Проект 5

Установка и запуск чата

1.Вам нужно установить Node.js (<https://nodejs.org/en/>), выбирайте рекомендуемый установщик.

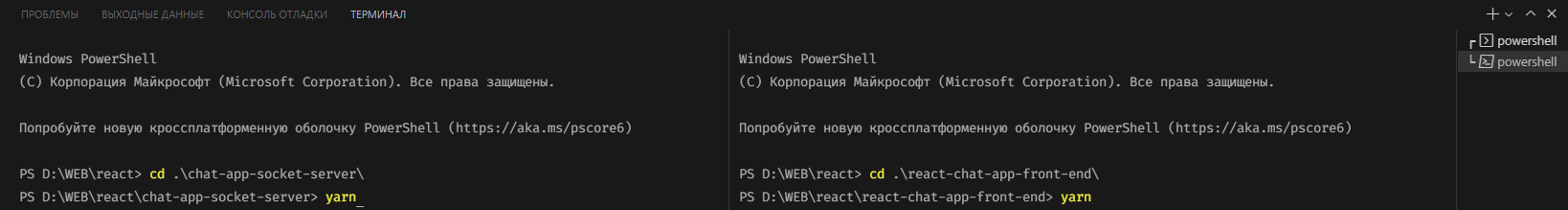
2. Чтобы скачать чат вы должны скачать zip файл в Teams или перейти на репозиторий чата и с клонировать его.

Репозиторий: <https://github.com/KozinAlexandr/Chat_react-soket.io>

git clone <https://github.com/KozinAlexandr/Chat_react-soket.io>

3. После того как вы с клонировали или скачали чат нужно открыть его в Visual Studio Code и открыть терминал.

Далее вы прописываете в терминале команду “cd” чтобы перейти из корневой папки в нужную и устанавливаете общие зависимости, а также зависимости сервера и клиента с помощью команд “yarn” или “npm i”.

На скриншоте показан терминал:  


4. Чтобы запустить чат нужно написать, находясь в папках react-chat-app-front-end и chat-app-socket-server “yarn start” или “npm start”

Далее вас должно перенести в браузер с работающим чатом.

Теория к установке и запуску чата

**React** — это JavaScript-библиотека для создания пользовательских интерфейсов. Обратите внимание, что это именно библиотека, а не фреймворк. React часто называют фреймворком, но это ошибка. Во-первых, его использование ни к чему вас не обязывает, не формирует «фрейм» проекта. Во-вторых, React выполняет единственную задачу: показывает на странице компонент интерфейса, синхронизируя его с данными приложения, и только этой библиотеки в общем случае недостаточно для того, чтобы полностью реализовать проект.

**Node.js** — это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом, которая выполняет код JavaScript вне браузера.  
Он используется для работы с React приложением и без него вы не сможете запустить чат.

Node Package Manager (**npm**) — менеджер пакетов, входящий в состав Node.js.  
Инфраструктура пакетного менеджера состоит из консольного клиента (npm) и доступной онлайн базы данных публичных и приватных пакетов — npm registry. Клиент позволяет скачивать пакеты из реестра и устанавливать их. Для хранения списка используемых в проекте пакетов используется файл package.json

**Yarn** — это альтернативный npm-клиент для работы в качестве пакетного менеджера JavaScript, совместно созданный Facebook, Google, Exponent и Tilde. Этот менеджер пакетов ускоряет сборку пакетов и делает её более безопасной.

NPM в файле package.json фиксирует не конкретную версию используемых пакетов, а их диапазон. Таким образом, стратегия стандартного клиента npm может привести к тому, что на основе идентичных файлов package.json в разное время будут установлены различные версии пакетов. Yarn решает эту проблему, так как позволяет точно зафиксировать зависимости в файле yarn.lock

Последовательная установка пакетов через npm заметно замедляет работу. Yarn же поддерживает параллельную установку, что обычно в несколько раз быстрее.

Структура чата

Чат представлен двумя папками chat-app-socket-server и react-chat-app-front-end.

В chat-app-socket-server представлена back-end часть чата.

В файле server.js написан весь код back-end и этот файл самый важный в этой папке.

Я не буду подробно расписывать, что и как происходит в файле server.js, потому что нужная часть кода закомментирована и в файле ничего сложного для понимания нету.

В файле react-chat-app-front-end представлена front-end часть чата.

