
Newton
Quem se prepara, não para.

Arquitetura de Aplicações Web

5º período

Professora: Michelle Hanne

Sumário

- Websocket
- O que é o NextJS
- O que é o Firebase Google
- Criando Chat com NextJS e Firebase

WebSocket

Com a criação da plataforma Node.js, foi possível que surgissem também servidores web nos mais diversos protocolos de comunicação (HTTP, HTTPS, FTP, dentre vários outros) e dentre um deles, está o WebSocket, um protocolo de comunicação suportado por browsers exatamente com o propósito de estabelecer entre o navegador e o servidor uma comunicação bidirecional e em tempo real, **possibilitando uma troca de mensagens mais ágil sem o refresh de página e um tempo de espera demorado.**

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

WebSocket

Newton
Quem se prepara, não para.

Visando atingir a todos esses problemas, foi criado o módulo do Node.JS, **que cuida de todos os protocolos de transporte que podem servir de plano B para os navegadores mais antigos.** Que seriam os seguintes (nesta ordem): Adobe Flash Socket, AJAX long polling, AJAX multipart streaming, Forever iframe e o JSONP Polling.

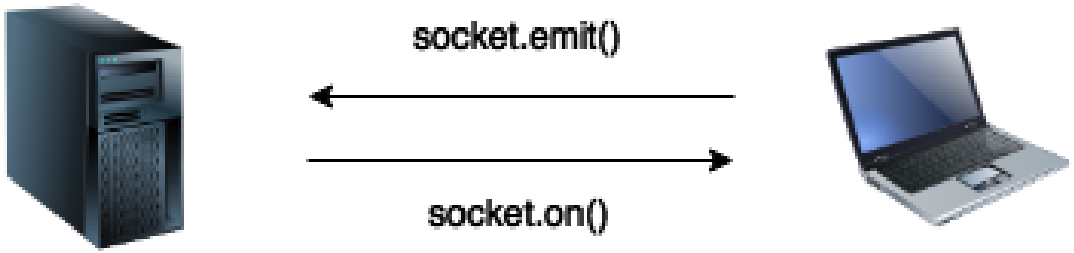
O próprio modulo fica a cargo de realizar a comunicação com o servidor pelo protocolo de transporte que lhe for mais conveniente.

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

WebSocket

Newton
Quem se prepara, não para.

Este módulo do Node.js traz uma forma de conexão direta do browser do cliente com o servidor de aplicação**. A biblioteca funciona através de eventos, ou seja, o servidor ou o cliente irão disparar eventos para que haja respostas de uma das partes, veja uma exemplificação na Figura 1.**



WebSocket

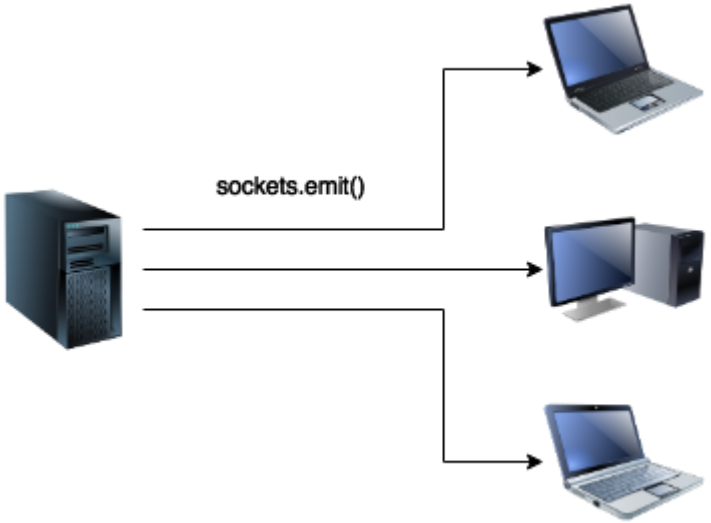
Newton
Quem se prepara, não para.

De certa forma, vamos usar dois métodos muitos básicos do módulo, que são o **emit** e o **on**. Um serve para efetuar a emissão do evento e o outro para receber a resposta do mesmo. Cada um dos lados da aplicação, portanto, terão a **biblioteca Socket.IO** adicionada.

