

# Ionic com Firebase

O **Firebase** é uma plataforma de desenvolvimento de aplicativos móveis e web, mantida pelo Google. Ele funciona como um **Backend as a Service (BaaS)**, o que significa que ele oferece um conjunto de serviços e ferramentas de backend prontos para uso, permitindo que desenvolvedores criem aplicativos de alta qualidade de forma mais rápida e eficiente, sem a necessidade de gerenciar sua própria infraestrutura de servidor.



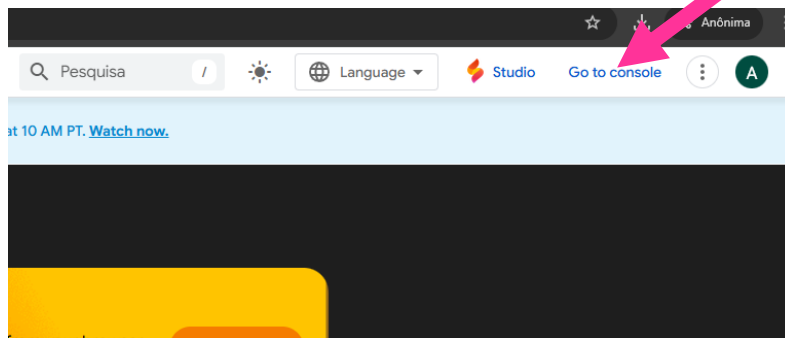
krivago



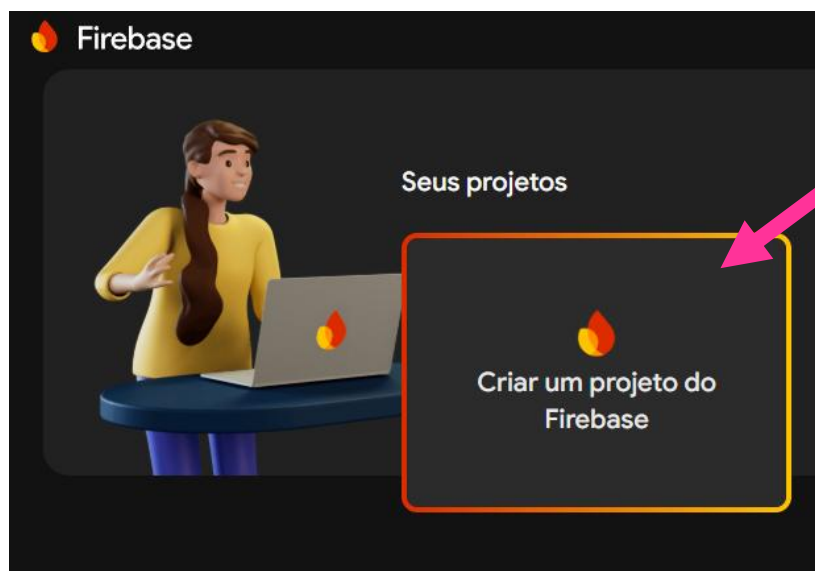
## **Passo 1: Criando o projeto no Firebase**

Acesse o link: <https://firebase.google.com/>

1. Faça login com uma conta Google válida e clique no link **GO TO CONSOLE**



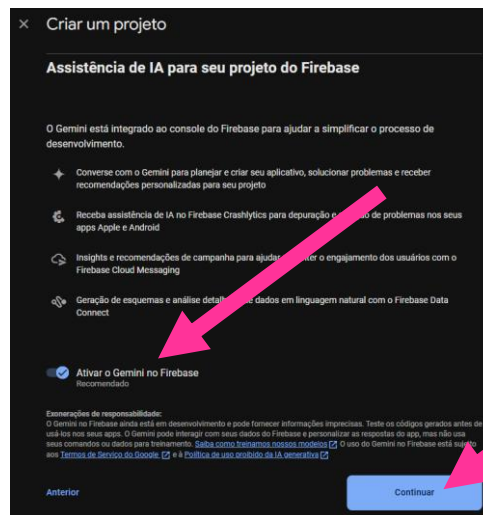
2. Clique em **CRIAR UM PROJETO DO FIREBASE**



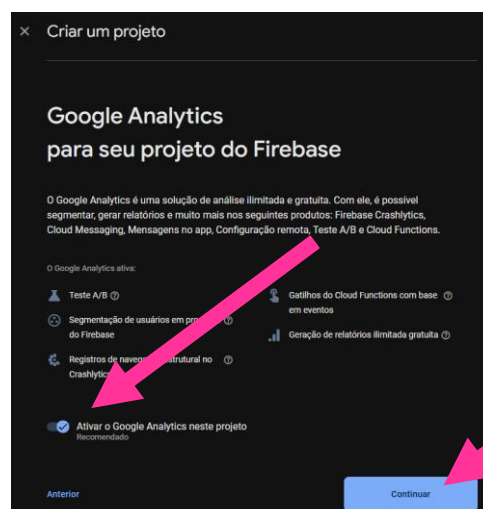
3. Dê um nome ao projeto e clique em **CONTINUAR**



