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

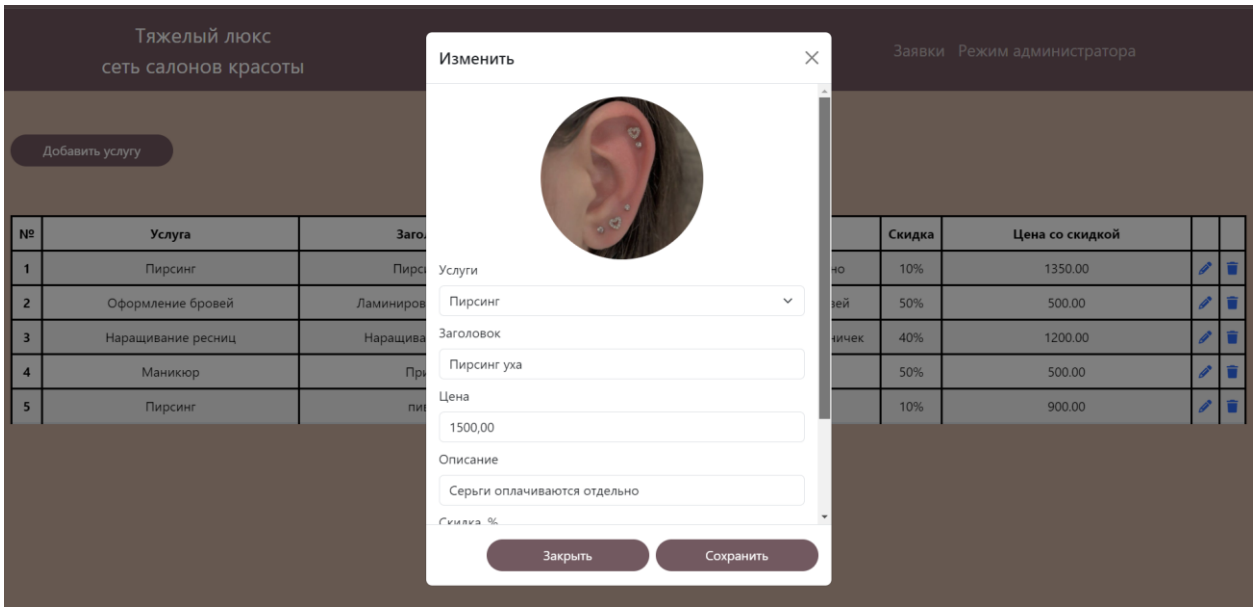
AJAX, REST API

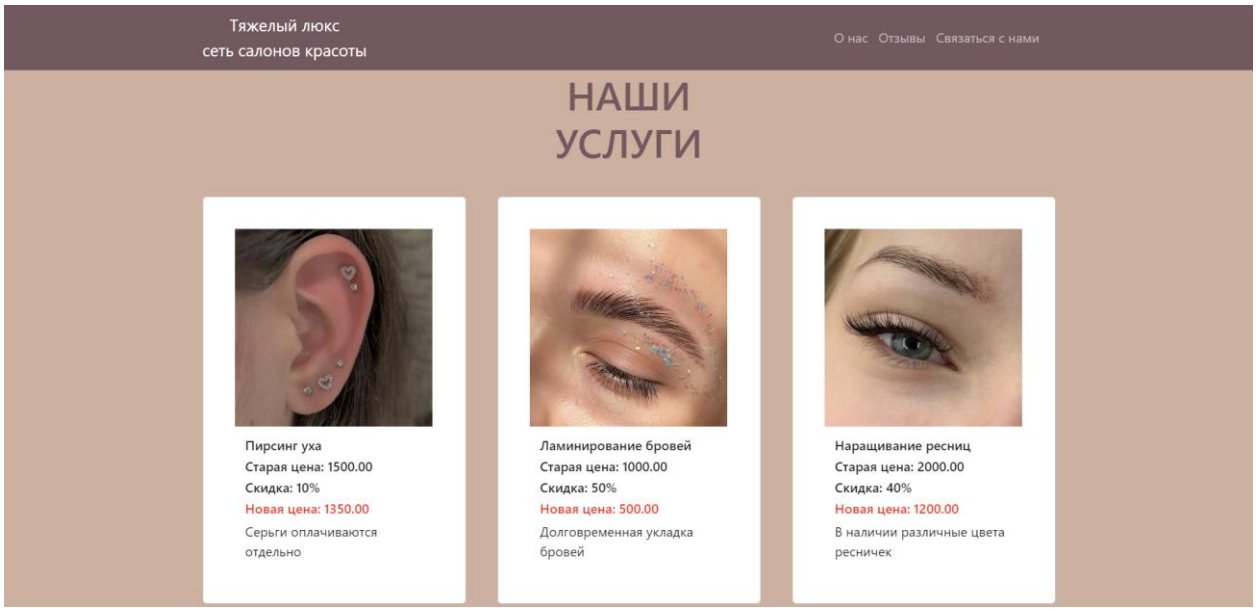
Выполнил студен группы ПИбд-22
Булатова К.Р.