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

WebSocket

Além de permitir a troca direta de mensagens entre dois dispositivos, o Socket.IO também permite o broadcast de mensagens, o envio de um evento a todos os outros usuários conectados. O broadcast pode ser tanto do cliente quanto do servidor, conforme demonstrado na Figura 2.



Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

WebSocket

Newton
Quem se prepara, não para.

Quando o usuário acessar a página, um socket é criado com o servidor e é através deste socket que é realizada a troca de mensagens entre um cliente e um servidor. **Este, por sua vez, pode tanto emitir um evento para um único Socket como para todos os sockets conectados a ele, o que chamamos de broadcast de mensagem.**

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Inicializando o projeto

Newton
Quem se prepara, não para.

Instalar o node.js
Instalar o npm

Criar uma pasta: **app-chat**
Inicializar: npm init –y
Criando o projeto com o express
npm i express

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Criando projeto

Newton
Quem se prepara, não para.

1- Criando o arquivo app.js (servidor)

```
var app = require('http').createServer(resposta);
var fs = require('fs');

app.listen(3000);
console.log("Aplicação está em execução...");
function resposta (req, res) {
     fs.readFile(__dirname + '/index.html',
     function (err, data) {
         if (err) {
              res.writeHead(500);
              return res.end('Erro ao carregar o arquivo index.html');
         }

         res.writeHead(200);
         res.end(data);
     });
}
```

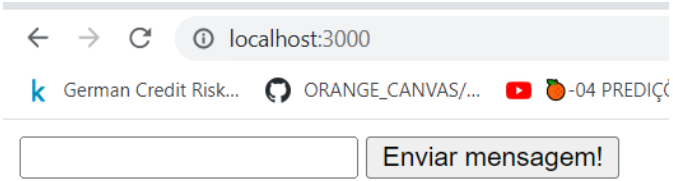
Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Criando o projeto

Newton
Quem se prepara, não para.

2- Criar o arquivo index.html

```
<!DOCTYPE html>
<html>
<head>
     <title>ChatJS - FrontEnd Magazine - DevMedia</title>
     <link rel="stylesheet" type="text/css"
href="/css/style.css" />
</head>
<body>
     <div id="historico_mensagens"></div>
     <form id='chat'>
         <input type='text' id='texto_mensagem'
name='texto_mensagem' />
         <input type='submit' value='Enviar mensagem!' />
     </form>
</body>
</html>
```



3- Testar o funcionamento do servidor: **node app.js**

Criando o projeto

Newton
Quem se prepara, não para.

4- Criar o arquivo style.css

```
<!DOCTYPE html>
<html>
<head>
     <title>ChatJS - FrontEnd Magazine - DevMedia</title>
     <link rel="stylesheet" type="text/css"
href="/css/style.css" />
</head>
<body>
     <div id="historico_mensagens"></div>
     <form id='chat'>
         <input type='text' id='texto_mensagem'
name='texto_mensagem' />
         <input type='submit' value='Enviar mensagem!' />
     </form>
</body>
</html>
```

Criando o projeto

5- Alterar o arquivo app.js

Vamos alterar o arquivo app.js para que ele carregue os arquivos **que são passados na URL da solicitação**, ao invés de colocarmos cada uma das URLs manualmente.

```
var app = require('http').createServer(resposta);
var fs = require('fs');

app.listen(3000);
console.log("Aplicação está em execução...");

function resposta (req, res) {
     var arquivo = "";
     if(req.url == "/"){
          arquivo = __dirname + '/index.html';
     }else{
          arquivo = __dirname + req.url;
     }
     fs.readFile(arquivo,
          function (err, data) {
               if (err) {
                    res.writeHead(404);
                    return res.end('Página ou arquivo não
encontrados');
               }

               res.writeHead(200);
               res.end(data);
          }
     );
}
```

Enviando Mensagens

Newton
Quem se prepara, não para.

Agora temos o servidor funcionando, o estilo CSS na nossa página e toda a estrutura HTML pronta. Vamos a partir de agora trabalhar na função de envio de mensagens.

Nossa aplicação vai funcionar se comunicando com o servidor node através da biblioteca client-side do Socket.IO com o jQuery fazendo a interação com a página.

Para isso vamos alterar o arquivo **app.js,** incluir uma linha de um comando **require logo no começo do arquivo informando que estamos incluindo na aplicação o Socket.IO e que o módulo será armazenado na variável socket.**

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Enviando Mensagens

Newton
Quem se prepara, não para.

