ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

Лабораторная работа № 1

Дисциплина: API-технологии

Тема: Основы работы с REST API

Выполнил: студент группы 221-375

Яковлев Р. А.

(Фамилия И.О.)

Дата, подпись _____ _____

(Дата) (Подпись)

Проверил: _____ _____

(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____

(Дата) (Подпись)

Замечания:

_____

_____

Москва

2025

Поиск книг через Open Library API

lr1.js

1. Инициализация элементов

В начале кода происходит получение ссылок на элементы DOM, с которыми будет производиться взаимодействие:

```
const searchButton = document.getElementById('search');
const queryInput = document.getElementById('query');
const resultsDiv = document.getElementById('results');
const errorDiv = document.getElementById('error');
```

2. Обработка события нажатия на searchButton

```
searchButton.addEventListener('click', async () => {
  const query = queryInput.value.trim();
  if (query === '') {
    showError('Please enter a search query.');
    return;
  }

  clearResults();
  showError('');

  try {
    const data = await searchBooks(query);
    if (data.numFound === 0) {
      showError('No results found.');
    } else {
      displayResults(data.docs);
    }
  } catch (error) {
    showError('An error occurred. Please try again later.');
  }
```

});

1. Получение запроса из поля ввода:
    - query — значение, введенное пользователем, очищенное от лишних пробелов с помощью метода trim().
    - Если запрос пустой, вызывается функция showError('Please enter a search query.');, которая отображает сообщение об ошибке, и выполнение функции прекращается.
2. Очистка предыдущих результатов и сообщений об ошибках:
    - clearResults() — очищает блок resultsDiv, удаляя предыдущие результаты.
    - showError('') — очищает сообщение об ошибке.
3. Асинхронный запрос к API:
    - Запрос к API выполняется с помощью функции searchBooks(query). Эта функция возвращает данные, которые затем обрабатываются.
    - Если данных не найдено (data.numFound === 0), отображается сообщение: No results found..
    - В случае успешного поиска, данные передаются в функцию displayResults(data.docs), которая отображает результаты на странице.
4. Обработка ошибок:
    - Если во время выполнения запроса возникает ошибка, выполнение переходит в блок catch, где вызывается функция

3. Отправка запроса к API через searchBooks(query)

```
async function searchBooks(query) {
  const url = `https://openlibrary.org/search.json?q=${encodeURIComponent(query)}`;
  const response = await fetch(url);
  if (!response.ok) {
    throw new Error('Failed to fetch data');
  }
```

```
  return response.json();
}
```

1. Формирование URL:
    - `url` — строка запроса, в которой поисковый запрос кодируется с помощью `encodeURIComponent(query)`, чтобы корректно обработать возможные спецсимволы.
2. Отправка запроса:
    - Используется функция `fetch(url)` для отправки GET-запроса к API.
    - Если ответ от сервера не успешный (`!response.ok`), выбрасывается ошибка `Failed to fetch data`.
3. Возврат данных:
    - В случае успешного ответа, данные преобразуются в JSON-формат с помощью `response.json()` и возвращаются из функции.

4. Отображение результатов через displayResults(books)

```
function displayResults(books) {
  books.forEach((book) => {
    const card = document.createElement('div');
    card.className = 'card';

    const cover = document.createElement('img');
    cover.className = 'cover-image';
    if (book.cover_i) {
      cover.src = `https://covers.openlibrary.org/b/id/${book.cover_i}-M.jpg`;
    } else {
      cover.src = 'https://via.placeholder.com/128x193?text=No+Cover';
    }
    cover.alt = book.title || 'Book cover';

    const title = document.createElement('div');
```

```
    title.className = 'card-title';
    title.textContent = book.title || 'No title available';

    const author = document.createElement('div');
    author.className = 'card-author';
    author.textContent = book.author_name
      ? `Author: ${book.author_name.join(', ')}`
      : 'Author: Unknown';

    const date = document.createElement('div');
    date.className = 'card-date';
    date.textContent = book.first_publish_year
      ? `First published: ${book.first_publish_year}`
      : 'No publication date';

    card.appendChild(cover);
    card.appendChild(title);
    card.appendChild(author);
    card.appendChild(date);
    resultsDiv.appendChild(card);
  });
}
```

1. Создание карточки для каждой книги:
    - Для каждой книги из массива `books` создается элемент `div` с классом `card`.
2. Загрузка обложки книги:
    - Если у книги есть идентификатор обложки (`book.cover_i`), то она загружается с сервиса covers.openlibrary.org
    - Если обложка отсутствует, используется изображение-заглушка с сервиса via.placeholder.com

3. Отображение названия, автора и даты публикации:
    - Название: Если название книги доступно, оно отображается, иначе — "No title available".
    - Автор: Если имя автора доступно, оно отображается в формате `Author: {Имя автора}`. Если автор неизвестен, отображается "Author: Unknown".
    - Дата публикации: Если дата публикации доступна, она отображается в формате `First published: {Год}`. В противном случае — "No publication date".
4. Добавление карточки в блок результатов:
    - Карточка книги добавляется в элемент `resultsDiv`, где отображаются все результаты поиска.

5. Вспомогательные функции clearResults() и showError(message)

```
function clearResults() {
  resultsDiv.innerHTML = '';
}

function showError(message) {
  errorDiv.textContent = message;
}
```

lr1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Book Finder</title>
  <style>
```

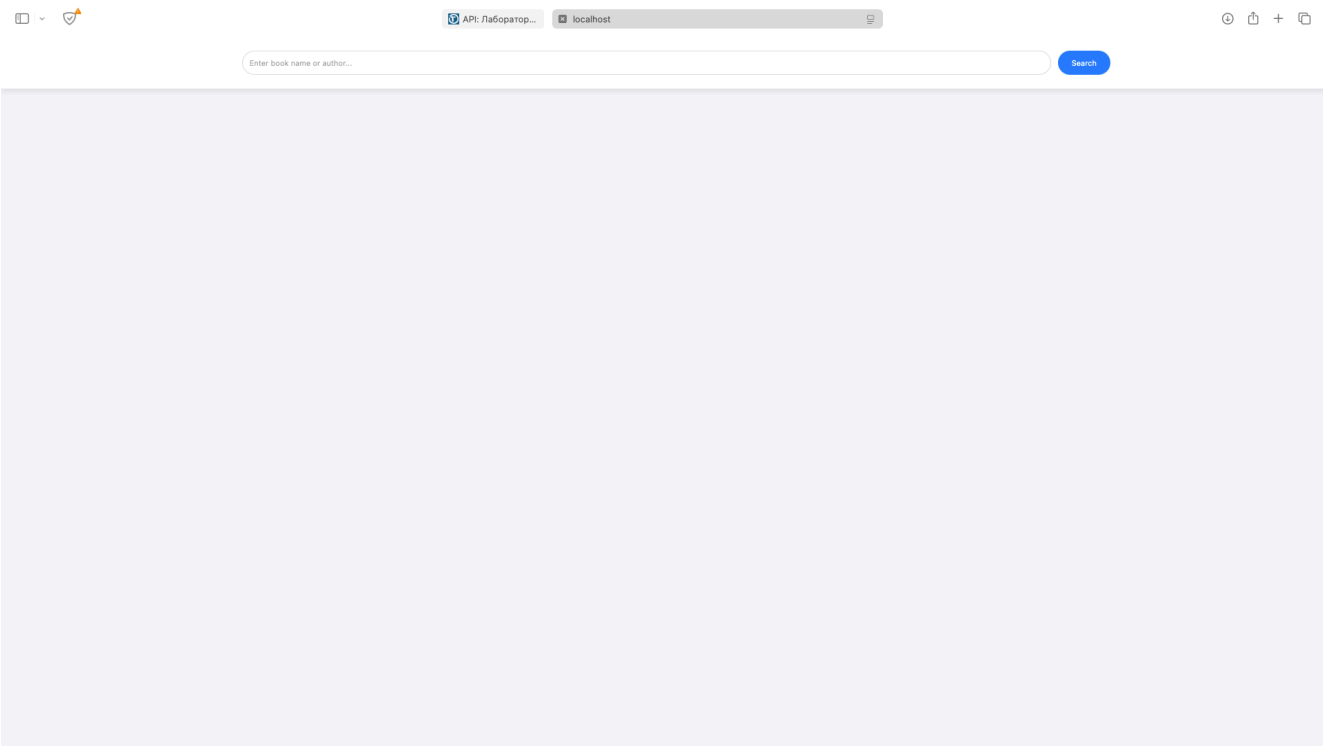
```css
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f9;
}

.search-container {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: #ffffff;
  padding: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  z-index: 1000;
}

.search-bar {
  display: flex;
  justify-content: center;
}

input[type="text"] {
  padding: 10px;
  border: 1px solid #ddd;
  border-radius: 20px;
  outline: none;
  width: 60%;
}
```

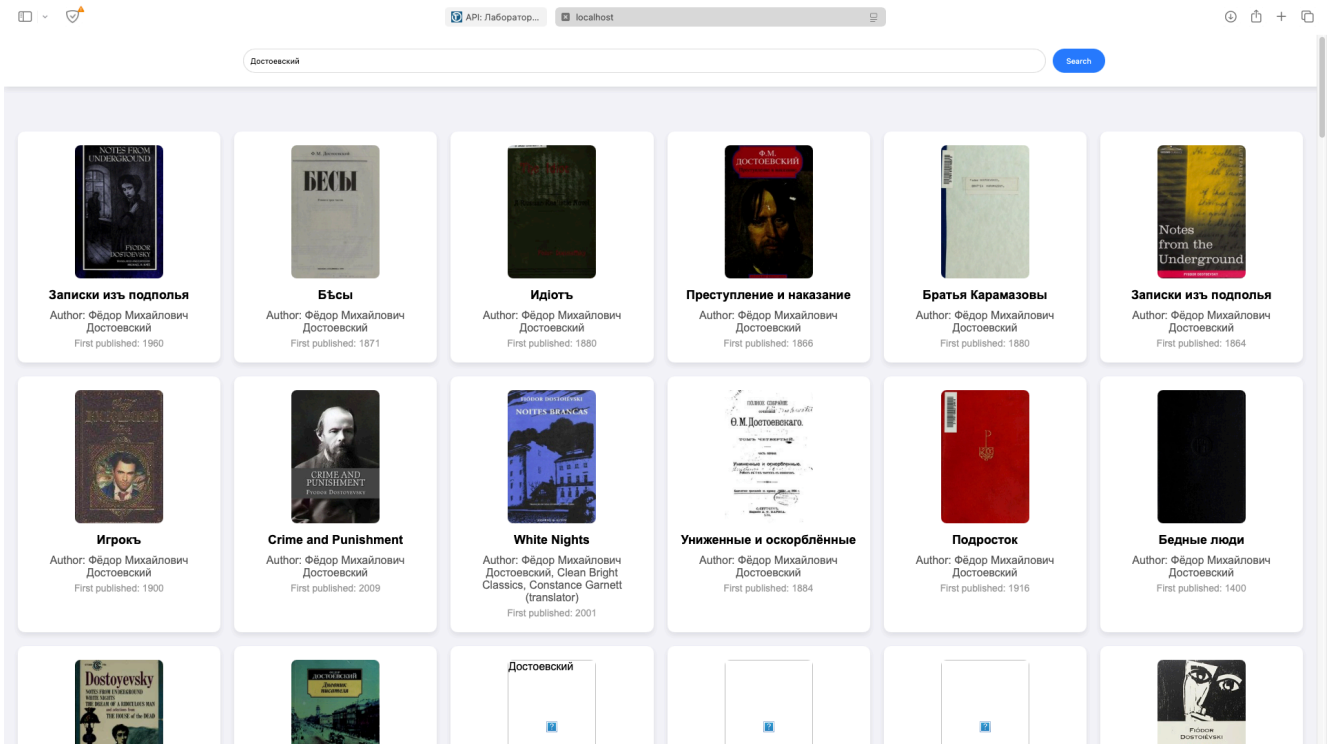
```
button {
  padding: 10px 20px;
  color: #fff;
  background-color: #007bff;
  border: none;
  border-radius: 20px;
  margin-left: 10px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #0056b3;
}

.content {
  margin-top: 120px;
  padding: 20px;
}

.results-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
}

.card {
  background-color: #ffffff;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

```
  text-align: center;
}

.cover-image {
  width: 128px;
  height: 193px;
  border-radius: 8px;
  margin-bottom: 10px;
  object-fit: cover;
}

.card-title {
  font-size: 18px;
  font-weight: bold;
  margin-bottom: 10px;
}

.card-author {
  font-size: 16px;
  color: #555;
  margin-bottom: 5px;
}

.card-date {
  font-size: 14px;
  color: #999;
}

.error {
  color: red;
  font-size: 16px;
```

```
    margin-top: 20px;
   }
  </style>
</head>
<body>
  <div class="search-container">
   <div class="search-bar">
    <input type="text" id="query" placeholder="Enter book name or author...">
    <button id="search">Search</button>
   </div>
  </div>
  <div class="content">
   <div class="results-grid" id="results"></div>
   <div class="error" id="error"></div>
  </div>
  <script src="lr1.js"></script>
</body>
</html>
```

Картиночки





Ссылка на репозиторий с кодом:

https://github.com/Ry0u14iY0Ru/APIshechki.git