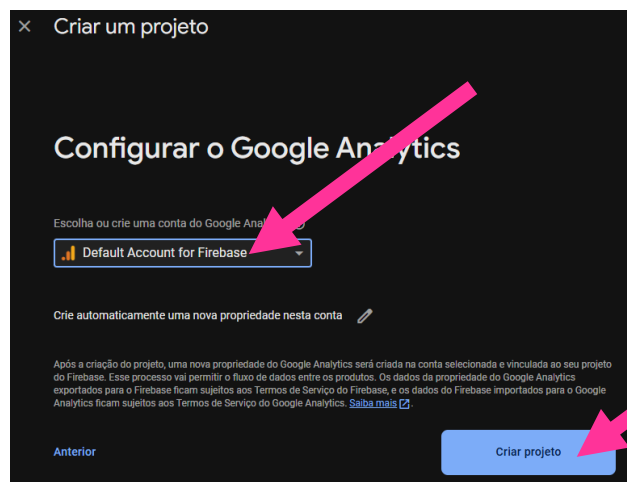
4. Permita o uso de IA no seu projeto. Pode ser útil no futuro, para uma integração com o Gemini



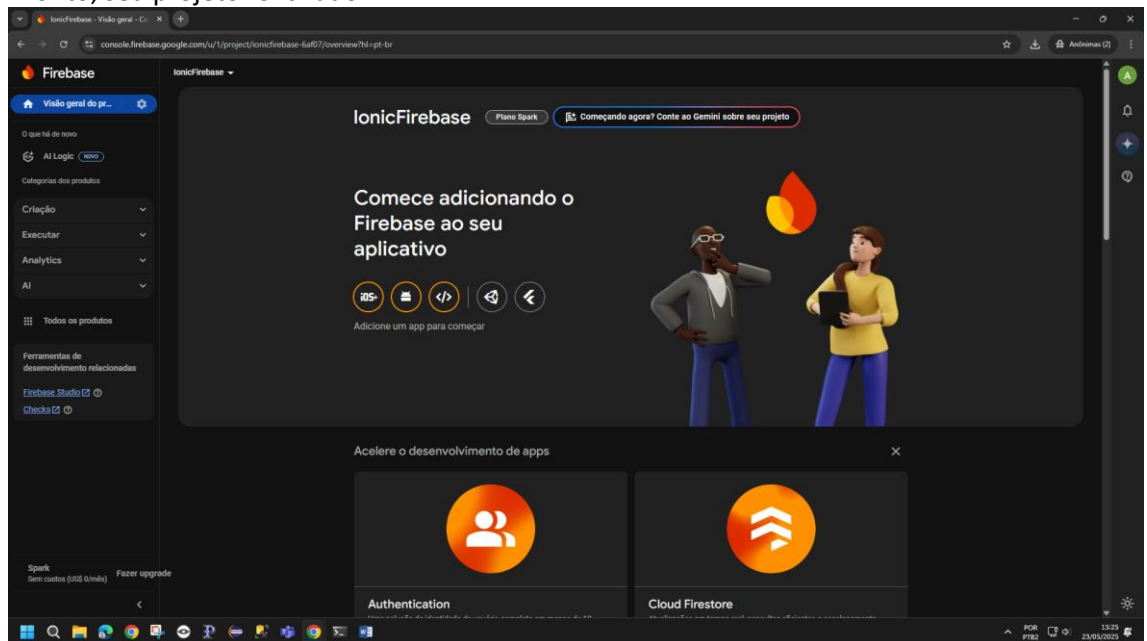
5. Deixe ativado o Google Analytics. O **Google Analytics para Firebase** é uma solução de análise de aplicativos gratuita que fornece insights detalhados sobre o uso de apps e o engajamento do usuário. Ele é a ferramenta de análise central do Firebase, uma plataforma de desenvolvimento de aplicativos móveis e web do Google.