Так же, как и в back-end части большая часть кода закомментирована и была прокомментирована при сдаче чата лично, поэтому объяснять использования хуков и других функций не вижу смысла.

Используемые библиотеки

Back-end:  
1. **Express** - самый популярный веб-фреймворк для Node. Он является базовой библиотекой для ряда других популярных веб-фреймворков Node. Он предоставляет следующие механизмы:

* Написание обработчиков для запросов с различными HTTP-методами в разных URL-адресах (маршрутах).
* Интеграцию с механизмами рендеринга «view», для генерации ответов, вставляя данные в шаблоны.
* Установка общих параметров веб-приложения, такие как порт для подключения, и расположение шаблонов, которые используются для отображения ответа.
* «промежуточное ПО» для дополнительной обработки запроса в любой момент в конвейере обработки запросов.

2. **Socket.io** — это библиотека, которая обеспечивает двустороннюю и основанную на событиях связь в режиме реального времени между браузером и сервером. Она состоит из:

* Сервера Node.js
* Клиентской библиотеки Javascript для браузера (которую также можно запустить из Node.js):

Его основными особенностями являются:

* Надежность
* Поддержка автоматического переподключения
* Обнаружение разъединения
* Поддержка бинарных типов данных
* Поддержка мультиплексирования
* Поддержка комнат (room)

3. **Nodemon -** заключает в оболочку ваше приложение Node, наблюдает за файловой системой и автоматически перезапускает процесс.

В Node.js для вступления изменений в силу необходимо перезапустить процесс. Это добавляет в рабочий процесс дополнительный шаг, необходимый для внесения изменений. Вы можете устранить этот дополнительный шаг, используя nodemon для автоматического перезапуска процесса.

Front-end:

1. React – js библиотека

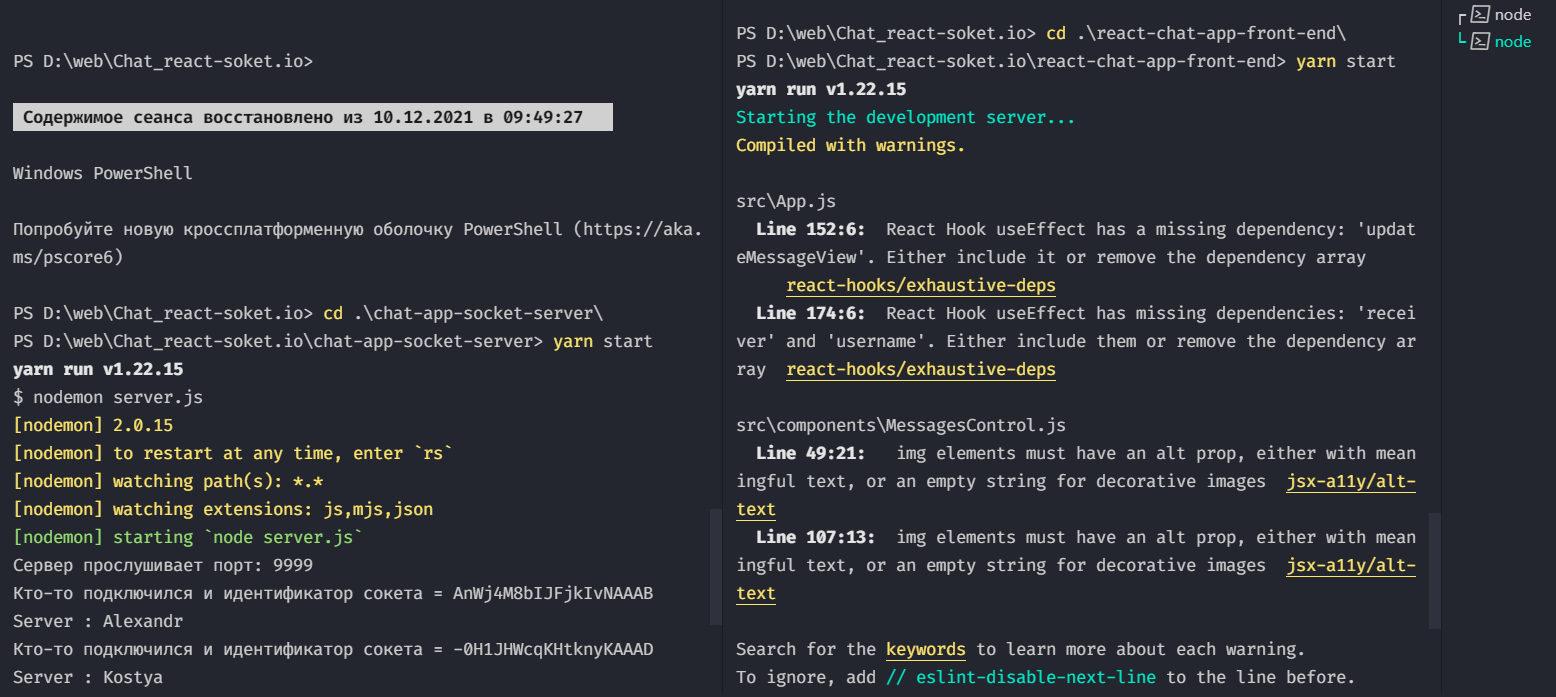
2. Socket.io-client – сокеты для клиента

3. Jest-dom – добавляет пользовательские сопоставители jest для утверждения на узлах DOM

Демонстрация работы

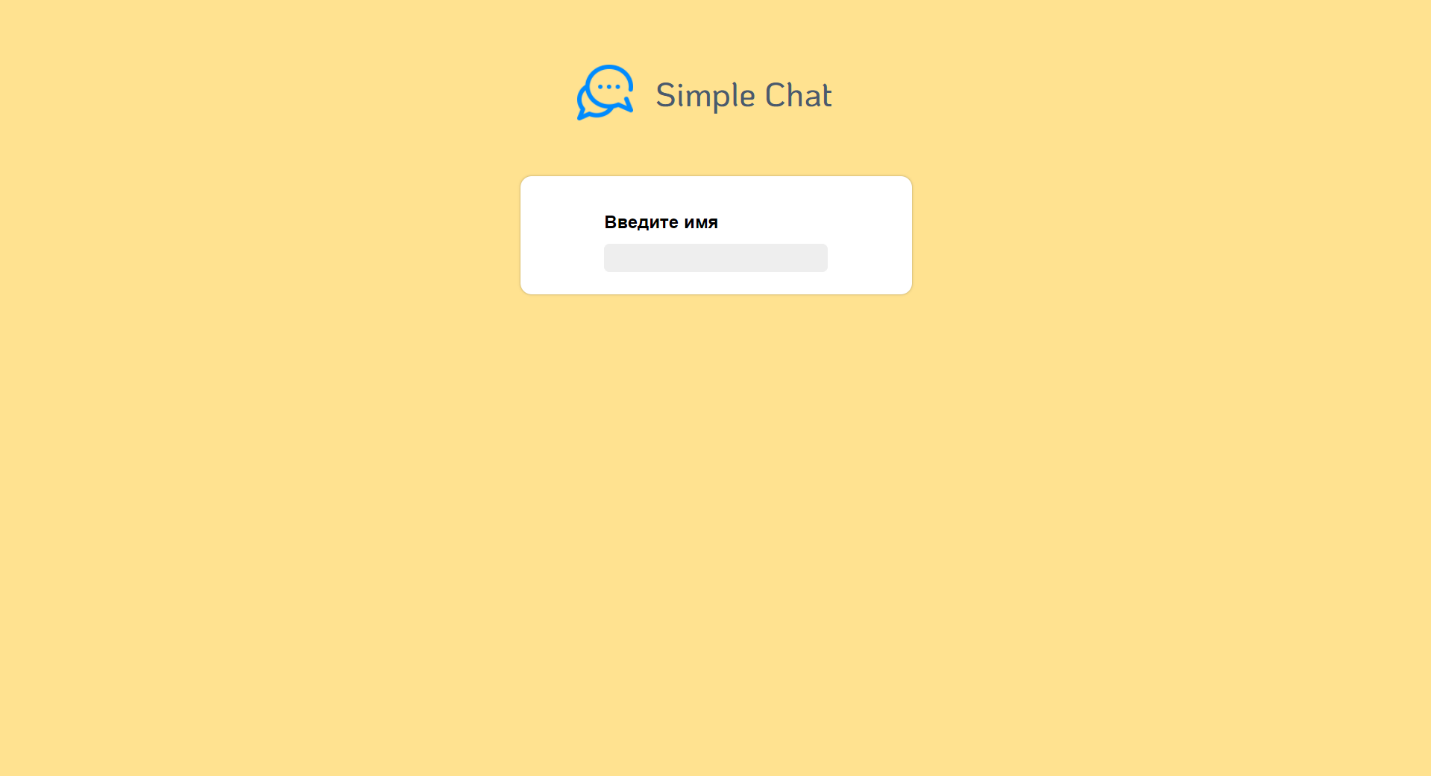
После того как вы напишете в терминале “yarn start” или “npm start” вас перекинет в браузер и откроется чат

На скриншоте ниже показано, как запускается проект, что после подключения клиента к серверу создается индификатор сокета и выводится введное имя пользователем.