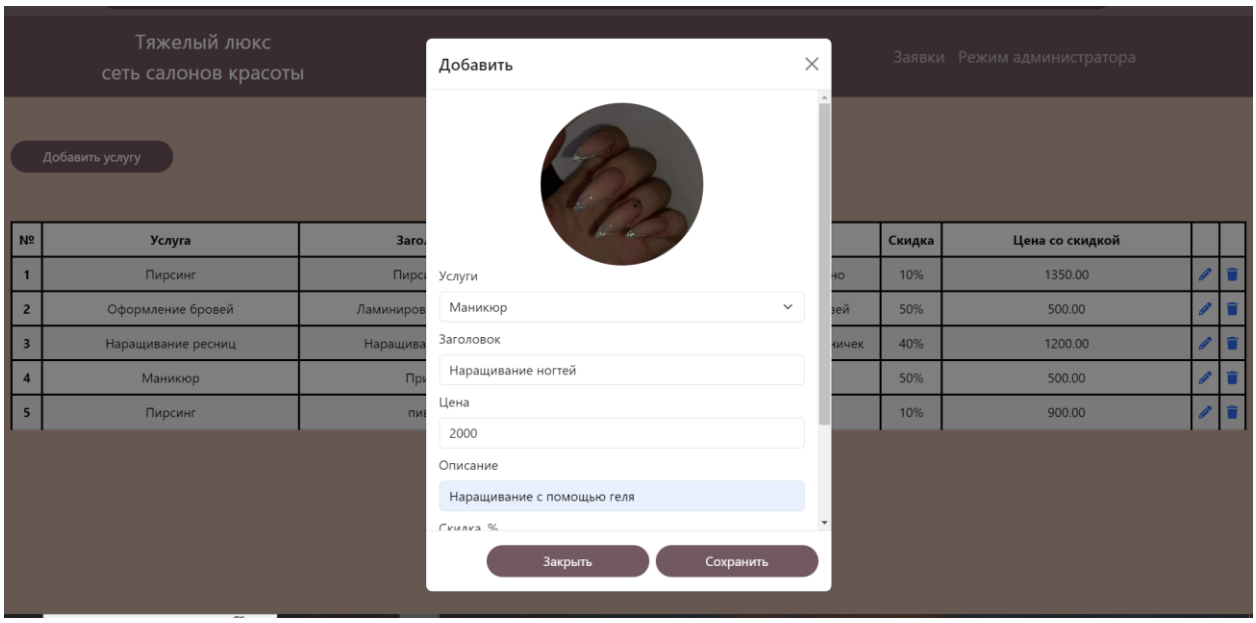
Проверил доцент кафедры
«Информационные системы»
Филиппов А.А.

Ульяновск, 2023

Тема сайта: Салон красоты







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.title.value = "";
  cntrls.price.value = parseFloat(0).toFixed(2);
  cntrls.description.value = "";
  cntrls.discountsType.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 async 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.title.value = item.title;
    cntrls.price.value = item.price;
    cntrls.description.value = item.description;
    cntrls.discountsType.value = item.discountsId;
    // 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, title, price, description, discounts, image) {
  return {
    itemsId: item,
    title,
    price: parseFloat(price).toFixed(2),
    newPrice: parseFloat(price * (1 - discounts * 0.1)).toFixed(2),
    description,
    discountsId: discounts,
    image,
  };
}

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

```
    throw response.statusText;
  }
  console.log(response);
  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&_expand=discounts`
  );
  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,
  title,
  price,
  description,
  discounts,
  newPrice,
  image
) {
  const itemObject = createLineObject(
    item,
    title,
    price,
    description,
    discounts,
```

```
    newPrice,
    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,
  title,
  price,
  description,
  discounts,
  newPrice,
  image
) {
  const itemObject = createLineObject(
    item,
```

```
    title,
    price,
    description,
    discounts,
    newPrice,
    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 = {
  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"),
  title: document.getElementById("title"),
  price: document.getElementById("price"),
  description: document.getElementById("description"),
  discountsType: document.getElementById("discounts"),
  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>
export 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, 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.title));
  row.appendChild(createTableColumn(parseFloat(item.price).toFixed(2)));
  row.appendChild(createTableColumn(item.description));
  row.appendChild(createTableColumn(item.discounts.name + "%"));
  row.appendChild(createTableColumn(parseFloat(item.newPrice).toFixed(2)));

  // редактировать в модальном окне
  row.appendChild(createTableAnchor("fa-pencil", editCallback));
  // удаление
  row.appendChild(createTableAnchor("fa-trash", deleteCallback));

  return row;
}
```