6. Selecione **DEFAULT ACCOUNT FOR FIREBASE**. Isso criará um projeto padrão. As configurações deste projeto podem ser modificadas posteriormente



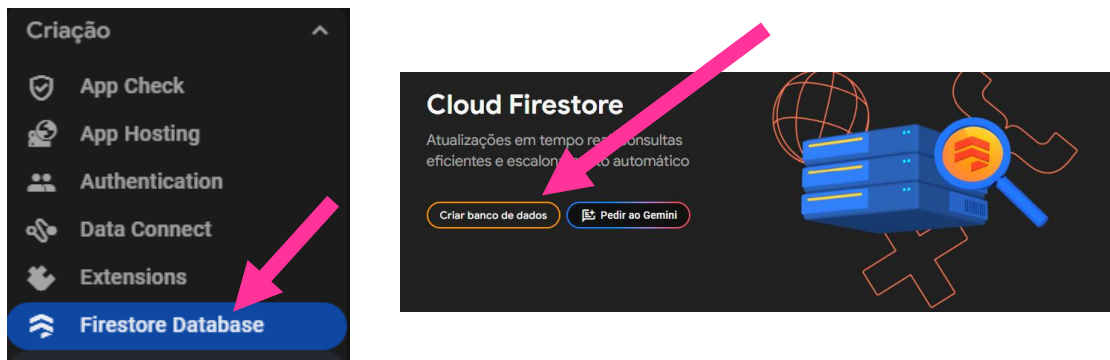
7. Pronto, seu projeto foi criado.



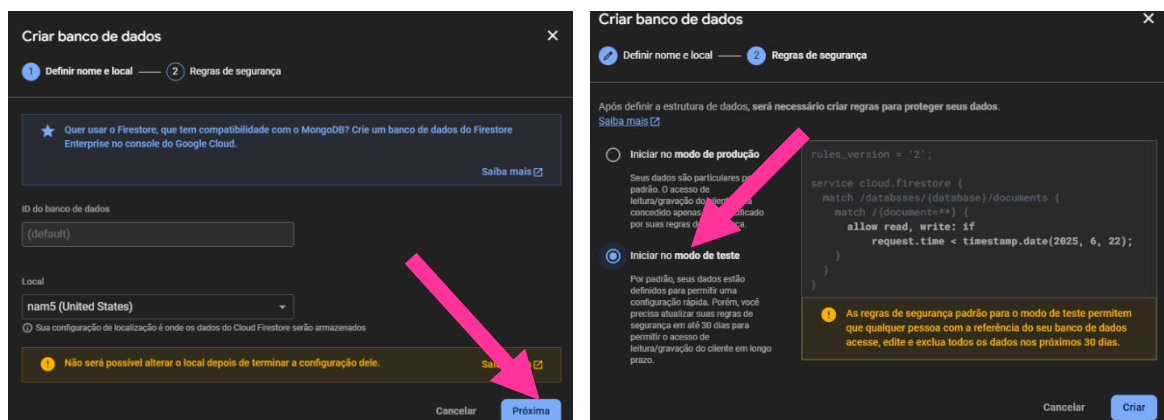
## Passo 2: Criando o banco de dados Firestore Database

O **Cloud Firestore** é um banco de dados de documentos NoSQL flexível, escalável e de alto desempenho, desenvolvido pelo Google como parte da plataforma Firebase e Google Cloud. Ele é uma evolução do Realtime Database e foi projetado para armazenar, sincronizar e consultar dados para aplicativos móveis, web e de servidor em escala global.

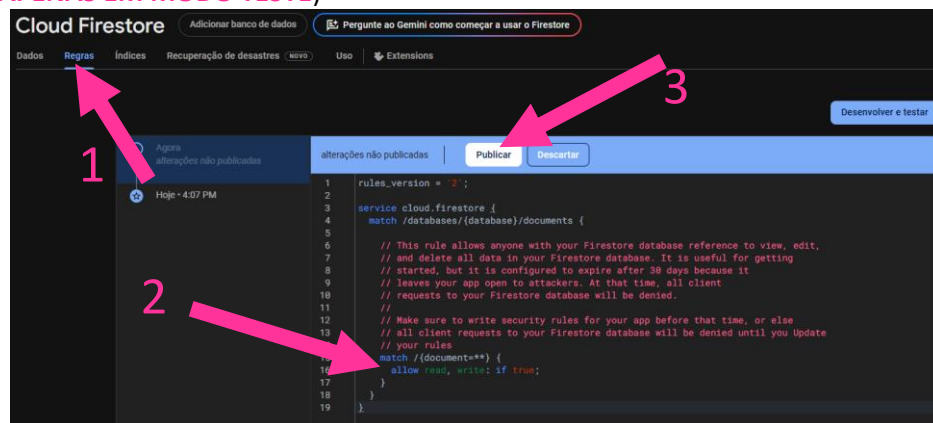
1. No menu lateral (lado esquerdo) selecione **CRIAÇÃO ▢ FIRESTORE DATABASE** em seguida clique em **CRIAR BANCO DE DADOS**



