MERN base project

Создаем новую директорию с именем `backend`, которая будет использоваться для хранения серверной части приложения.

```
mkdir backend
```

Переходим в директорию `backend`, чтобы выполнять дальнейшие команды в контексте этой папки.

```
cd .\backend
```

Устанавливаем необходимые пакеты через npm:

- `express`: веб-фреймворк для Node.js.
- `mongoose`: библиотека для работы с MongoDB.
- `dotenv`: позволяет загружать переменные окружения из файла .env.
- `cors`: middleware для включения CORS (Cross-Origin Resource Sharing).

```
npm i express mongoose dotenv cors
```

Выводим список установленных модулей в директории `node_modules`, чтобы проверить, что все необходимые пакеты были успешно установлены.

```
ls .\node_modules\
```

Открываем файл `.env` в редакторе кода для редактирования переменных окружения.

```
code .env
```

Пример файла `.env` в папке `backend`

```
MONGODB_URI="mongodb+srv://<USERNAME>:<PASSWORD>@<PATH_TO_DB>"
PORT=4000
FRONTEND_URL="http://localhost:5173"
```

Открываем файл `server.js` в редакторе кода для написания серверного кода.

```
code server.js
```

Создадим простой сервер с подключением к `MongoDB`.

```
const app = express();
const cors = require("cors");
require("dotenv").config();

// middleware
app.use(express.json());
app.use(cors({
    origin: process.env.FRONTEND_URL
}));

// connect MongoDB
mongoose.connect(process.env.MONGODB_URI).then(() => {
    const PORT = process.env.PORT
    app.listen(PORT, () => {
        console.log(`App is Listening on PORT ${PORT}`);
    })
}).catch(err => {
    console.log(err);
});

// route
app.get("/", (req, res) => {
    res.status(201).json({message: "Connected to Backend!"});
});
```

Запускаем сервер, используя Node.js, и выполняем код, написанный в файле `server.js`.

```
node .\server.js
```

Возвращаемся на один уровень вверх в файловой системе, чтобы перейти к родительской директории.

```
cd ..
```

Создаем новый проект с использованием Vite и называем его `frontend`.

```
npm create vite frontend
```

Переходим в директорию `frontend`, чтобы работать с клиентской частью приложения.

```
cd frontend
```

Устанавливаем все зависимости, указанные в `package.json` для проекта `frontend`.

```
npm install
```

Запускаем сервер разработки для проекта `frontend`, чтобы можно было тестировать приложение в режиме разработки.

```
npm run dev
```

Открываем файл `.env` в редакторе кода для редактирования переменных окружения.

```
code .env
```

Пример файла `.env` в папке `frontend`.

```
VITE_API_URL='http://localhost:4000/'
```

Открываем файл `App.jsx` в директории `src` в редакторе кода для редактирования клиентского кода.

```
code src\App.jsx
```

Пример кода для главной страницы сайта.

```
import { useEffect, useState } from "react";

const apiUrl = import.meta.env.VITE_API_URL;

function App() {
  const [message, setMessage] = useState("");

  // Получим сообщение с сервера
  useEffect(() => {
    fetch(apiUrl)
      .then((res) => res.json())
      .then((data) => setMessage(data.message));
  }, []);

  return (
    <div className="App">
```

```
        <h1>{message}</h1>
      </div>
    );
  }

  export default App;
```

Снова возвращаемся на один уровень вверх в файловой системе, чтобы перейти к родительской директории.

```
cd ..
```

Инициализируем новый проект npm в текущей директории, создавая файл `package.json` с настройками по умолчанию.

```
npm init -y
```

Открываем файл `package.json` для редактирования.

```
code package.json
```

Добавляем в файл `package.json` необходимые скрипты.

```
{
  "name": "mern-base",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "build-server": "cd backend && npm install",
    "server": "cd backend && nodemon server.js",
    "start-server": "cd backend && node server.js",
    "build-client": "cd frontend && npm install && npm run build",
    "client": "cd frontend && npm run dev"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Открываем файл `.gitignore` в редакторе кода для редактирования, чтобы указать, какие файлы и директории должны быть проигнорированы системой контроля версий Git.

```
code .gitignore
```

Содержимое файла `.gitignore`

```
node_modules

dist
build

.env
.DS_Store
.env.*

*.log*
```

Создаём новый репозиторий Git в корневой директории проекта

```
git init
```

Добавим все файлы проекта в индекс Git

```
git add .
```

Создадим коммит с сообщением, описывающим изменения.

```
git commit -m "Первый коммит"
```

Выкладываем проект на GitHub.