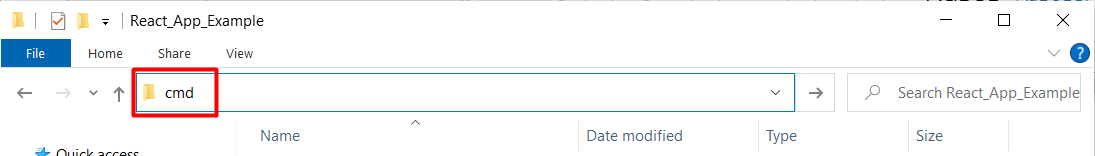
**Лаб. работа №3. Создание фронтенд-приложения на React.js**

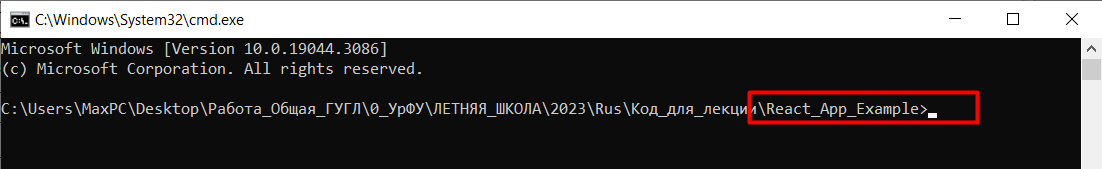
Создайте новую папку на рабочем столе с именем "**React\_App\_Example**".

Откройте командную строку из текущей папки.

Чтобы сделать это, вы можете просто ввести команду «**cmd**» в строке поиска текущей папки и нажать Enter.



После этого откроется командная строка:



Теперь, когда подготовка закончена, давайте создадим приложение React, которое подключается к API приложения Django и выполняет некоторые операции с записями.

**Шаг 1: Настройка приложения React**

1. Убедитесь, что у вас установлен **Node.js** и **npm** на локальном компьютере.
2. Запустите следующую команду, чтобы создать новое приложение React:

npx create-react-app my-app

Как только команда завершит выполнение, перейдите во вновь созданный каталог:

cd my-app

**Шаг 2: Установим необходимые пакеты**

Установите пакет **axios**, который является популярным HTTP-клиентом для выполнения запросов API:

npm install axios

Мы также установим дополнительный модуль "**react-router-dom**" для организации навигации в приложении:

npm install react-router-dom

Теперь нам нужно открыть проект в VS Code:

* + закройте командную строку,
  + запустите VS Code,
  + откройте папку проекта через меню File -> Open Folder.
  + запустите терминал из текущего проекта. Это можно сделать через меню Terminal -> New Terminal.

В терминале нам нужно перейти в папку проекта, чтобы дальнейшие команды выполнялись в соответствующей папке.

Чтобы сделать это, введите команду в терминале:

cd my-app

**Шаг 3: Создадим компоненты**

Создайте новую папку «**components**» в папке «**src**».

Создайте **Home.js** компонент (основная страница приложения) в каталоге **src/components** и вставьте туда приведенный ниже код:

import React from 'react'

function Home() {

  return (

    <div>

        <h1>Main Page</h1>

        <p>Here you can place some content of the main page {`:)`}</p>

    </div>

    )

}

export default Home

Создайте **PostList.js** компонент (в него мы будем выводить список всех постов из базы данных) в каталоге **src/components** и вставьте туда приведенный ниже код:

import React, { useEffect, useState } from 'react';

import axios from 'axios';

import { Link } from 'react-router-dom';

const PostList = () => {

  const [posts, setPosts] = useState([]);

  useEffect(() => {

    fetchPosts();

  }, []);

  const fetchPosts = async () => {

    try {

      const response = await axios.get('http://localhost:8000/posts/');

      setPosts(response.data);

      console.log(response.data);

    } catch (error) {

      console.error(error);

    }

  };

  return (

    <div>

      <h1>Posts</h1>

      <ul>

        {posts.map((post) => (

          <li key={post.id}>

            {post.title}

          </li>

        ))}

      </ul>

      <Link to="/posts/create">Create New Post</Link>

    </div>

  );

};

export default PostList;

Создать **PostCreate.js** компонент (здесь будет находится форма для создания новых постов) в каталоге **src/components**:

import React, { useState } from 'react';

import axios from 'axios';

const PostCreate = () => {

  const [title, setTitle] = useState('');

  const [content, setContent] = useState('');

  const handleSubmit = async (e) => {

    e.preventDefault();

    try {

      await axios.post('http://127.0.0.1:8000/posts/', { title, content });

      setTitle('');

      setContent('');

      console.log('Post created successfully.');

    } catch (error) {

      console.error(error);

    }

  };

  return (

    <div>

      <h1>Create Post</h1>

      <form onSubmit={handleSubmit}>

        <div>

          <label>Title:</label>

          <input type="text" value={title} onChange={(e) => setTitle(e.target.value)} />

        </div>

        <div>

          <label>Content:</label>

          <textarea value={content} onChange={(e) => setContent(e.target.value)}></textarea>

        </div>

        <button type="submit">Create</button>

      </form>

    </div>

  );

};

export default PostCreate;

**Шаг 4: Создадим основной компонент приложения**

Замените содержимое файла **src/App.js** следующим кодом:

import React from "react";

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

import PostList from "./components/PostList";

import PostCreate from "./components/PostCreate";

import Home from "./components/Home";

const App = () => {

  return (

    <Router>

      <div className="container">

        <h1>React Post App</h1>

        <nav>

          <ul>

            <li>

              <Link to="/">Home</Link>

            </li>

            <li>

              <Link to="/posts">Posts</Link>

            </li>

          </ul>

        </nav>

        <Routes>

          <Route exact path="/" element={<Home />} />

          <Route exact path="/posts" element={<PostList />} />

          <Route exact path="/posts/create" element={<PostCreate />} />

        </Routes>

      </div>

    </Router>

  );

};

export default App;

В приведенном выше коде дополнительный модуль React с именем "**react-router-dom**" используется для навигации между страницами. Это позволяет создавать навигацию по различным частям приложения. Изначально приложение React - это приложение SAP (SPA - одностраничное приложение). С помощью этого дополнительного модуля мы можем создать видимость перехода по страницам, но на самом деле мы просто переходим от одного компонента к другому, не перерисовывая всю страницу целиком.

Давайте добавим стили CSS на главную страницу и страницу с публикациями.

**Для этого:**

В папке **src/components** создайте файл **PostList.module.css** и добавьте в него следующие стили:

.wraper{

    padding: 20px;

    display: flex;

    flex-direction: column;

    align-items: center;

}

.header{

    text-align: center;

    margin-bottom: 20px;

}

.post{

    width: 70%;

    /\* border: 1px solid gray; \*/

    border-radius: 5px;

    padding: 5px;

    margin-bottom: 10px;

    box-shadow: 0px 0px 10px 10px rgba(194, 194, 198, 0.4);

}

.postTitle{

    color: #000;

    font-weight: 600;

    margin-bottom: 10px;

    padding: 5px;

}

.postContent{

    background-color:bisque;

    border-radius: 5px;

    padding: 10px;

}

.createNew{

    display: flex;

    justify-content: center;

    align-items: center;

    width: 150px;

    border-radius: 5px;

    text-decoration: none;

    color: #000;

    font-weight: bold;

    height: 50px;

    background-color: rgb(202, 86, 4);

    margin-top: 20px;

    margin-left: 10px;

    margin-right: 10px;

    transition: all 0.2s;

}

.createNew:hover {

    box-shadow: 0px 0px 10px 10px rgba(155, 155, 157, 0.8);

}

.buttons{

    display: flex;

    flex-direction: row;

    justify-content: space-between;

    align-items: center;

}

Добавим css-классы в код компонента **src/components/PostList.js** для минимальной стилизации страницы. Замените весь код компонента приведенным ниже кодом:

import React, { useEffect, useState } from 'react';

import axios from 'axios';

import { Link } from 'react-router-dom';

import styles from '../components/PostList.module.css';

const PostList = () => {

  const [posts, setPosts] = useState([]);

  useEffect(() => {

    fetchPosts();

  }, []);

  const fetchPosts = async () => {

    try {

      const response = await axios.get('http://localhost:8000/posts/');

      setPosts(response.data);

      console.log(response.data);

    } catch (error) {

      console.error(error);

    }

  };

  return (

    <div className={styles.wraper}>

      <h1 className={styles.header}>Posts</h1>

        {posts.map((post) => (

          <div className={styles.post} key={post.id}>

            <div className={styles.postTitle}>

              {post.title}

            </div>

            <div className={styles.postContent}>

              {post.content}

            </div>

          </div>

        ))}

      <div className={styles.buttons}>

        <Link className={styles.createNew} to="/posts/create">Create New Post</Link>

        <Link className={styles.createNew} to="/">Main page</Link>

      </div>

    </div>

  );

};

export default PostList;

В папке **src** создайте файл **App.module.css** и добавьте в него следующие стили:

.container{

    width: 95%;

    margin: 0 auto;

  }

  h1 {

    text-align: center;

  }

  nav {

    display: flex;

    flex-direction: row;

    justify-content: left;

    align-items: center;

  }

  .navLink {

    color: white;

    text-decoration: none;

    margin-left: 10px;

    font-weight: bold;

  }

Добавим css-классы в код компонента **src/App.js**. Замените весь код компонента приведенным ниже кодом:

import React from "react";

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

import PostList from "./components/PostList";

import PostCreate from "./components/PostCreate";

import Home from "./components/Home";

import styles from '../src/App.module.css';

const App = () => {

  return (

    <Router>

      <div className={styles.container}

        <h1 >React Post App</h1>

        <nav>

          <div><Link className={styles.navLink} to="/">Home</Link></div>

          <div><Link className={styles.navLink} to="/posts">Posts</Link></div>

        </nav>

        <Routes>

          <Route exact path="/" element={<Home />} />

          <Route exact path="/posts" element={<PostList />} />

          <Route exact path="/posts/create" element={<PostCreate />} />

        </Routes>

      </div>

    </Router>

  );

};

export default App;

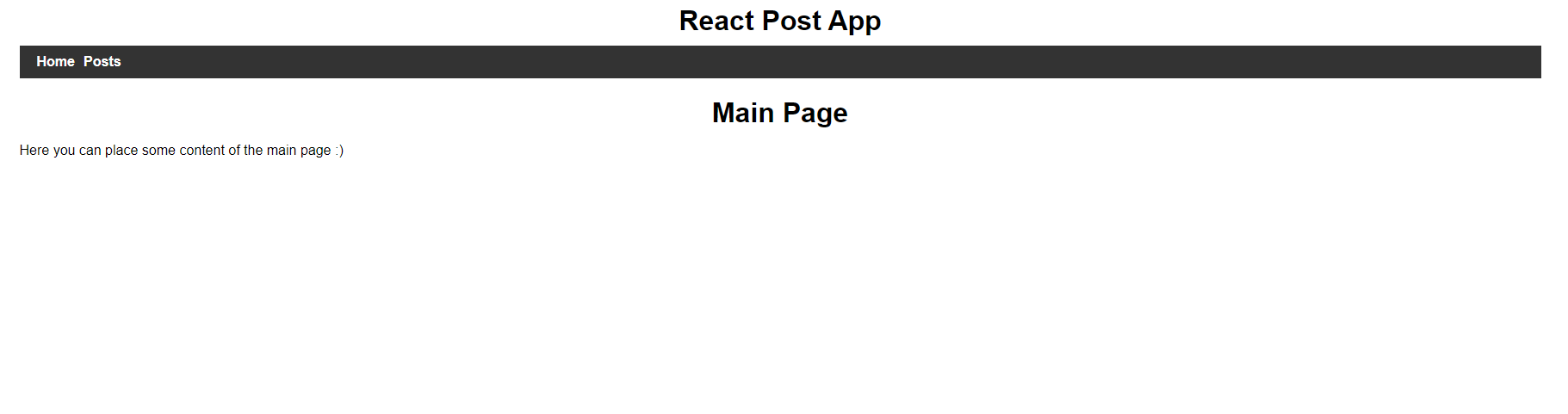
**Шаг 5: Запустите сервер разработки React**

**Сохраните** все файлы и запустите следующую команду в терминале:

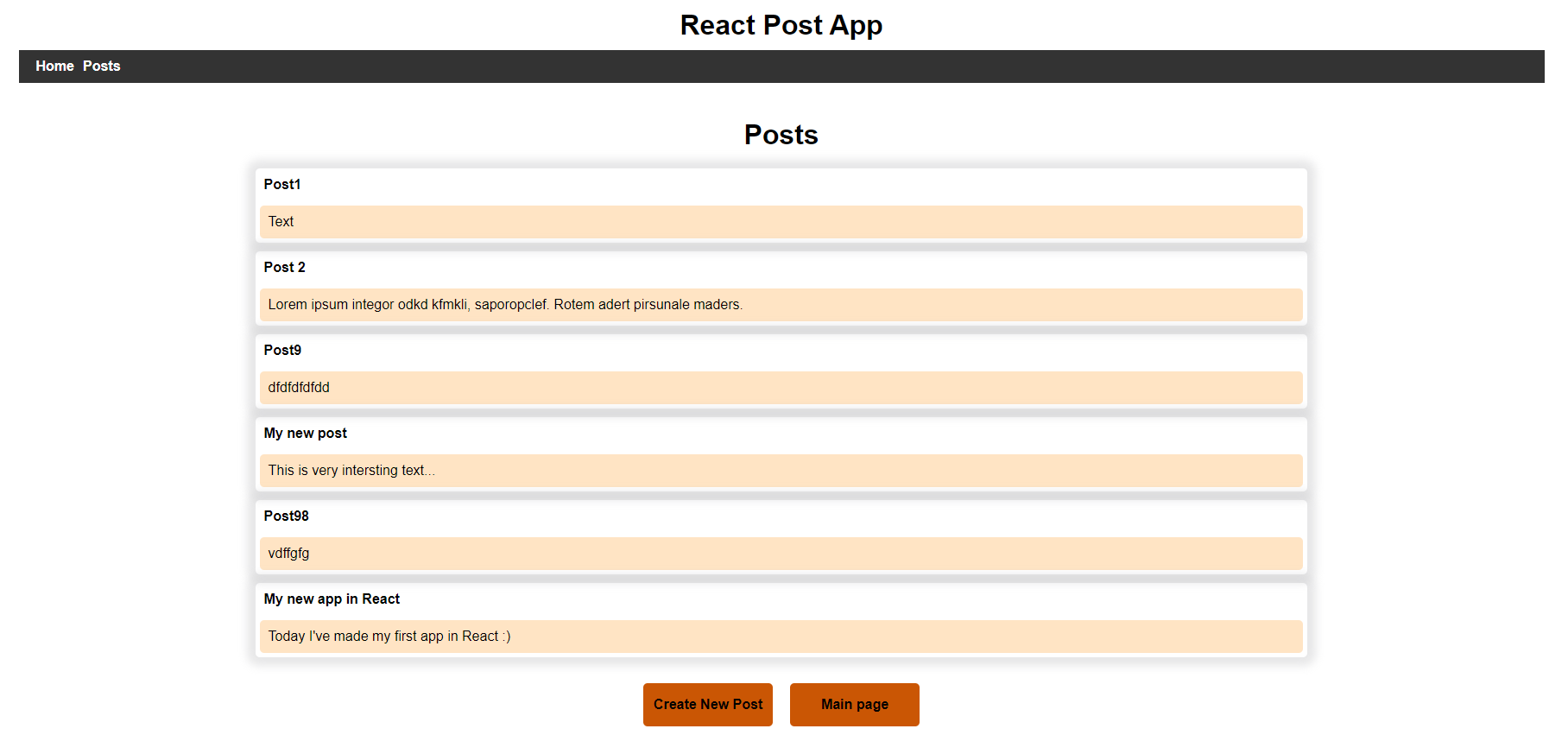
npm start

Приложение React будет запущено на [**http://localhost:3000**](http://localhost:3000). Вы можете получить к нему доступ в своем браузере.

***Примечание:*** *Для работы этого приложения необходимо запустить проект приложения Post на Django Framework, который мы создали в ходе последней лабораторной работы. Этот проект будет работать как удаленный API для нашего приложения React.*



Если нажать на ссылку "Post", приложение отправит запрос на сервер (которым сейчас является наше предыдущее приложение Django), получит данные из базы данных и отобразит их на экране.



Если нажать на кнопку "Create new post", мы перейдем на страницу, где есть возможность добавить новую запись и сохранить ее в базе данных.