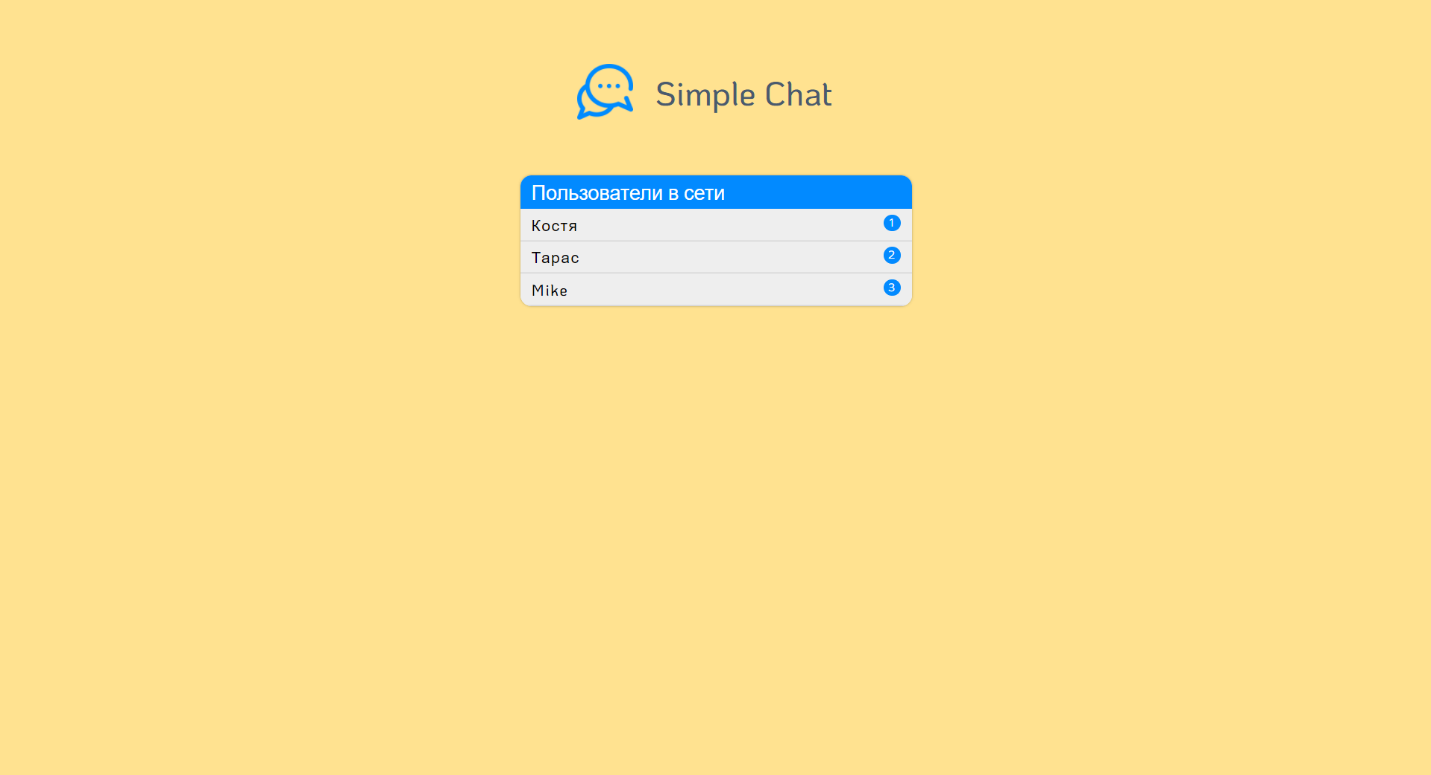
Начальная страница чата:

При вводе имени может произойти такое, что человек попытается войти с именем, которое есть у другого пользователя. В таком случае он зайдет в аккаунт пользователя с таким же именем.