Lines.js

```
// модуль с логикой
```

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

async function drawItemsSelect() {
  console.log(111);
  // вызов метода REST API для получения списка типов товаров
  const data = await getAllItemTypes();
  const data2 = await getAllDiscountsTypes();
  // очистка содержимого select
  // удаляется все, что находится между тегами <select></select>
  // но не атрибуты
  cntrls.itemsType.innerHTML = "";
  cntrls.discountsType.innerHTML = "";
  // пустое значение
  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));
  });
  data2.forEach((item) => {
    cntrls.discountsType.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),
        () => removeLine(item.id)
      )
    );
```

```
  });
}

async function addLine(item, title, price, description, discounts, image) {
  console.info("Try to add item");
  // вызов метода REST API для добавления записи
  const data = await createLine(
    item,
    title,
    price,
    description,
    discounts,
    image
  );
  console.info("Added");
  console.info(data);
  // загрузка и заполнение table
  drawLinesTable();
}

async function editLine(id, item, title, price, description, discounts, image) {
  console.info("Try to update item");
  // вызов метода REST API для обновления записи
  const data = await updateLine(
    id,
    item,
    title,
    price,
    description,
    discounts,
    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(isCheckEvent = true) {
  console.info("linesForm");

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

  if (isCheckEvent === true) {
```

```
  cntrls.image.addEventListener("change", () => updateImagePreview());
  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()) {
    console.log("MYsubmit", cntrls);
    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
    console.log("MYsubmit", cntrls);
    if (!currentId) {
      await addLine(
        cntrls.itemsType.value,
        cntrls.title.value,
        cntrls.price.value,
        cntrls.description.value,
        cntrls.discountsType.value,
        imageBase64
      );
    } else {
      await editLine(
        currentId,
        cntrls.itemsType.value,
        cntrls.title.value,
        cntrls.price.value,
        cntrls.description.value,
        cntrls.discountsType.value,
        imageBase64
      );
    }

    // после выполнения добавления/обновления модальное окно скрывается
    hideUpdateModal();
  });
}

export async function createCards() {
```

```
console.info("createCards");
try {
  const container = document.getElementById("cards-container");

  const data = await getAllLines();
  console.log("data", data);

  data.forEach((line, index) => {
    console.log("K", line, index);
    const cardDiv = document.createElement("div");
    cardDiv.className = "col";

    const card = document.createElement("div");
    card.className = "card";

    const img = document.createElement("img");
    img.src = line.image !== "" ? line.image : "images/centerpicture.jpg";
    img.className = "rounded-img";
    img.alt = "";
    img.style.maxHeight = "500px";

    const cardBody = document.createElement("div");
    cardBody.className = "card-body";

    const title = document.createElement("h5");
    title.className = "card-title";
    title.textContent = line.title;

    const price = document.createElement("h5");
    price.className = "card-title";
    price.textContent = "Старая цена: " + line.price;

    const discounts = document.createElement("h6");
    discounts.classList.add("card-text");
    discounts.innerText = "Скидка: " + line.discounts.name + "%";
```

```
      const newPrice = document.createElement("h5");
      newPrice.className = "card-title";
      newPrice.textContent = "Новая цена: " + line.newPrice;
      newPrice.style.color = "red";

      const description = document.createElement("p");
      description.className = "card-text";
      description.innerHTML = line.description;

      cardBody.appendChild(title);
      cardBody.appendChild(price);
      cardBody.appendChild(discounts);
      cardBody.appendChild(newPrice);
      cardBody.appendChild(description);

      card.appendChild(img);
      card.appendChild(cardBody);

      cardDiv.appendChild(card);

      container.appendChild(cardDiv);
    });
  } catch (error) {
    console.error("Error fetching data:", error);
  }
}
```

Validation.js

