

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Таначев Егор

Группа К33412

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Привязать проект из Лабораторной работы №1 к внешнему API средствами fetch/axios/xhr.

Ход работы

В этой лабораторной работе мы будем использовать json-server и json-server для авторизации, которые предварительно установим с помощью команд *npm install -g json-server* и *npm install -D json-server json-server-auth*. Теперь при запуске у нас будет крутиться сервер с базой данных *data.json* на локальном хосте с портом 3000.

Наша база данных имеет 3 категории: users (данные о пользователях), favorites (товары, которые пользователь добавил в избранные) и products (все товары сайта). Структура файла представлена на Рисунке 1.

```
1  [
2    "users": [
3      {
4        "email": "e.tanachev@inbox.ru",
5        "password": "$2a$10$JIMKSMK.xq4PksHmu0a3UeEYzLX0F6AK",
6        "name": "Егор Таначев",
7        "phone": "8998120077",
8        "id": 1
9      },
10      {
11        "email": "e.tanachev@inbox.ru2",
12        "password": "$2a$10$JIMKSMK.xq4PksHmu0a3UeEYzLX0F6AK",
13        "name": "Егор Таначев",
14        "phone": "8998120077",
15        "id": 2
16      }
17    ],
18    "favorites": [
19      {
20        "cardId": "7",
21        "userId": 1,
22        "image": "assets/img/cards/card7.jpg",
23        "title": "Пиджак черного цвета",
24        "price": "12.000 ₽",
25        "id": 8
26      }
27    ]
28  ]
```

Рисунок 1 – Файл *data.json*

Теперь реализуем авторизацию и регистрацию с помощью скриптом, которые будут отправлять POST-запросы на json сервер с url: */login* и */signup*, как показано на Рисунках 2-3.

При отправке сервер нам возвращает ответ, в котором будет содержаться accessToken для отправки следующих запросов. Этот токен и информацию о пользователе мы сохраняем в localStorage для того, чтобы не нужно было каждый раз авторизовываться, а также для вывода информации в личном кабинете. Данные о пароле кстати шифруются в data.json с помощью хэширования, поэтому безопасность системы повышается.

```
62  async function login(event) {
63    event.preventDefault();
64
65    const inputs = Array.from(event.target.querySelectorAll('input'));
66
67    const loginData = {};
68
69    for (const input of inputs) {
70      loginData[input.name] = input.value;
71    }
72
73    const response = await fetch('http://localhost:3000/login', {
74      method: 'POST',
75      body: JSON.stringify(loginData),
76      headers: {
77        'Content-Type': 'application/json'
78      }
79    );
80
81    const responseJson = await response.json();
82
83    if (response.status === 200) {
84      // Авторизация успешна
85      const { accessToken, user } = responseJson;
86
87      localStorage.accessToken = accessToken;
88      localStorage.user = JSON.stringify(user);
89
90      window.location.href = '../../../../../account.html';
91    } else {
92      // Ошибка авторизации
93      alert('Ошибка авторизации. Пожалуйста, проверьте введенные данные.');
94    }
95 }
```

Рисунок 2 – Авторизация пользователя

```
98 // РЕГИСТРАЦИЯ
99
100 async function registration(event) {
101     event.preventDefault()
102
103     const inputs = Array.from(event.target.querySelectorAll('input'))
104
105     const registerData = {}
106
107     for (const input of inputs) {
108         registerData[input.name] = input.value
109     }
110
111     const response = await fetch('http://localhost:3000/signup', {
112         method: 'POST',
113         body: JSON.stringify(registerData),
114         headers: {
115             'Content-Type': 'application/json'
116         }
117     })
118
119     const responseJson = await response.json()
120
121     if (response.status === 201) {
122         // Регистрация успешна
123         const { accessToken, user } = responseJson;
124
125         localStorage.accessToken = accessToken;
126         localStorage.user = JSON.stringify(user);
127
128         window.location.href = "../../account.html"
129     }
130 }
```

Рисунок 3 – Регистрация пользователя

Скриншот личного кабинета, а также скрипт для заполнения данных представлены на Рисунках 4-5.

