

Comunicação com o Backend utilizando o HTTP

Introdução

- A maioria dos aplicativos front-end precisa se comunicar com um servidor através do protocolo HTTP;
- Através de pedidos HTTP, é possível fazer download ou upload de dados e acessar outros serviços do back-end;
- Angular fornece a API HTTP cliente para manipular pedidos HTTP: a classe de serviço **HttpClient** em `@angular/common/http`.

Protocolo HTTP

- Para a API ser RESTFull ela deve ser capaz de lidar com todos esses métodos:
 - GET: para recuperar/buscar dados;
 - POST: para criar ou atualizar dados;
 - PUT: para atualização de dados;
 - DELETE: Para exclusão de dados;
 - PATCH: Atualização de dados;

1.10_restfull\app.ts

<https://jasonwatmore.com/post/2021/09/06/fetch-http-get-request-examples>

Funcionalidades do serviço de cliente HTTP

- A capacidade de solicitar objetos de resposta;
 - `HttpClient.get()` - método para buscar dados de um servidor;
 - Envia uma solicitação HTTP e retorna um `Observable` que emite os dados solicitados quando a resposta é recebida.
- Tratamento de erros simplificado;
 - Se a solicitação falhar no servidor, o `HttpClient` retornará um objeto de erro em vez de uma resposta bem-sucedida.
- Recursos de testabilidade;
- Interceptação de solicitação e resposta (Interceptor);
 - Declaração de interceptadores que inspecionam e transformam solicitações HTTP do seu aplicativo para um servidor;
 - Os mesmos interceptadores também podem inspecionar e transformar as respostas de um servidor no caminho de volta para a aplicação.

HTTP: configuração para comunicação com o servidor

- Antes de poder usar o HttpClient, é preciso importar o HttpClientModule em @angular/common/http no app.module.ts;
- Injetar o serviço HttpClient como uma dependência de uma classe de aplicativo no construtor:

```
constructor(private http: HttpClient) { }
```

Método HttpClient.post()

- Uma solicitação POST é usada para enviar dados ao servidor;
 - Ex: dados de um formulário enviados para um servidor.

```
addTicket(ticketData: {  NomePassageiro: string,  
                           numeroVoo: string,  
                           dataPartida: string,  
                           dataChegada: string,  
                           aeroportoPartida: string,  
                           aeroportoChegada: string }) {  
  
  this.http.post(  
    'https://aula13-3a92f-default-rtdb.firebaseio.com/posts.json',  
    ticketData)  
    .subscribe(responseData => {  
      console.log(responseData);  
    });  
}
```

URL



Método HttpClient.get()

- Uma solicitação GET é usada para recuperar dados do servidor.
- O método `get(url, options)` recebe dois argumentos:
 - A URL do endpoint da string a partir do qual buscar;
 - Um objeto de opções opcionais para configurar a solicitação.

```
getTicket2() {  
  return this.http.get('https://aula13-3a92f-default-rtdb.firebaseio.com/posts.json',  
    {  
      params: new HttpParams().set('print', 'pretty')  
    })  
};
```

Inspetor

Console

Debugger

Rede

Editor de estilos

Desempenho

Memória

Filtrar URLs

☐ Desativar cache

Sem limitação

TudoHTMLCSSJSXHRFontesImagensMídiaWSOutros

Status	Método	Domínio	Arquivo	Iniciador	Tipo	Transferido	Ta...	0 ms
200	GET	localhost:4200	/	document	html	853 B (raced)	56...	1003 ms
304	GET	localhost:4200	styles.css	stylesheet	css	em cache	77...	1 ms
200	GET	localhost:4200	runtime.js	script	js	6,98 kB	6,...	1 ms
304	GET	localhost:4200	polyfills.js	script	js	em cache	0 B	1 ms
304	GET	localhost:4200	styles.js	script	js	em cache	0 B	3 ms
304	GET	localhost:4200	vendor.js	script	js	em cache	0 B	6 ms
200	GET	localhost:4200	main.js	script	js	31,29 kB	30...	7 ms
101	GET	localhost:4200	ng-cli-ws	polyfills.js:5320 (...)	plain	129 B	0 B	2059 ms
200	OPTIO...	aula13-3a92f-def...	posts.json?print=pretty	xhr	json	382 B	0 B	182 ms
	GET	localhost:4200	favicon.ico	/:1 (img)	vnd.m...	NS_BINDING_AB...	94...	0 ms
200	GET	localhost:4200	favicon.ico	FaviconLoader.s...	vnd.m...	em cache	94...	0 ms
200	GET	aula13-3a92f-def...	posts.json?print=pretty	polyfills.js:10424...	json	518 B	20...	200 ms

