

Frontend Engineer — тестовое задание Предлагаем вам решить тестовое задание — реализовать макет нового функционала, который впоследствии предстоит внедрить в наш сервис.

У нас есть такие понятия, как админ, бот и пользователь бота.

* Админ — тот, кто создает бота.
* Пользователь — кто общается с ботом.
* Бот — это сущность, которая может автоматически общаться с пользователем.

Администратор бота хочет иметь возможность поиска по базе данных пользователей бота.

Приложение должно состоять из одной страницы которая отображает

* список пользователей бота
* для каждого пользователя отображается аватар и имя
* строка ввода поискового запроса

Особенности:

* Поиск в данной версии осуществляется только по имени пользователя. В дальнейшем возможно расширение на остальные атрибуты пользователей
* В одном боте может быть больше миллиона пользователей

Поведение:

* По мере того как Админ вводит поисковый запрос, список пользователей должен автоматически фильтроваться. Хочется избежать необходимости нажимать кнопку “поиск” либо вообще убрать ее из интерфейса.

Так как фича новая, то API на бекенде пока нет, но разработчики обещают дать следующее API:

typescript

Копировать код

GET -> /api/users?searchTerm={value} -> {result: [{id:string, name:string, avatarUrl: string}], nextPageUrl?:string, previousPageUrl?:string }

Можно использовать любой набор инструментов, фреймворков, библиотек и даже языков. Детали, которых не хватает в задаче, можно выбрать на свое усмотрение, включая дизайн.

Решение можно прислать в виде ссылки на любой публичный git-репозиторий: GitHub, Bitbucket, GitLab и т.п.

Желаем успехов!

ChatGPT

Для реализации данного тестового задания предлагаю использовать следующие технологии и инструменты:

1. **React** — для создания пользовательского интерфейса.
2. **Axios** — для выполнения HTTP-запросов.
3. **Styled-Components** — для стилизации компонентов.
4. **React-Debounce-Input** — для реализации функции debounce при вводе поискового запроса.

**Основные шаги реализации:**

1. **Создание проекта**:
   * Инициализируем новый проект с помощью Create React App.
   * Устанавливаем необходимые зависимости: Axios, Styled-Components, React-Debounce-Input.
2. **Структура проекта**:
   * components — папка для компонентов.
   * services — папка для файлов, связанных с API-запросами.
   * App.js — основной компонент приложения.
3. **Создание компонентов**:
   * UserList — компонент, отображающий список пользователей.
   * UserItem — компонент, отображающий информацию о пользователе (аватар и имя).
   * SearchBar — компонент для ввода поискового запроса.
4. **API-запросы**:
   * Создаем сервис для работы с API, который будет выполнять запросы и возвращать результаты поиска.
5. **Основной компонент**:
   * App.js будет содержать основную логику для получения данных, обновления состояния и рендеринга компонентов.

**Пример кода:**

**1. Инициализация проекта и установка зависимостей:**

bash

Копировать код

npx create-react-app bot-user-search

cd bot-user-search

npm install axios styled-components react-debounce-input

**2. Структура проекта:**

lua

Копировать код

src/

|-- components/

| |-- SearchBar.js

| |-- UserItem.js

| `-- UserList.js

|-- services/

| `-- api.js

|-- App.js

|-- index.js

|-- App.css

**3. Реализация компонентов:**

**services/api.js**:

javascript

Копировать код

import axios from 'axios';

const API\_URL = '/api/users';

export const fetchUsers = async (searchTerm) => {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}`);

return response.data;

};

**components/SearchBar.js**:

javascript

Копировать код

import React from 'react';

import { DebounceInput } from 'react-debounce-input';

import styled from 'styled-components';

const Input = styled(DebounceInput)`

padding: 10px;

width: 100%;

box-sizing: border-box;

font-size: 16px;

`;

const SearchBar = ({ onSearch }) => (

<Input

minLength={2}

debounceTimeout={300}

onChange={(e) => onSearch(e.target.value)}

placeholder="Search users..."

/>

);

export default SearchBar;

**components/UserItem.js**:

javascript

Копировать код

import React from 'react';

import styled from 'styled-components';

const UserContainer = styled.div`

display: flex;

align-items: center;

padding: 10px;

border-bottom: 1px solid #ddd;

`;

const Avatar = styled.img`

width: 50px;

height: 50px;

border-radius: 50%;

margin-right: 10px;

`;

const UserItem = ({ user }) => (

<UserContainer>

<Avatar src={user.avatarUrl} alt={user.name} />

<span>{user.name}</span>

</UserContainer>

);

export default UserItem;

**components/UserList.js**:

javascript

Копировать код

import React from 'react';

import UserItem from './UserItem';

import styled from 'styled-components';

const ListContainer = styled.div`

max-height: 80vh;

overflow-y: auto;

`;

const UserList = ({ users }) => (

<ListContainer>

{users.map(user => (

<UserItem key={user.id} user={user} />

))}

</ListContainer>

);

export default UserList;

**App.js**:

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

if (searchTerm) {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

} else {

setUsers([]);

}

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**Заключение**

Эта реализация предоставляет администратору возможность искать пользователей бота по имени в режиме реального времени без необходимости нажимать кнопку "поиск". Дальнейшее расширение функционала может включать поиск по другим атрибутам и улучшение дизайна интерфейса.

Проект можно разместить на любом публичном репозитории (например, GitHub), чтобы легко поделиться результатом.