```
// модуль используется для валидации форма на странице

function validation() {
    // поиск всех форма с классом .needs-validation
    const forms = document.querySelectorAll("form.needs-validation");

    for (let i = 0; i < forms.length; i += 1) {
```

```
      const form = forms[i];
      // для каждой формы добавляется обработчик события отправки
      form.addEventListener("submit", (event) => {
        // если форма не прошла валидацию
        // то выключить стандартное действие
        if (!form.checkValidity()) {
          event.preventDefault();
          // предотвращает распространение preventDefault
          // на другие объекты
          event.stopPropagation();
        }
        // добавляет к форме класс was-validated
        form.classList.add("was-validated");
      });
  }
}

export default validation;
```

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

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Булатова К. Сайт компании</title>
  <link
   href="node_modules/bootstrap/dist/css/bootstrap.min.css"
   rel="stylesheet"
  />
  <script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>
  <link
   href="./node_modules/@fortawesome/fontawesome-free/css/all.min.css"
   rel="stylesheet"
  />
```

```
  <link rel="stylesheet" href="css/style.css" />
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark">
    <div class="container">
      <a class="navbar-brand" href="index.html"
        >Тяжелый люкс <br />сеть салонов красоты</a
      >
      <button
        class="navbar-toggler"
        type="button"
        data-bs-toggle="collapse"
        data-bs-target="#navbarNav"
        aria-controls="navbarNav"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav justify-content-end flex-grow-1 pe-3">
          <li class="nav-item">
            <a class="nav-link" href="applications.html">Заявки</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="admin.html">Режим администратора</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
  <main class="container-fluid p-2">
    <div class="btn-group" role="group">
      <button id="items-add" class="btn btn-info">Добавить услугу</button>
    </div>
```

```
  <div>
    <table id="items-table" class="table table-striped">
      <thead>
        <th scope="col">№</th>
        <th scope="col" class="w-25">Услуга</th>
        <th scope="col" class="w-25">Заголовок</th>
        <th scope="col" class="w-20">Цена</th>
        <th scope="col" class="w-25">Описание</th>
        <th scope="col" class="w-20">Скидка</th>
        <th scope="col" class="w-25">Цена со скидкой</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 modal-dialog-scrollable">
    <form1 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
   class="kartochka"
   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">Услуги</label>
  <select
   id="item"
   class="form-select"
   name="selected"
   required
  ></select>
 </div>
 <div class="mb-2">
  <label class="form-label" for="description">Заголовок</label>
  <input
   id="title"
   name="title"
   class="form-control"
   type="text1"
   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="0.50"
    required
   />
  </div>
  <div class="mb-2">
   <label class="form-label" for="description">Описание</label>
   <input
    id="description"
    name="description"
    class="form-control"
    type="text1"
    required
   />
  </div>
  <div class="mb-2 width_add_object">
   <label for="item" class="form-label">Скидка, %</label>
   <select
    id="discounts"
    class="form-select"
    name="selected"
    required
   ></select>
  </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="button"
            class="btn btn-secondary"
            data-bs-dismiss="modal"
          >
            Закрыть
          </button>
          <button type="submit" class="btn btn-primary">Сохранить</button>
        </div>
      </div>
    </form1>
  </div>
</div>
<script type="module">
  import validation from "./js/validation";
  import { linesForm } from "./js/lines";

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

Script для вывода товаров на страницу каталога

```
<script type="module">
  import { createCards } from "./js/lines";
  import validation from "./js/validation";
  import { linesForm } from "./js/lines";
```

```
    document.addEventListener("DOMContentLoaded", () => {
      validation();
      // linesForm(false);
      createCards();
    });
  </script>
<div class="about text-center my-20 mx-auto">
      <h2>
        НАШИ <br />
        УСЛУГИ
      </h2>
    </div>
    <div class="container my-5">
      <div
        class="row row-cols-1 row-cols-md-2 row-cols-lg-3 g-5"
        id="cards-container"
      ></div>
    </div>
    <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"
      style="width: 200px"
      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">Название</label>
    <select
      id="item"
      class="form-select"
      name="selected"
      required
    ></select>
  </div>
  <div class="mb-2">
    <label class="form-label" for="description">Заголовок</label>
    <input
      id="title"
      name="title"
      class="form-control"
      type="text1"
      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="0.50"
  required
 />
</div>
<div class="mb-2">
 <label class="form-label" for="description">Описание</label>
 <input
  id="description"
  name="description"
  class="form-control"
  type="text1"
  required
 />
</div>
<div class="mb-2 width_add_object">
 <label for="item" class="form-label">Скидка</label>
 <select
  id="discounts"
  class="form-select"
  name="selected"
  required
 ></select>
</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="button"
          class="btn btn-secondary"
          data-bs-dismiss="modal"
        >
          Закрыть
        </button>
        <button type="submit" class="btn btn-primary">Сохранить</button>
      </div>
    </div>
  </form>
</div>
```