
    class="form-control"
    type="number"
    value="0"
    min="1000"
    step="100"
    required
   />
  </div>
  <div class="mb-2">
   <label class="form-label" for="count">Количество</label>
   <input
    id="count"
    name="count"
    class="form-control"
    type="number"
    value="0"
    min="1"
    step="1"
    required
   />
   <div class="mb-2">
    <label class="form-label" for="color">Цвет</label>
    <input
     id="color"
     name="color"
     class="form-control"
     type="color"
     maxlength="100"
     required
    />
   </div>
   <div class="mb-2">
    <label class="form-label" for="size">Размеры</label>
    <input
     id="size"
     name="size"
     class="form-control"
     type="text"
     maxlength="100"
     required
```

```
            />
          </div>
        </div>
        <div class="mb-2">
          <label class="form-label" for="image">Изображение</label>
          <input
            id="image"
            type="file"
            name="image"
            class="form-control"
            accept="image/*"
          />
        </div>
      </div>
      <div class="modal-footer fixed-footer">
        <button
          type="submit-grey"
          class="btn btn-secondary"
          data-bs-dismiss="modal"
        >
          Закрыть
        </button>
        <button type="submit" class="btn btn-success">Сохранить</button>
      </div>
    </div>
  </form>
 </div>
</div>
```

Добавление диалогового окна для редактирования таблицы с товарами, таблица с товарами

```
<html lang="ru">
 <head>
  <meta charset="utf-8" />
  <title>Список товаров</title>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <script
   type="module"
   src="./node_modules/bootstrap/dist/js/bootstrap.min.js"
  ></script>
  <link
   href="./node_modules/bootstrap/dist/css/bootstrap.min.css"
   rel="stylesheet"
  />
  <link
   href="./node_modules/@fortawesome/fontawesome-free/css/all.min.css"
   rel="stylesheet"
  />
  <link rel="stylesheet" href="./css/style.css" />
```

```
</head>

<body class="h-100 d-flex flex-column">
  <header>
    <a class="navbar-brand p-0" href="catalog.html">
      <img src="images/iconCatalog.png" alt="Каталог" width="45" />
    </a>
    <a class="nav-link active" aria-current="page" href="index.html">ANNA</a>
    <nav>
      <a class="navbar-brand p-0" href="cart.html">
        <img src="images/iconsCor.png" alt="Корзина" width="37" />
      </a>
      <a class="navbar-brand p-0" href="account.html">
        <img src="images/iconsAcc.png" alt="Аккаунт" width="38" />
      </a>
    </nav>
  </header>
  <main class="container-fluid p-2">
    <button id="items-add" type="submit" class="btn btn-success">
      Добавить товар (диалог)
    </button>

    <div>
      <table id="items-table" class="table table-light table-striped">
        <thead>
          <th scope="col">№</th>
          <th scope="col" class="w-25">Категория</th>
          <th scope="col" class="w-25">Название</th>
          <th scope="col" class="w-25">Цена</th>
          <th scope="col" class="w-25">Количество</th>
          <th scope="col" class="w-25">Цвет</th>
          <th scope="col" class="w-25">Размеры</th>
          <th scope="col"></th>
          <th scope="col"></th>
          <th scope="col"></th>
        </thead>
        <tbody></tbody>
      </table>
    </div>
  </main>
  <div
    id="items-update"
    class="modal fade"
    tabindex="-1"
    data-bs-backdrop="static"
    data-bs-keyboard="false"
  >
    <div class="modal-dialog">
```

```
<form id="items-form" class="needs-validation" novalidate>
  <div class="modal-content">
    <div class="modal-header">
      <h1 class="modal-title fs-5" id="items-update-title"></h1>
      <button
        type="button"
        class="btn-close"
        data-bs-dismiss="modal"
        aria-label="Close"
      ></button>
    </div>
    <div class="modal-body">
      <div class="text-center">
        <img
          id="image-preview"
          src="https://via.placeholder.com/200"
          class="rounded rounded-circle"
          alt="placeholder"
        />
      </div>
      <input id="items-line-id" type="number" hidden />
      <div class="mb-2">
        <label for="item" class="form-label">Категория</label>
        <select
          id="item"
          class="form-select"
          name="selected"
          required
        ></select>
      </div>
      <div class="mb-2">
        <label class="form-label" for="description">Название</label>
        <input
          id="name"
          name="name"
          class="form-control"
          type="text"
          maxlength="100"
          required
        />
      </div>
      <div class="mb-2">
        <label class="form-label" for="price">Цена</label>
        <input
          id="price"
          name="price"
          class="form-control"
          type="number"
```

```
    value="0,00"
    min="1000,00"
    step="0,5"
    required
   />
  </div>
  <div class="mb-2">
   <label class="form-label" for="count">Количество</label>
   <input
    id="count"
    name="count"
    class="form-control"
    type="number"
    value="0"
    min="1"
    step="1"
    required
   />
   <div class="mb-2">
    <label class="form-label" for="color">Цвет</label>
    <input
     id="color"
     name="color"
     class="form-control"
     type="color"
     maxlength="100"
     required
    />
   </div>
   <div class="mb-2">
    <label class="form-label" for="size">Размеры</label>
    <input
     id="size"
     name="size"
     class="form-control"
     type="text"
     maxlength="100"
     required
    />
   </div>
  </div>
  <div class="mb-2">
   <label class="form-label" for="image">Изображение</label>
   <input
    id="image"
    type="file"
    name="image"
    class="form-control"
```

```
                accept="image/*"
              />
            </div>
          </div>
          <div class="modal-footer">
            <button
              type="submit-grey"
              class="btn btn-secondary"
              data-bs-dismiss="modal"
            >
              Закрыть
            </button>
            <button type="submit" class="btn btn-success">Сохранить</button>
          </div>
        </div>
      </form>
    </div>
  </div>
  <script type="module">
    import validation from "./js/validation";
    import { linesForm } from "./js/lines";

    document.addEventListener("DOMContentLoaded", () => {
      validation();
      linesForm();
    });
  </script>
 </body>
</html>
```