2. Não altere nada na tela seguinte e depois selecione **MODOS DE TESTE**



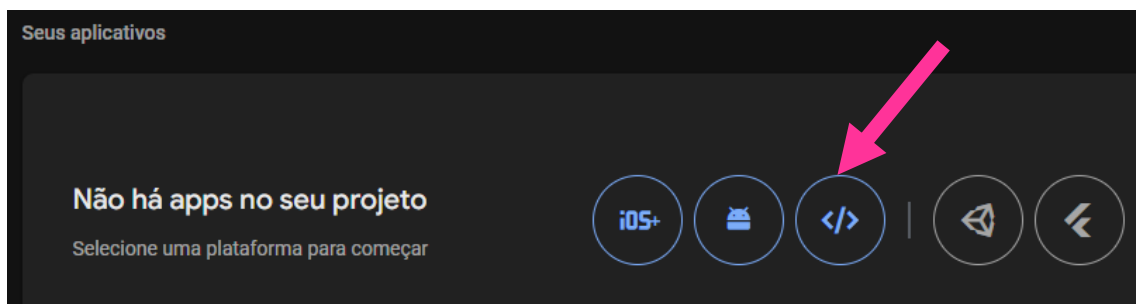
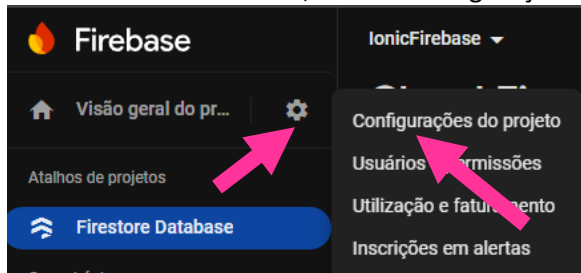
3. Com o banco criado, vamos alterar algumas políticas de segurança (**RECOMENDÁVEL APENAS EM MODOS DE TESTE**)



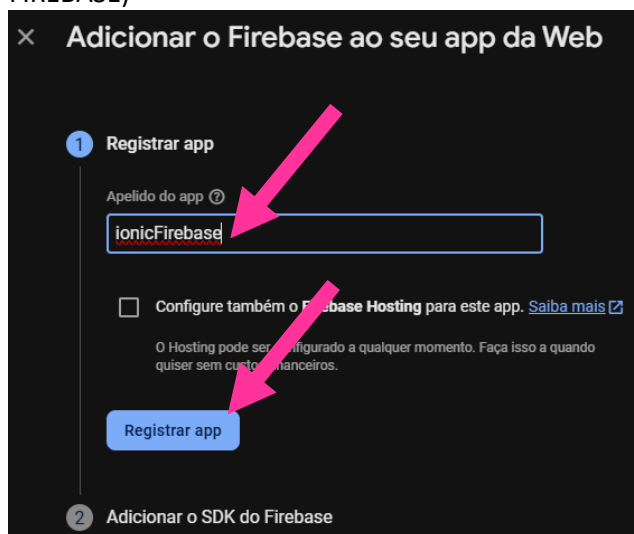
### Passo 3: Criar o projeto WEB

Agora vamos criar o projeto WEB para obter as credenciais que serão utilizadas no projeto IONIC.

1. No menu lateral, abra as configurações do projeto



2. Identifique o projeto (não há necessidade de usar o mesmo nome para o projeto FIREBASE)



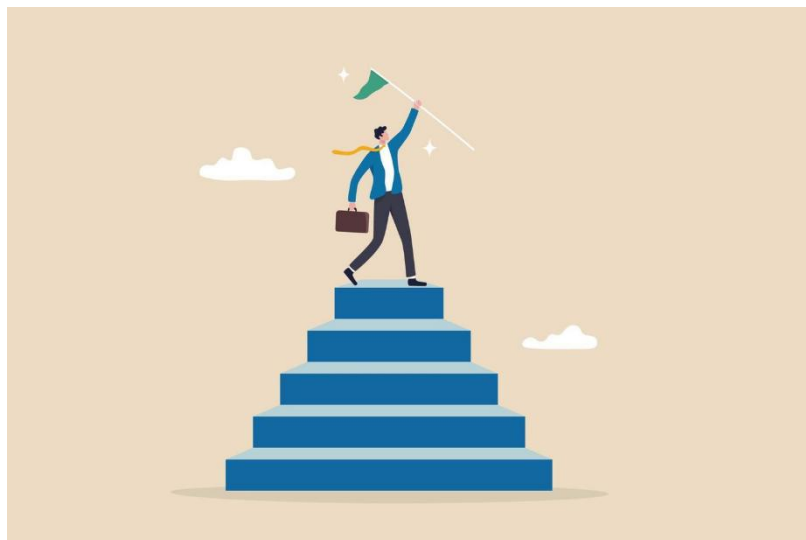
3. Guarde o trecho de código em destaque. Ele será usado no IONIC. Estas são as credenciais de acesso ao projeto FIREBASE. **JAMAIS COMPARTILHE ESTAS INFORMAÇÕES!**

