Aula 11: Aplicativo de chat com Websockets

**Importante!** Ao longo do curso, você encontrará aulas fortemente práticas. Isso significa que a maioria dos slides reflete as etapas que o professor seguirá ao longo da aula. Continue com toda a atenção e siga o professor no código!

aplicativo de bate-papo do websocket

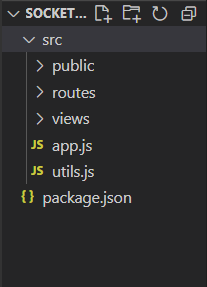
Como aprendemos na última aula, os aplicativos de websocket são bastante extensos. Uma das melhores maneiras de entender seu aplicativo é participando de um bate-papo da comunidade.

Nosso bate-papo da comunidade contará com:

* Uma visão que tem um formulário para se identificar. O usuário poderá escolher o nome de usuário com o qual aparecerá no chat.
* Uma caixa de entrada na qual o usuário pode digitar a mensagem.
* Um painel onde todos os usuários conectados podem visualizar as mensagens em tempo real
* Assim que este aplicativo for desenvolvido, enviaremos nosso código para glitch.com, para que todos possam usá-lo.

## Vamos começar o projeto *do zero*

1. **Tenha um servidor Express pronto**

Para trabalhar com websockets no Express, precisamos de um servidor para que eles funcionem juntos, então vamos construir um servidor expresso como já conhecemos.

Usaremos a mesma estrutura do template trabalhado com Handlebars, então devemos ter a arquitetura

A estrutura inicial deve ser a indicada na imagem:

1. **Faça as instalações**

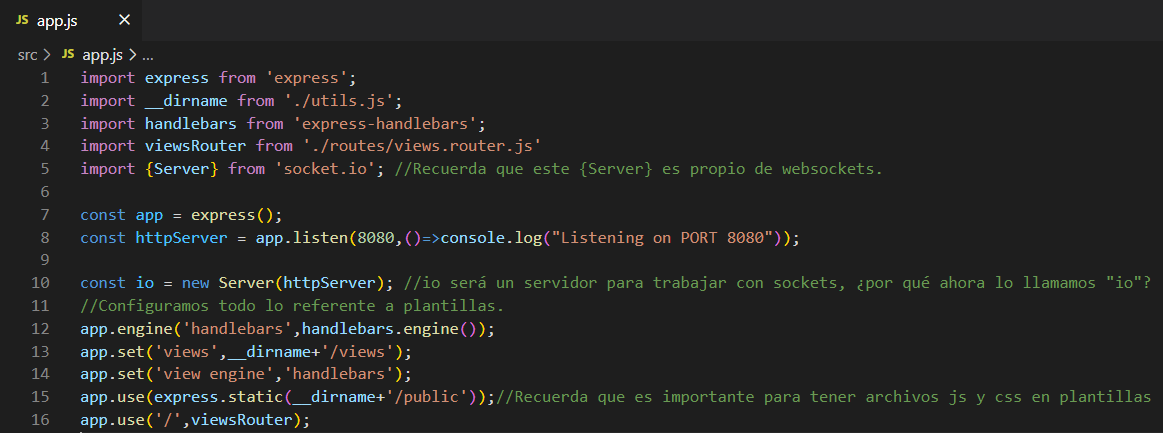
Assim que tivermos a estrutura inicial de pastas, instalaremos nossos elementos cruciais para trabalhar com websockets.

* **express:** Nosso servidor principal.
* **express-handlebars:** Para os templates onde colocaremos o socket do lado do cliente.
* **socket.io:** Para trabalhar com websockets, tanto para cliente quanto para servidor.



**3. Configurando nosso servidor expresso com Handlebars + socket.io**

1. **app.js**

****

**Importante!** Você provavelmente está pensando: “Por que o socketServer agora é chamado de 'io'?”

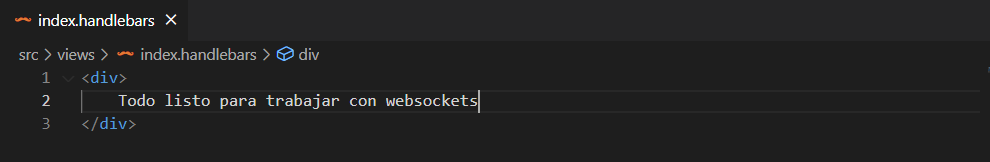
O nome, como você sabe, não influencia em nada, queremos apenas mostrar o nome com o qual é recomendável nomeá-lo "por convenção". Ou seja, queremos que você se acostume com o nome que encontrará em outros projetos.