Método HttpClient.put()

- Uma solicitação PUT normalmente é usada para atualizar um recurso no servidor.

```
editarTicket(id:string, ticketData: { NomePassageiro: string,
                                     numeroVoo: string,
                                     dataPartida: string,
                                     dataChegada: string,
                                     aeroportoPartida: string,
                                     aeroportoChegada: string
                                   }) {
  return this.http.put(`https://aula13-3a92f-default-rtdb.firebaseio.com/posts/${id}.json`, ticketData, {observe: 'response'});
}
```

Método HttpClient.delete()

- O método delete do HttpClient é usado para enviar uma solicitação DELETE para um servidor;
- Uma solicitação DELETE é usada para excluir um recurso no servidor.

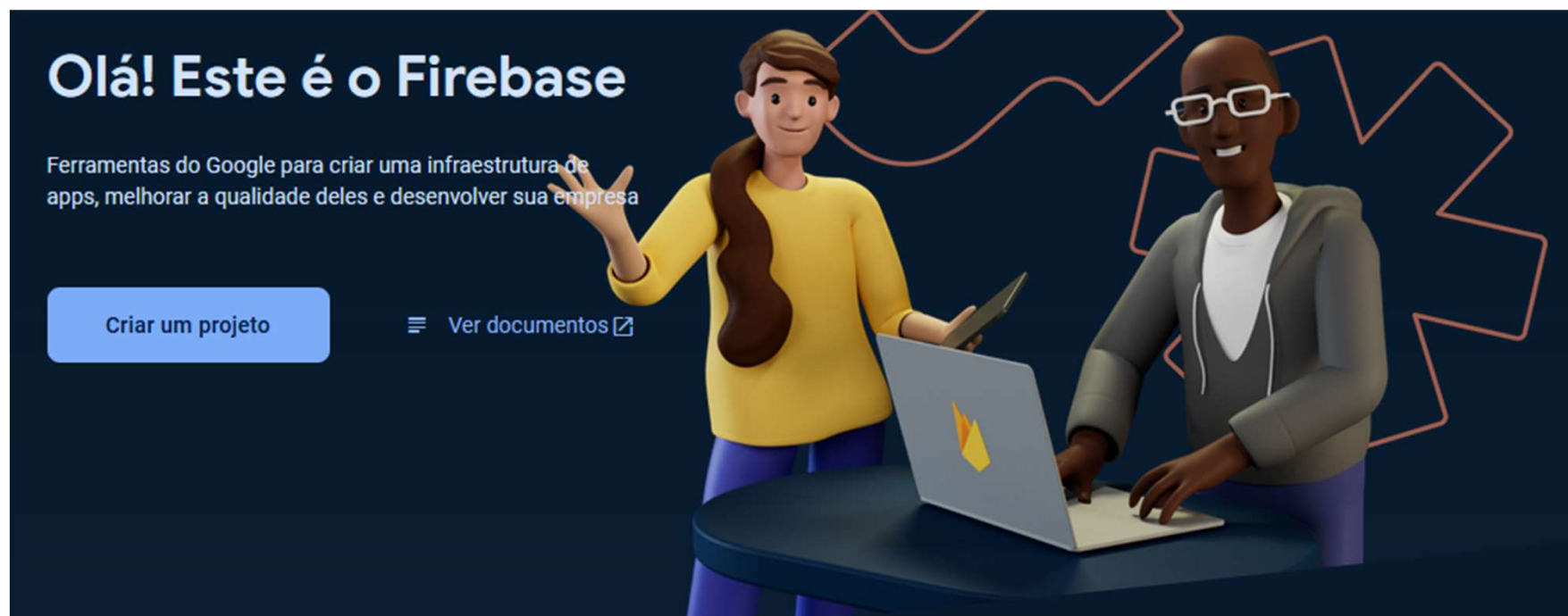
```
apagarTodosTickets() {  
  return this.http.delete('https://aula13-3a92f-default-rtdb.firebaseio.com/posts.json');  
}
```

Criação da parte do backend com o Firebase

- <https://console.firebase.google.com>

Configurando o firebase

- Utilize uma conta do google;
- Ir no endereço: <https://console.firebase.google.com>




Configurando
o firebase

× Criar um projeto(Passo 1 de 3)

Vamos começar nomeando
o projeto[?]

