perplexity

создай и пиши подробно, что делать. мне нужно создать простенький проект с двуми страницами, где бекенд - fastapi, а фронтенд - react. напиши все структуру файлов. у меня linux manjaro

Для создания простого проекта с двумя страницами, где бэкенд реализован на FastAPI, а фронтенд — на React, на Linux Manjaro, можно следовать следующему подробному плану с описанием структуры файлов и шагов.

1. Создание структуры проекта

Рекомендуется разделить проект на две отдельные папки:

- `backend` — для FastAPI
- `frontend` — для React

Пример структуры:

```
fastapi-react-project/
├── backend/
│   ├── app/
│   │   ├── __init__.py
│   │   ├── main.py
│   │   ├── api.py
│   │   ├── models.py
│   │   ├── database.py
│   │   └── routers/
│   │       ├── __init__.py
│   │       └── pages.py
│   ├── venv/  (виртуальное окружение)
│   └── requirements.txt
└── frontend/
    ├── node_modules/
    ├── public/
    ├── src/
    │   ├── components/
    │   │   ├── Page1.jsx
    │   │   └── Page2.jsx
    │   ├── api.js
    │   ├── App.jsx
    │   └── main.jsx
    ├── package.json
    └── vite.config.js (или webpack.config.js)
```

2. Настройка Backend (FastAPI)

Шаги:

1. Создайте папку `backend` и перейдите в неё:

```
mkdir -p fastapi-react-project/backend
cd fastapi-react-project/backend
```

2. Создайте виртуальное окружение и активируйте его:

```
python3 -m venv venv
source venv/bin/activate
```

3. Установите зависимости:

```
pip install fastapi uvicorn sqlalchemy pydantic
```

4. Создайте файл `requirements.txt`:

```
fastapi
uvicorn
sqlalchemy
pydantic
```

5. Создайте структуру папок и файлов, например:

- `app/main.py` — точка входа приложения FastAPI
- `app/api.py` — API роуты
- `app/models.py` — модели данных (например, SQLAlchemy модели)
- `app/database.py` — настройка базы данных (например, SQLite)
- `app/routers/pages.py` — роуты для страниц (если нужно отдавать шаблоны или API для страниц)

Пример `main.py`:

```
from fastapi import FastAPI
from app.routers import pages

app = FastAPI()

app.include_router(pages.router)

@app.get("/")
async def root():
    return {"message": "Hello from FastAPI backend!"}
```

Пример `routers/pages.py`**:**

```
from fastapi import APIRouter

router = APIRouter()

@router.get("/page1")
async def get_page1():
    return {"page": "This is page 1"}

@router.get("/page2")
async def get_page2():
    return {"page": "This is page 2"}
```

6. Запуск FastAPI сервера:

```
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

3. Настройка Frontend (React)

Шаги:

1. В корне проекта создайте папку `frontend` и перейдите в неё:

```
mkdir ../frontend
cd ../frontend
```

2. Инициализируйте React проект с помощью Vite (быстрый сборщик):

```
npm create vite@latest . --template react
npm install
```

3. Установите axios для запросов к API:

```
npm install axios
```

4. Структура папки `src`:

- `src/App.jsx` — основной компонент приложения с маршрутизацией на две страницы
- `src/components/Page1.jsx` — компонент первой страницы
- `src/components/Page2.jsx` — компонент второй страницы
- `src/api.js` — конфигурация для запросов к FastAPI

Пример `src/api.js`**:**

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:8000',  // адрес вашего FastAPI сервера
});

export default api;
```

Пример `src/components/Page1.jsx`**:**

```
import React, { useEffect, useState } from 'react';
import api from '../api';

function Page1() {
  const [data, setData] = useState(null);

  useEffect(() =&gt; {
    api.get('/page1').then(response =&gt; setData(response.data.page));
  }, []);

  return (
    <div>
      <h1>Page 1</h1>
      <p>{data}</p>
    </div>
  );
}

export default Page1;
```

Пример `src/components/Page2.jsx`**:**

```
import React, { useEffect, useState } from 'react';
import api from '../api';

function Page2() {
  const [data, setData] = useState(null);

  useEffect(() =&gt; {
    api.get('/page2').then(response =&gt; setData(response.data.page));
  }, []);

  return (
    <div>
      <h1>Page 2</h1>
      <p>{data}</p>
    </div>
  );
}
```

```
export default Page2;
```

Пример `src/App.jsx` с React Router:

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Page1 from './components/Page1';
import Page2 from './components/Page2';

function App() {
  return (
    &lt;Router&gt;
      &lt;nav&gt;
        &lt;Link to="/page1"&gt;Page 1&lt;/Link&gt; | &lt;Link to="/page2"&gt;Page 2&lt;/
      &lt;/nav&gt;
      &lt;Routes&gt;
        &lt;Route path="/page1" element={&lt;Page1 /&gt;} /&gt;
        &lt;Route path="/page2" element={&lt;Page2 /&gt;} /&gt;
      &lt;/Routes&gt;
    &lt;/Router&gt;
  );
}

export default App;
```

5. Установите React Router:

```
npm install react-router-dom
```

6. Запустите React приложение:

```
npm run dev
```

4. Итог

- FastAPI сервер слушает на `localhost:8000`
- React приложение — на `localhost:3000`
- React делает запросы к FastAPI через axios (CORS можно настроить в FastAPI, если нужно)

5. Настройка CORS в FastAPI (если React и FastAPI на разных портах)

Добавьте в `main.py`:

```
from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
```

```
    allow_origins=["http://localhost:3000"],  # адрес React
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Этот план и структура файлов основаны на лучших практиках и примерах из руководств по FastAPI + React [1] [2]. Такой подход позволит быстро создать и запустить простой проект с двумя страницами и разделённым бэкендом и фронтендом.

⁂

1. https://300.ya.ru/v_LJHSSqiw
2. https://testdriven.io/blog/fastapi-react/