1. **utils.js**

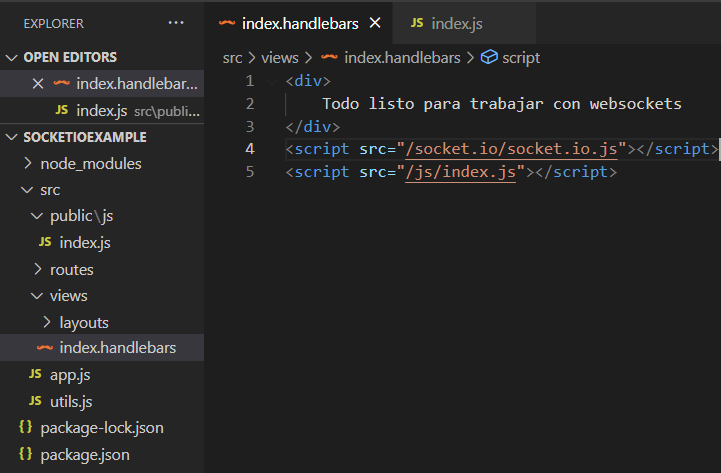
(válido apenas se trabalharmos com o tipo: módulo)



1. **index.handlebars**

****

**4. Adicione js à pasta pública e ao nosso index.handlebars**

Criamos um index.js na pasta public/js/ e o referenciamos em nosso index.handlebars (linha 5)

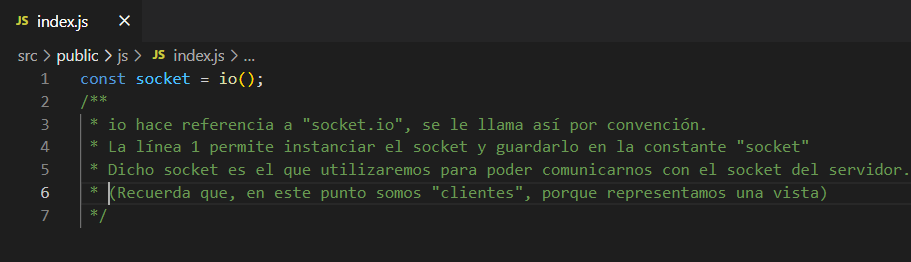
Como mencionamos, o cliente também precisa instanciar seu socket, então colocamos em um script com a sintaxe indicada na imagem. (linha 4)

**O script de soquete deve sempre vir antes do próprio script**

**5. Aumente nosso soquete do lado do cliente em index.js**

Depois de importar o socket.io de nosso script do lado do cliente, podemos testá-lo em nosso arquivo index.js.

Neste arquivo index.js é onde teremos o socket/cliente para conectar com o socket/servidor



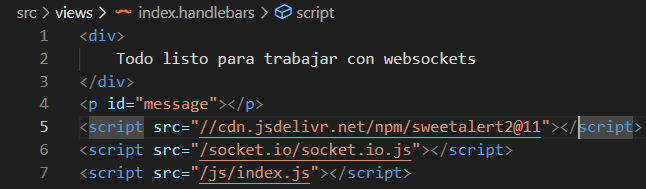
# Começamos com as configurações: Sweetalert2



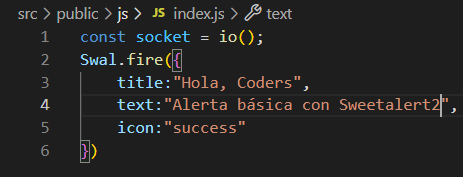
Sweetalert2 nos permitirá usar alertas mais estéticos e com mais funcionalidades. Neste caso, vamos usá-lo para duas coisas particulares:

* Para bloquear a tela de bate-papo até que o usuário faça login
* Para notificar os usuários quando alguém se conecta ao chat