4o

Инициализируем новый проект с помощью Create React App. как это сделать на гитхаб

ChatGPT

Для создания нового проекта с помощью Create React App и размещения его на GitHub, выполните следующие шаги:

**1. Создание нового проекта с Create React App**

1. **Инициализация нового проекта:**

bash

Копировать код

npx create-react-app bot-user-search

cd bot-user-search

1. **Создание локального репозитория Git:**

bash

Копировать код

git init

git add .

git commit -m "Initial commit"

**2. Создание нового репозитория на GitHub**

1. **Зайдите на GitHub и создайте новый репозиторий:**
   * Перейдите на GitHub (<https://github.com/>).
   * В верхнем правом углу страницы нажмите на иконку + и выберите New repository.
   * Введите имя репозитория (например, bot-user-search), добавьте описание, если необходимо, и нажмите кнопку Create repository.
2. **Подключение локального репозитория к GitHub:**
   * После создания репозитория на GitHub, вы увидите инструкции по его подключению к существующему локальному репозиторию. Выполните команды, указанные в разделе "…or push an existing repository from the command line":

bash

Копировать код

git remote add origin https://github.com/YOUR\_GITHUB\_USERNAME/bot-user-search.git

git branch -M main

git push -u origin main

Замените YOUR\_GITHUB\_USERNAME на ваше имя пользователя GitHub.

**3. Установка зависимостей и структура проекта**

1. **Установите необходимые зависимости:**

bash

Копировать код

npm install axios styled-components react-debounce-input

1. **Создайте структуру проекта и файлы:**
   * src/components/SearchBar.js
   * src/components/UserItem.js
   * src/components/UserList.js
   * src/services/api.js
   * src/App.js

**4. Реализация кода**

**services/api.js:**

javascript

Копировать код

import axios from 'axios';

const API\_URL = '/api/users';

export const fetchUsers = async (searchTerm) => {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}`);

return response.data;

};

**components/SearchBar.js:**

javascript

Копировать код

import React from 'react';

import { DebounceInput } from 'react-debounce-input';

import styled from 'styled-components';

const Input = styled(DebounceInput)`

padding: 10px;

width: 100%;

box-sizing: border-box;

font-size: 16px;

`;

const SearchBar = ({ onSearch }) => (

<Input

minLength={2}

debounceTimeout={300}

onChange={(e) => onSearch(e.target.value)}

placeholder="Search users..."

/>

);

export default SearchBar;

**components/UserItem.js:**

javascript

Копировать код

import React from 'react';

import styled from 'styled-components';

const UserContainer = styled.div`

display: flex;

align-items: center;

padding: 10px;

border-bottom: 1px solid #ddd;

`;

const Avatar = styled.img`

width: 50px;

height: 50px;

border-radius: 50%;

margin-right: 10px;

`;

const UserItem = ({ user }) => (

<UserContainer>

<Avatar src={user.avatarUrl} alt={user.name} />

<span>{user.name}</span>

</UserContainer>

);

export default UserItem;

**components/UserList.js:**

javascript

Копировать код

import React from 'react';

import UserItem from './UserItem';

import styled from 'styled-components';

const ListContainer = styled.div`

max-height: 80vh;

overflow-y: auto;

`;

const UserList = ({ users }) => (

<ListContainer>

{users.map(user => (

<UserItem key={user.id} user={user} />

))}

</ListContainer>

);

export default UserList;

**App.js:**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

if (searchTerm) {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

} else {

setUsers([]);

}

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**5. Публикация на GitHub**

1. **Добавьте изменения и сделайте коммит:**

bash

Копировать код

git add .

git commit -m "Implemented bot user search feature"

1. **Отправьте изменения на GitHub:**

bash

Копировать код

git push origin main

Теперь ваш проект доступен на GitHub по адресу https://github.com/YOUR\_GITHUB\_USERNAME/bot-user-search. Замените YOUR\_GITHUB\_USERNAME на ваше имя пользователя GitHub.

4o

Создание нового проекта с Create React App как это сделать

ChatGPT

Для создания нового проекта с помощью Create React App, выполните следующие шаги:

**1. Установите Node.js и npm**

Убедитесь, что у вас установлены Node.js и npm. Если нет, скачайте и установите их с [официального сайта Node.js](https://nodejs.org/).

**2. Создайте новый проект с Create React App**

1. **Откройте терминал или командную строку.**
2. **Введите следующую команду для создания нового проекта:**

bash

Копировать код

npx create-react-app my-app

Замените my-app на желаемое имя вашего проекта.

1. **Перейдите в созданную директорию проекта:**

bash

Копировать код

cd my-app

**3. Запуск проекта**

1. **В той же директории выполните команду для запуска локального сервера разработки:**

bash

Копировать код

npm start

После этого откроется новое окно браузера с вашим React приложением по адресу http://localhost:3000.

**4. Подключение проекта к GitHub**

1. **Создайте новый репозиторий на GitHub:**
   * Перейдите на [GitHub](https://github.com/).
   * В правом верхнем углу нажмите на кнопку + и выберите New repository.
   * Введите имя репозитория (например, my-app) и нажмите Create repository.
2. **Подключите локальный репозиторий к удаленному репозиторию на GitHub:**

В терминале, находясь в корневой директории проекта (my-app), выполните следующие команды:

bash

Копировать код

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin https://github.com/YOUR\_GITHUB\_USERNAME/my-app.git

git push -u origin main

Замените YOUR\_GITHUB\_USERNAME на ваше имя пользователя GitHub и my-app на имя вашего репозитория.

Теперь ваш проект создан с помощью Create React App и размещен на GitHub.

**Дополнительно: Полный пример реализации**

Чтобы выполнить ваше задание по созданию страницы с поиском пользователей бота, следуйте инструкциям ниже.

**5. Установка дополнительных зависимостей**

1. **Установите необходимые зависимости:**

bash

Копировать код

npm install axios styled-components react-debounce-input

**6. Создание структуры проекта**

1. **Создайте следующую структуру директорий и файлов:**

plaintext

Копировать код

src/

├── components/

│ ├── SearchBar.js

│ ├── UserItem.js

│ └── UserList.js

├── services/

│ └── api.js

├── App.js

└── index.js

**7. Реализация кода**

**src/services/api.js:**

javascript

Копировать код

import axios from 'axios';

const API\_URL = '/api/users';

export const fetchUsers = async (searchTerm) => {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}`);

return response.data;

};

**src/components/SearchBar.js:**

javascript

Копировать код

import React from 'react';

import { DebounceInput } from 'react-debounce-input';

import styled from 'styled-components';

const Input = styled(DebounceInput)`

padding: 10px;

width: 100%;

box-sizing: border-box;

font-size: 16px;

`;

const SearchBar = ({ onSearch }) => (

<Input

minLength={2}

debounceTimeout={300}

onChange={(e) => onSearch(e.target.value)}

placeholder="Search users..."

/>

);

export default SearchBar;

**src/components/UserItem.js:**

javascript

Копировать код

import React from 'react';

import styled from 'styled-components';

const UserContainer = styled.div`

display: flex;

align-items: center;

padding: 10px;

border-bottom: 1px solid #ddd;

`;

const Avatar = styled.img`

width: 50px;

height: 50px;

border-radius: 50%;

margin-right: 10px;

`;

const UserItem = ({ user }) => (

<UserContainer>

<Avatar src={user.avatarUrl} alt={user.name} />

<span>{user.name}</span>

</UserContainer>

);

export default UserItem;

**src/components/UserList.js:**

javascript

Копировать код

import React from 'react';

import UserItem from './UserItem';

import styled from 'styled-components';

const ListContainer = styled.div`

max-height: 80vh;

overflow-y: auto;

`;

const UserList = ({ users }) => (

<ListContainer>

{users.map(user => (

<UserItem key={user.id} user={user} />

))}

</ListContainer>

);

export default UserList;

**src/App.js:**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

if (searchTerm) {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

} else {

setUsers([]);

}

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**8. Коммит и пуш изменений на GitHub**

1. **Добавьте изменения и сделайте коммит:**

bash

Копировать код

git add .

git commit -m "Implemented bot user search feature"

1. **Отправьте изменения на GitHub:**

bash

Копировать код

git push origin main

Теперь ваше приложение готово и доступно на GitHub.

4o

Создайте новый проект с Create React App Откройте терминал или командную строку. как это сделать

ChatGPT

Для создания нового проекта с помощью Create React App, выполните следующие шаги:

**1. Установите Node.js и npm**

Убедитесь, что у вас установлены Node.js и npm. Если они не установлены, скачайте и установите их с [официального сайта Node.js](https://nodejs.org/).

**2. Создание нового проекта с Create React App**

**На Windows:**

1. **Откройте командную строку:**
   * Нажмите Win + R, введите cmd и нажмите Enter.

**На macOS:**

1. **Откройте терминал:**
   * Нажмите Cmd + Space, введите Terminal и нажмите Enter.

**На Linux:**

1. **Откройте терминал:**
   * Нажмите Ctrl + Alt + T или найдите терминал через меню приложений.

**3. Инициализация нового проекта**

1. **Введите следующую команду для создания нового проекта:**

bash

Копировать код

npx create-react-app my-app

Замените my-app на желаемое имя вашего проекта. Команда npx загрузит и запустит Create React App, создавая новый проект в папке my-app.

1. **Перейдите в созданную директорию проекта:**

bash

Копировать код

cd my-app

1. **Запустите локальный сервер разработки:**

bash

Копировать код

npm start

После этого откроется новое окно браузера с вашим React приложением по адресу http://localhost:3000.

**4. Создание нового репозитория на GitHub**

1. **Зайдите на GitHub и создайте новый репозиторий:**
   * Перейдите на GitHub (<https://github.com/>).
   * В верхнем правом углу страницы нажмите на иконку + и выберите New repository.
   * Введите имя репозитория (например, my-app), добавьте описание, если необходимо, и нажмите кнопку Create repository.

**5. Подключение локального репозитория к удаленному репозиторию на GitHub**

1. **В терминале, находясь в корневой директории проекта (my-app), выполните следующие команды:**

bash

Копировать код

git init

git add .

git commit -m "Initial commit"

git branch -M main

git remote add origin https://github.com/YOUR\_GITHUB\_USERNAME/my-app.git

git push -u origin main

Замените YOUR\_GITHUB\_USERNAME на ваше имя пользователя GitHub и my-app на имя вашего репозитория.

Теперь ваш проект создан с помощью Create React App и размещен на GitHub. Вы можете перейти к дальнейшей разработке вашего приложения, включая реализацию необходимых компонентов и функциональности.

4o

C:\Users\Workgroup>npx create-react-app my-app npm error code ENOENT npm error syscall lstat npm error path C:\Users\Workgroup\AppData\Roaming\npm npm error errno -4058 npm error enoent ENOENT: no such file or directory, lstat 'C:\Users\Workgroup\AppData\Roaming\npm' npm error enoent This is related to npm not being able to find a file. npm error enoent

npm error A complete log of this run can be found in: C:\Users\Workgroup\AppData\Local\npm-cache\_logs\2024-06-17T23\_13\_16\_587Z-debug-0.log

ChatGPT

Ошибка ENOENT указывает на то, что npm не может найти файл или директорию. Это может происходить по нескольким причинам, но в данном случае проблема может быть связана с отсутствием необходимых директорий или проблемами с правами доступа.

**Шаги по исправлению ошибки:**

1. **Проверьте установку Node.js и npm:**

Убедитесь, что Node.js и npm установлены корректно. Вы можете проверить это, выполнив следующие команды:

bash

Копировать код

node -v

npm -v

Эти команды должны вывести версии установленных Node.js и npm. Если они не установлены, скачайте и установите Node.js с [официального сайта](https://nodejs.org/).

1. **Создайте отсутствующие директории вручную:**

Создайте необходимые директории вручную, если они отсутствуют:

bash

Копировать код

mkdir C:\Users\Workgroup\AppData\Roaming\npm

mkdir C:\Users\Workgroup\AppData\Roaming\npm-cache

1. **Переустановите npm:**

Если директории уже существуют, попробуйте переустановить npm:

bash

Копировать код

npm install -g npm

1. **Очистите кеш npm:**

Попробуйте очистить кеш npm:

bash

Копировать код

npm cache clean --force

1. **Используйте другой пакетный менеджер:**

Если проблема не решается, можно попробовать использовать yarn для создания проекта. Установите yarn и используйте его для создания нового проекта:

bash

Копировать код

npm install -g yarn

yarn create react-app my-app

**Создание проекта с Create React App с использованием yarn**

1. **Установите yarn:**

bash

Копировать код

npm install -g yarn

1. **Создайте новый проект:**

bash

Копировать код

yarn create react-app my-app

1. **Перейдите в директорию проекта и запустите его:**

bash

Копировать код

cd my-app

yarn start

Если после выполнения этих шагов ошибка сохраняется, пожалуйста, предоставьте больше информации из лог-файла C:\Users\Workgroup\AppData\Local\npm-cache\\_logs\2024-06-17T23\_13\_16\_587Z-debug-0.log, чтобы можно было более точно диагностировать проблему.

4o

\*\*\* Please tell me who you are.

Run

git config --global user.email "you@example.com" git config --global user.name "Your Name"

to set your account's default identity. Omit --global to set the identity only in this repository.

ChatGPT

Эти сообщения появляются, когда Git запрашивает информацию о пользователе для выполнения коммитов. Если вы не настроили глобальные параметры для Git, вам нужно это сделать, чтобы продолжить работу с вашим проектом.

**Настройка Git пользователя**

Откройте ваш терминал или командную строку и выполните следующие команды, заменяя you@example.com и Your Name на ваш фактический адрес электронной почты и имя:

bash

Копировать код

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

**Полный процесс создания нового проекта с Create React App и загрузка его на GitHub**

Давайте пройдем через весь процесс с самого начала:

1. **Создание нового проекта с помощью Create React App:**

Откройте терминал или командную строку и выполните:

bash

Копировать код

npx create-react-app my-app

1. **Переход в директорию проекта:**

bash

Копировать код

cd my-app

1. **Настройка Git пользователя (если еще не выполнено):**

bash

Копировать код

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

1. **Инициализация нового репозитория Git:**

bash

Копировать код

git init

1. **Добавление всех файлов в репозиторий и создание первого коммита:**

bash

Копировать код

git add .

git commit -m "Initial commit"

1. **Создание нового репозитория на GitHub:**
   * Перейдите на [GitHub](https://github.com/).
   * Нажмите на кнопку + в правом верхнем углу и выберите New repository.
   * Введите имя репозитория (например, my-app), добавьте описание и нажмите Create repository.
2. **Подключение локального репозитория к удаленному репозиторию на GitHub:**

Выполните команды, которые GitHub предложит после создания нового репозитория. Например:

bash

Копировать код

git remote add origin https://github.com/YOUR\_GITHUB\_USERNAME/my-app.git

git branch -M main

git push -u origin main

Замените YOUR\_GITHUB\_USERNAME на ваше имя пользователя GitHub и my-app на имя вашего репозитория.

1. **Запуск проекта:**

bash

Копировать код

npm start

Теперь ваш проект создан с помощью Create React App и загружен на GitHub. Вы можете продолжить разработку и вносить изменения в свой проект, коммитить их и пушить на GitHub.

4o

C:\Users\Workgroup\admin-user-bot>git push -u origin main To <https://github.com/DmitryDanilin/admin-user-bot.git> ! [rejected] main -> main (fetch first) error: failed to push some refs to '<https://github.com/DmitryDanilin/admin-user-bot.git>' hint: Updates were rejected because the remote contains work that you do not hint: have locally. This is usually caused by another repository pushing to hint: the same ref. If you want to integrate the remote changes, use hint: 'git pull' before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details.

ChatGPT

Это сообщение об ошибке говорит о том, что на удаленном репозитории уже есть изменения, которых у вас нет локально. Это может произойти, например, если кто-то другой внес изменения и уже запустил git push, или если вы внесли изменения в другом месте и не обновили свой локальный репозиторий.

Чтобы решить эту проблему и успешно выполнить git push, вам нужно сначала получить (fetch) изменения с удаленного репозитория и затем интегрировать их с вашими локальными изменениями.

**Шаги для решения проблемы:**

1. **Получение изменений с удаленного репозитория:**

bash

Копировать код

git pull origin main

Эта команда загрузит (fetch) изменения с удаленного репозитория и попытается автоматически слить их с вашими локальными изменениями.

1. **Разрешение конфликтов (если они есть):**

Если git pull обнаруживает конфликты, вам придется разрешить их вручную. Git покажет вам файлы, в которых есть конфликты, и вы сможете внести изменения вручную, чтобы разрешить эти конфликты.

1. **Добавление и коммит изменений после разрешения конфликтов (если они были):**

bash

Копировать код

git add .

git commit -m "Merge changes from remote repository"

1. **Повторная попытка выполнения git push:**

bash

Копировать код

git push origin main

Теперь после успешного слияния изменений и коммита, git push должен выполниться без ошибок.

Если у вас возникают трудности при разрешении конфликтов или при выполнении любых из вышеперечисленных команд, не стесняйтесь обращаться за помощью!

3.5

C:\Users\Workgroup\admin-user-bot>git push origin main To <https://github.com/DmitryDanilin/admin-user-bot.git> ! [rejected] main -> main (non-fast-forward) error: failed to push some refs to '<https://github.com/DmitryDanilin/admin-user-bot.git>' hint: Updates were rejected because the tip of your current branch is behind hint: its remote counterpart. If you want to integrate the remote changes, hint: use 'git pull' before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details.

ChatGPT

Это сообщение означает, что ваш локальный main-ветка отстает от удаленной main-ветки, поэтому Git отклоняет вашу попытку прямого (non-fast-forward) пуша. Для успешного пуша вам нужно сначала получить (fetch) изменения с удаленного репозитория, затем интегрировать их с вашими локальными изменениями и только после этого выполнить пуш.

**Шаги для решения проблемы:**

1. **Получение изменений с удаленного репозитория:**

bash

Копировать код

git pull origin main

Эта команда загрузит (fetch) изменения с удаленного репозитория и автоматически попытается слить их с вашими локальными изменениями.

1. **Разрешение конфликтов (если они есть):**

Если git pull обнаруживает конфликты, вам придется разрешить их вручную. Git покажет вам файлы, в которых есть конфликты, и вы сможете внести изменения вручную, чтобы разрешить эти конфликты.

1. **Добавление и коммит изменений после разрешения конфликтов (если они были):**

bash

Копировать код

git add .

git commit -m "Merge changes from remote repository"

1. **Повторная попытка выполнения git push:**

bash

Копировать код

git push origin main

Теперь после успешного слияния изменений и коммита, git push должен выполниться без ошибок.

Если у вас возникают трудности при разрешении конфликтов или при выполнении любых из вышеперечисленных команд, не стесняйтесь обращаться за помощью!

3.5

1. Создание структуры проекта Создайте следующую структуру директорий и файлов:

plaintext Копировать код src/ ├── components/ │ ├── SearchBar.js │ ├── UserItem.js │ └── UserList.js ├── services/ │ └── api.js ├── App.js └── index.js

ChatGPT

Для создания указанной структуры проекта в React с помощью Create React App, выполните следующие шаги:

**1. Создание и переход в новый проект**

Если вы еще не создали проект с помощью Create React App, создайте его и перейдите в директорию проекта:

bash

Копировать код

npx create-react-app my-app

cd my-app

**2. Создание структуры директорий и файлов**

Создайте нужную структуру директорий и файлов. Вы можете сделать это вручную или с помощью команд в терминале.

**Вручную**

1. Перейдите в директорию src.
2. Создайте папки components и services.
3. В папке components создайте файлы SearchBar.js, UserItem.js, UserList.js.
4. В папке services создайте файл api.js.
5. В корне папки src создайте или отредактируйте файлы App.js и index.js.

**С помощью команд в терминале**

Выполните следующие команды в терминале, находясь в директории проекта:

bash

Копировать код

mkdir -p src/components src/services

touch src/components/SearchBar.js src/components/UserItem.js src/components/UserList.js src/services/api.js src/App.js src/index.js

**3. Реализация кода**

**src/services/api.js**

javascript

Копировать код

import axios from 'axios';

const API\_URL = '/api/users';

export const fetchUsers = async (searchTerm) => {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}`);

return response.data;

};

**src/components/SearchBar.js**

javascript

Копировать код

import React from 'react';

import { DebounceInput } from 'react-debounce-input';

import styled from 'styled-components';

const Input = styled(DebounceInput)`

padding: 10px;

width: 100%;

box-sizing: border-box;

font-size: 16px;

`;

const SearchBar = ({ onSearch }) => (

<Input

minLength={2}

debounceTimeout={300}

onChange={(e) => onSearch(e.target.value)}

placeholder="Search users..."

/>

);

export default SearchBar;

**src/components/UserItem.js**

javascript

Копировать код

import React from 'react';

import styled from 'styled-components';

const UserContainer = styled.div`

display: flex;

align-items: center;

padding: 10px;

border-bottom: 1px solid #ddd;

`;

const Avatar = styled.img`

width: 50px;

height: 50px;

border-radius: 50%;

margin-right: 10px;

`;

const UserItem = ({ user }) => (

<UserContainer>

<Avatar src={user.avatarUrl} alt={user.name} />

<span>{user.name}</span>

</UserContainer>

);

export default UserItem;

**src/components/UserList.js**

javascript

Копировать код

import React from 'react';

import UserItem from './UserItem';

import styled from 'styled-components';

const ListContainer = styled.div`

max-height: 80vh;

overflow-y: auto;

`;

const UserList = ({ users }) => (

<ListContainer>

{users.map(user => (

<UserItem key={user.id} user={user} />

))}

</ListContainer>

);

export default UserList;

**src/App.js**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

if (searchTerm) {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

} else {

setUsers([]);

}

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**src/index.js**

javascript

Копировать код

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**4. Запуск проекта**

Теперь вы можете запустить проект и убедиться, что все работает корректно:

bash

Копировать код

npm start

Проект должен запуститься в браузере по умолчанию по адресу http://localhost:3000.

4o

GET -> /api/users?searchTerm={value} -> {result: [{id

, name

, avatarUrl: string}], nextPageUrl?

, previousPageUrl?

}

ChatGPT

**Реализация компонента для поиска пользователей с пагинацией**

Для реализации поиска пользователей с поддержкой пагинации, нужно немного изменить структуру и логику проекта. Мы будем использовать следующий API:

plaintext

Копировать код

GET -> /api/users?searchTerm={value} -> {result: [{id:string, name:string, avatarUrl: string}], nextPageUrl?:string, previousPageUrl?:string }

**Структура проекта**

plaintext

Копировать код

src/

├── components/

│ ├── SearchBar.js

│ ├── UserItem.js

│ └── UserList.js

├── services/

│ └── api.js

├── App.js

└── index.js

**Реализация кода**

**src/services/api.js**

javascript

Копировать код

import axios from 'axios';

const API\_URL = '/api/users';

export const fetchUsers = async (searchTerm, pageUrl = '') => {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}&pageUrl=${pageUrl}`);

return response.data;

};

**src/components/SearchBar.js**

javascript

Копировать код

import React from 'react';

import { DebounceInput } from 'react-debounce-input';

import styled from 'styled-components';

const Input = styled(DebounceInput)`

padding: 10px;

width: 100%;

box-sizing: border-box;

font-size: 16px;

`;

const SearchBar = ({ onSearch }) => (

<Input

minLength={2}

debounceTimeout={300}

onChange={(e) => onSearch(e.target.value)}

placeholder="Search users..."

/>

);

export default SearchBar;

**src/components/UserItem.js**

javascript

Копировать код

import React from 'react';

import styled from 'styled-components';

const UserContainer = styled.div`

display: flex;

align-items: center;

padding: 10px;

border-bottom: 1px solid #ddd;

`;

const Avatar = styled.img`

width: 50px;

height: 50px;

border-radius: 50%;

margin-right: 10px;

`;

const UserItem = ({ user }) => (

<UserContainer>

<Avatar src={user.avatarUrl} alt={user.name} />

<span>{user.name}</span>

</UserContainer>

);

export default UserItem;

**src/components/UserList.js**

javascript

Копировать код

import React from 'react';

import UserItem from './UserItem';

import styled from 'styled-components';

const ListContainer = styled.div`

max-height: 80vh;

overflow-y: auto;

`;

const UserList = ({ users, loadMoreUsers }) => (

<ListContainer>

{users.map(user => (

<UserItem key={user.id} user={user} />

))}

{loadMoreUsers && (

<button onClick={loadMoreUsers}>Load More</button>

)}

</ListContainer>

);

export default UserList;

**src/App.js**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

const [nextPageUrl, setNextPageUrl] = useState(null);

useEffect(() => {

const getUsers = async () => {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

setNextPageUrl(result.nextPageUrl);

};

if (searchTerm) {

getUsers();

} else {

setUsers([]);

setNextPageUrl(null);

}

}, [searchTerm]);

const loadMoreUsers = async () => {

if (nextPageUrl) {

const result = await fetchUsers(searchTerm, nextPageUrl);

setUsers([...users, ...result.result]);

setNextPageUrl(result.nextPageUrl);

}

};

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} loadMoreUsers={nextPageUrl ? loadMoreUsers : null} />

</AppContainer>

);

};

export default App;

**src/index.js**

javascript

Копировать код

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**Запуск проекта**

Теперь вы можете запустить проект и убедиться, что все работает корректно:

bash

Копировать код

npm start

Проект должен запуститься в браузере по умолчанию по адресу http://localhost:3000.

Эта реализация включает компонент поиска с поддержкой автоматического обновления списка пользователей по мере ввода поискового запроса и поддержкой загрузки дополнительных пользователей с помощью кнопки "Load More".

4o

Uncaught runtime errors: × ERROR Request failed with status code 404 AxiosError: Request failed with status code 404 at settle (<http://localhost:3000/static/js/bundle.js:44769:12>) at XMLHttpRequest.onloadend (<http://localhost:3000/static/js/bundle.js:43435:66>) at Axios.request (<http://localhost:3000/static/js/bundle.js:43922:41>) at async fetchUsers (<http://localhost:3000/static/js/bundle.js:564:20>) at async getUsers (<http://localhost:3000/static/js/bundle.js:47:22>)

ChatGPT

Ошибка Request failed with status code 404 указывает на то, что запрашиваемый ресурс не найден. В данном случае проблема может быть связана с неправильным URL для API или с отсутствием нужного API на сервере.

**Шаги для устранения ошибки:**

1. **Проверьте правильность URL для API:**

Убедитесь, что URL API правильный и доступен. В нашем случае это:

javascript

Копировать код

const API\_URL = '/api/users';

Если ваш сервер находится на другом порту или домене, укажите полный URL, например:

javascript

Копировать код

const API\_URL = 'http://localhost:5000/api/users';

1. **Проверьте наличие API на сервере:**

Убедитесь, что серверная часть приложения запущена и может обрабатывать запросы по указанному URL.

1. **Проверьте обработку запросов на сервере:**

Убедитесь, что серверная часть корректно обрабатывает запросы типа GET /api/users?searchTerm={value}.