Nome do projeto

 residencia-ccdd7

Continuar

Configurando o firebase

Google Analytics para seu projeto do Firebase

O Google Analytics é uma solução de análise ilimitada e sem custos financeiros. Com ele, é possível segmentar, gerar relatórios e muito mais nos seguintes produtos: Firebase Crashlytics, Cloud Messaging, Mensagens no app, Configuração remota, Teste A/B e Cloud Functions.

O Google Analytics ativa:



Teste A/B ?



Segmentação de usuários em produtos do Firebase ?



Usuários sem falhas ?



Gatilhos do Cloud Functions com base em eventos ?



Geração de relatórios ilimitada gratuita ?



Ativar o Google Analytics neste projeto
Recomendado

Anterior

Continuar

Configurando o firebase

✕ Criar um projeto(Passo 3 de 3)

Configurar o Google Analytics

Localização do Analytics ⓘ

Brasil ▾

O Google Analytics é uma ferramenta de negócios. Use-a exclusivamente para fins relacionados ao seu comércio, negócio, ofício ou profissão.

Configurações de compartilhamento de dados e termos do Google Analytics

☒ Usar as configurações padrão para o compartilhamento de dados do Google Analytics. [Saiba mais](#) ⓘ

✕

Compartilhe seus dados do Analytics com o Google para melhorar os produtos e serviços da empresa

✓

Compartilhe seus dados do Analytics com o Google para ativar o Comparativo de mercado

✓

Compartilhe seus dados do Analytics com o Google para ativar o suporte técnico

✓

Compartilhe seus dados do Analytics com os especialistas em contas do Google

☒ Eu aceito os [Termos do Google Analytics](#) ⓘ

Após a criação do projeto, uma nova propriedade do Google Analytics será criada e vinculada ao seu projeto do Firebase. Esse processo permitirá o fluxo de dados entre os produtos. Os dados da propriedade do Google Analytics exportados para o Firebase ficam sujeitos aos Termos de Serviço do Firebase, e os dados do Firebase importados para o Google Analytics ficam sujeitos aos Termos de Serviço do Google Analytics. [Saiba mais](#) ⓘ.