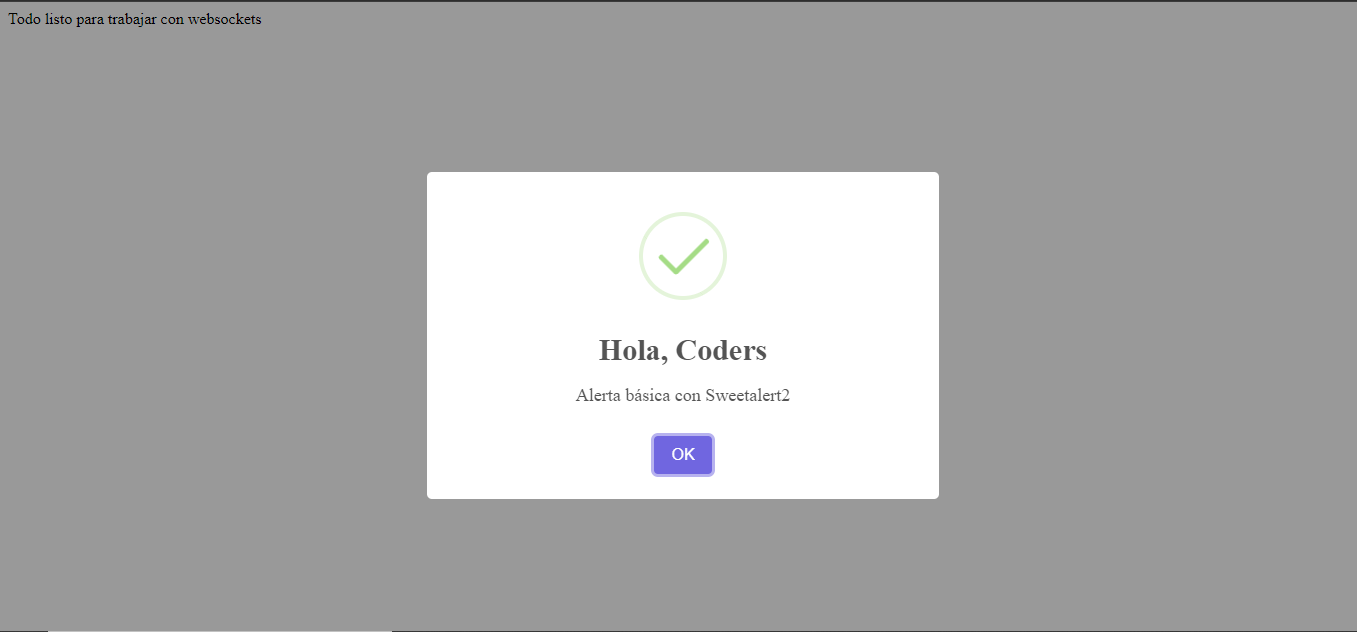
1. **Instalamos o Sweetalert em nossa view “index.handlebars”**



1. **Usamos o objeto “Swal” em nosso index.js**



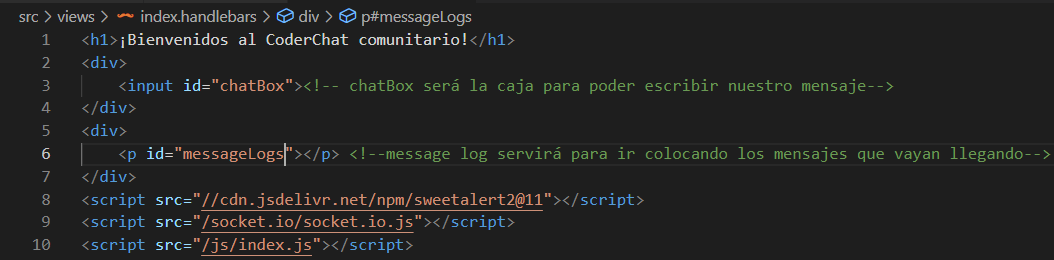
**Resultado: um alerta apresentável, profissional e personalizável**