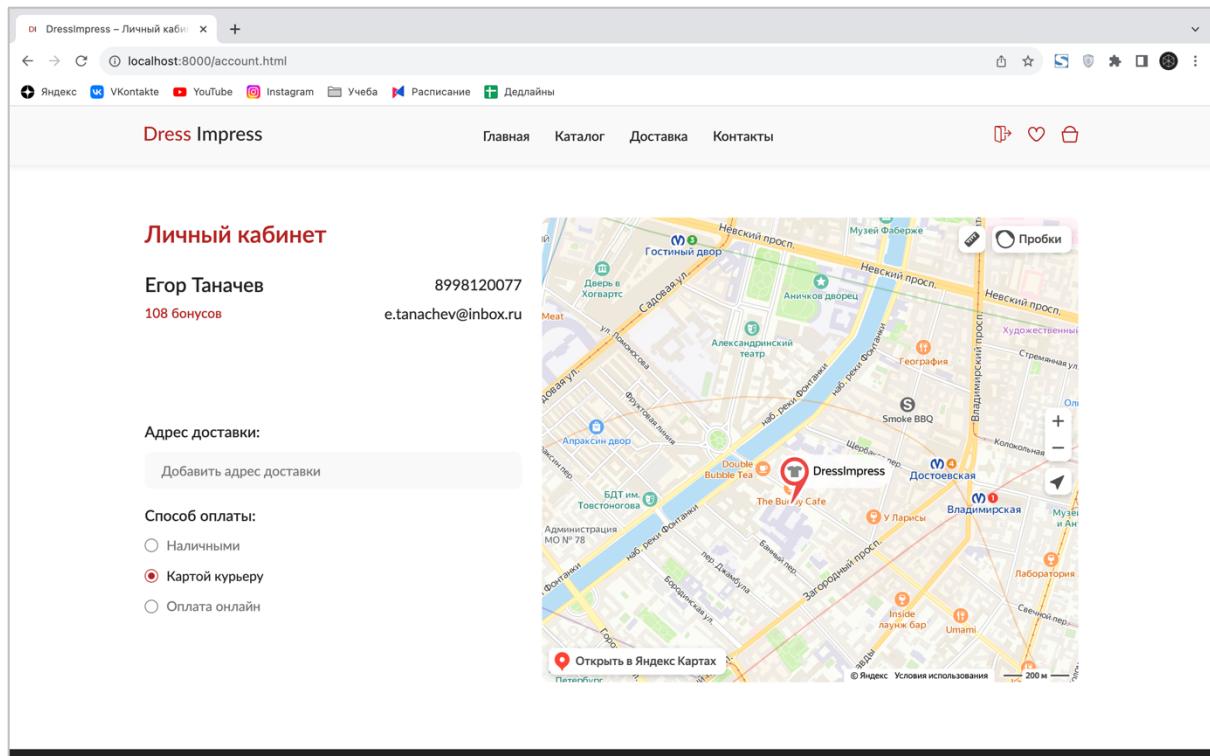


Рисунок 4 – Личный кабинет

```

<script>
    // Получение пользователя из локального хранилища
    const user = JSON.parse(localStorage.getItem('user'));

    // Добавление имени пользователя
    const accountNameElement = document.querySelector('.section__account-name');
    accountNameElement.textContent = user.name;
    // Добавление телефона пользователя
    const accountPhoneElement = document.querySelector('.section__account-phone');
    accountPhoneElement.textContent = user.phone;
    // Добавление почты пользователя
    const accountMailElement = document.querySelector('.section__account-mail');
    accountMailElement.textContent = user.email;

    // Выход из личного кабинета
    function logout() {
        localStorage.clear()
        window.location.href = "./catalog.html"
    }
</script>

```

Рисунок 5 – Генерация данных

После этого реализуем вывод и фильтрацию товаров с помощью отправление GET-запросов на сервер и инструмента фильтрации у json сервера. Скриншоты скриптов и страницы представлены на Рисунках 6-8.

```

59 // ВЫВОД ТОВАРОВ В КАТАЛОГ
60
61 document.addEventListener('DOMContentLoaded', async function () {
62     const catalogElement = document.querySelector('#catalog');
63
64     function getCardHtml({ id, title, price, image }) {
65         return `
66             <div class="section__catalog-card" data-card-id="${id}">
67                 
68                 <h3 class="section__catalog-card-title">${title}</h3>
69                 <p class="section__catalog-card-price">${price}</p>
70                 <button class="section__catalog-card-btn" onclick="addCardFavorite(event);">Добавить в избранное</button>
71             </div>
72     `;
73 }
74
75     async function loadAndDisplayCards(category = "Все товары") {
76         try {
77             // Формируем URL для запроса на сервер с учетом выбранной категории
78             const url = category === "Все товары"
79                 ? 'http://localhost:3000/products'
80                 : `http://localhost:3000/products?category=${encodeURIComponent(category)}`;
81
82             // Загружаем данные из JSON-сервера
83             const response = await fetch(url);
84             if (!response.ok) {
85                 throw new Error('Ошибка загрузки данных');
86             }
87
88             const data = await response.json(); // Получаем данные JSON
89
90             // Очищаем контейнер перед загрузкой новых карточек
91             catalogElement.innerHTML = '';
92
93             // Генерируем карточки товаров из данных
94             data.forEach((item) => { // Обращаемся к массиву data
95                 const cardHtml = getCardHtml(item);
96                 catalogElement.insertAdjacentHTML('beforeend', cardHtml);
97             });
98         } catch (error) {
99             console.error('Ошибка загрузки данных:', error);
100        }
101    }
102
103 // Загрузка и отображение карточек товаров при загрузке страницы
104 loadAndDisplayCards();
105
106 // Добавляем обработчик события для выбора категории
107 const categoryLinks = document.querySelectorAll('.header__categories-type');
108 categoryLinks.forEach(link => {
109     link.addEventListener('click', function(e) {
110         e.preventDefault(); // Предотвращаем переход по ссылке
111         const selectedCategory = this.textContent;
112         loadAndDisplayCards(selectedCategory);
113     });
114 });
115 });