Anterior

Criar projeto

Configurando
o firebase



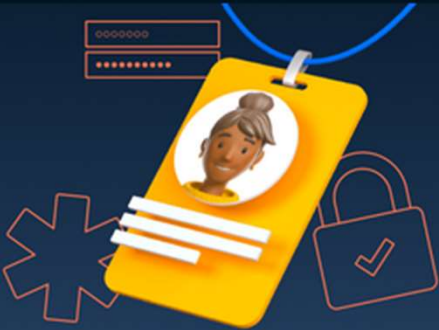
residencia

✓ Your Firebase project is ready


Continuar

Configurar o banco de dados

Armazene e sincronize dados de app em milissegundos ×



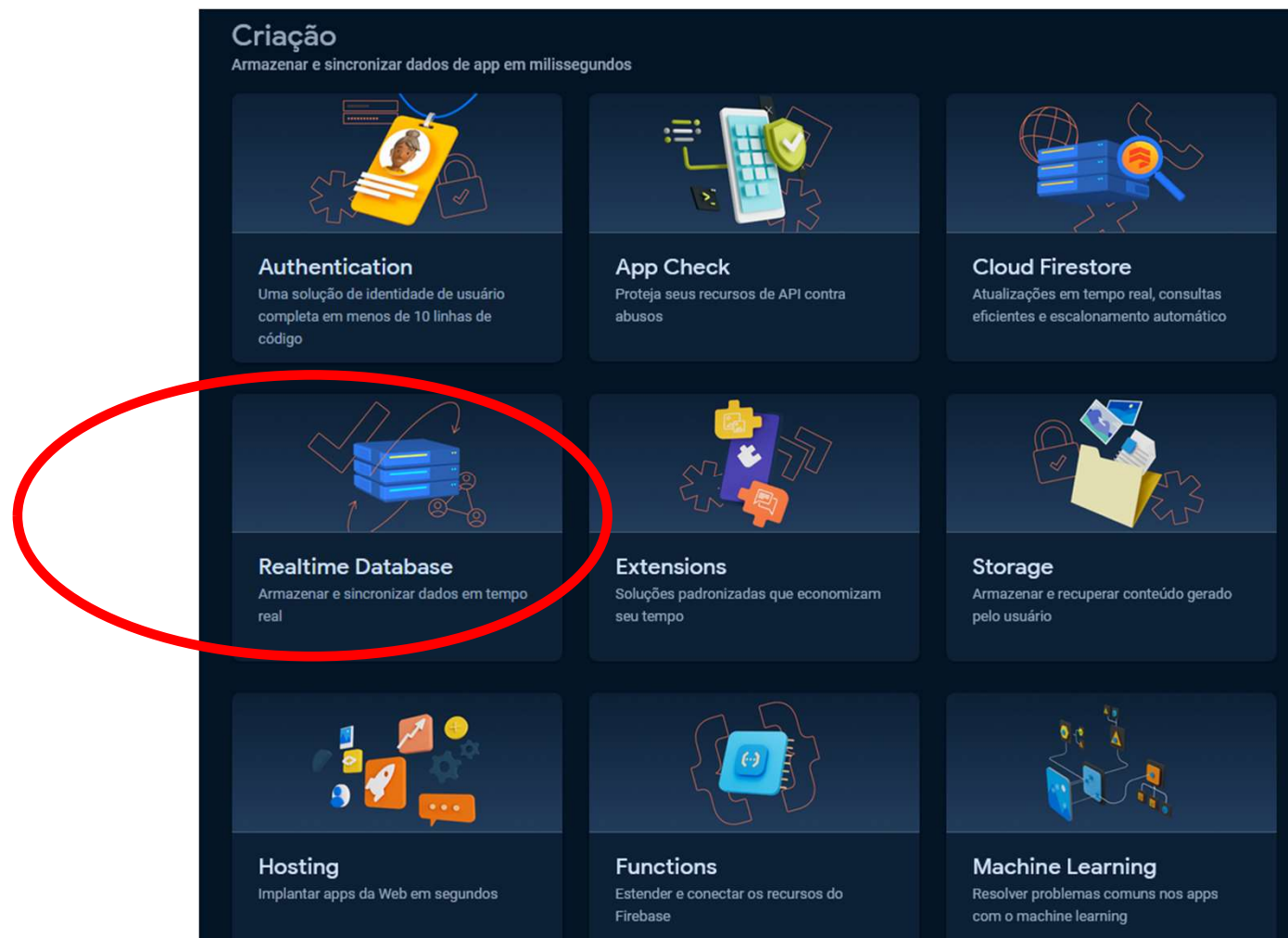
Authentication
Autenticar e gerenciar usuários