6- Acrescentando as variáveis que serão utilizadas no projeto no arquivo app.js.

```
var app = require('http').createServer(resposta);
var fs = require('fs');
var io = require('socket.io')(app);
```

Podemos ver que o require claramente chama o módulo socket.io e estamos passando a variável app (referente ao nosso servidor)

7- Para funcionar temos que instalar as dependências do socket.io

npm install socket.io

Enviando Mensagens

Newton
Quem se prepara, não para.

7- O jQuery facilitará o nosso processo de desenvolvimento e o **Socket.IO** para o client-side. Acesse o arquivo index.html e altere:

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
```

A biblioteca Socket.IO vai entender este caminho automaticamente e irá trazer para nossa aplicação a biblioteca client-side por si só.

Enviando Mensagens

Newton
Quem se prepara, não para.

7- Primeiramente, vamos abrir uma **tag script (desta vez sem o atributo src)** onde estará todo o nosso código do lado cliente da nossa aplicação. Para começar temos que programar um evento de envio de mensagem, criando assim uma função que será atrelada ao submit do formulário de mensagem:

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
<script type="text/javascript">
        var socket = io.connect();

        $("form#chat").submit(function(e){
            e.preventDefault();
            socket.emit("enviar mensagem", $(this).find("#texto_mensagem").val(), function(){
                $("form#chat #texto_mensagem").val("");
            });
        });

</script>
</body>
</html>
```

Enviando Mensagens

Newton
Quem se prepara, não para.

No código JavaScript da página declaramos uma variável socket que é referente à biblioteca Socket.IO, que será responsável por todas as funcionalidades do socket. A seguir declaramos um evento submit do nosso formulário em jQuery e passamos um preventDefault< para que o formulário não prossiga ao action do formulário, já que nós é quem vamos cuidar da resposta do formulário.

Em seguida, podemos ver que é invocado o método emit da biblioteca, no qual passamos como parâmetros três coisas: o nome do evento (isso será útil no servidor), os dados que estamos emitindo (no caso só estamos enviando o conteúdo do campo mensagem) e por último o call-back, uma função que vai ser executada assim que o evento for emitido. Este último, em específico, servirá apenas para limpar o campo de mensagem, assim o usuário não tem que ficar excluindo a mensagem depois que mandá-la.

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Recebendo Mensagens no Servidor

Newton
Quem se prepara, não para.

8- Para isso, edite o arquivo app.js colocando o código mostrado no fim do mesmo.

```
io.on("connection", function(socket){
    socket.on("enviar mensagem", function(mensagem_enviada, callback){
        mensagem_enviada = "[ " + pegarDataAtual() + " ]: " + mensagem_enviada;

        io.sockets.emit("atualizar mensagens", mensagem_enviada);
        callback();
    });
});
function pegarDataAtual(){
 var dataAtual = new Date();
 var dia = (dataAtual.getDate()<10 ? '0' : '') + dataAtual.getDate();
 var mes = ((dataAtual.getMonth() + 1)<10 ? '0' : '') + (dataAtual.getMonth() + 1);
 var ano = dataAtual.getFullYear();
 var hora = (dataAtual.getHours()<10 ? '0' : '') + dataAtual.getHours();
 var minuto = (dataAtual.getMinutes()<10 ? '0' : '') + dataAtual.getMinutes();
 var segundo = (dataAtual.getSeconds()<10 ? '0' : '') + dataAtual.getSeconds();

 var dataFormatada = dia + "/" + mes + "/" + ano + " " + hora + ":" + minuto + ":" + segundo;
 return dataFormatada;
}
```

Recebendo Mensagens no Servidor

Newton
Quem se prepara, não para.

Criamos um método que atuará em resposta à conexão do cliente ao servidor. Quando o cliente acessa a página ela dispara este método no servidor e **quando este socket receber um método Enviar Mensagem acionamos um método que tem como parâmetros os dados enviados** (o campo mensagem) e o call-back que criamos no lado cliente.

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Recebendo Mensagens no Servidor

Newton
Quem se prepara, não para.

Dentro deste método colocamos a segunda parte da funcionalidade: o módulo vai emitir para todos os sockets conectados com o servidor (todos os usuários, por assim dizer) o evento *Atualizar Mensagens* e passará também qual mensagem nova foi enviada, com uma formatação de data e hora entre colchetes. Para fornecer a data e hora criamos uma função a parte porque ainda utilizaremos este método mais algumas vezes ao longo do desenvolvimento. Logo em seguida chamamos o call-back que criamos no lado cliente, que é o método para limpar os campos.

Recebendo Mensagens no Servidor

Newton
Quem se prepara, não para.

9- Finalmente, edite também o arquivo **index.html** e **crie o método que vai atualizar as mensagens para os usuários.** A ideia é bem simples: vamos dar um **append** na **div historico_mensagens.** As linhas a seguir devem ser inseridas logo depois do processamento do **submit** do formulário.

```
io.on("connection", function(socket){
    socket.on("enviar mensagem", function(mensagem_enviada, callback){
        mensagem_enviada = "[ " + pegarDataAtual() + " ]: " + mensagem_enviada;

        io.sockets.emit("atualizar mensagens", mensagem_enviada);
        callback();
    });
});
function pegarDataAtual(){
 var dataAtual = new Date();
 var dia = (dataAtual.getDate()<10 ? '0' : '') + dataAtual.getDate();
 var mes = ((dataAtual.getMonth() + 1)<10 ? '0' : '') + (dataAtual.getMonth() + 1);
 var ano = dataAtual.getFullYear();
 var hora = (dataAtual.getHours()<10 ? '0' : '') + dataAtual.getHours();
 var minuto = (dataAtual.getMinutes()<10 ? '0' : '') + dataAtual.getMinutes();
 var segundo = (dataAtual.getSeconds()<10 ? '0' : '') + dataAtual.getSeconds();

 var dataFormatada = dia + "/" + mes + "/" + ano + " " + hora + ":" + minuto + ":" + segundo;
 return dataFormatada;
}
```

Fonte: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Continuação: Atribuindo nome aos usuários

Disponível no link: https://www.devmedia.com.br/como-criar-um-chat-com-node-js/33719

Firebase Google – Criando um Projeto

Newton
Quem se prepara, não para.

O que é o Firebase de Google?

- O Firebase de Google é uma plataforma digital utilizada para facilitar o desenvolvimento de aplicativos web ou móveis, de uma forma efetiva, rápida e simples. Graças às suas diversas funções, é utilizado como uma técnica de Marketing Digital, com a finalidade de aumentar a base de usuários e gerar maiores benefícios econômicos.

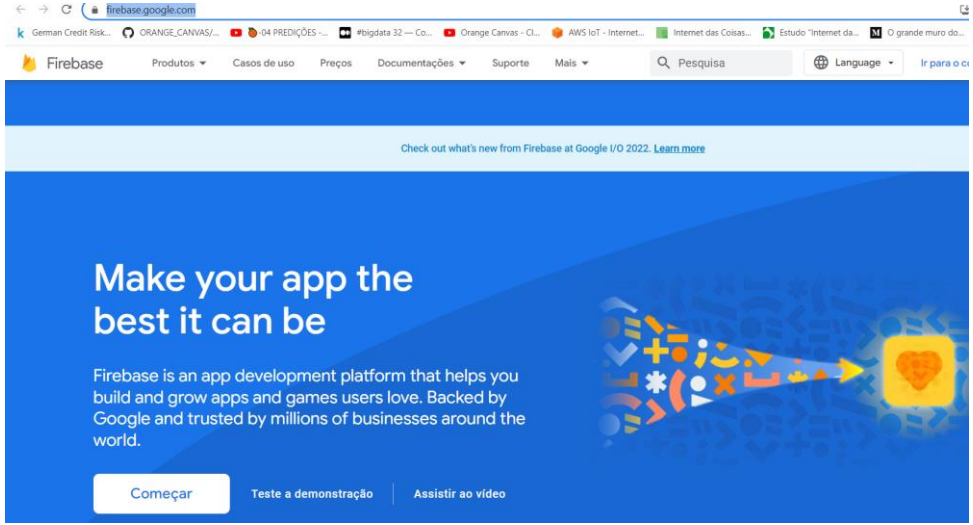
- Seu principal objetivo é melhorar o rendimento dos apps mediante a implementação de diversas funcionalidades que farão do aplicativo um instrumento muito mais maleável, seguro e de fácil acesso para os usuários.