```

Рисунок 6 – Вывод и фильтрация товаров

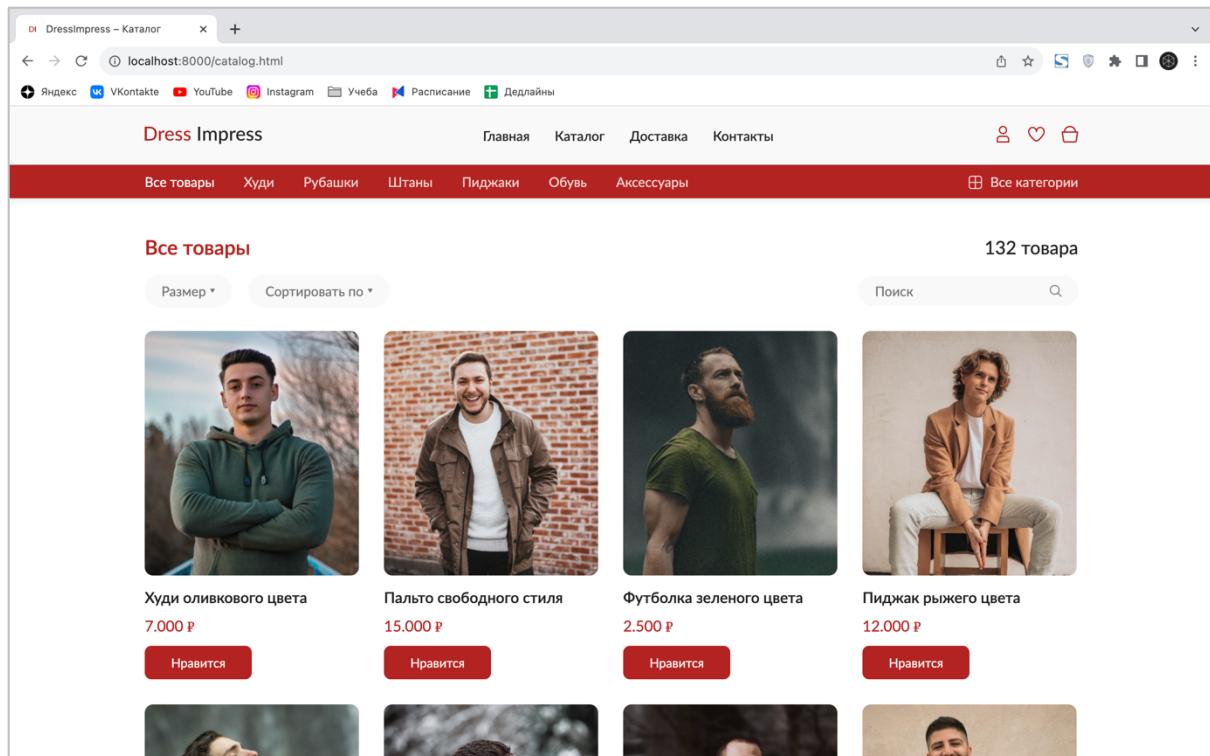


Рисунок 7 – Вывод всех товаров

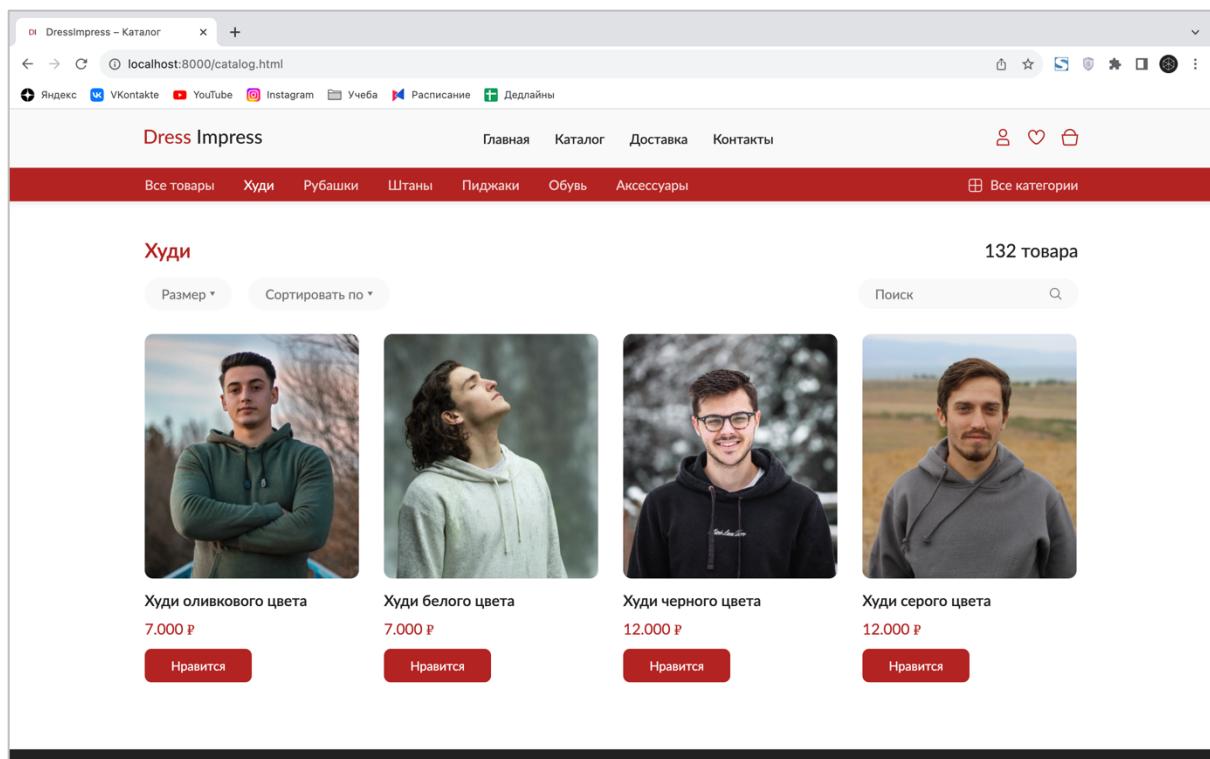


Рисунок 8 – Вывод товаров в категории "Худи"