****

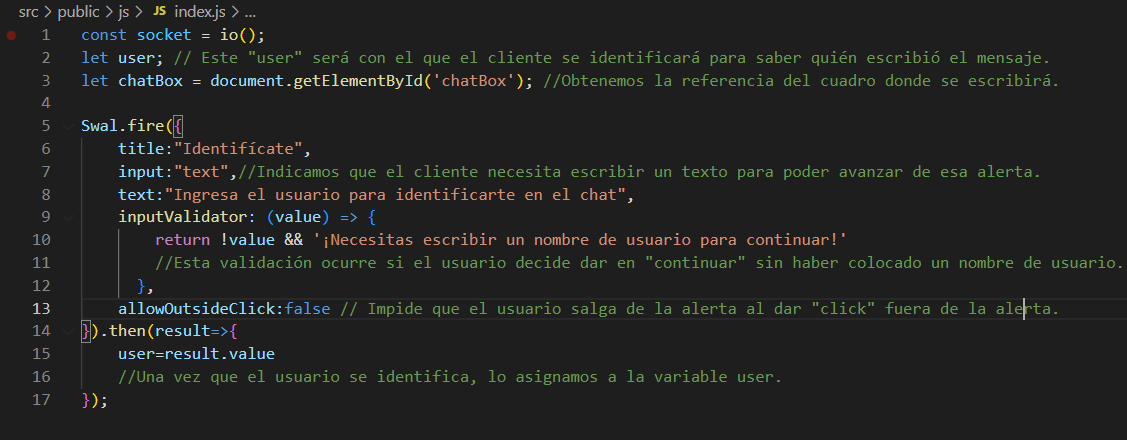
## Visualização de autenticação com Sweetalert2

**Mudando nosso index.handlebars**

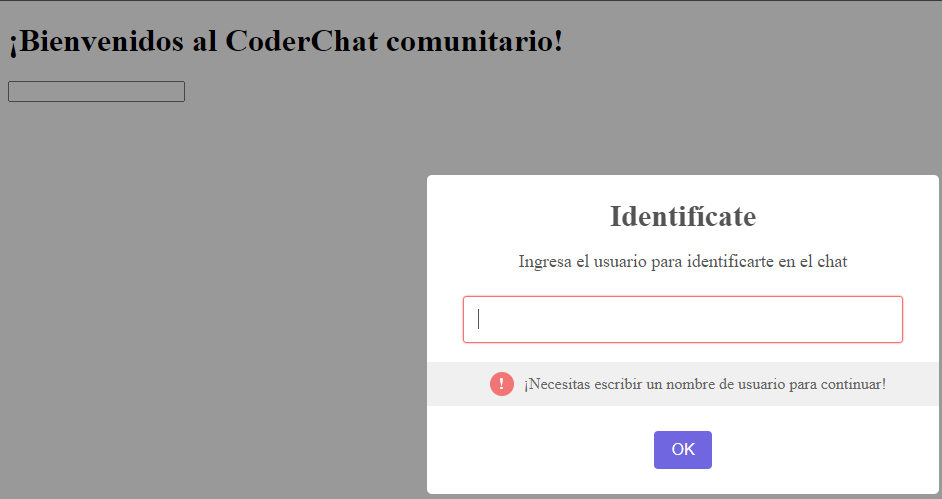
Configuramos nosso arquivo index.handlebars para que desta vez ele não apenas nos cumprimente. Colocaremos uma mensagem de boas-vindas, mas colocaremos dois rótulos adicionais.

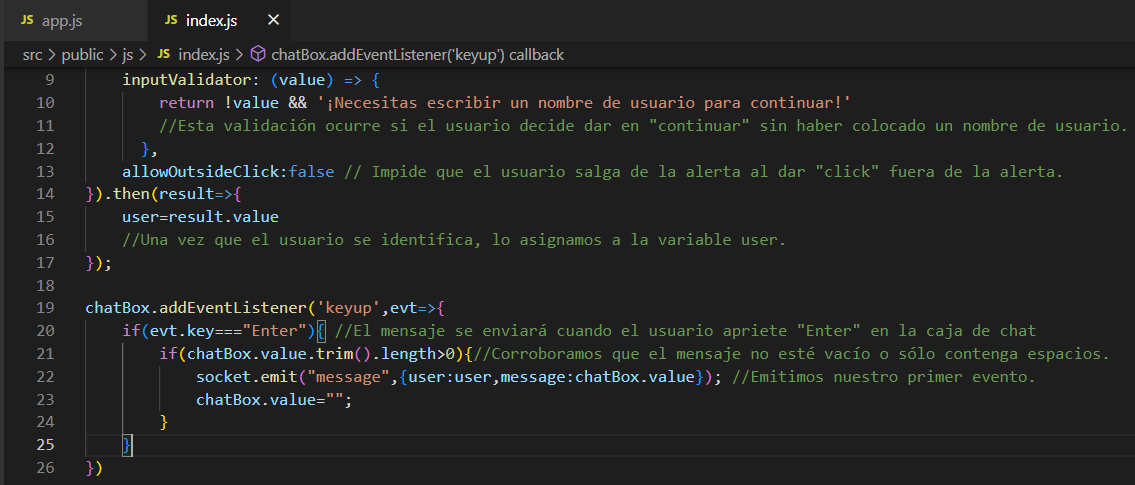


**Reestruturamos nosso index.js para um sistema de autenticação com Sweetalert2**

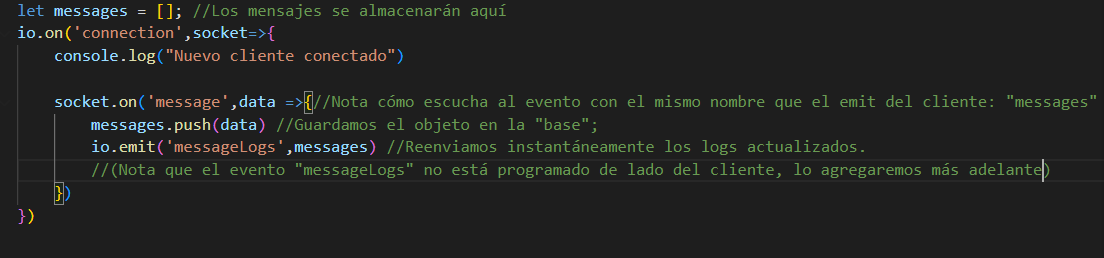


**Veja se não quero me identificar**

****

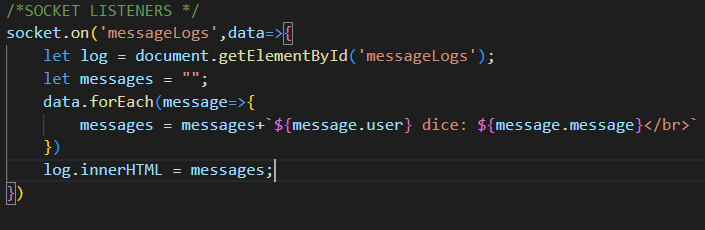
**Adicionamos ao nosso chatBox o evento socket**

**Recebemos o evento “mensagem”, armazenamos a mensagem no array e instantaneamente encaminhamos a mensagem para todos os usuários conectados**

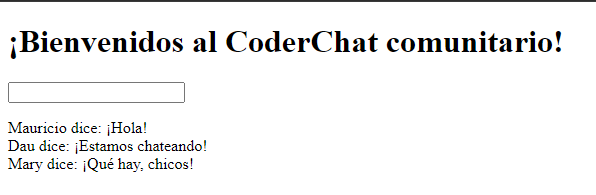


**Configuramos index.js novamente para adicionar o Listener do evento "messageLogs" e exibi-lo na página**

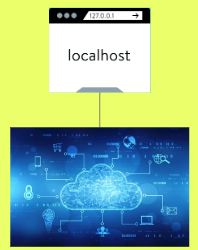
Colocamos em nosso próprio index.js um "listener" para ouvir o evento messageLogs e o mostramos no rótulo que referenciamos para colocar os logs



**Resultado testando o chat com três navegadores diferentes**



# Nossa primeira implantação com Glitch.com

Implantar?

* Fazer "Deploy" de nosso aplicativo significa que ele pode ser acessado para uso geral por todos os usuários que tenham acesso ao link onde reside nosso servidor.
* Atualmente nosso aplicativo vive em **localhost** , que, como o nome sugere, é **apenas local, apenas para mim** .
* Uma vez implantado nosso aplicativo, ele ficará na nuvem e poderemos compartilhá-lo com quem quisermos.

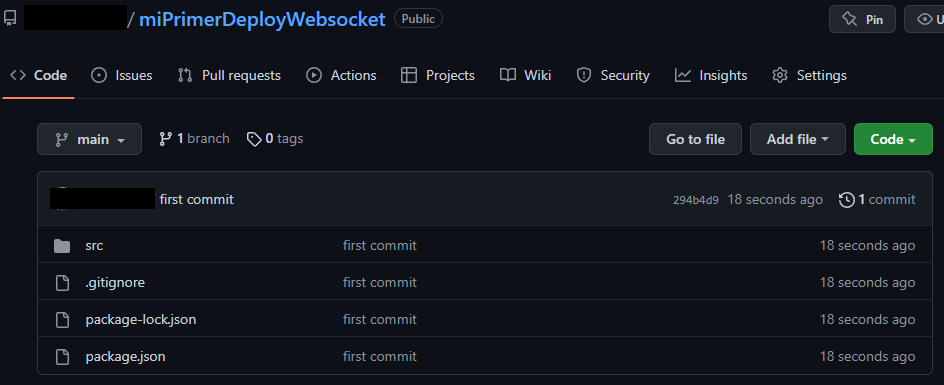
Glitch. com?

* Glitch.com é uma plataforma para implantar nossos aplicativos.
* O Glitch.com foi construído com base no conceito de tentar tornar a experiência de implantação de um aplicativo cada vez mais fácil.
* Embora não seja o mais recomendado para implantações profissionais, ele nos ajudará a entender o conceito e aplicá-lo em nosso bate-papo.
* Podemos conectá-lo ao nosso repositório Github.

## Processo de implantação do nosso aplicativo

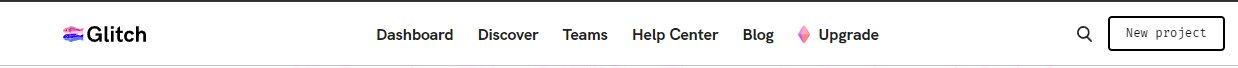
1. **Carregue nosso código em um repositório do Github**

O Glitch.com, como qualquer outro sistema de implantação decente, é muito amigável para os gerentes de repositório, então a melhor maneira de entregar nosso projeto ao Glitch.com é se ele já estiver em um repositório.

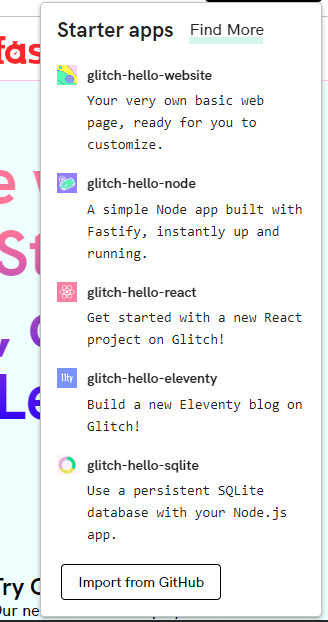
* O repositório deve ser público para que Glitch.com possa acessá-lo.
* Lembre-se de não carregar os node\_modules
* Lembre-se de incluir o script “start” em package.json

**2. Crie uma conta em Glitch.com**

O processo é bastante simples, uma vez finalizado o processo de registro e com nossa conta glitch.com pronta, veremos o seguinte painel:

