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

Лабораторная работа № 3

по дисциплине «Интернет программирование»

Выполнил студент группы ПИбд-21
Разубаев С.М.

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

Ульяновск, 2023

Задание:

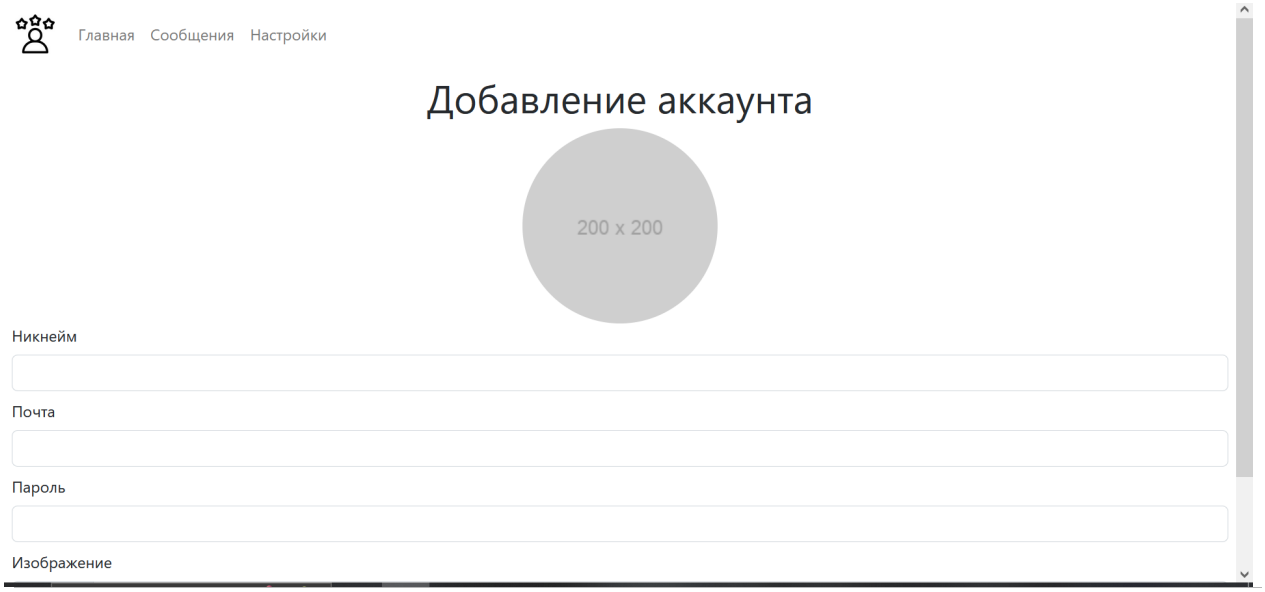
1. По результатам лабораторной работы 2 необходимо добавить CRUD-операции для страницы администратора. На данной странице администратор должен иметь возможность добавлять, редактировать, удалять и просматривать записи о некоторых объектах. Для создания и редактирования записей необходимо использовать отдельную страницу. Для просмотра сведений о выбранном объекте также необходимо использовать отдельную таблицу. В отдельных случаях для редактирования/добавления (только для этих операций) записей можно использовать диалоговые (модальные) окна. См. пример для лекции № 3.
2. В качестве сервера необходимо использовать пакет json-server. Для работы с сервером необходимо создать отдельный js-модуль с набором необходимых асинхронных функций (async/await). Запросы к серверу необходимо выполнять через Fetch API. См. пример для лекции № 3.
3. Для запуска проекта необходимо использовать задачу (скрипт) из файла package.json, которая будет запускать vite и json-server. См. пример для лекции № 3.
4. Разработку необходимо осуществлять в VS Code с установленным плагином ESLint. Более подробно см. в файле readme.md в примере для лекции № 3. Проект следует открыть в VS Code через меню «File -> Open Folder». В диалоге необходимо выбрать каталог (папку) с проектом. За основу лучше взять пример для лекции № 3. Пример загружен в LMS в раздел лекции № 3.
5. Файлы проекта и отчет должны быть загружены в git-репозиторий на сервере http://student.git.athene.tech/