Теперь реализуем добавление товаров в избранные с привязкой конкретному пользователю. Для этого напишем три скрипта, которые будут

выполнять функции добавления, генерации и удаления карточки из избранных. Для генерации мы опять будем использовать шаблонизацию и очищать блок с карточками перед загрузкой. Скриншоты скриптов и работы представлены на Рисунках 9-13.

```
29 // // ИЗБРАННЫЕ ТОВАРЫ
30
31 // Находим контейнер для карточек
32 const favoritesContainer = document.getElementById("favorites-cards");
33
34 // Получаем Id из localStorage
35 const user = JSON.parse(localStorage.getItem('user'));
36 const userId = user ? user.id : null;
37
38 // Основная функция для генерации и заполнения карточек
39 function generateFavoriteCards(data) {
40     favoritesContainer.innerHTML = '';
41
42     data.forEach((item) => {
43         // Создаем шаблон карточки
44         const cardTemplate = `
45             <div class="modal__favorites-card" data-card-id="${item.id}">
46                 
47                 <div class="modal__favorites-info">
48                     <h3 class="modal__favorites-title">${item.title}</h3>
49                     <p class="modal__favorites-price">${item.price}</p>
50                     <button class="modal__favorites-btn" onclick="deleteFavoriteCard(${item.id})">Удалить</button>
51                 </div>
52             </div>
53         `;
54
55         // Создаем элемент карточки из шаблона и добавляем его в контейнер
56         const cardElement = document.createElement('div');
57         cardElement.innerHTML = cardTemplate;
58         favoritesContainer.appendChild(cardElement.firstChild);
59     });
60 }
```

```

62  async function fetchData() {
63    try {
64      if (!userId) {
65        console.error('ID пользователя не найден в localStorage');
66        return;
67      }
68
69      const response = await fetch(`http://localhost:3000/favorites?userId=${userId}`, {
70        method: "GET",
71        headers: {
72          'Content-Type': 'application/json',
73          'Authorization': `Bearer ${getAuthToken()}`
74        }
75      );
76
77      if (!response.ok) {
78        throw new Error('Ошибка при выполнении запроса');
79      }
80
81      const data = await response.json();
82      // Вызываем функцию с данными из сервера
83      generateFavoriteCards(data);
84    } catch (error) {
85      console.error('Произошла ошибка при загрузке данных:', error);
86    }
87  }

```

Рисунок 9 – Генерация избранных товаров

```

89 // ДОБАВЛЕНИЕ ТОВАРА В ИЗБРАННЫЕ
90
91 function getAuthToken() {
92   return localStorage.accessToken
93 }
94
95 function addCardFavorite(event) {
96   event.preventDefault();
97   const { dataset: { btnId } } = event.target;
98
99   const cardId = btnId;
100  const userId = JSON.parse(localStorage.getItem('user')).id;
101  const image = document.querySelector(`.section__catalog-card[data-card-id='${btnId}'] .image`);
102  const h3 = document.querySelector(`.section__catalog-card[data-card-id='${btnId}'] h3`);
103  const p = document.querySelector(`.section__catalog-card[data-card-id='${btnId}'] p`);
104
105  // Создаем объект cardData и заполняем его данными
106  const cardData = {
107    cardId,
108    userId,
109    image,
110    title: h3,
111    price: p
112  };
113
114  // Отправляем данные на сервер
115  sendCardDataToServer(cardData);
116 }

```

```

117
118     async function sendCardDataToServer(cardData) {
119         const response = await fetch('http://localhost:3000/600/favorites', {
120             method: "POST",
121             body: JSON.stringify(cardData),
122             headers: {
123                 'Content-Type': 'application/json',
124                 'Authorization': `Bearer ${getAuthToken()}`
125             }
126         });
127     }

