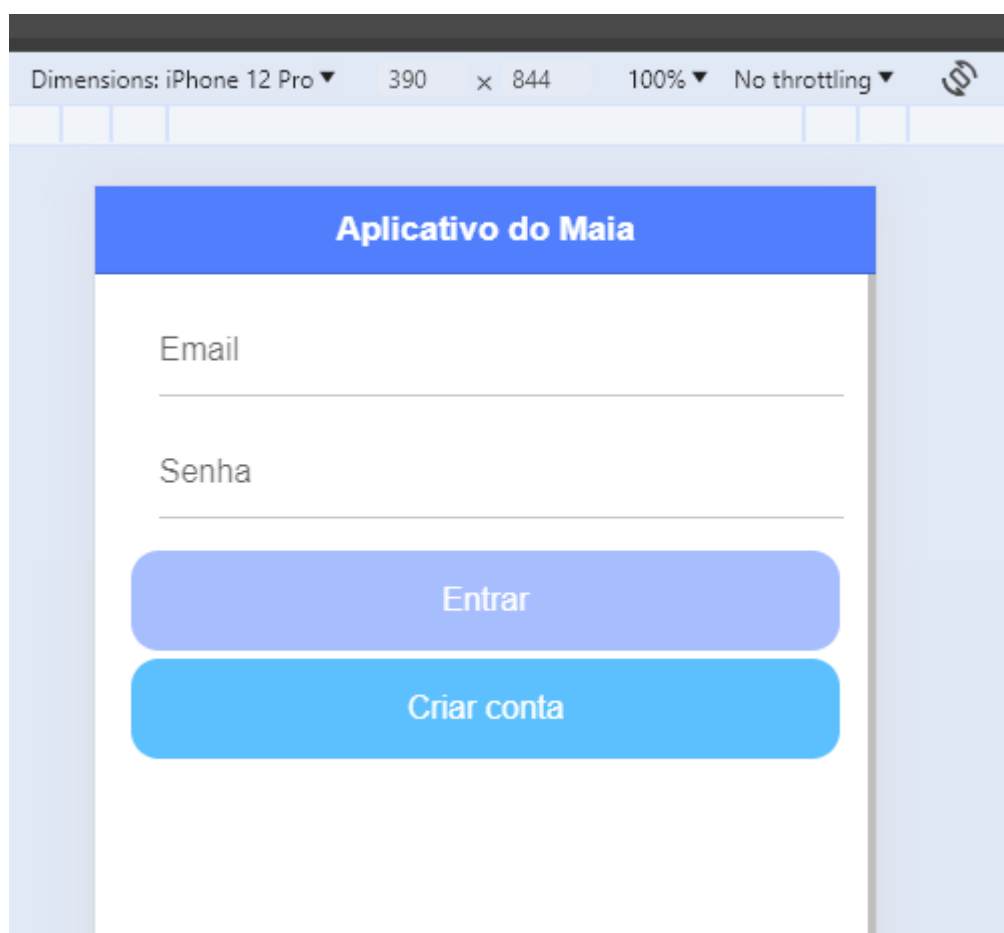


Roteiro: Tela de login integrado com o Firebase



Entre no site:

<https://console.firebase.google.com/?pli=1>



Clique em criar um projeto

X

Criar um projeto(Passo 1 de 3)

Vamos começar nomeando o projeto[?]

Nome do projeto

appfire

✎ appfire-d8e74

📁

Selecionar recurso pai

☒ Aceito os [Termos do Firebase](#)

☒ Confirmo que vou usar o Firebase exclusivamente para fins rel empresa, ofício ou profissão.


Continuar

 Firebase

🏠 Visão geral do p...

⚙️

O que há de novo

 Extensions NOVO

 Release Moni... NOVO

Categorias dos produtos

Criação

Liberar e monitorar

Analytics

Engajamento

⋮

Todos os produtos

appfire

Receber atualizações por e-mail sobre novos recursos, pesquisas e eventos do Firebase [Inscreva-se](#)

appfire

Plano Spark

Comece adicionando o Firebase ao seu aplicativo

iOS





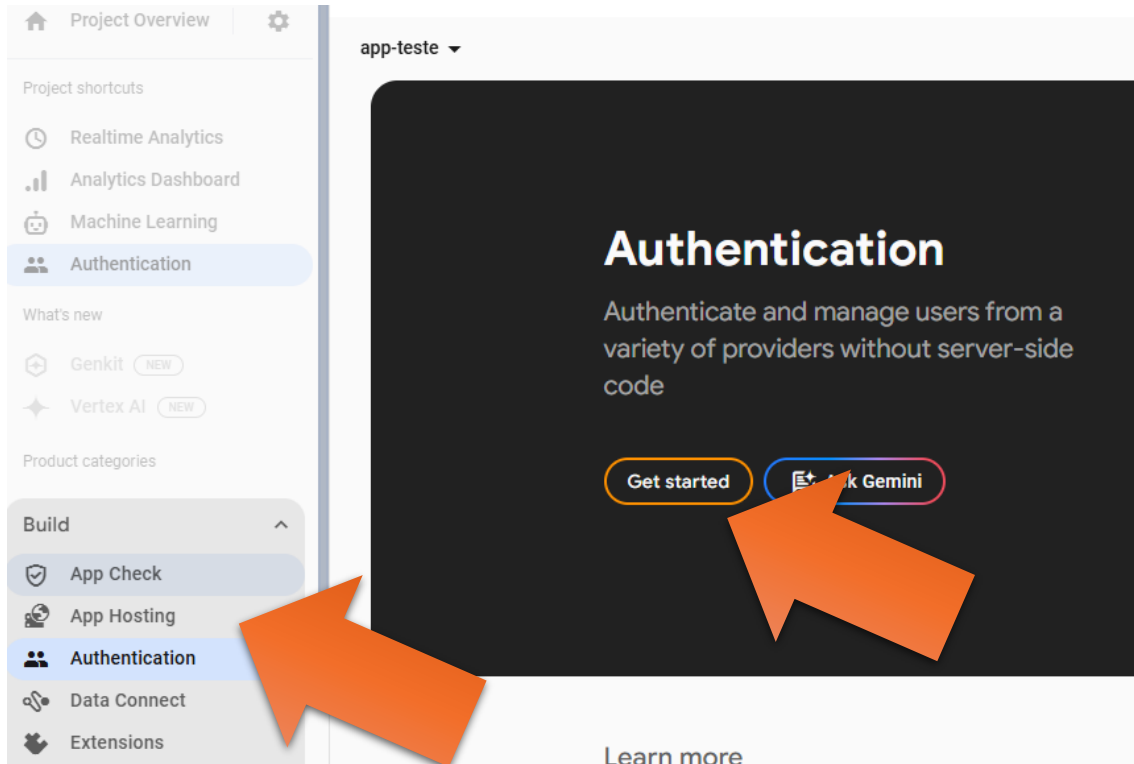




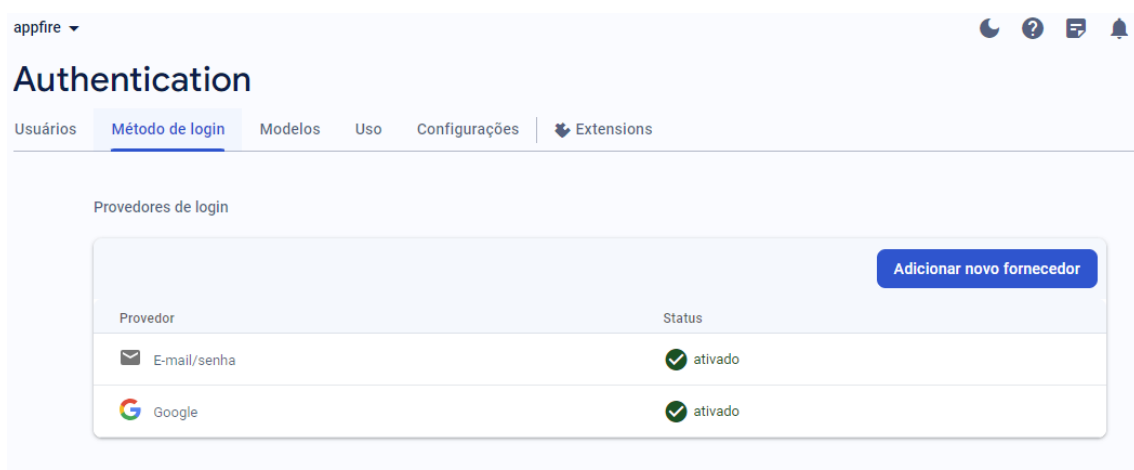
Adicione um app para começar



No menu do lado esquerdo clique em Build -> authentication e depois em “get started”.



Adicione (habilite) dois provedores de autenticação o de e-mail senha e o do Google conforme imagens abaixo:





Ativar

Importante: para ativar o Login do Google nos seus apps Android, é preciso fornecer a [impressão digital da versão SHA-1](#) em cada app (acesse a seção [Configurações do projeto](#) > *Seus apps*).



Atualize abaixo a [configuração no nível do projeto](#) para continuar

Nome público do projeto ⓘ

project-984368944885

E-mail de suporte do projeto ⓘ

luiz.maia@fumec.br

Adicionar IDs de cliente à lista de permissões usando projetos externos (opcional) ⓘ

Configuração do SDK da Web ⓘ

Cancelar

Salvar

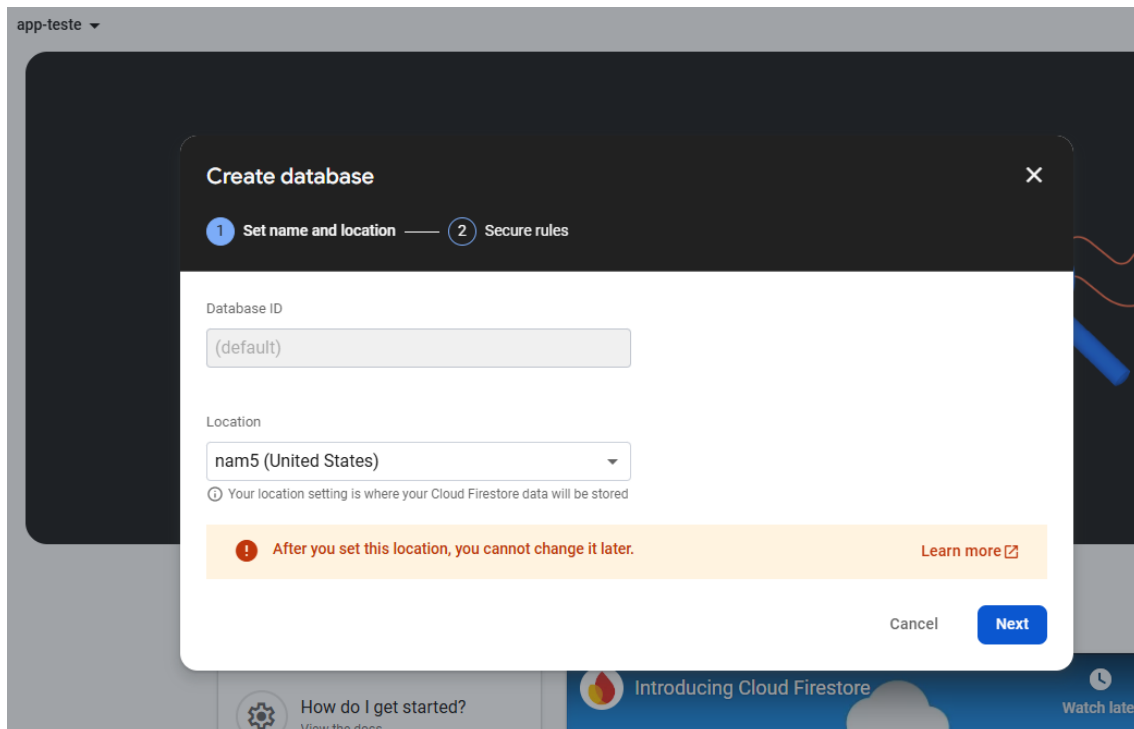
Criando um banco de dados para o projeto:

Cloud Firestore

Atualizações em tempo real, consultas eficientes e escalonamento automático

Criar banco de dados





Após definir a estrutura, é preciso criar regras para proteger seus dados.

[Saiba mais](#)



Iniciar no modo de produção

Seus dados são particulares por padrão. O acesso de leitura/gravação do cliente vai ser 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 o acesso de leitura/gravação do cliente em longo prazo.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2024, 4, 13);
    }
  }
}
```



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

Criar

Escolha o modo de teste. Na estrutura de arquivos do projeto do Ionic existem environment diferentes para cada forma de execução. Esteja atento a esta diferença ao colar as configurações de acesso (segurança) de seu projeto.

Volte a tela inicial e clique em WEB:



× Adicionar o Firebase ao seu app da Web

1 Registrar app

Apelido do app ?

appmaia

☒ Configure também o **Firebase Hosting** para este app. [Saiba mais](#) ?

O Hosting pode ser configurado a qualquer momento sem custos financeiros.

appmaia

.web.app

Registrar app

2 Adicionar o SDK do Firebase

3 Instalar a CLI do Firebase

4 Implantar no Firebase Hosting

2 Adicionar o SDK do Firebase

☒ Usar o npm ☐ Usar a tag <script>

Se você já estiver usando o [npm](#) e um bundler de módulos, como [webpack](#) ou [Rollup](#), execute o seguinte comando para instalar o SDK mais recente ([saiba mais](#)):

```
$ npm install firebase
```



Depois inicialize o Firebase e comece a usar os SDKs dos produtos.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// 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
const firebaseConfig = {
  apiKey: "AIzaSyDexsyk-J6NgVo516-7yxQ6ntnbBAHMLeA",
  authDomain: "appfire-d8e74.firebaseio.com",
  databaseURL: "https://appfire-d8e74-default-rtdb.firebaseio.com",
  projectId: "appfire-d8e74",
  storageBucket: "appfire-d8e74.appspot.com",
  messagingSenderId: "984368944885",
  appId: "1:984368944885:web:b115f6ecbf7da1071d97bc"
};

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



COPIE OS DADOS DE AUTENTICAÇÃO ACIMA EM UM ARQUIVO TEXTO. ELE DEVERÁ SER INSERIDO NO CÓDIGO DO PROJETO IONIC POSTERIORMENTE.

Parte do IONIC

Abra o ícone do prompt dentro da pasta Node.JS no menu Iniciar

Verifique se o ionic encontra-se instalado em seu ambiente digitando *ionic*. Caso responda algo como comando não reconhecido, você deve instalar novamente os pacotes do Node utilizando os comandos:

```
npm install -g @angular/cli  
npm install -g typescript  
npm install -g @ionic/cli
```

- Caso esteja utilizando um projeto já existente:
Mude para o diretório do projeto
Exemplo:
C:
CD \DesMobile\Projetos\appfire

Importante: Para execução deste roteiro, precisamos de um projeto que tenha sido criado utilizando NgModules.

- Caso esteja criando um novo projeto (**ionic start**), em um drive local, digite o comando para criar um aplicativo. Lembre-se de não utilizar unidades em rede como por exemplo o H:.

Escolha como template inicial a opção blank e use o ngmodules.

Rode o comando abaixo para adicionar as bibliotecas de integração ao seu projeto

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

Adicione a página de login

```
ionic g page login
```

Após gerar a página adicione o módulo do firebase:

```
ng add @angular/fire
```

```

C:\Users\1P0411461\Documents\fireauth\appfire>ng add @angular/fire
Skipping installation: Package already installed
UPDATE package.json (2099 bytes)
✓ Packages installed successfully.
? What features would you like to setup? (Press <space> to select, <a>
<enter> to proceed)
  ( ) Cloud Storage
  ( ) Remote Config
  ( ) Vertex AI
> (*) Authentication
  ( ) Google Analytics
  ( ) App Check
  ( ) Firestore
(Move up and down to reveal more choices)

```

Marque a opção de autenticação.

Volte ao Visual Studio Code e continue as edições:

Para facilitar e evitar a definição dos tipos mude o arquivo `tsconfig.json` passando o “strict” para false:



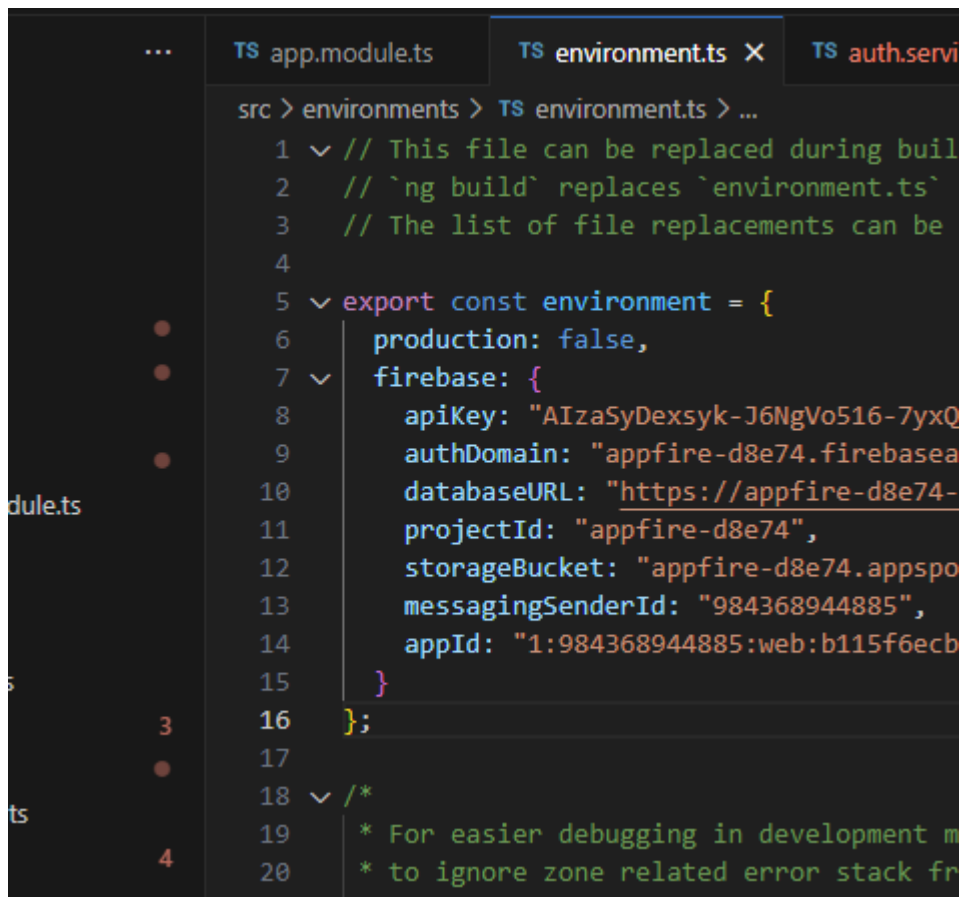
```

{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "strictPropertyInitialization": false,
    "outDir": "./dist/out-tsc",
    "forceConsistentCasingInFileNames": true,
    "strict": false,
    "noImplicitOverride": true,
    "noPropertyAccessFromIndexSignature": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true,
    "sourceMap": true,
    "declaration": false,
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2022",
    "module": "es2020",
    "lib": [
      "es2018",
      "dom"
    ],
    "useDefineForClassFields": false
  }
}

```

Abra o arquivo

environments/environment.ts e coloque suas credenciais do firebase obtidas anteriormente. O formato e uso das chaves é um pouco diferente, realize as adaptações necessárias.



```
src > environments > TS environment.ts > ...
1  // This file can be replaced during build
2  // `ng build` replaces `environment.ts`
3  // The list of file replacements can be
4
5  export const environment = {
6    production: false,
7    firebase: {
8      apiKey: "AIzaSyDexsyk-J6NgVo516-7yxQ",
9      authDomain: "appfire-d8e74.firebaseio.com",
10     databaseURL: "https://appfire-d8e74.firebaseio.com",
11     projectId: "appfire-d8e74",
12     storageBucket: "appfire-d8e74.appspot.com",
13     messagingSenderId: "984368944885",
14     appId: "1:984368944885:web:b115f6ecb",
15   }
16 };
17
18 /*
19  * For easier debugging in development mode, you can
20  * to ignore zone related error stack frames.
```

Arquivo `src/app/app.module.ts`

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

import { AppComponent } from './app.component';
import { AppRoutingModuleModule } from './app-routing.module';
import { initializeApp, provideFirebaseApp } from '@angular/fire/app';
import { environment } from '../environments/environment';
import { provideAuth, getAuth } from '@angular/fire/auth';
import { provideFirestore, getFirestore } from '@angular/fire/firestore';
import { provideStorage, getStorage } from '@angular/fire/storage';
import { ReactiveFormsModule } from '@angular/forms';
```

```

@NgModule({
  declarations: [AppComponent],
  //entryComponents: [],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    IonicModule.forRoot(),
    AppRoutingModule
  ],
  providers: [{
    provide: RouteReuseStrategy,
    useClass: IonicRouteStrategy},
    provideFirebaseApp(() => initializeApp(environment.firebase)),
    provideAuth(() => getAuth()),
    provideFirestore(() => getFirestore()),
    provideStorage(() => getStorage())
  ],
  bootstrap: [AppComponent]
})

export class AppModule {}

```

Ainda não implementamos a autenticação, mas já podemos usar os protetores de autenticação AngularFire de duas maneiras interessantes:

Proteja o acesso a páginas “internas” redirecionando usuários não autorizados para o login

Impedir o acesso à página de login de usuários previamente autenticados, para que sejam automaticamente encaminhados para a área “interna” do app

Isso é feito com os pipes e serviços de ajuda do AngularFire que agora você pode adicionar dentro de `src/app/app-routing.module.ts`:

```

import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from
 '@angular/router';
import { redirectUnauthorizedTo, redirectLoggedInTo, canActivate }
from '@angular/fire/auth-guard';

const redirectUnauthorizedToLogin = () =>
redirectUnauthorizedTo(['']);
const redirectLoggedInToHome = () => redirectLoggedInTo(['home']);

const routes: Routes = [
  {
    path: '',
    loadChildren: () =>
import('./login/login.module').then(m => m.LoginPageModule),
    ...canActivate(redirectLoggedInToHome)
  }
]

```

```

    },
    {
      path: 'home',
      loadChildren: () => import('./home/home.module').then((m)
=> m.HomePageModule),
      ...canActivate(redirectUnauthorizedToLogin)
    },
    {
      path: '**',
      redirectTo: '',
      pathMatch: 'full'
    }
  ]
];

@NgModule({
  imports: [RouterModule.forRoot(routes, { preloadingStrategy:
PreloadAllModules })],
  exports: [RouterModule]
})
export class AppRoutingModule {}

```

Construindo a lógica de autenticação

Toda a lógica estará em um serviço separado, e precisamos apenas de três funções que simplesmente chamem a função correspondente do Firebase para criar um novo usuário, fazer login em um usuário ou encerrar a sessão atual.

Para todas essas chamadas você precisa adicionar a referência Auth, que injetamos dentro do construtor.

CRIE um serviço no ionic chamado auth (comando **ionic generate**).

Como essas chamadas às vezes falham e eu não fiquei muito feliz com o tratamento de erros, envolvi-as em blocos try/catch para que tenhamos mais facilidade quando chegarmos à nossa página real.

Vamos começar com `src/app/services/auth.service.ts` agora e alterá-lo para:

```

import { Injectable } from '@angular/core';
import {
  Auth,
  signInWithEmailAndPassword,
  createUserWithEmailAndPassword,
  signOut
} from '@angular/fire/auth';

@Injectable({
  providedIn: 'root'

```

```

}))
export class AuthService {
  constructor(private auth: Auth) {}

  async register({ email, password }) {
    try {
      const user = await createUserWithEmailAndPassword(this.auth, email,
password);
      return user;
    } catch (e) {
      return null;
    }
  }

  async login({ email, password }) {
    try {
      const user = await signInWithEmailAndPassword(this.auth, email,
password);
      return user;
    } catch (e) {
      return null;
    }
  }

  logout() {
    return signOut(this.auth);
  }
}

```

Agora precisamos capturar as informações do usuário para o cadastro, e por isso importamos o `ReactiveFormsModule` em nosso `src/app/login/login.module.ts` agora:

```
... TS app.module.ts TS environment.ts 1 TS auth.service.ts 4 TS login.module.ts TS
src > app > login > TS login.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
4
5 import { IonicModule } from '@ionic/angular';
6
7 import { LoginPageRoutingModule } from './login-routing.module';
8
9 import { LoginPage } from './login.page';
10
11 @NgModule({
12   imports: [
13     CommonModule,
14     FormsModule,
15     IonicModule,
16     LoginPageRoutingModule,
17     ReactiveFormsModule
18   ],
19   declarations: [LoginPage]
20 })
21 export class LoginPageModule {}
22
23
24
```

Como queremos facilitar, cuidaremos do registro e da inscrição com o mesmo formulário em uma página única de login.

Mas como já adicionamos toda a lógica a um serviço, não nos resta muito o que fazer além de mostrar um indicador de carregamento casual ou apresentar um alerta caso a ação falhe.

Se o registro ou login for bem-sucedido e recebermos de volta um objeto de usuário, encaminhamos imediatamente o usuário para nossa área interna.

src/app/login/login.page.ts:

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AlertController, LoadingController } from '@ionic/angular';
import { AuthService } from '../services/auth.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.page.html',
  styleUrls: ['./login.page.scss'],
  standalone: false
})
```

```

}))
export class LoginPage implements OnInit {
  credentials: FormGroup;

  constructor(
    private fb: FormBuilder,
    private loadingController: LoadingController,
    private alertController: AlertController,
    private authService: AuthService,
    private router: Router
  ) {}

  // Easy access for form fields
  get email() {
    return this.credentials.get('email');
  }

  get password() {
    return this.credentials.get('password');
  }

  ngOnInit() {
    this.credentials = this.fb.group({
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(6)]]
    });
  }

  async register() {
    const loading = await this.loadingController.create();
    await loading.present();

    const user = await this.authService.register(this.credentials.value);
    await loading.dismiss();

    if (user) {
      this.router.navigateByUrl('/home', { replaceUrl: true });
    } else {
      this.showAlert('Falha no registro', 'Tente novamente!');
    }
  }

  async login() {
    const loading = await this.loadingController.create();
    await loading.present();

    const user = await this.authService.login(this.credentials.value);
    await loading.dismiss();
  }
}

```

```

    if (user) {
      this.router.navigateByUrl('/home', { replaceUrl: true });
    } else {
      this.showAlert('Falha no registro', 'Tente novamente!');
    }
  }
}

async showAlert(header, message) {
  const alert = await this.alertController.create({
    header,
    message,
    buttons: ['OK']
  });
  await alert.present();
}
}

```

A última peça que falta agora é a nossa view, que conectamos com o `formGroup` que definimos em nossa página.

Apenas certifique-se de que um botão dentro do formulário tenha o tipo de envio e, portanto, acione a ação `ngSubmit`, enquanto o outro tenha o botão de tipo, caso deva apenas acionar seu evento de clique conectado!

Abra o `src/app/login/login.page.html` agora e altere-o para:

```

<ion-header>
  <ion-toolbar color="primary">
    <ion-title>App de Senha do Maia</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <form (ngSubmit)="login()" [formGroup]="credentials">
    <ion-item fill="solid" class="ion-margin-bottom">
      <ion-input type="email" placeholder="Email"
formControlName="email"></ion-input>
      <ion-note slot="error" *ngIf="(email.dirty || email.touched) &&
email.errors"
        >Email inválido</ion-note>
    </ion-item>
    <ion-item fill="solid" class="ion-margin-bottom">
      <ion-input type="password" placeholder="Password"
formControlName="password"></ion-input>

```

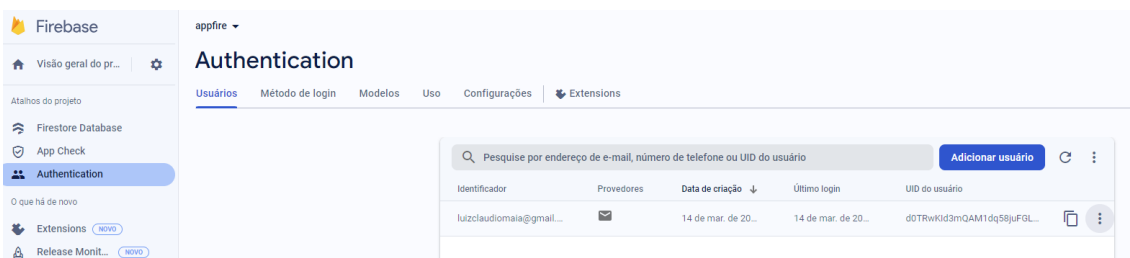
```

        <ion-note slot="error" *ngIf="(password.dirty || password.touched)
&& password.errors"
        >Senhas de seis caracteres</ion-note>
    >
</ion-item>

    <ion-button type="submit" expand="block"
[disabled]="!credentials.valid">Log in</ion-button>
    <ion-button type="button" expand="block" color="secondary"
(click)="register()"
    >Criar conta</ion-button>
    >
</form>
</ion-content>

```

Crie um usuário e verifique se ok.



Entregue um print da tela de login com seu nome e outro print da tela do firebase com o usuário criado conforme acima.

TAREFA OPCIONAL:

Seguindo a chamada dos botões da tela de login como exemplo, crie um botão na tela de HOME com a chamada ao método de logout(). O método de logout já está implementado no serviço de autenticação.

Referências:

<https://devdactic.com/ionic-firebase-auth-upload>
<https://github.com/baumblatt/capacitor-firebase-auth>
<https://www.positronx.io/ionic-firebase-authentication-tutorial-with-examples/>