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyA...PQ",
  authDomain: "ion...com",
  projectId: "ion...07",
  storageBucket: "ion...ge.app",
  messagingSenderId: "96...5",
  appId: "1:96793...bfb56",
  measurementId: "G-...LS"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Com isso concluímos a configuração do **Firebase** e **do Firestore Database**. Agora vamos para o projeto IONIC



### Passo 4: Criar o App Ionic

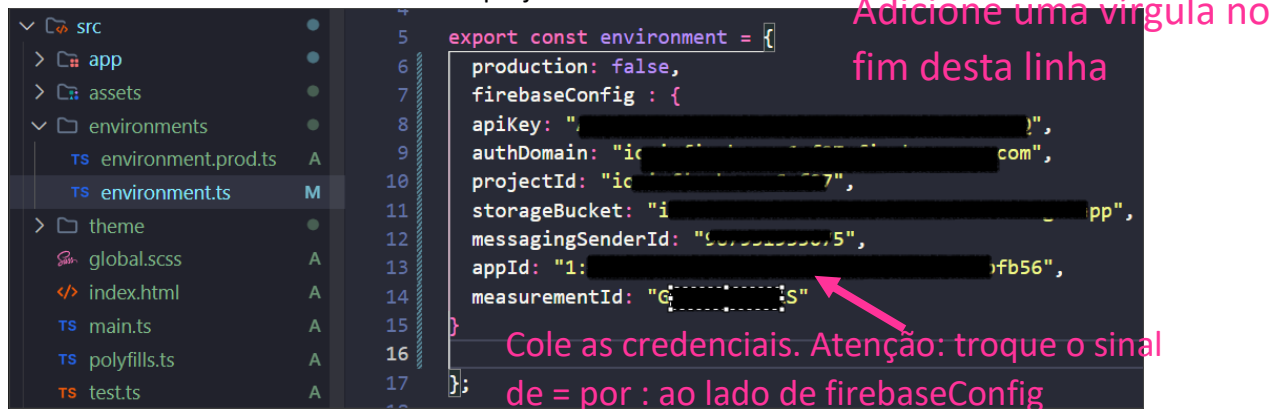
Crie um projeto **IONIC**, usando o comando **ionic start** no console. Lembre-se que a framework é **ANGULAR** que vamos usar o **NgModules**.

Após a criação, **entre na pasta do projeto** e adicione os módulos do firebase, digitando o seguinte comando no console:

```
npm install @angular/fire firebase
```

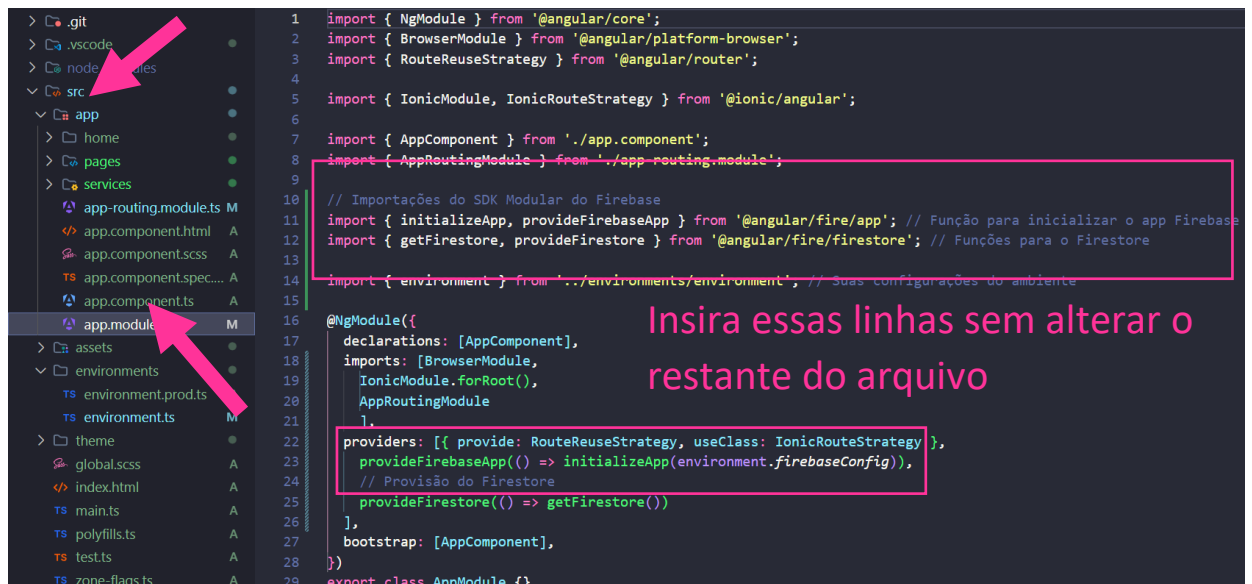
Abra o Visual Studio Code para iniciar o código.

### Inserindo as credenciais do Firebase no projeto:



Agora vamos alterar o arquivo **APP.MODULE.TS**.

**ATENÇÃO:** Não altere todo o arquivo, apenas as linhas em destaque



## Passo 5 - Criação do Serviço

---

No console, dentro da pasta do projeto, digite:

```
ionic g service services/data
```

Código do arquivo `data.service.ts`

```
1  import { Injectable } from '@angular/core';
2
3  // Importações do SDK Modular do Firestore
4  import {
5    Firestore, // O serviço principal do Firestore
6    collection, // Para obter uma referência a uma coleção
7    doc, // Para obter uma referência a um documento
8    collectionData, // Para obter dados de uma coleção como Observable
9    docData, // Para obter dados de um documento como Observable
10   addDoc, // Para adicionar um novo documento
11   updateDoc, // Para atualizar um documento
12   deleteDoc, // Para deletar um documento
13   query, // Para construir consultas (ex: ordenar)
14   orderBy // Para ordenar resultados
15 } from '@angular/fire/firestore';
16 import { Observable } from 'rxjs';
17
18 // Interface para o nosso item
19 export interface Item {
20   id?: string;
21   name: string;
22   description: string;
23   createdAt?: number;
24 }
25
26 @Injectable({
27   providedIn: 'root'
28 })
```



```

29 export class DataService {
30
31     constructor(private firestore: Firestore){
32     }
33
34     // Retorna todos os itens
35     getItems(): Observable<Item[]> {
36         // Cria uma referência para a coleção 'items'
37         const itemsCollectionRef = collection(this.firestore, 'items');
38         // Cria uma query para ordenar por 'createdAt' em ordem decrescente
39         const q = query(itemsCollectionRef, orderBy('createdAt', 'desc'));
40         // Retorna os dados da coleção como um Observable, incluindo o ID do documento
41         return collectionData(q, { idField: 'id' }) as Observable<Item[]>;
42     }
43
44     // Retorna um item específico pelo ID
45     getItem(id: string): Observable<Item | undefined> {
46         // Cria uma referência para o documento específico
47         const itemDocRef = doc(this.firestore, `items/${id}`);
48         // Retorna os dados do documento como um Observable, incluindo o ID do documento
49         return docData(itemDocRef, { idField: 'id' }) as Observable<Item | undefined>;
50     }
51
52     // Adiciona um novo item
53     addItem(item: Item) {
54         const itemsCollectionRef = collection(this.firestore, 'items');
55         // Adiciona um novo documento à coleção
56         return addDoc(itemsCollectionRef, { ...item, createdAt: Date.now() });
57     }
58
59     // Atualiza um item existente
60     updateItem(item: Item) {
61         // Cria uma referência para o documento específico
62         const itemDocRef = doc(this.firestore, `items/${item.id}`);
63         // Atualiza o documento
64         return updateDoc(itemDocRef, { name: item.name, description: item.description });
65     }
66
67     // Deleta um item pelo ID
68     deleteItem(id: string) {
69         // Cria uma referência para o documento específico
70         const itemDocRef = doc(this.firestore, `items/${id}`);
71         // Deleta o documento
72         return deleteDoc(itemDocRef);
73     }
74 }

```

```
1 <ion-header>
2   <ion-toolbar color="primary">
3     <ion-title>
4       Meus Itens
5     </ion-title>
6     <ion-buttons slot="end">
7       <ion-button (click)="addItem()">
8         <ion-icon name="add"></ion-icon>
9       </ion-button>
10    </ion-buttons>
11  </ion-toolbar>
12 </ion-header>
13
14 <ion-content class="ion-padding">
15   <ion-list>
16     <ion-item *ngIf="items.length === 0">
17       <ion-label>Nenhum item cadastrado ainda.</ion-label>
18     </ion-item>
19
20     <ion-item-sliding *ngFor="let item of items">
21       <ion-item lines="full" button (click)="editItem(item)">
22         <ion-label>
23           <h2>{{ item.name }}</h2>
24           <p>{{ item.description }}</p>
25         </ion-label>
26       </ion-item>
27
28       <ion-item-options side="end">
29         <!--A exclamação aqui indica que o item.id jamais será nulo" -->
30         <ion-item-option color="danger" (click)="deleteItem(item.id!)">
31           <ion-icon slot="icon-only" name="trash"></ion-icon>
32         </ion-item-option>
33       </ion-item-options>
34     </ion-item-sliding>
35   </ion-list>
36 </ion-content>
```

```

1  import { Component } from '@angular/core';
2
3  import { DataService, Item } from '../services/data.service';
4  import { Router } from '@angular/router';
5  import { AlertController } from '@ionic/angular';
6
7  @Component({                                Não esquecer os imports
8      selector: 'app-home',
9      templateUrl: 'home.page.html',
10     styleUrls: ['home.page.scss'],
11     standalone: false,
12 })
13 export class HomePage {
14
15     items: Item[] = [];
16
17     constructor(
18         private dataService: DataService,
19         private router: Router,
20         private alertController: AlertController
21     ) {}
22
23     ngOnInit() {
24         this.dataService.getItems().subscribe(res => {
25             this.items = res;
26         });
27     }
28
29     addItem() {
30         this.router.navigateByUrl('/item-detail');
31     }

```

```

32
33     editItem(item: Item) {
34         this.router.navigateByUrl(`/item-detail/${item.id}`);
35     }
36
37     async deleteItem(id: string) {
38         const alert = await this.alertController.create({
39             header: 'Confirmar exclusão',
40             message: 'Tem certeza que deseja excluir este item?',
41             buttons: [
42                 {
43                     text: 'Cancelar',
44                     role: 'cancel',
45                     cssClass: 'secondary',
46                 },
47                 {
48                     text: 'Excluir',
49                     handler: () => {
50                         this.dataService.deleteItem(id);
51                     },
52                 },
53             ],
54         });
55         await alert.present();
56     }
57 }

```

### Passo 7: item-detail.page.html e item.detail.page.ts

No console, dentro da pasta do projeto, digite:

```
ionic g page pages/page-detail
```

```
1 <ion-header>
2   <ion-toolbar color="primary">
3     <ion-buttons slot="start">
4       <ion-back-button defaultHref="/home"></ion-back-button>
5     </ion-buttons>
6     <ion-title>{{ isNewItem ? 'Adicionar Novo Item' : 'Editar Item' }}</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content class="ion-padding">
11   <form #form="ngForm" (ngSubmit)="saveItem()">
12     <ion-item>
13       <ion-label position="floating">Nome</ion-label>
14       <ion-input type="text" [(ngModel)]="item.name" name="name" required></ion-input>
15     </ion-item>
16
17     <ion-item>
18       <ion-label position="floating">Descrição</ion-label>
19       <ion-textarea rows="4" [(ngModel)]="item.description" name="description"></ion-textarea>
20     </ion-item>
21
22     <ion-button expand="full" type="submit" [disabled]="!form.valid" class="ion-margin-top">
23       Salvar
24     </ion-button>
25   </form>
26 </ion-content>
```

```

1 import { Component, OnInit } from '@angular/core';
2
3 import { ActivatedRoute, Router } from '@angular/router';
4 import { DataService, Item } from '../../services/data.service';
5 import { LoadingController, ToastController } from '@ionic/angular';
6
7
8 @Component({
9   selector: 'app-item-detail',
10  templateUrl: './item-detail.page.html',
11  styleUrls: ['./item-detail.page.scss'],
12  standalone: false,
13 })
14 export class ItemDetailPage implements OnInit {
15
16   // Declara uma propriedade 'item' do tipo Item, inicializada com valores vazios.
17   // Será usada para vincular os dados do formulário (nome e descrição).
18   item: Item = {
19     name: '',
20     description: ''
21   };
22   // Declara 'itemId' que pode ser uma string ou null. Armazenará o ID do item se estivermos editando.
23   itemId: string | null = null;
24   // Uma flag booleana para verificar se estamos criando um novo item (true) ou editando um existente (false).
25   isNewItem = true;
26
27   constructor(private route: ActivatedRoute, private dataService: DataService, private router: Router,
28     | | | | private loadingController: LoadingController, private toastController: ToastController) { }
29
30   ngOnInit() {
31     this.itemId = this.route.snapshot.paramMap.get('id');
32     // Verifica se um ID de item foi encontrado na URL.
33     if (this.itemId) {
34       // Se um ID existe, significa que estamos editando um item existente.
35       this.isNewItem = false;
36       // Chama o método para carregar os dados do item.
37       this.loadItem();
38     }
39   }
40
41   async loadItem() {
42     // Cria um controle de carregamento (loading spinner) com uma mensagem.
43     const loading = await this.loadingController.create({
44       message: 'Carregando item...'
45     });
46     // Apresenta o loading spinner na tela.
47     await loading.present();
48     // Chama o método getItem do DataService para obter o item pelo ID.
49     // O '!' (non-null assertion operator) informa ao TypeScript que itemId não será null aqui.
50     this.dataService.getItem(this.itemId!).subscribe(res => {
51       // Dispensa o loading spinner assim que a resposta for recebida.
52       loading.dismiss();
53       // Verifica se o item foi encontrado.
54       if (res) {
55         // Se encontrado, atribui os dados retornados à propriedade 'item' do componente.
56         this.item = res;
57       } else {
58         // Se o item não for encontrado, exibe um toast de erro.
59         this.presentToast('Item não encontrado!', 'danger');
60         // Redireciona o usuário de volta para a página inicial.
61         this.router.navigateByUrl('/home');
62       }
63     }, err => { // Trata erros na requisição.
64       // Dispensa o loading spinner em caso de erro.
65       loading.dismiss();
66       // Exibe um toast de erro genérico.
67       this.presentToast('Erro ao carregar item.', 'danger');
68       // Redireciona o usuário de volta para a página inicial.
69       this.router.navigateByUrl('/home');
70     });
71   }