```

Рисунок 10 – Добавление в избранные

```

130 // УДАЛЕНИЕ КАРТОЧКИ
131
132 async function deleteFavoriteCard(cardId, event) {
133     event.preventDefault();
134
135     const response = await fetch(`http://localhost:3000/600/favorites/${cardId}`, {
136         method: 'DELETE',
137         headers: {
138             'Authorization': `Bearer ${getAuthToken()}`
139         }
140     );
141
142     if (response.status === 200) {
143         fetchData();
144     }
145 }

```

Рисунок 11 – Удаление из избранных

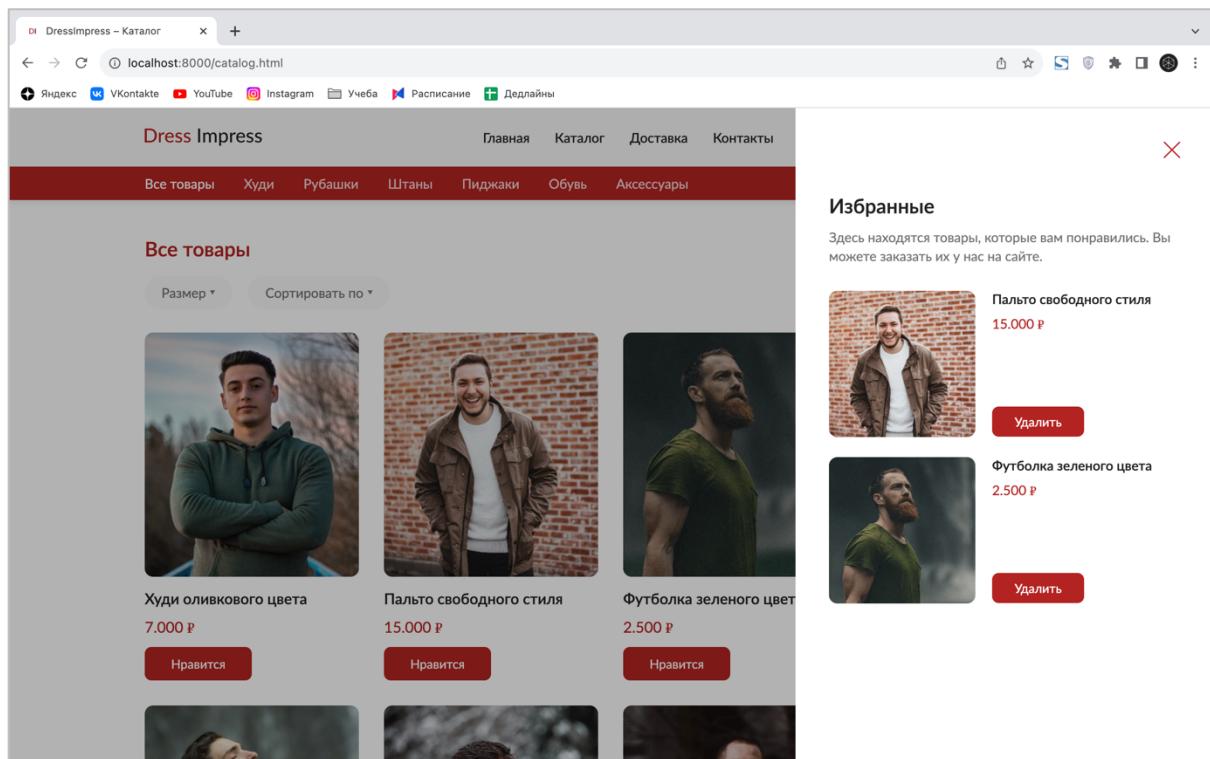


Рисунок 12 – Избранные товары

Вывод

В результате выполнения лабораторной работы было успешно решено основное задание - привязка проекта из Лабораторной работы №1 к внешнему API с использованием средств fetch/axios/xhr. В данной работе был использован json server и json server auth.