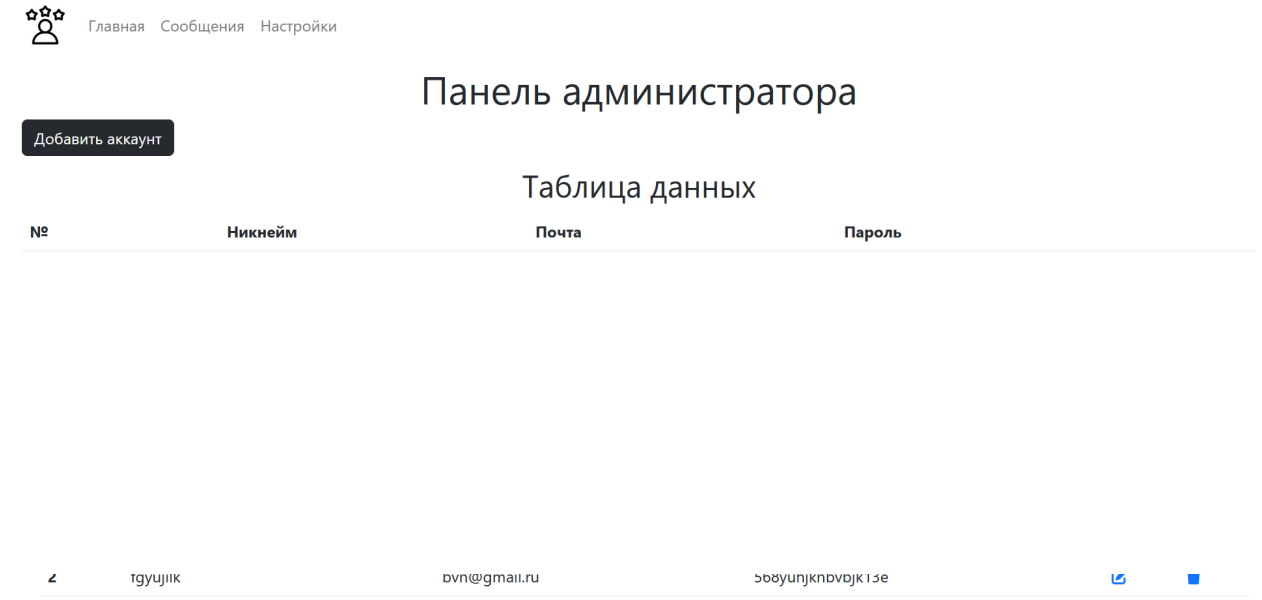
Описание решения:

Тема:

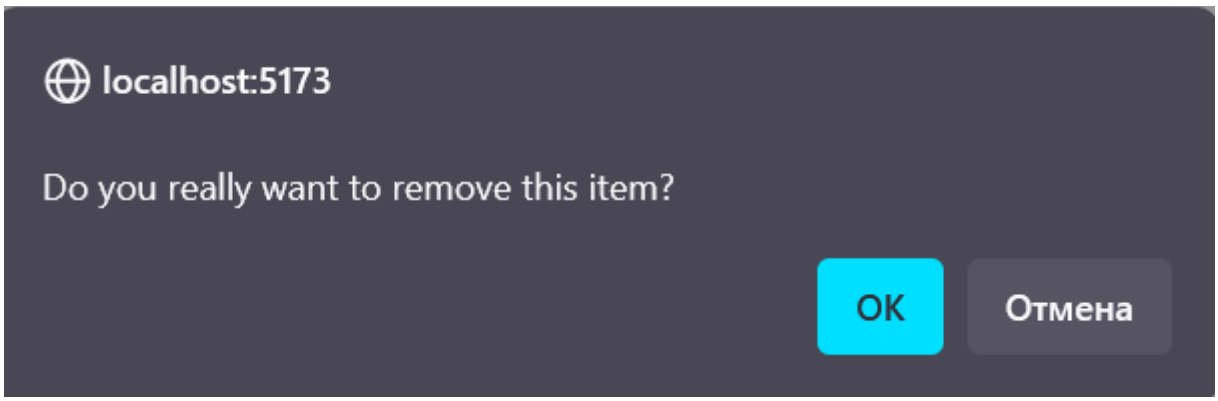
Сайт социальная сеть

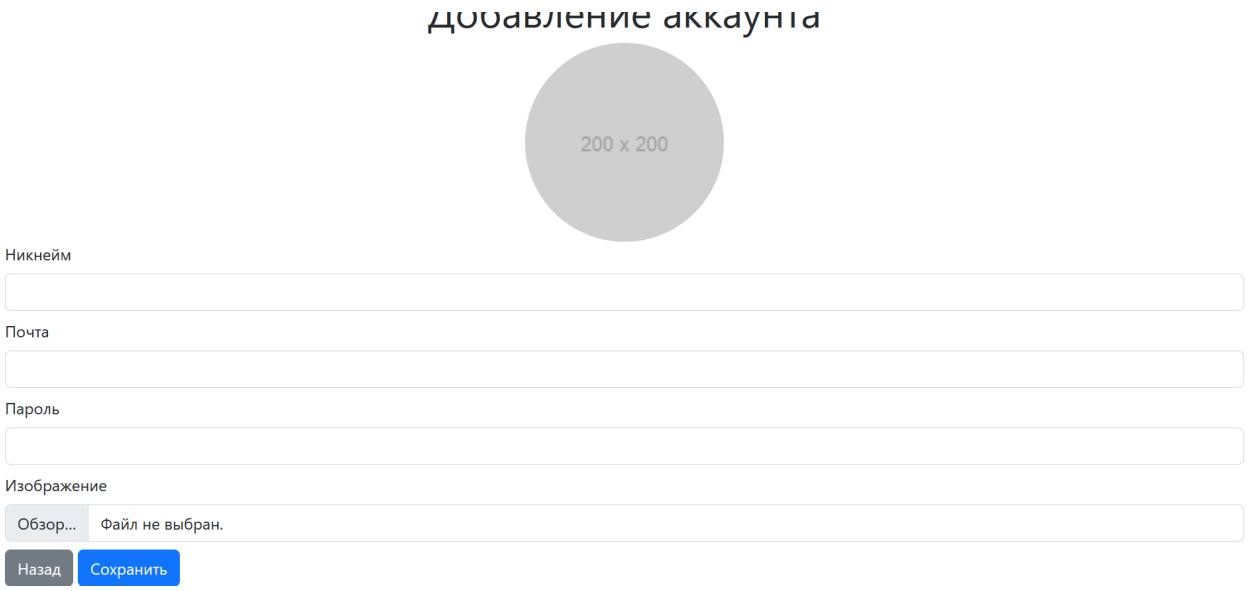
.Панель админа из 2 лабораторной

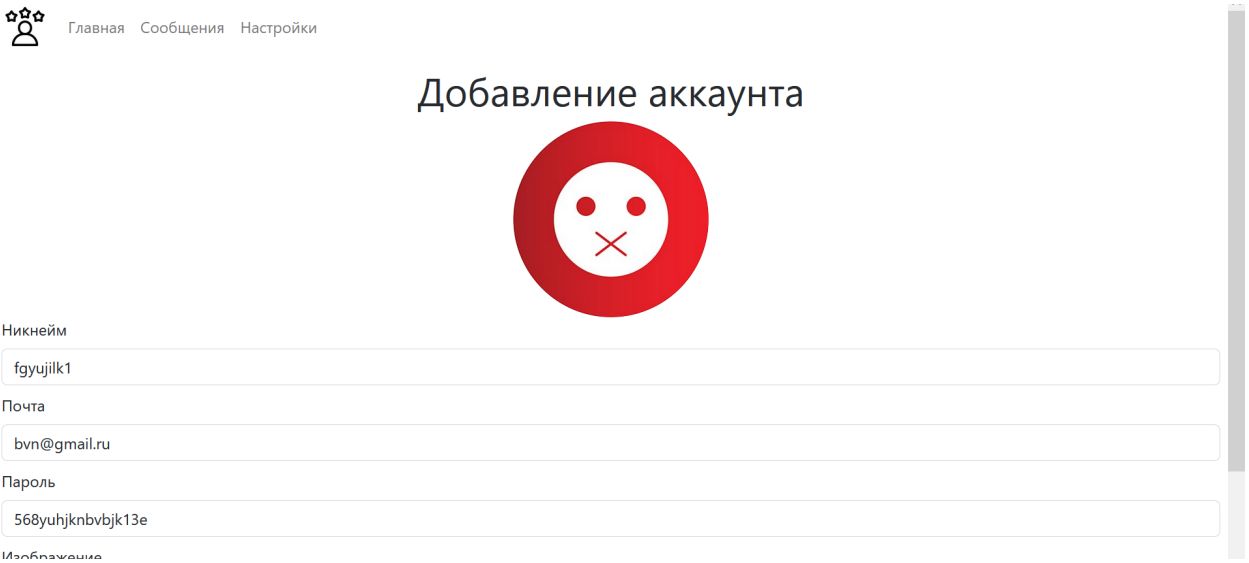




Созданы CRUD операции для панели администратора. Создание новых данных и редактирование сохраненных данных в таблице осуществляется с помощью отдельной страницы. Удаление происходит с помощью диалогового окна. Обновление данных происходит после каждой операции, а также при загрузке панели.







Код:

lines-rest-api.js

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


function createLineObject(itemName, itemEmail, itemPassword, image) {
```

```
  return {
    nickname: itemName,
    email: itemEmail,
    password: itemPassword,
    image,
  };
}

export async function getAllLines() {
  const response = await fetch(`${serverUrl}/lines`);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

export async function getLine(id) {
  const response = await fetch(`${serverUrl}/lines/${id}`);
  if (!response.ok) {
    throw response.statusText;
  }
  return response.json();
}

export async function createLine(itemName, itemEmail, itemPassword, image) {
  const itemObject = createLineObject(itemName, itemEmail, itemPassword, 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();
}

export async function updateLine(id, itemName, itemEmail, itemPassword, image) {
   const itemObject = createLineObject(itemName, itemEmail, itemPassword, 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;
   }
   return response.json();
}

export async function deleteLine(id) {
   const options = {
```

```
    method: "DELETE",
  };

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

Lines-ui.js

```
// таблица
export const cntrls = {
  table: document.querySelector("#items-table tbody"),
  form: document.getElementById("items-form"),
  lineId: document.getElementById("items-line-id"),
  nickname: document.getElementById("nickname"),
  email: document.getElementById("email"),
  password: document.getElementById("password"),
  image: document.getElementById("image"),
  imagePreview: document.getElementById("image-preview"),
};

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

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) => {
```

```
        event.preventDefault();
        event.stopPropagation();
        callback();
    };
    const td = document.createElement("td");