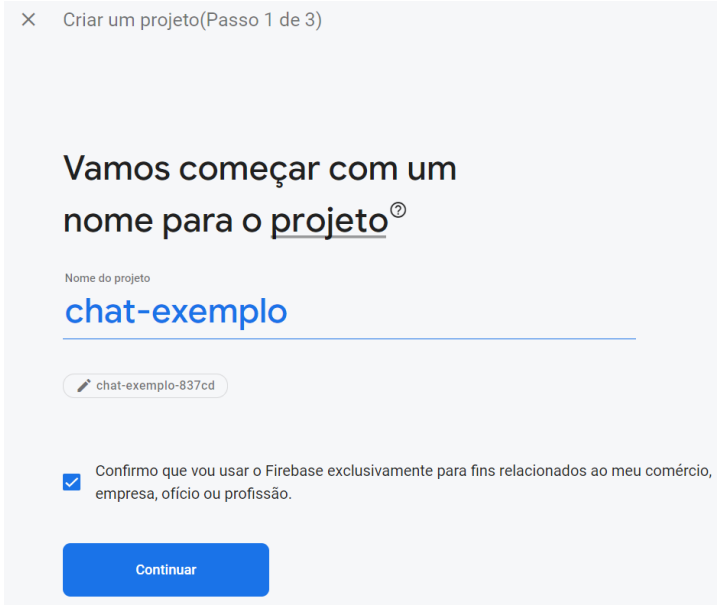
Fonte: https://rockcontent.com/br/blog/firebase/

Firebase Google – Criando um Projeto

Newton
Quem se prepara, não para.

Acesse o site do Firebase — https://firebase.google.com/— e clique no botão “Começar”, em seguida “Criar um Projeto”

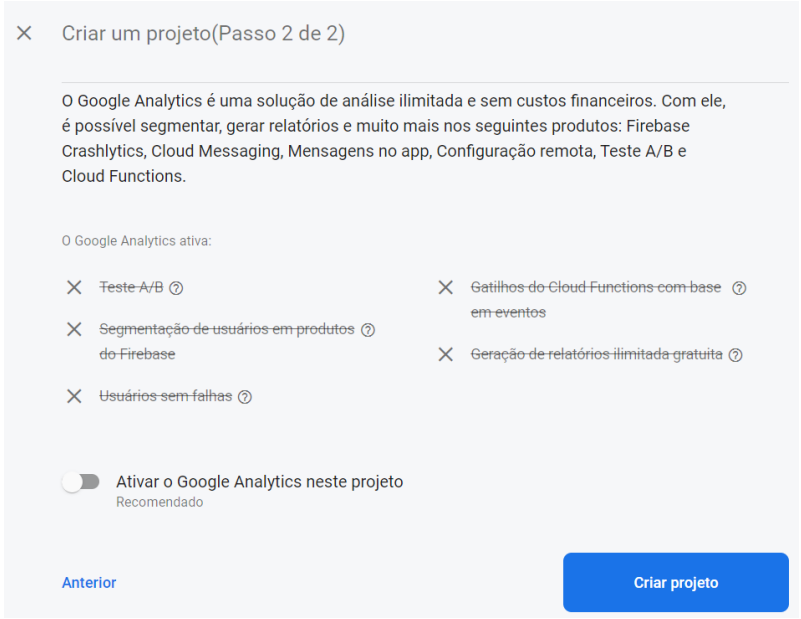




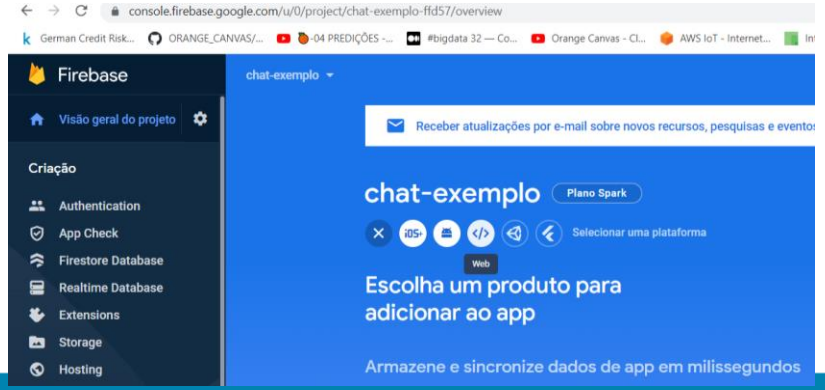
Fonte: https://medium.com/@adsonrocha/como-criar-um-projeto-no-firebase-273b984ea1b3

Firebase Google – Criando um Projeto

Newton
Quem se prepara, não para.