**Обновленный код для проверки:**

**src/services/api.js**

Если ваш API находится на другом порту, обновите URL:

javascript

Копировать код

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api/users'; // Убедитесь, что это правильный URL

export const fetchUsers = async (searchTerm, pageUrl = '') => {

try {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}&pageUrl=${pageUrl}`);

return response.data;

} catch (error) {

console.error("Error fetching users:", error);

throw error;

}

};

**Тестирование API:**

Используйте инструмент вроде Postman или curl, чтобы проверить, что ваш API работает как ожидалось. Например:

sh

Копировать код

curl -X GET 'http://localhost:5000/api/users?searchTerm=test'

Если API не отвечает или возвращает 404, проверьте настройки сервера.

**Если проблема сохраняется:**

1. **Логирование на стороне сервера:**

Добавьте логирование на стороне сервера, чтобы убедиться, что запросы приходят и обрабатываются корректно.

1. **Настройка CORS (если необходимо):**

Если ваш клиентский и серверный код находятся на разных портах или доменах, убедитесь, что сервер настроен для обработки CORS-запросов.

**Проверка обновленного проекта:**

После внесения всех изменений и проверок перезапустите клиентскую и серверную части приложения:

1. Запустите серверную часть (если это требуется):

bash

Копировать код

npm run server

1. Запустите клиентскую часть:

bash

Копировать код

npm start

**Дополнительная отладка:**

Если ошибка продолжает появляться, вы можете добавить больше отладочной информации в клиентский код для отслеживания проблемы.

**Обновленный fetchUsers с отладочной информацией:**

javascript

Копировать код

import axios from 'axios';

const API\_URL = 'http://localhost:5000/api/users'; // Убедитесь, что это правильный URL

export const fetchUsers = async (searchTerm, pageUrl = '') => {

try {

const response = await axios.get(`${API\_URL}?searchTerm=${searchTerm}&pageUrl=${pageUrl}`);

console.log('Fetched users:', response.data);

return response.data;

} catch (error) {

console.error("Error fetching users:", error);

throw error;

}

};

Это поможет вам понять, что именно идет не так в процессе запроса к API.

4o

как добавить вторую ветку на гитхабе и вместо апи сделать список пользователей текстовым файлом

ChatGPT

**Создание второй ветки на GitHub и замена API на текстовый файл**

Для того чтобы создать новую ветку на GitHub и заменить запросы к API на чтение данных из текстового файла, выполните следующие шаги:

**1. Создание новой ветки на GitHub**

Откройте терминал в корневой директории вашего проекта и выполните следующие команды:

bash

Копировать код

# Создание и переход на новую ветку

git checkout -b new-feature-branch

# Добавление и коммит изменений

git add .

git commit -m "Switched to reading user data from a text file"

# Пуш ветки на GitHub

git push origin new-feature-branch

**2. Создание текстового файла с пользователями**

Создайте файл users.txt в директории src и добавьте в него список пользователей в формате JSON:

**src/users.txt**

json

Копировать код

[

{ "id": "1", "name": "Alice", "avatarUrl": "https://example.com/avatar1.png" },

{ "id": "2", "name": "Bob", "avatarUrl": "https://example.com/avatar2.png" },

{ "id": "3", "name": "Charlie", "avatarUrl": "https://example.com/avatar3.png" }

]

**3. Обновление кода для чтения данных из файла**

Теперь обновим наш код, чтобы вместо запроса к API данные загружались из текстового файла.

**Установка библиотеки для чтения текстовых файлов**

bash

Копировать код

npm install axios

**src/services/api.js**

Измените api.js для чтения данных из текстового файла:

javascript

Копировать код

import axios from 'axios';

const USERS\_FILE = '/users.txt';

export const fetchUsers = async (searchTerm) => {

try {

const response = await axios.get(USERS\_FILE);

const users = response.data;

return {

result: users.filter(user => user.name.toLowerCase().includes(searchTerm.toLowerCase())),

};

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

};

**src/App.js**

Обновите код в App.js, чтобы использовать новые данные:

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**4. Тестирование проекта**

Запустите проект:

bash

Копировать код

npm start

Откройте проект в браузере и убедитесь, что поиск пользователей работает корректно, загружая данные из users.txt.

**5. Коммит и пуш изменений**

После внесения всех изменений выполните коммит и пуш:

bash

Копировать код

# Добавление и коммит изменений

git add .

git commit -m "Switched to reading user data from a text file"

# Пуш изменений на новую ветку

git push origin new-feature-branch

Теперь у вас есть новая ветка на GitHub с изменениями, которые переключают загрузку данных с API на текстовый файл.

4o

ERROR Request failed with status code 404 AxiosError: Request failed with status code 404 at settle (<http://localhost:3000/static/js/bundle.js:44762:12>) at XMLHttpRequest.onloadend (<http://localhost:3000/static/js/bundle.js:43428:66>) at Axios.request (<http://localhost:3000/static/js/bundle.js:43915:41>) at async fetchUsers (http://localhost:3000/static/js/bundle.js:550:22) at async getUsers (<http://localhost:3000/static/js/bundle.js:46:22>)

ChatGPT

**Ошибка 404 при загрузке текстового файла**

Ошибка 404 указывает на то, что файл users.txt не найден. Это может быть связано с тем, что файл не находится в правильной директории или путь к файлу указан неправильно.

Чтобы исправить эту проблему, убедитесь, что файл users.txt существует в правильной директории и путь к файлу указан корректно.

**Шаги для устранения ошибки:**

1. **Создайте файл users.txt в директории public:**

Переместите файл users.txt в директорию public вашего проекта. Директория public предназначена для статических файлов, которые будут доступны в корне вашего веб-приложения.

1. **Проверьте путь к файлу в api.js:**

Обновите путь к файлу в api.js, чтобы он указывал на правильное место.

**Пример:**

1. **Переместите файл users.txt в директорию public:**

Создайте файл public/users.txt и добавьте в него список пользователей:

json

Копировать код

[

{ "id": "1", "name": "Alice", "avatarUrl": "https://example.com/avatar1.png" },

{ "id": "2", "name": "Bob", "avatarUrl": "https://example.com/avatar2.png" },

{ "id": "3", "name": "Charlie", "avatarUrl": "https://example.com/avatar3.png" }

]

1. **Обновите src/services/api.js:**

Измените api.js, чтобы использовать правильный путь к файлу:

javascript

Копировать код

import axios from 'axios';

const USERS\_FILE = '/users.txt';

export const fetchUsers = async (searchTerm) => {

try {

const response = await axios.get(USERS\_FILE);

const users = response.data;

return {

result: users.filter(user => user.name.toLowerCase().includes(searchTerm.toLowerCase())),

};

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

};

**Полный проект:**

**src/services/api.js**

javascript

Копировать код

import axios from 'axios';

const USERS\_FILE = '/users.txt';

export const fetchUsers = async (searchTerm) => {

try {

const response = await axios.get(USERS\_FILE);

const users = response.data;

return {

result: users.filter(user => user.name.toLowerCase().includes(searchTerm.toLowerCase())),

};

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

};

**src/App.js**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**Запуск проекта:**

1. Убедитесь, что файл users.txt находится в директории public.
2. Запустите проект:

bash

Копировать код

npm start

Теперь проект должен корректно загружать данные пользователей из текстового файла, находящегося в директории public.

4o

users.filter is not a function TypeError: users.filter is not a function at fetchUsers (http://localhost:3000/static/js/bundle.js:553:21) at async getUsers (<http://localhost:3000/static/js/bundle.js:46:22>)

ChatGPT

Ошибка users.filter is not a function указывает на то, что данные, загруженные из файла, не являются массивом. Это может происходить по нескольким причинам:

1. **Файл users.txt не содержит корректный JSON.**
2. **Ответ от сервера не парсится как JSON.**

**Шаги для устранения ошибки:**

1. **Проверьте формат файла users.txt:**

Убедитесь, что содержимое файла users.txt корректно и соответствует формату JSON:

json

Копировать код

[

{ "id": "1", "name": "Alice", "avatarUrl": "https://example.com/avatar1.png" },

{ "id": "2", "name": "Bob", "avatarUrl": "https://example.com/avatar2.png" },

{ "id": "3", "name": "Charlie", "avatarUrl": "https://example.com/avatar3.png" }

]

1. **Обновите api.js, чтобы убедиться, что данные парсятся как JSON:**

Давайте добавим проверку типа данных в api.js, чтобы убедиться, что данные являются массивом.

**src/services/api.js**

javascript

Копировать код

import axios from 'axios';

const USERS\_FILE = '/users.txt';

export const fetchUsers = async (searchTerm) => {

try {

const response = await axios.get(USERS\_FILE);

let users = response.data;

// Проверяем, является ли users массивом

if (typeof users === 'string') {

users = JSON.parse(users);

}

if (!Array.isArray(users)) {

throw new Error('Users data is not an array');

}

return {

result: users.filter(user => user.name.toLowerCase().includes(searchTerm.toLowerCase())),

};

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

};

1. **Проверьте, правильно ли загружается файл:**

Убедитесь, что файл users.txt доступен по правильному пути. В случае разработки на локальном сервере путь /users.txt должен быть доступен без ошибок.

**Полный проект:**

**src/services/api.js**

javascript

Копировать код

import axios from 'axios';

const USERS\_FILE = '/users.txt';

export const fetchUsers = async (searchTerm) => {

try {

const response = await axios.get(USERS\_FILE);

let users = response.data;

// Проверяем, является ли users массивом

if (typeof users === 'string') {

users = JSON.parse(users);

}

if (!Array.isArray(users)) {

throw new Error('Users data is not an array');

}

return {

result: users.filter(user => user.name.toLowerCase().includes(searchTerm.toLowerCase())),

};

} catch (error) {

console.error('Error fetching users:', error);

throw error;

}

};

**src/App.js**

javascript

Копировать код

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

import styled from 'styled-components';

import SearchBar from './components/SearchBar';

import UserList from './components/UserList';

import { fetchUsers } from './services/api';

const AppContainer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

`;

const App = () => {

const [users, setUsers] = useState([]);

const [searchTerm, setSearchTerm] = useState('');

useEffect(() => {

const getUsers = async () => {

try {

const result = await fetchUsers(searchTerm);

setUsers(result.result);

} catch (error) {

console.error('Failed to fetch users:', error);

}

};

getUsers();

}, [searchTerm]);

return (

<AppContainer>

<h1>Bot User Search</h1>

<SearchBar onSearch={setSearchTerm} />

<UserList users={users} />

</AppContainer>

);

};

export default App;

**Запуск проекта:**

1. Убедитесь, что файл users.txt находится в директории public.
2. Запустите проект:

bash

Копировать код

npm start

Теперь проект должен корректно загружать данные пользователей из текстового файла, находящегося в директории public.

4o

Начало формы



Конец формы

ChatGPT может допускать ошибки. Рекомендуем проверять ва