    td.appendChild(a);
    return td;
}
function createTableColumn(value) {
    const td = document.createElement("td");
    td.textContent = value;
    return td;
}

export function createTableRow(item, index, 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.nickname));
    row.appendChild(createTableColumn(item.email));
    row.appendChild(createTableColumn(item.password));
    row.appendChild(createTableAnchor("fa-pen-to-square", editPageCallback));
    row.appendChild(createTableAnchor("fa-trash", deleteCallback));
    return row;
}
```

Lines.js

```
// модуль с логикой
import {
```

```
   createLine, getLine, deleteLine, getAllLines, updateLine,
} from "./lines-rest-api";
import {
   cntrls, createTableRow, imagePlaceholder,
} from "./lines-ui";


export 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
            () => location.assign(`page-edit.html?id=${item.id}`),
            () => removeLine(item.id),
         ),
      );
```

```
    });
}

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

async function editLine(id, itemName, itemEmail, itemPassword, image) {
    console.info("Try to update item");
    // вызов метода REST API для обновления записи
    const data = await updateLine(id, itemName, itemEmail, itemPassword, 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 страницу page-edit.html
// Если хотите делать через модальное окно, то удалите эту функцию
// eslint-disable-next-line import/prefer-default-export
export async function linesPageForm() {
  console.info("linesPageForm");

  // загрузка и заполнение select со списком товаров
  drawLinesTable();
  // func1 = (id) => {} аналогично function func1(id) {}
  const goBack = () => location.assign("/admin-page.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.nickname.value = line.nickname;
      cntrls.email.value = line.email;
      cntrls.password.value = line.password;
      // 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.nickname.value,
        cntrls.email.value,
```

```
            cntrls.password.value,
            imageBase64,
          );
      } else {
          await editLine(
            currentId,
            cntrls.nickname.value,
            cntrls.email.value,
            cntrls.password.value,
            imageBase64,
          );
      }
      // возврат к странице page4
      goBack();
    });
}
```