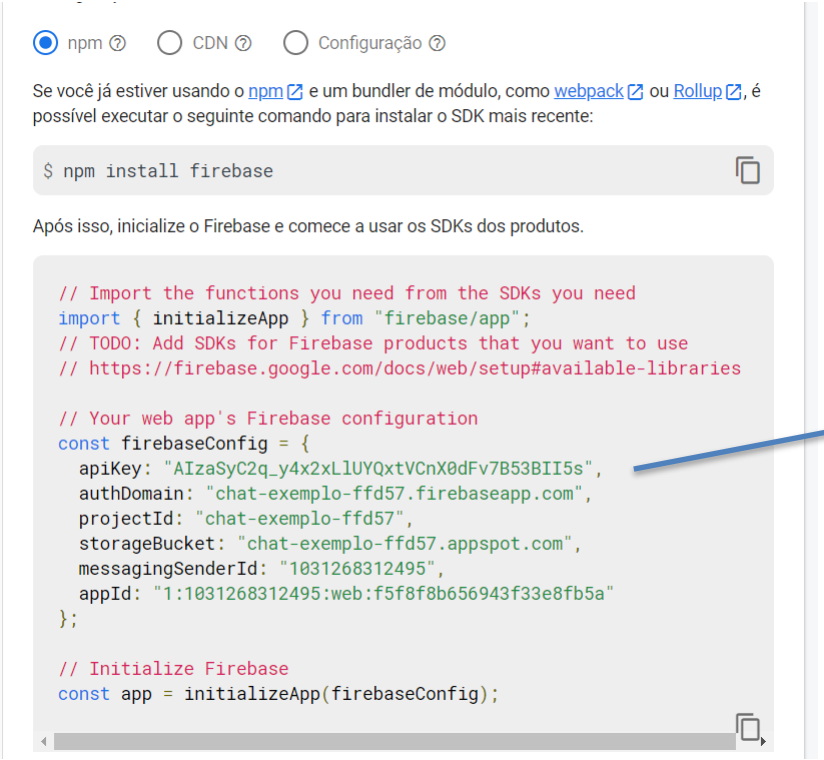
Após a conclusão do Projeto, será redirecionado para o painel de visão geral do seu novo projeto, na lateral esquerda há um menu com os serviços oferecidos pelo Firebase. Na parte central escolha a opção: “Adicionar o Firebase ao seu aplicativo da Web”,



Fonte: https://medium.com/@adsonrocha/como-criar-um-projeto-no-firebase-273b984ea1b3

Firebase Google – Criando um Projeto

Newton
Quem se prepara, não para.



Após a criação do Projeto será possível obter as configurações do Firebase.

Estes códigos serão copiados para o arquivo **index.js do seu projeto.**

Fonte: https://medium.com/@adsonrocha/como-criar-um-projeto-no-firebase-273b984ea1b3

Next.JS

Next.js é um framework para React. O que isso quer dizer? O React é uma biblioteca Javascript para construção de interfaces e o Next é considerado um framework pois adiciona várias funcionalidades em cima do React

https://blog.geekhunter.com.br/o-que-e-next-js/

Deploy na Vercel

Newton
Quem se prepara, não para.

Como fazer Deploy na Vercel:

https://gabrielcordeiro.dev/blog/como-fazer-deploy-na-vercel/

https://vercel.com/docs

Hospedagem Heroku

Newton
Quem se prepara, não para.

https://blog.geekhunter.com.br/heroku/

Fonte: https://blog.geekhunter.com.br/o-que-e-next-js/

Exemplos do Chat

Newton
Quem se prepara, não para.

https://www.cometchat.com/tutorials/how-to-build-a-chat-app-with-nextjs

https://adebola-niran.medium.com/building-a-serverless-realtime-chat-application-with-pusher-and-nextjs-fb92f451c75d

https://blog.logrocket.com/implementing-websocket-communication-next-js/

Fonte: https://blog.geekhunter.com.br/o-que-e-next-js/