```

```

71  async saveItem() {
72      // Cria um controle de carregamento (loading spinner) com uma mensagem.
73      const loading = await this.loadingController.create({
74          message: 'Salvando item...'
75      });
76      // Apresenta o loading spinner na tela.
77      await loading.present();
78
79      // Verifica se a flag 'isNewItem' é verdadeira, indicando que é um novo item.
80      if (this.isNewItem) {
81          // Se for um novo item, chama o método 'addItem' do DataService.
82          this.dataService.addItem(this.item).then(() => {
83              // Dispensa o loading spinner após o sucesso.
84              loading.dismiss();
85              // Exibe um toast de sucesso.
86              this.presentToast('Item adicionado com sucesso!', 'success');
87              // Redireciona o usuário para a página inicial.
88              this.router.navigateByUrl('/home');
89          }, err => { // Trata erros ao adicionar.
90              // Dispensa o loading spinner em caso de erro.
91              loading.dismiss();
92              // Exibe um toast de erro.
93              this.presentToast('Erro ao adicionar item.', 'danger');
94          });
95      } else { // Se 'isNewItem' for falsa, estamos atualizando um item existente.
96          // Chama o método 'updateItem' do DataService.
97          this.dataService.updateItem(this.item).then(() => {
98              // Dispensa o loading spinner após o sucesso.
99              loading.dismiss();
100             // Exibe um toast de sucesso.
101             this.presentToast('Item atualizado com sucesso!', 'success');
102             // Redireciona o usuário para a página inicial.
103             this.router.navigateByUrl('/home');
104         }, err => { // Trata erros ao atualizar.
105             // Dispensa o loading spinner em caso de erro.
106             loading.dismiss();
107             // Exibe um toast de erro.
108             this.presentToast('Erro ao atualizar item.', 'danger');
109         });
110     }
111 }
112
113 async presentToast(message: string, color: string = 'primary') {
114     // Cria um controle de toast com a mensagem, duração e cor.
115     const toast = await this.toastController.create({
116         message: message,
117         duration: 2000, // O toast desaparecerá após 2 segundos.
118         color: color // A cor do toast (pode ser primary, secondary, success, danger, etc.).
119     });
120     // Apresenta o toast na tela.
121     toast.present();
122 }
123 }

```

### Passo 7: Arquivo app-routing.module.ts

Insira no arquivo app-routing.module.ts uma nova rota para abrir a página de detalhes do item com passagem de parâmetro:

```
1 import { NgModule } from '@angular/core';
2 import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [
5   {
6     path: 'home',
7     loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
8   },
9   {
10    path: '',
11    redirectTo: 'home',
12    pathMatch: 'full'
13  },
14  {
15    path: 'item-detail',
16    loadChildren: () => import('./pages/item-detail/item-detail.module').then( m => m.ItemDetailPageModule)
17  },
18  {
19    path: 'item-detail/:id', // Rota para editar item existente
20    loadChildren: () => import('./pages/item-detail/item-detail.module').then( m => m.ItemDetailPageModule)
21  }
22 ];
23
24 @NgModule({
25   imports: [
26     RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
27   ],
28   exports: [RouterModule]
29 })
30 export class AppRoutingModule { }
```

Apenas acrescente a rota em destaque, sem alterar o restante do arquivo



**Finalizamos.** Agora é só testar o aplicativo (digitando ionic serve no console, dentro da pasta do projeto)!