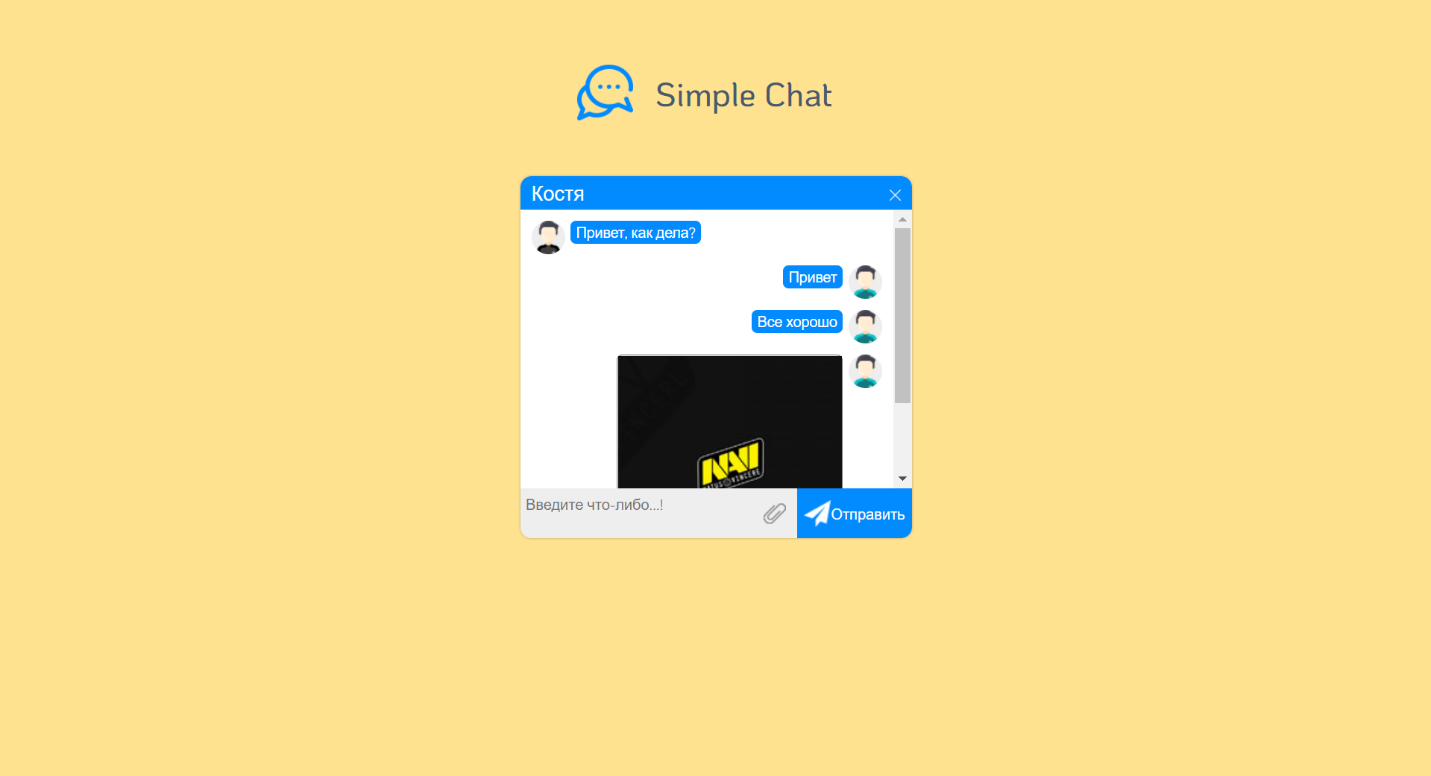
После ввода имени вы переходите в комнату, где показаны пользователи в сети.

Можно выбрать любого пользователя в сети и написать ему сообщение, если пользователь не находится с вами в чате, но вы ему написали сообщение ему приходит уведомление в виде:

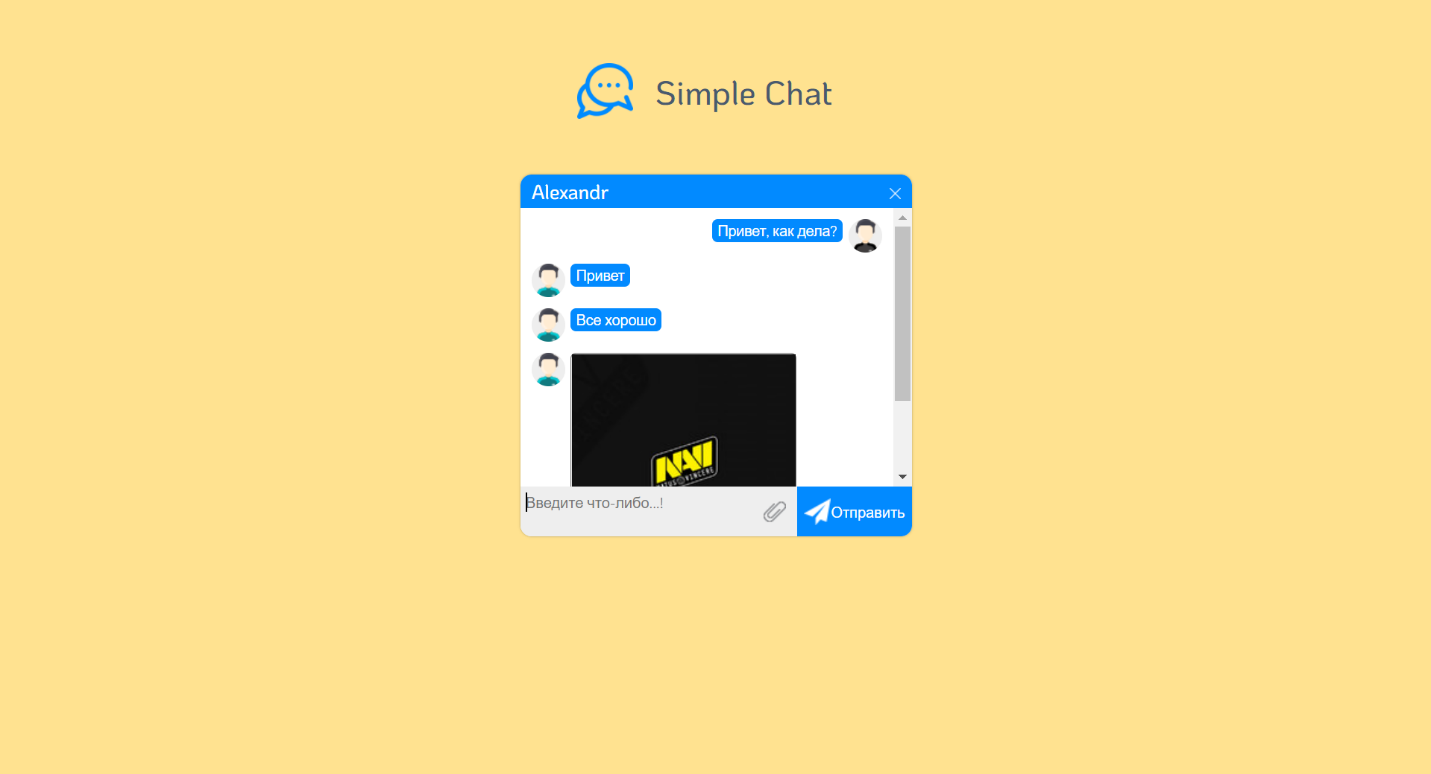


Приватный чат выглядит так

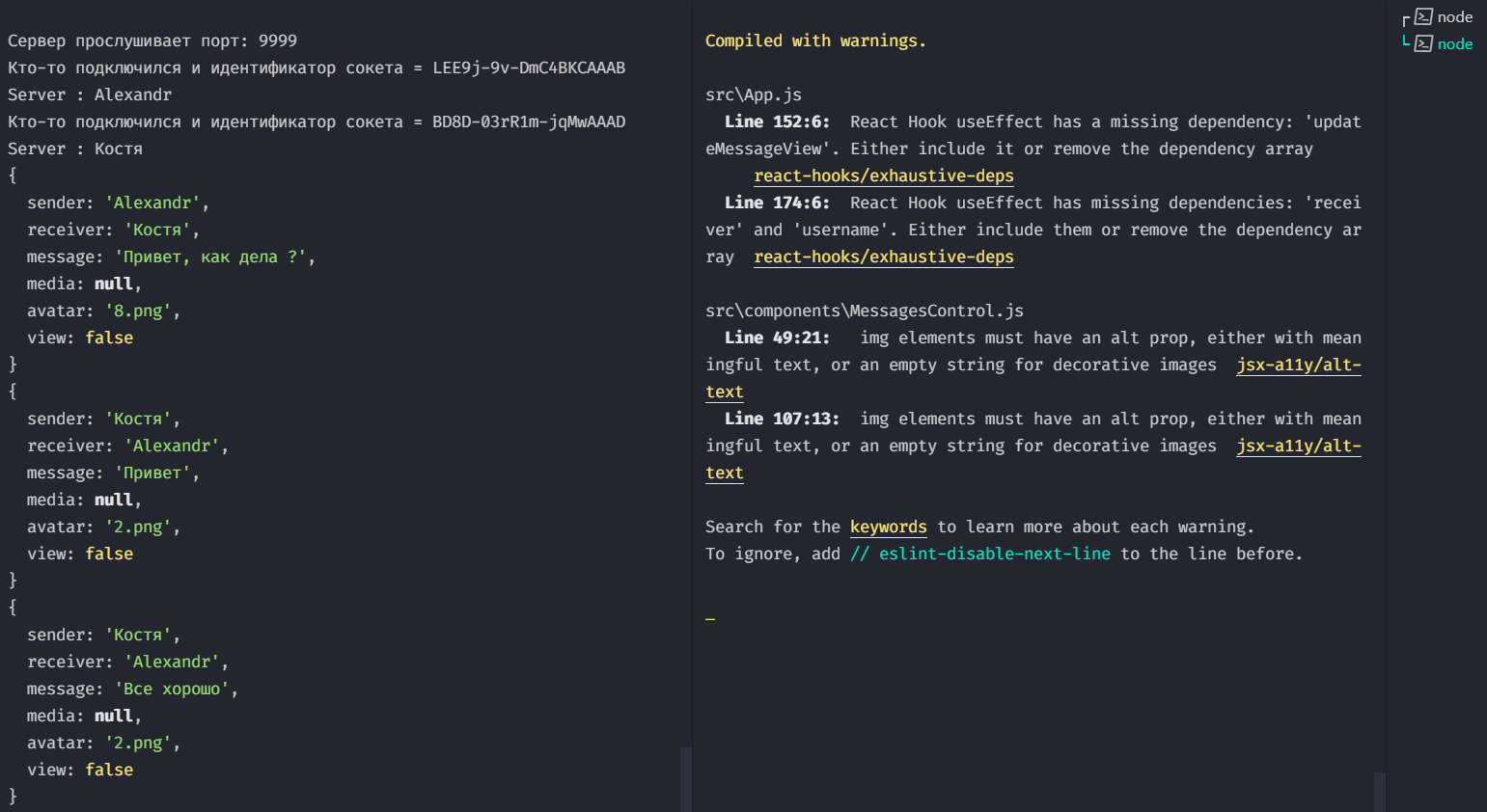
Со стороны Alexandr:



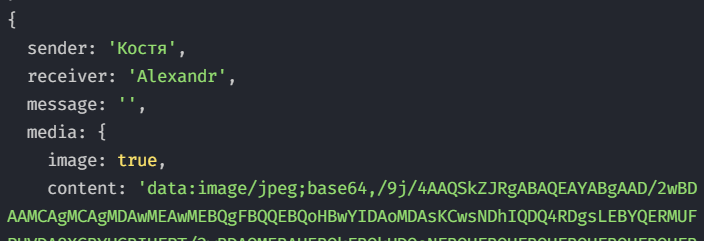
Со стороны Кости:

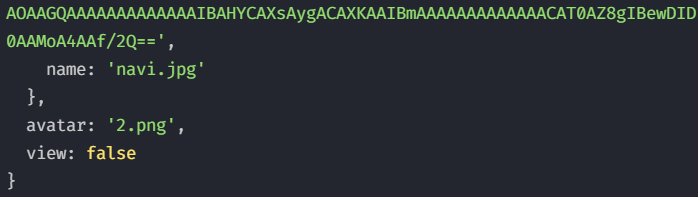


Терминал при отправке сообщений выглядит так:



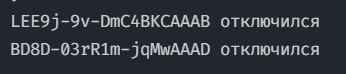
Отправка файла выглядит в терминале так:





Я первый раз пытался передать файл в чате, поэтому с отправкой файлов могут быть баги.

Если пользователь прекращает соединение с чатом в терминале выводится текст о том, что он отключился



Чтобы закончить работу чата нужно нажать сочетания клавиш “ctrl+c” в терминалах.