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;
```

Admin-page.html

```
<!DOCTYPE html>
<html lang="ru" class="h-100">
<head>
  <meta charset="UTF-8">
  <title>Моя страница</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="style.css">
</head>
<body class="d-flex flex-column h-100">
  <header>
    <nav class="navbar navbar-expand-md">
      <div class="container-fluid">
        <a class="navbar-brand" href="/">
          <img src="Images/icon.jpg" alt="logo" width="40">
        </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="navbar-collapse collapse justify-content-start" id="navbarNav">
            <div class="navbar-nav">
              <a class="nav-link" href="home.html">Главная</a>
              <a class="nav-link" href="messages.html">Сообщения</a>
              <a class="nav-link" href="settings.html">Настройки</a>
            </div>
          </div>
        </div>
      </nav>
    </header>
<main class="container-fluid p-2">
    <h1 class=" text-center font-weight-bold">Панель администратора</h1>
    <div class="btn-group" role="group">
      <a class="btn btn-dark" href="page-edit.html">Добавить аккаунт(страница)</a>
    </div>
    <div>
      <h2 class=" text-center font-weight-bold" style="padding-top: 10px;">Таблица данных</h2>
      <table id="items-table" class="table table-striped">
        <thead>
          <th scope="col">№</th>
          <th scope="col" class="w-25">Никнейм</th>
          <th scope="col" class="w-25">Почта</th>
          <th scope="col" class="w-10">Пароль</th>
          <th scope="col"></th>
            <th scope="col"></th>
            <th scope="col"></th>
        </thead>
        <tbody></tbody>
```

```
      </table>
   </div>
</main>
 <script type="module">
      import validation from "./js/validation";
      import {  drawLinesTable as drawTable } from "./js/lines";

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

Page-edit.html

```
<html lang="ru">

<head>
   <meta charset="utf-8">
   <title>Моя страница</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="style.css">
</head>

<body class="h-100 d-flex flex-column">
   <header>
      <nav class="navbar navbar-expand-md">
         <div class="container-fluid">
```

```
          <a class="navbar-brand" href="/">
            <img src="Images/icon.jpg" alt="logo" width="40">
          </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="navbar-collapse collapse justify-content-start" id="navbarNav">
            <div class="navbar-nav">
              <a class="nav-link" href="home.html">Главная</a>
              <a class="nav-link" href="messages.html">Сообщения</a>
              <a class="nav-link" href="settings.html">Настройки</a>
            </div>
          </div>
        </div>
      </nav>
    </header>
    <main class="container-fluid p-2">
      <h1 class=" text-center font-weight-bold">Добавление аккаунта</h1>
      <div class="text-center">
        <img id="image-preview" src="https://via.placeholder.com/200" class="rounded rounded-
circle"
          alt="placeholder">
      </div>
      <form id="items-form" class="needs-validation" novalidate>
        <div class="mb-2">
          <label for="nickname" class="form-label">Никнейм</label>
          <input id="nickname" name="nickname" class="form-control" type="text"
            required>
        </div>
        <div class="mb-2">
```

```
            <label class="form-label" for="email">Почта</label>
            <input id="email" name="email" class="form-control" type="email"
                required>
        </div>
        <div class="mb-2">
            <label class="form-label" for="password">Пароль</label>
            <div class="d-flex flex-row input-group">
            <input id="password" name="password" class="form-control" type="password" required>
            <a href ="#" class =" input-group-text password-control" onclick="return
showPassword(this)"></a>
            <script>
              function showPassword(target) {
              const input = document.getElementById("password");
             if (input.getAttribute("type") === "password") {
                target.classList.add("view");
             input.setAttribute("type", "text");
             } else {
            target.classList.remove("view");
            input.setAttribute("type", "password");
             }
             return false;
             }
            </script>
        </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>
        <a href="admin-page.html" class="btn btn-secondary">Назад</a>
        <button type="submit" class="btn btn-primary">Сохранить</button>
    </form>
```

```
  </main>


  <script type="module">
    import validation from "./js/validation";
    import { linesPageForm, showPassword } from "./js/lines"


    document.addEventListener('DOMContentLoaded', () => {
      validation();
      linesPageForm();
    });
  </script>
</body>



</html>
```