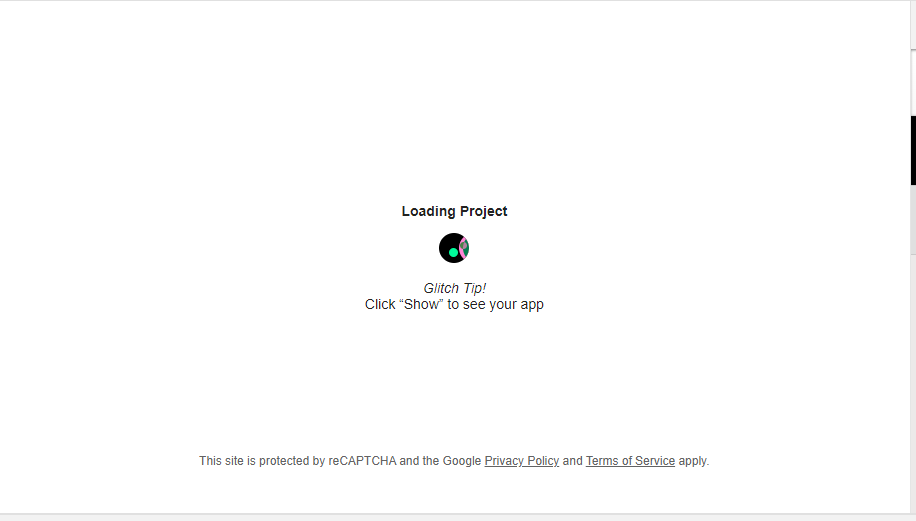


**3. Crie um novo projeto no Github**

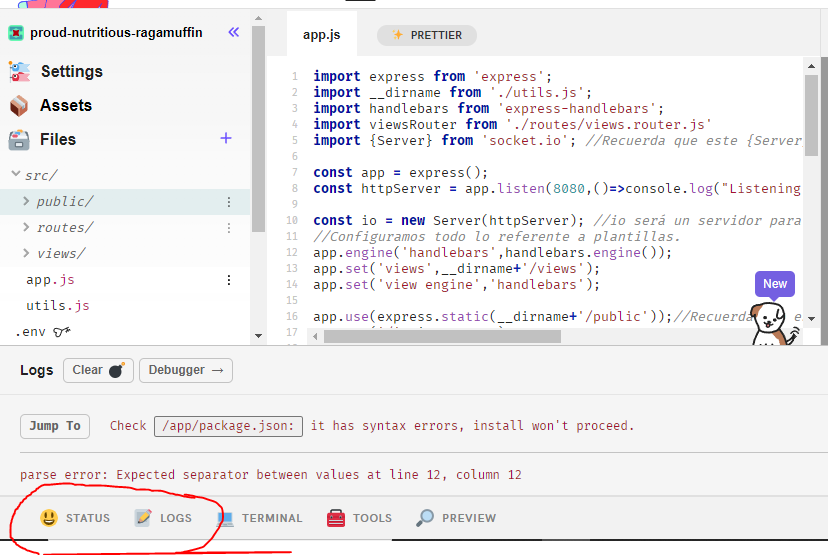
Clicar em "novo projeto" exibirá uma lista de modelos de projeto. Vamos selecionar o botão abaixo de “Importar do Github”.

Posteriormente, colocaremos o link que aponta para o nosso repositório Github.

**Glitch.com começará a criar o projeto e ajustá-lo**

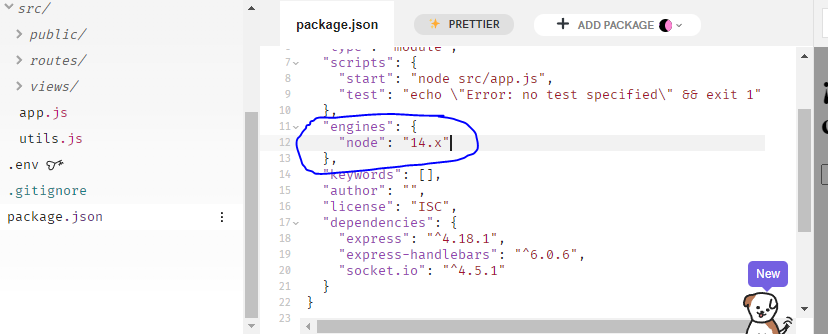


**Verifique se o status está OK, caso contrário, verifique os LOGS**

****

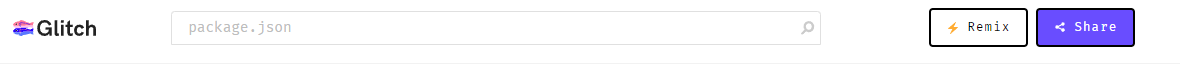
**Talvez tenha havido um engano...**

Se os logs mostrarem um erro, provavelmente é porque o núcleo do nodejs não está correto, tente alterar a versão do mecanismo do nodejs para 14.x ou 15.x do package.json



**Por fim, compartilhamos o link com quem quisermos**

Assim que o projeto estiver em execução no Glitch.com, clicaremos no botão “ **Compartilhar** ”.

Isso exibirá uma lista de informações, dentro da qual os **links do projeto** contêm o que nos interessa: a opção **Site ao vivo** é o link que fará com que qualquer pessoa a quem você passar o link tenha acesso ao chat em grupo.