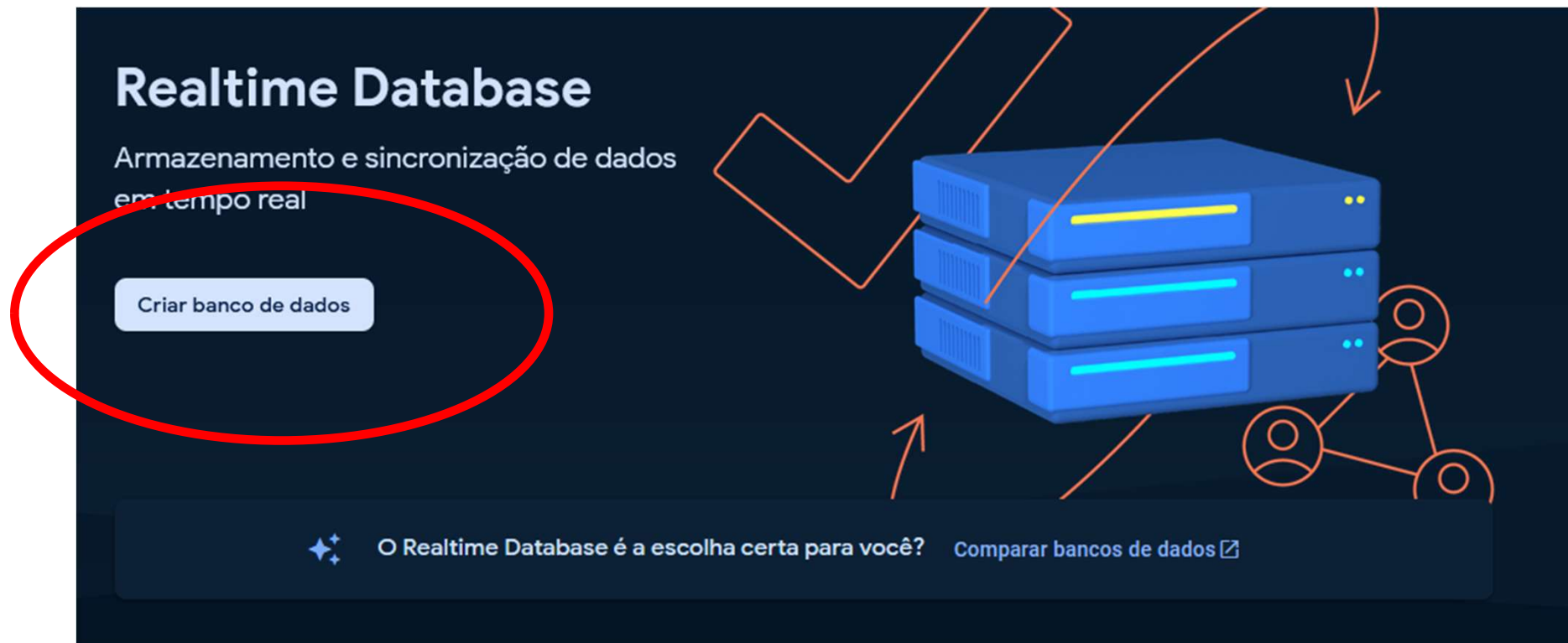
Cloud Firestore
Atualizações em tempo real, consultas eficientes e escalonamento automático

[Veja todos os Criação recursos](#)

Configurar o banco de dados



Configurar o banco de dados



Configurar o banco de dados

Configurar banco de dados

1

Opções de banco de dados

2

Regras de segurança

Sua configuração de local é onde os dados do Realtime Database serão armazenados.

Local do Realtime Database

Estados Unidos (us-central1)

Cancelar

Próxima

Configurar o banco de dados

Configurar banco de dados

Opções de banco de dados

2 Regras de segurança

Após definir a estrutura de dados, escreva regras para proteger os dados.
[Saiba mais](#)

☐ Iniciar no modo bloqueado

Seus dados são privados por padrão. O acesso de leitura/ gravação do cliente será concedido apenas se especificado por suas regras de segurança.

☒ Iniciar no modo de teste

Por padrão, seus dados estão definidos para permitir uma configuração rápida. Porém, você precisa atualizar suas regras de segurança em até 30 dias para permitir em longo prazo o acesso de leitura/gravação do cliente.

```
{
  "rules": {
    ".read": "now < 1709694000000", // 2024-3-6
    ".write": "now < 1709694000000", // 2024-3-6
  }
}
```

!

As regras de segurança padrão para o modo de teste permitem que qualquer pessoa com a referência do seu banco de dados acesse, edite e exclua todos os dados nos próximos 30 dias


Cancelar


Ativar


URL




Realtime Database

[Dados](#)[Regras](#)[Backups](#)[Uso](#)[Extensões](#)

 Proteja os recursos do Realtime Database de abusos, como fraude de faturamento ou phishing

[Configurar o App Check](#) 

 <https://aula13-3a92f-default-rtdb.firebaseio.com>



`https://aula13-3a92f-default-rtdb.firebaseio.com/: null`

Exercício 01

- Crie um projeto no Firebase;
- Crie um banco realTime database;
- Crie um service com os métodos: Adicionar, Editar, Listar, Consultar;
 - Faça as consultas em cada método para acessar o realTime database no firebase.
- Escolha um tema de interesse e crie uma aplicação:
 - PetShop, Lojinha online, Oficina Mecânica, etc.
- Crie componentes relacionados para alimentar o banco de dados através do service.

Referências

- <https://angular.io/guide/understanding-communicating-with-http>
- <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers>