

Programação Móvel

Firestore

Prof. Dr. Diego R. Antunes

drantunes@utfpr.edu.br

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Anteriormente

Bancos de Dados Móveis

Introdução e Tipos

Introdução ao SQLite

Ionic SqlStorage + SQLite

Dicas

Ionic e SQLite

1. Executar o comando `ionic platform add android`
2. Crie páginas através de `ionic g page MinhaPagina`
3. Usar `INTEGER` ao invés de `INT` para criar tabelas



Firebase

Firebase

É um serviço online do Google que funciona como um backend (servidor) para sua aplicação móvel.











Uma grande vantagem é que o Firebase torna desnecessário o uso de servidor ou de infra-estrutura. Basta criar um aplicativo e se conectar por meio do SDK.

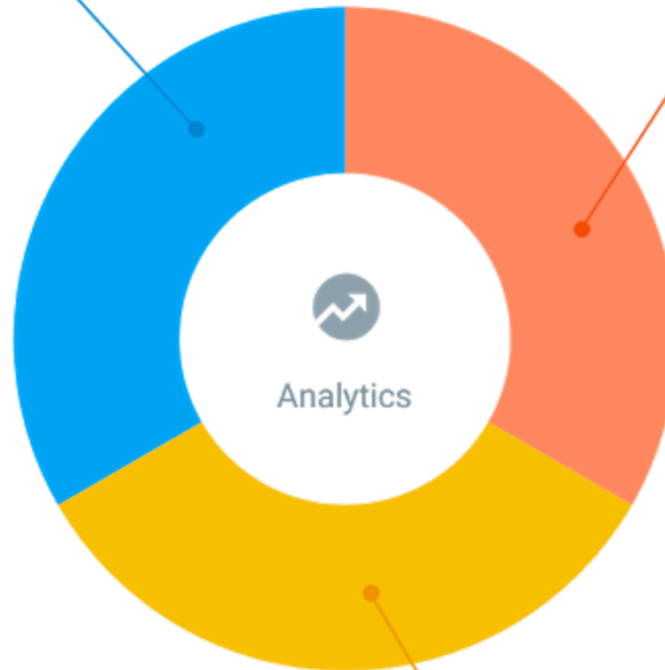
Firestore

Consiste de um serviço pago, porém disponibiliza uma camada gratuita para apps que não tenham muito tráfego.






Serviços

DEVELOP

-  Realtime Database
-  Authentication
-  Cloud Messaging
-  Storage
-  Hosting
-  Remote Config
-  Test Lab
-  Crash Reporting



GROW

-  Notifications
-  App Indexing
-  Dynamic Links
-  Invites
-  AdWords

EARN

Usos

O Firebase pode ser utilizado para Android, iOS e para qualquer Aplicação Web. Desta forma, podemos utilizar em aplicativos híbridos, assim como em páginas web.

Documentação

1. [Autenticação](#)
2. [Realtime Database](#)
3. [Instalação para Web](#)
4. [Página com Exemplos](#)

Serviços

Dos serviços fornecidos vamos estudar o Banco de Dados em Realtime e a Autenticação de Usuários.

Importante

Chamadas Assíncronas

Assim como o SQLite, o Firebase trabalha com chamadas assíncronas. Isto significa que ao realizar uma requisição, ela pode ser processada em paralelo ou posterior à renderização da view (HTML).

Chamadas Assíncronas

Isto pode ser um problema, pois a view (HTML) pode carregar e somente depois de um tempo os dados podem ser retornados do servidor.

Chamadas Assíncronas

*Para minimizar este problema, você pode trabalhar com o **ngIf** do Angular e inserir um loading para fornecer feedback ao usuário e evitar erros de processamento (principalmente em listas vazias).*

Firebase + Ionic

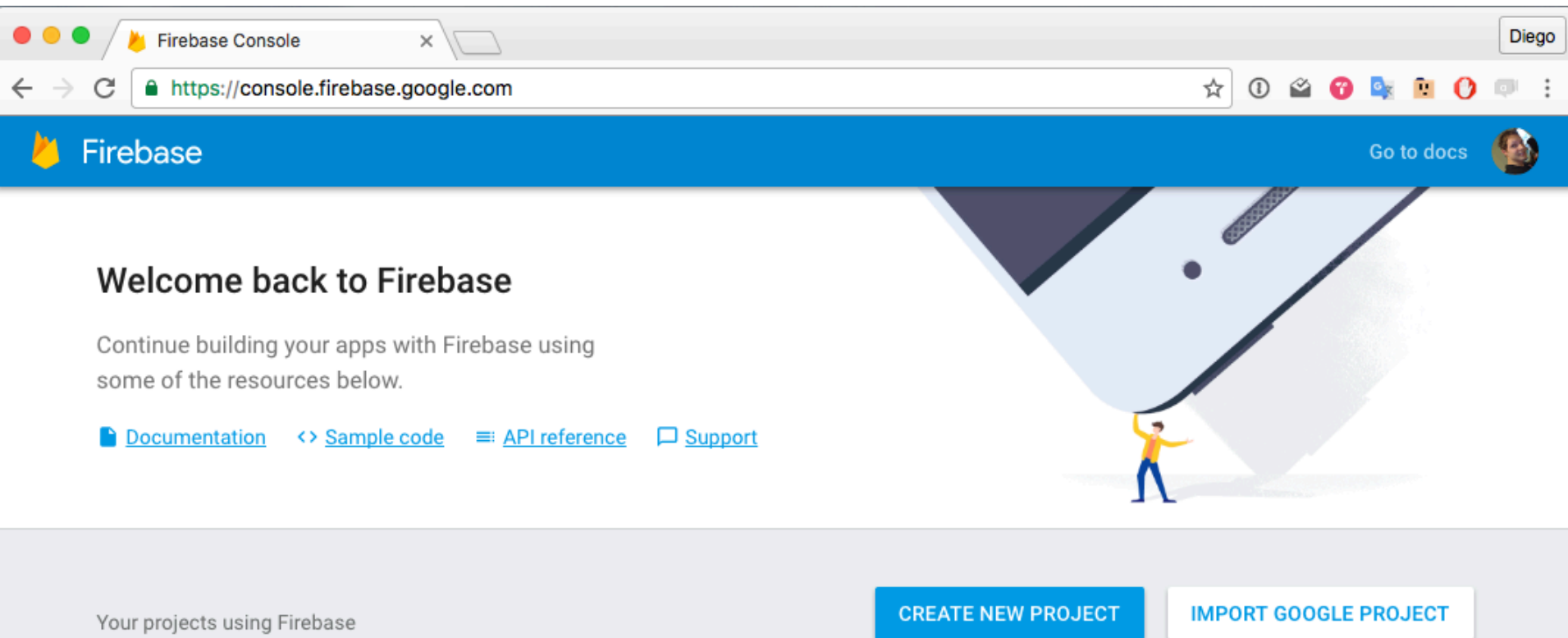
Firestore + Ionic

*Como estamos desenvolvendo aplicativos híbridos, vamos utilizar o **SDK Web** do Firestore.*

Obter os Dados de Acesso

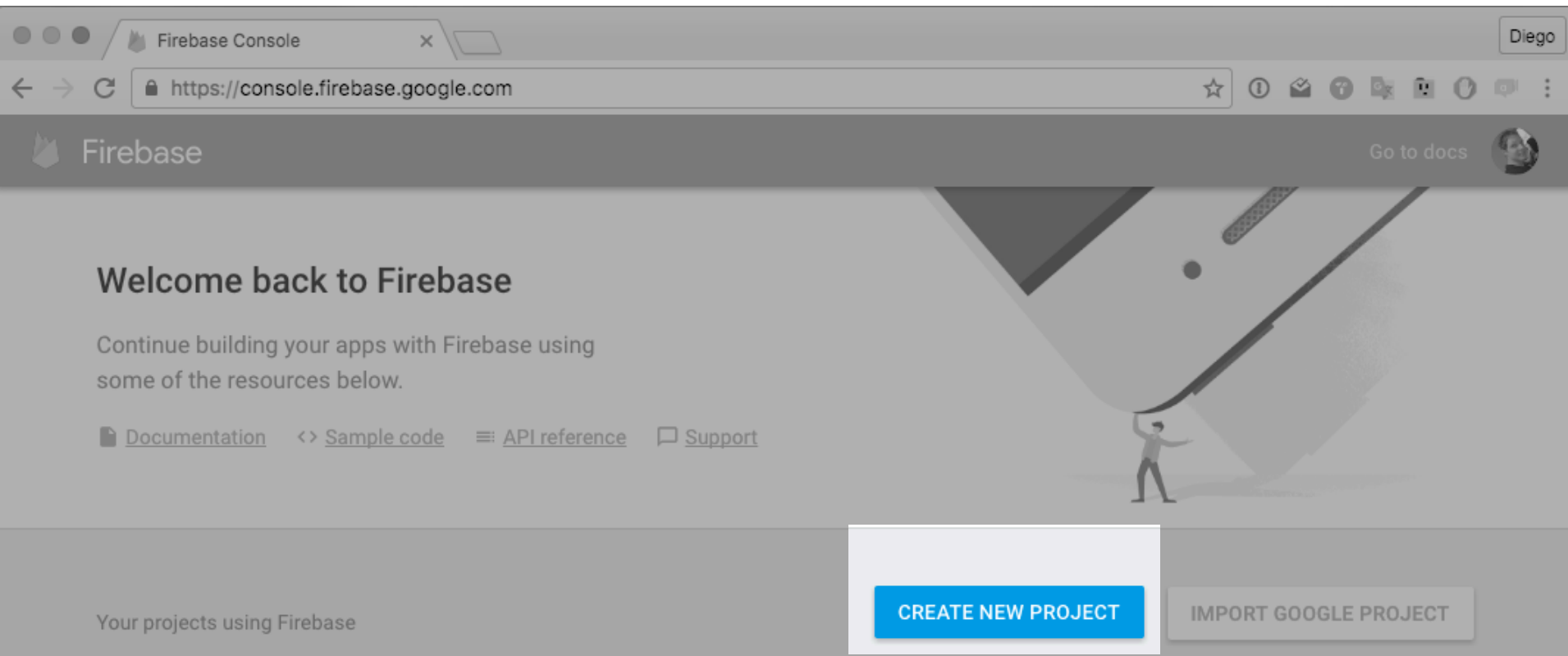
Configurações

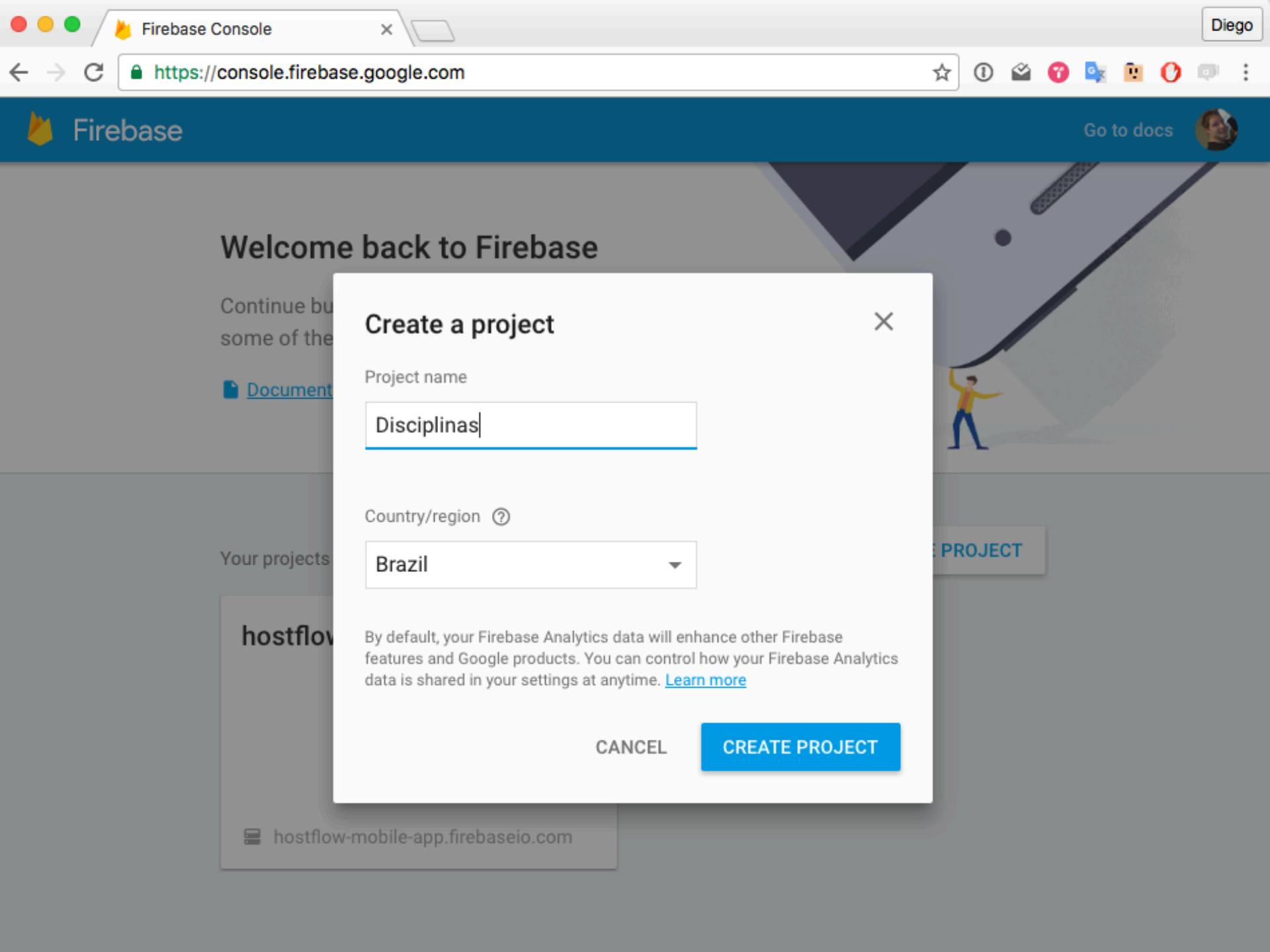
Você deve entrar em console.firebase.google.com para criar um novo projeto Firebase.



Configurações

Então você deve clicar em *Create New Project*, para gerar um novo projeto Firebase.






Welcome back to Firebase

Continue by
some of the

 [Document](#)

Your projects

hostflow

 hostflow-mobile-app.firebaseio.com

Create a project

Project name

Country/region ?

Brazil

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. [Learn more](#)

CANCEL

CREATE PROJECT

Diego

https://console.firebase.google.com/project/disciplinas-e4e86/overview

Firestore

Disciplinas

Go to docs

Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Overview

?

Welcome to Firebase! Get started here.

iOS

Add Firebase to your iOS app

Add Firebase to your Android app

</>

Add Firebase to your web app

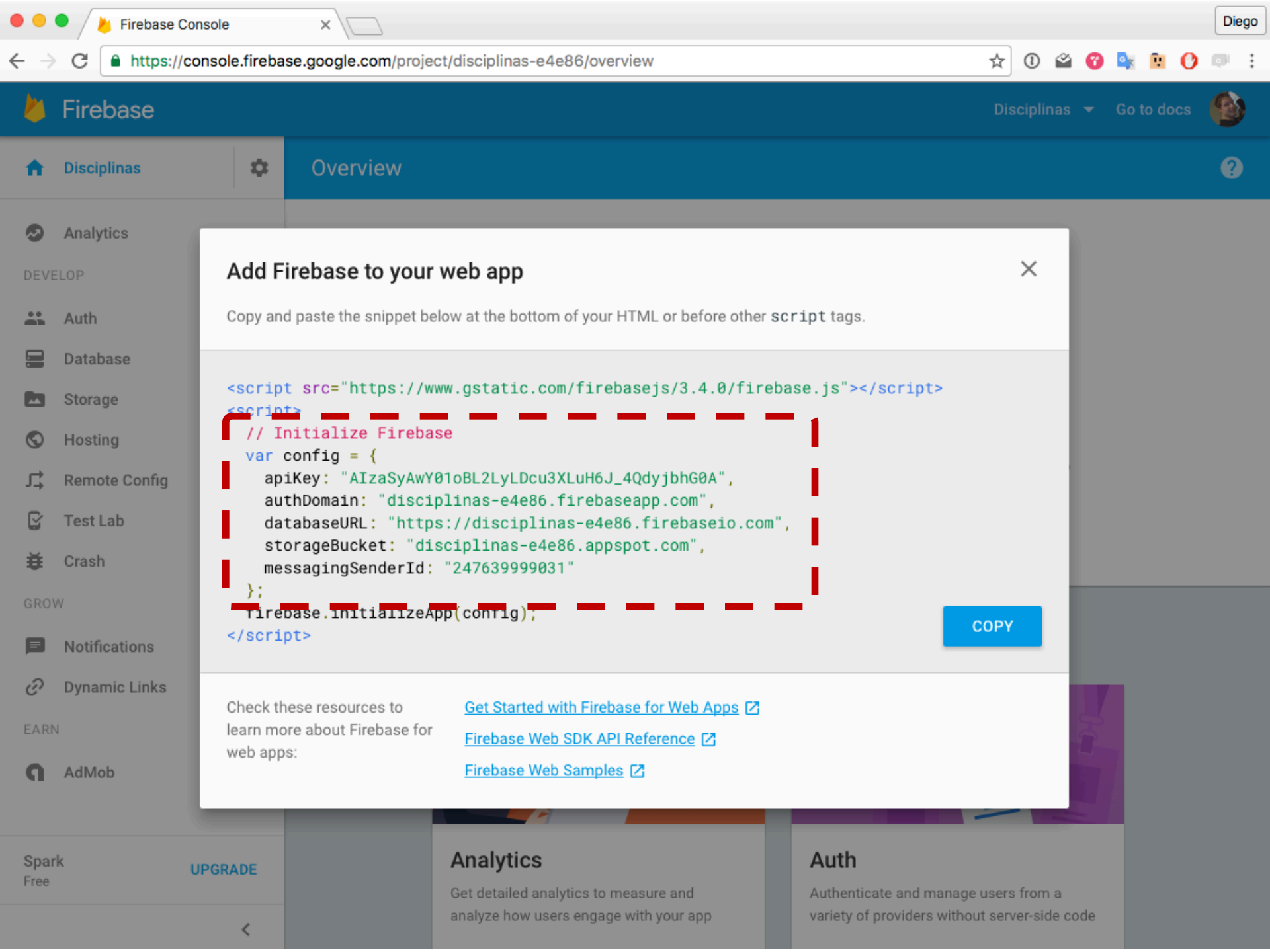
Discover Firebase

Analytics

Get detailed analytics to measure and analyze how users engage with your app

Auth

Authenticate and manage users from a variety of providers without server-side code



Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML or before other `script` tags.

```
<script src="https://www.gstatic.com/firebasejs/3.4.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyAwY01oBL2LyLDcu3XLuH6J_4QdyjbhG0A",
    authDomain: "disciplinas-e4e86.firebaseio.com",
    databaseURL: "https://disciplinas-e4e86.firebaseio.com",
    storageBucket: "disciplinas-e4e86.appspot.com",
    messagingSenderId: "247639999031"
  };
  firebase.initializeApp(config);
</script>
```

COPY

Check these resources to learn more about Firebase for web apps:

[Get Started with Firebase for Web Apps](#)

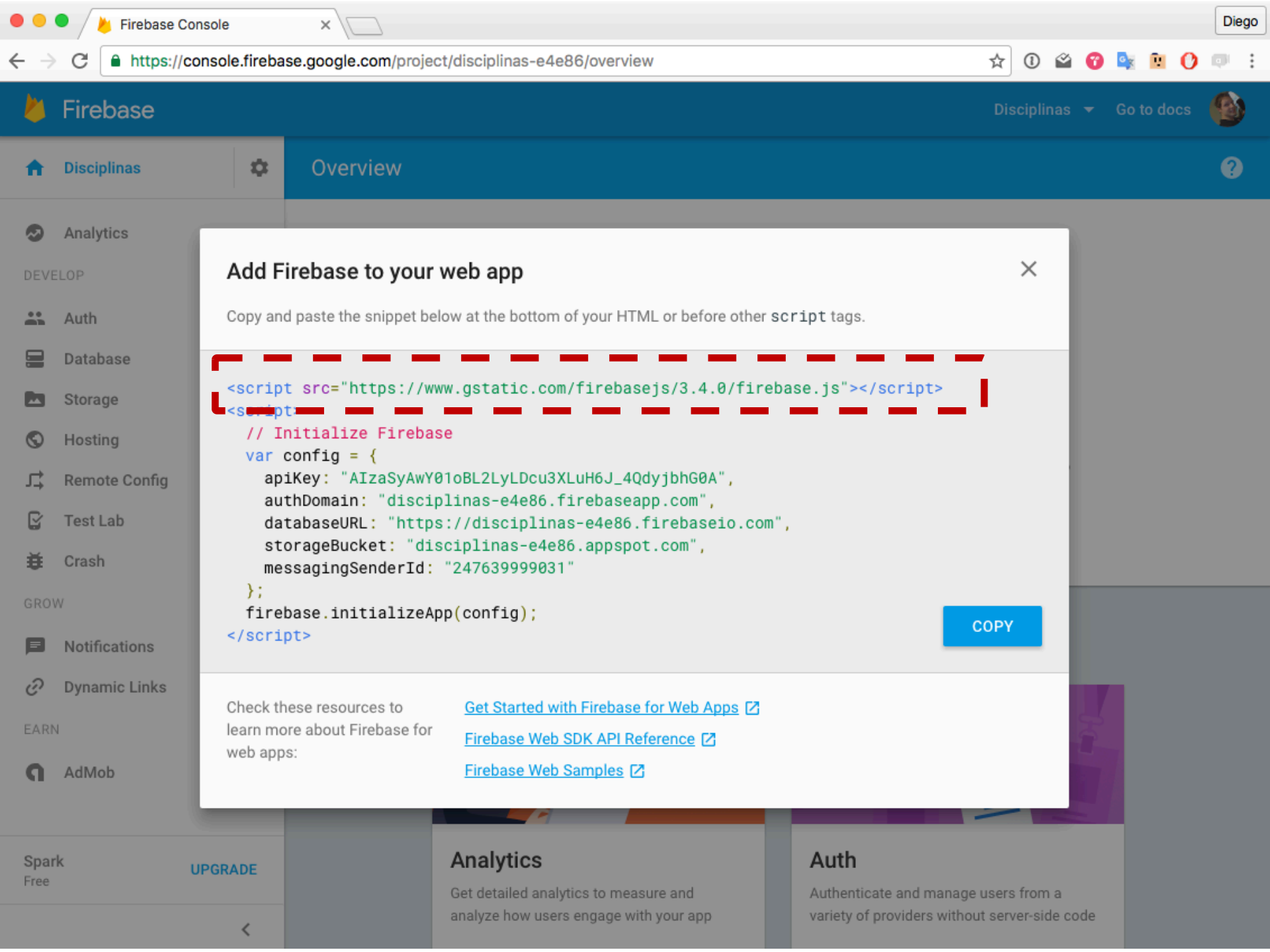
[Firebase Web SDK API Reference](#)

[Firebase Web Samples](#)

Configurar o Ionic

Inserir o SDK

Primeiro, vamos editar o arquivo [www/index.html](#) para incluir o script com o SDK do Firebase.



Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML or before other `script` tags.

```
<script src="https://www.gstatic.com/firebasejs/3.4.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyAwY01oBL2LyLDcu3XLuH6J_4QdyjbhG0A",
    authDomain: "disciplinas-e4e86.firebaseio.com",
    databaseURL: "https://disciplinas-e4e86.firebaseio.com",
    storageBucket: "disciplinas-e4e86.appspot.com",
    messagingSenderId: "247639999031"
  };
  firebase.initializeApp(config);
</script>
```

COPY

Check these resources to learn more about Firebase for web apps:

[Get Started with Firebase for Web Apps](#)

[Firebase Web SDK API Reference](#)

[Firebase Web Samples](#)

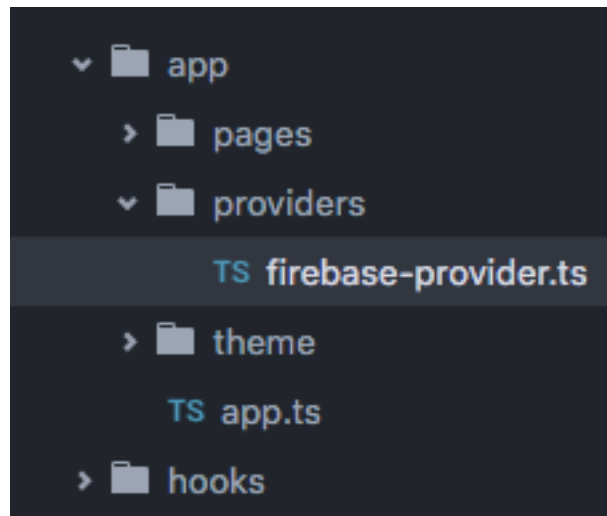
```
25 </head>
26
27 <body>
28   <ion-app></ion-app>
29
30   <!-- cordova.js required for cordova apps -->
31   <script src="cordova.js"></script>
32   <!-- Polyfill needed for platforms without Promise and Collection support -->
33   <script src="build/js/es6-shim.min.js"></script>
34   <!-- Zone.js and Reflect-metadata -->
35   <script src="build/js/Reflect.js"></script>
36   <script src="build/js/zone.js"></script>
37   <!-- the bundle which is built from the app's source code -->
38   <script src="build/js/app.bundle.js"></script>
39
40   <script src="https://www.gstatic.com/firebasejs/3.4.0/firebase.js"></script>
41
42 </body>
43
44 </html>
```

TypeScript

*Como o Firebase SDK consiste de uma biblioteca JavaScript Pura, ela não tem suporte ao TypeScript e neste caso não conseguimos utilizar o **import**.*

TypeScript

*Para resolver este problema, você pode baixar do Moodle a classe chamada **firebase-provider.ts**. Então você pode incluí-la em uma pasta chamada **providers**, dentro do diretório **app**.*



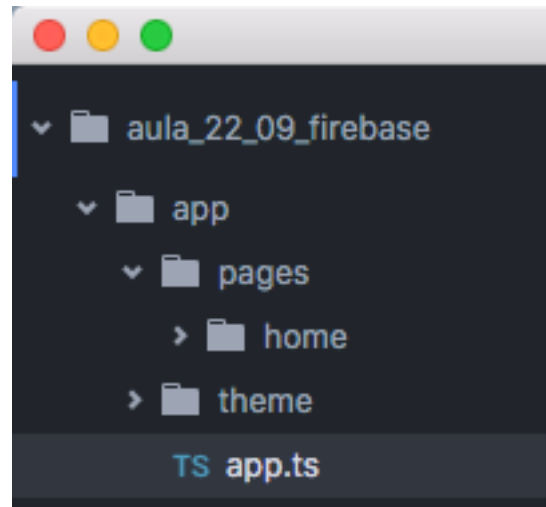
Configuração

*Então você deve editar o arquivo **firebase-provider.ts** e inserir as suas credenciais geradas no console do Firebase, conforme a imagem a seguir.*


```
1  import { Injectable } from '@angular/core';
2  import 'rxjs/add/operator/map';
3
4  declare var firebase: any;
5
6  @Injectable()
7  export class FirebaseProvider {
8
9      constructor() {
10         // Inserir as suas credenciais geradas no console.firebase.com
11         let config = {
12             apiKey: "<sua_chave>",
13             authDomain: "<seu_dominio>",
14             databaseURL: "<sua_url>",
15             storageBucket: "<seu_storage>",
16             messagingSenderId: "<seu_id>"
17         };
18         // Inicializar o firebase com suas credenciais
19         firebase.initializeApp(config);
20     }
21 }
```

Configuração

*Agora, vamos abrir e editar o arquivo **app.ts** para informar ao Ionic que o Firebase é um módulo disponível em todo o aplicativo como um provider (fornecedor de serviço).*



```
1  import { Component } from '@angular/core';
2  import { ionicBootstrap, Platform } from 'ionic-angular';
3  import { StatusBar } from 'ionic-native';
4
5  import { FirebaseProvider } from '../providers/firebase-provider';
6
7  import { LoginPage } from '../pages/login/login';
8
9  @Component({
10     template: '<ion-nav [root]="rootPage"></ion-nav>'
11 })
12 export class MyApp {
13     rootPage: any = LoginPage;
```

```
12  export class MyApp {
13      rootPage: any = LoginPage;
14
15      constructor(public platform: Platform) {
16          platform.ready().then(() => {
17              // Okay, so the platform is ready and our plugins are available.
18              // Here you can do any higher level native things you might need.
19              StatusBar.styleDefault();
20          });
21      }
22  }
23
24  ionicBootstrap(MyApp, [FirebaseProvider]);
```

Autenticação

Autenticação

Como estamos trabalhando com um banco de dados online, precisamos criar uma maneira de autenticar os usuários para filtrar as informações (dados) por usuário.

Autenticação

O Firebase fornece diversos providers para autenticação, por exemplo, o Facebook, Twitter, Google e Password.

Autenticação

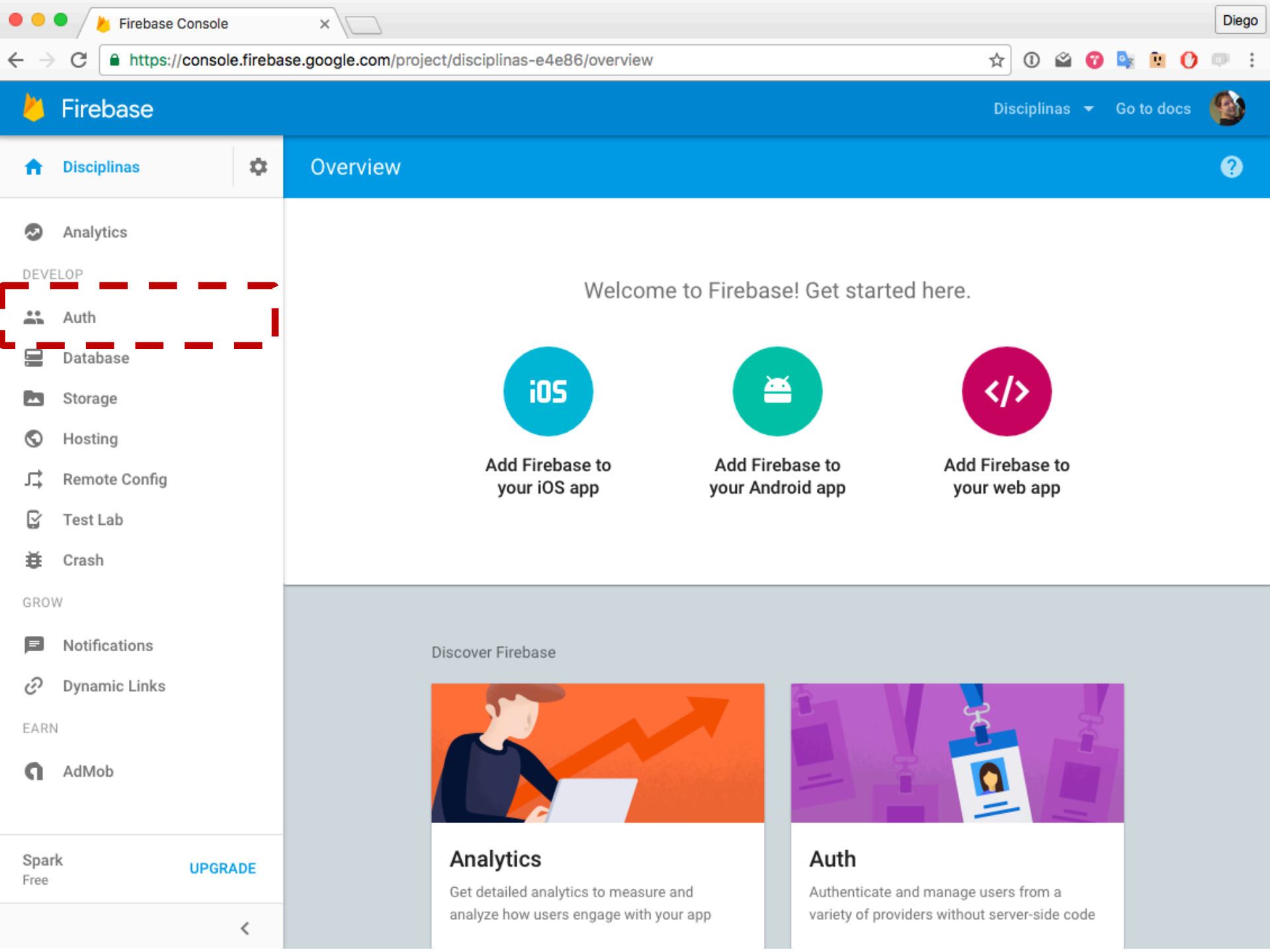
Neste momento, vamos aprender a utilizar o provider Password, no qual o usuário precisa se registrar informando um email e uma senha.

Autenticação

Vamos trabalhar com os providers sociais, como o Facebook, juntamente com os recursos nativos, para realizar a autenticação por meio do aplicativo nativo.

Como configurar

*Primeiramente, precisamos **habilitar** o método de login que desejamos utilizar **no Firebase Console**.*



Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Authentication


USERS

SIGN-IN METHOD

EMAIL TEMPLATES

Search by exact email address (email@domain.com) or user UID

ADD USER

Email	Providers	Created	Signed In	User UID
<div><div></div><div><div>Authenticate and manage users from a variety of providers without server-side code</div><div>Learn more</div><div>SET UP SIGN-IN METHOD</div></div></div>				

Need help setting up your app? iOS

Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Disciplinas

Go to docs

Diego

Authentication

USERS

SIGN-IN METHOD

EMAIL TEMPLATES

WEB SETUP

Sign-in providers

Provider	Status
Email/Password	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

OAuth redirect domains

Authorized domain	Type
localhost	Default

ADD DOMAIN

Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Disciplinas

Go to docs

WEB SETUP

Authentication

USERS

SIGN-IN METHOD

EMAIL TEMPLATES

Sign-in providers

Provider	Status
Email/Password	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

OAuth redirect domains

Authorized domain	Type
localhost	Default

Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Authentication

USERS

SIGN-IN METHOD

EMAIL TEMPLATES

Sign-in providers

Provider	Status
Email/Password	<div>Enable <input type="checkbox"/></div> <div>Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more</div>
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

WEB SETUP

Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark Free

UPGRADE

Disciplinas

Authentication

WEB SETUP

USERS

SIGN-IN METHOD

EMAIL TEMPLATES

Sign-in providers

Provider	Status
Email/Password	Enabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

OAuth redirect domains

Authorized domain	Type
localhost	Default

ADD DOMAIN

Vamos codificar

Agora com o método de autenticação habilitado no console do Firebase, podemos começar a codificar os métodos de login.

Página de Login

Usando o comando *ionic g page login*, vamos criar uma página específica para realizar o registro e a autenticação dos usuários.

Página de Login

No *login.ts* vamos importar o Firebase e adicionar ao construtor da classe para termos uma referência ao SDK na classe *LoginPage*.

```
1  import { Component } from '@angular/core';
2  import { NavController } from 'ionic-angular';
3
4  import { FirebaseProvider } from '../providers/firebase-provider';
5
6  @Component({
7      templateUrl: 'build/pages/login/login.html',
8  })
9  export class LoginPage {
10
11      constructor(private nav: NavController, private fire: FirebaseProvider) { }
12
13  }
14
```

```
1  import { Component } from '@angular/core';
2  import { NavController } from 'ionic-angular';
3
4  import { FirebaseProvider } from '../../../providers/firebase-provider';
5
6  @Component({
7      templateUrl: 'build/pages/login/login.html',
8  })
9  export class LoginPage {
10
11      constructor(private nav: NavController, private fire: FirebaseProvider) { }
12
13  }
14
```

Página de Login

No *login.html* vamos criar uma página simples para informar campos de *email* e *senha*, e dois botões para *entrar* e *registrar* um novo usuário.

```
1  <ion-header>
2    <ion-navbar primary>
3      <ion-title>Login</ion-title>
4    </ion-navbar>
5  </ion-header>
6
7  <ion-content padding>
8    <ion-label floating>Email</ion-label>
9    <ion-input type="email" [(ngModel)]="usuario.email"></ion-input>
10
11    <ion-label floating>Senha</ion-label>
12    <ion-input type="password" [(ngModel)]="usuario.senha"></ion-input>
13
14    <button block primary (click)="entrar()">Entrar</button>
15    <button block outline (click)="registrar()">Registrar</button>
16
17  </ion-content>
18  |
```

```
1  <ion-header>
2    <ion-navbar primary>
3      <ion-title>Login</ion-title>
4    </ion-navbar>
5  </ion-header>
6
7  <ion-content padding>
8    <ion-label floating>Email</ion-label>
9    <ion-input type="email" [(ngModel)]="usuario.email"></ion-input>
10
11    <ion-label floating>Senha</ion-label>
12    <ion-input type="password" [(ngModel)]="usuario.senha"></ion-input>
13
14    <button block primary (click)="entrar()">Entrar</button>
15    <button block outline (click)="registrar()">Registrar</button>
16
17  </ion-content>
```



```
1  <ion-header>
2    <ion-navbar primary>
3      <ion-title>Login</ion-title>
4    </ion-navbar>
5  </ion-header>
6
7  <ion-content padding>
8    <ion-label floating>Email</ion-label>
9    <ion-input type="email" [(ngModel)]="usuario.email"></ion-input>
10
11    <ion-label floating>Senha</ion-label>
12    <ion-input type="password" [(ngModel)]="usuario.senha"></ion-input>
13
14    <button block primary (click)="entrar()">Entrar</button>
15    <button block outline (click)="registrar()">Registrar</button>
16
17  </ion-content>
```

Página de Login

Agora vamos codificar os métodos *entrar* e *registrar* usando a API do Firebase no *login.ts*. Primeiro, definimos as variáveis que serão utilizadas.

```
10  export class LoginPage {
11
12      protected usuario = { email: '', senha: '' };
13      protected firebase: any;
14
15      constructor(private nav: NavController, private fire: FirebaseProvider) {
16          this.firebase = this.fire.db();
17      }
18
```

Página de Login

Depois, no construtor, inicializamos a variável firebase com a instância do banco de dados.

```
10  export class LoginPage {
11
12      protected usuario = { email: '', senha: '' };
13      protected firebase: any;
14
15      constructor(private nav: NavController, private fire: FirebaseProvider) {
16          this.firebase = this.fire.db();
17      }
18  }
```

Página de Login

Agora, vamos implementar um método para registrar um novo usuário no Firebase. Após criar um novo usuário, o Firebase faz seu login automático no sistema.

```
19  registrar() {
20      // Checa se os dados não estão vazios
21      if (this.usuario.email !== '' && this.usuario.senha !== '') {
22          // Tenta criar o cadastro
23          this.firebase.auth().createUserWithEmailAndPassword(this.usuario.email, this.usuario.senha)
24              // Retorna um erro se não conseguir criar
25              .catch((error) => { console.log(error.message) });
26      }
27  }
```

Login

Email

drantunes@gmail.com

Senha

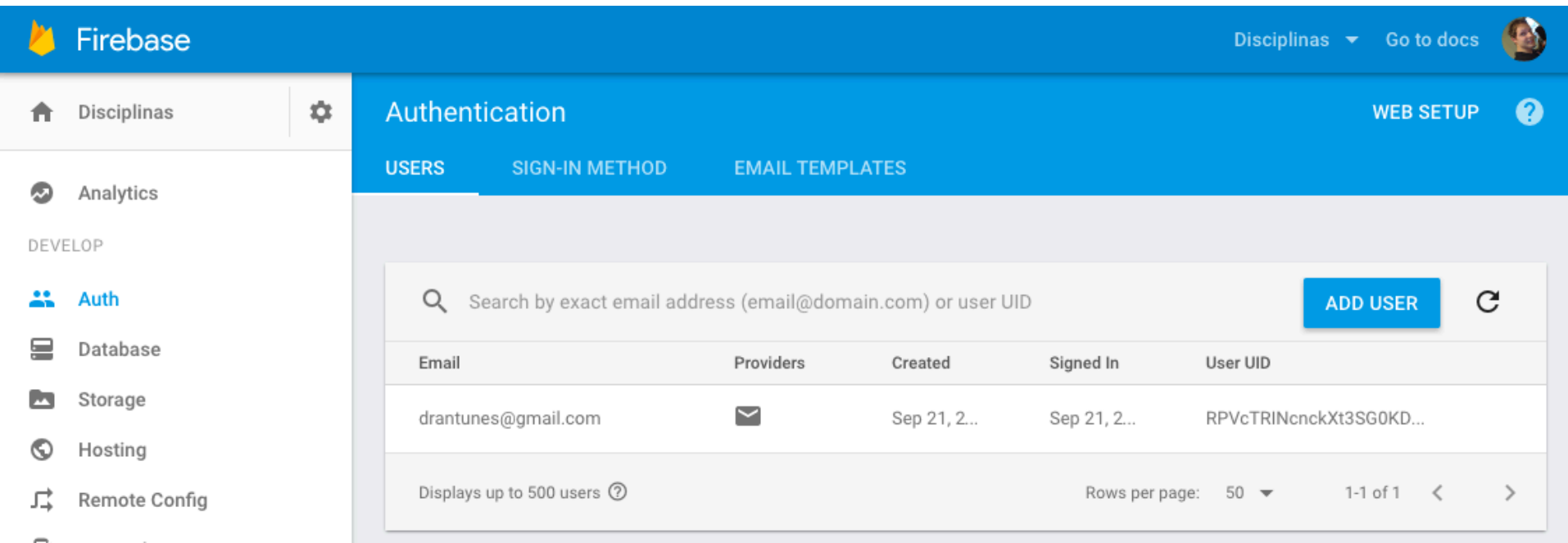
.....|

ENTRAR

REGISTRAR

Página de Login

Ao clicar em registrar, o Firebase insere um novo usuário na base de dados.




The screenshot shows the Firebase Authentication console. The left sidebar contains navigation links for Disciplinas, Analytics, DEVELOP, Auth, Database, Storage, Hosting, and Remote Config. The main content area is titled 'Authentication' and has tabs for USERS, SIGN-IN METHOD, and EMAIL TEMPLATES. The USERS tab is active, displaying a table with one user. Above the table is a search bar and an 'ADD USER' button. The table has columns for Email, Providers, Created, Signed In, and User UID. The user listed is drantunes@gmail.com, created on Sep 21, 2023, and signed in on the same date. The User UID is RPVcTRINcnckXt3SG0KD... At the bottom, it indicates 'Displays up to 500 users' and 'Rows per page: 50'.

Disciplinas Go to docs

Authentication WEB SETUP

USERS SIGN-IN METHOD EMAIL TEMPLATES

Search by exact email address (email@domain.com) or user UID **ADD USER**

Email	Providers	Created	Signed In	User UID
drantunes@gmail.com		Sep 21, 2...	Sep 21, 2...	RPVcTRINcnckXt3SG0KD...

Displays up to 500 users Rows per page: 50 1-1 of 1

Pergunta.

Como verificar se o usuário já está logado?

Por exemplo, no app.ts poderíamos verificar se o usuário já está logado no Firebase e então redirecionar ele para outra tela diferente do Login.

Resposta

No app.ts podemos fazer uma verificação no construtor.

```
16     constructor(public platform: Platform, public fire: FirebaseProvider) {  
17         // Fica aguardando um evento de mudança de estado do Login  
18         this.fire.db().auth().onAuthStateChanged((user) => {  
19             // Se houver um usuário envia par ao H0mePage  
20             if (user) {  
21                 this.rootPage = HomePage;  
22             } // Senão, envia para o Login.  
23             else {  
24                 this.rootPage = LoginPage;  
25             }  
26         });
```


Resposta

No app.ts podemos fazer uma verificação no construtor.

```
16 constructor(public platform: Platform, public fire: FirebaseProvider) {  
17     // Fica aguardando um evento de mudança de estado do Login  
18     this.fire.db().auth().onAuthStateChanged((user) => {  
19         // Se houver um usuário envia par ao H0mePage  
20         if (user) {  
21             this.rootPage = HomePage;  
22         } // Senão, envia para o Login.  
23         else {  
24             this.rootPage = LoginPage;  
25         }  
26     });
```

Resposta

No app.ts podemos fazer uma verificação no construtor.

```
16     constructor(public platform: Platform, public fire: FirebaseProvider) {
17         // Fica aguardando um evento de mudança de estado do Login
18         this.fire.db().auth().onAuthStateChanged((user) => {
19             // Se houver um usuário envia par ao H0mePage
20             if (user) {
21                 this.rootPage = HomePage;
22             } // Senão, envia para o Login.
23             else {
24                 this.rootPage = LoginPage;
25             }
26         });
```

Método de Logout

Podemos criar um método de logout simplesmente executando a instrução:

```
16     sair() {  
17         // Método para Deslogar  
18         this.firebase.auth().signOut().then(() => {  
19             // Muda a navegação para Login  
20             this.nav.setRoot(LoginPage);  
21         }, (error) => {  
22             console.log(error);  
23         });  
24     }
```

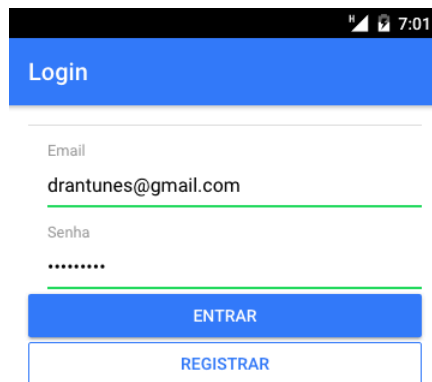
Método de Login

*O método de login (entrar) com usuário e senha na classe **login.ts**, funciona similar ao registrar:*

```
30  entrar() {  
31      // Checa se os dados não estão vazios  
32      if (this.usuario.email !== '' && this.usuario.senha !== '') {  
33          // Tenta logar o usuário  
34          this.firebase.auth().signInWithEmailAndPassword(this.usuario.email, this.usuario.senha)  
35          // Retorna um erro se não conseguir criar  
36          .catch((error) => { console.log(error.message) });  
37      }  
38  }
```

Como pegar dados do usuário

Em diversas páginas, como o home.ts, podemos desejar mostrar informações do usuário logado. Para isso fazemos:



Mockup of a mobile app login screen. The status bar at the top shows signal strength, battery, and the time 7:01. Below the status bar is a blue header with the word "Login" in white. The main area contains two input fields: "Email" with the value "drantunes@gmail.com" and "Senha" with masked characters "*****". Below the input fields are two buttons: a blue "ENTRAR" button and a white "REGISTRAR" button with a blue border. The bottom of the screen features a black navigation bar with three white icons: a back arrow, a circle, and a square.



Mockup of a mobile app home screen. The status bar at the top shows signal strength, battery, and the time 7:01. Below the status bar is a light gray header with the word "Home" and a close icon (X). The main area displays the text "Olá, drantunes@gmail.com". The bottom of the screen features a black navigation bar with three white icons: a back arrow, a circle, and a square.

Como pegar dados do usuário

Em diversas páginas, como o [home.ts](#), podemos desejar mostrar informações do usuário logado. Para isso fazemos:

```
9  export class HomePage {
10      protected firebase: any;
11      protected usuario: any = [];
12
13      constructor(private nav: NavController, private fire: FirebaseProvider) {
14          this.firebase = this.fire.db();
15
16          this.usuario = this.firebase.auth().currentUser;
17      }
```

Como pegar dados do usuário

No home.html podemos mostrar os dados do usuário. Para mais detalhes como atualizar um usuário, remover um usuário, entre outros, acessar:

<https://firebase.google.com/docs/auth/web/manage-users>

<https://firebase.google.com/docs/auth/web/password-auth>

```
14 <ion-content padding>
15   Olá, {{ usuario.email }}
16 </ion-content>
```

Banco de Dados

Estrutura de Dados

No Firebase os dados são estruturados como objetos JSON, estruturados na forma de uma “árvore”. Esses objetos sempre são estruturados como chave-valor.

 <https://disciplinas-e4e86.firebaseio.com/>



- ★ Default security rules require users to be authenticated

[LEARN MORE](#)

DISMISS

disciplinas-e4e86:	null
--------------------	------



Store and sync data in realtime across all connected clients

 [Learn more](#)

Realtime

Os dados no Firebase são armazenados em um banco NoSQL e são sincronizados em tempo real entre todos os clientes conectados no banco.

Como estruturar os dados

No geral, devemos estruturar os nossos dados baseado em identificadores únicos de cada usuário. Assim torna-se mais simples recuperar e gerenciar dados para cada usuário.

<https://firebase.google.com/docs/database/web/structure-data>

Como manipular os dados

*No Firebase precisamos sempre acessar os dados como referências, indicando o “caminho” para encontrar o dado que desejamos manipular. Por exemplo: **/users/id/email***

<https://firebase.google.com/docs/database/web/structure-data>

Gravando Dados

Method	Common uses
set()	Write or replace data to a defined path, such as users/<user-id>/<username> .
push()	Add to a list of data. Every time you call push() , Firebase generates a unique key that can also be used as a unique identifier, such as user-posts/<user-id>/<unique-post-id> .
update()	Update some of the keys for a defined path without replacing all of the data.
transaction()	Update complex data that could be corrupted by concurrent updates.

<https://firebase.google.com/docs/database/web/save-data>

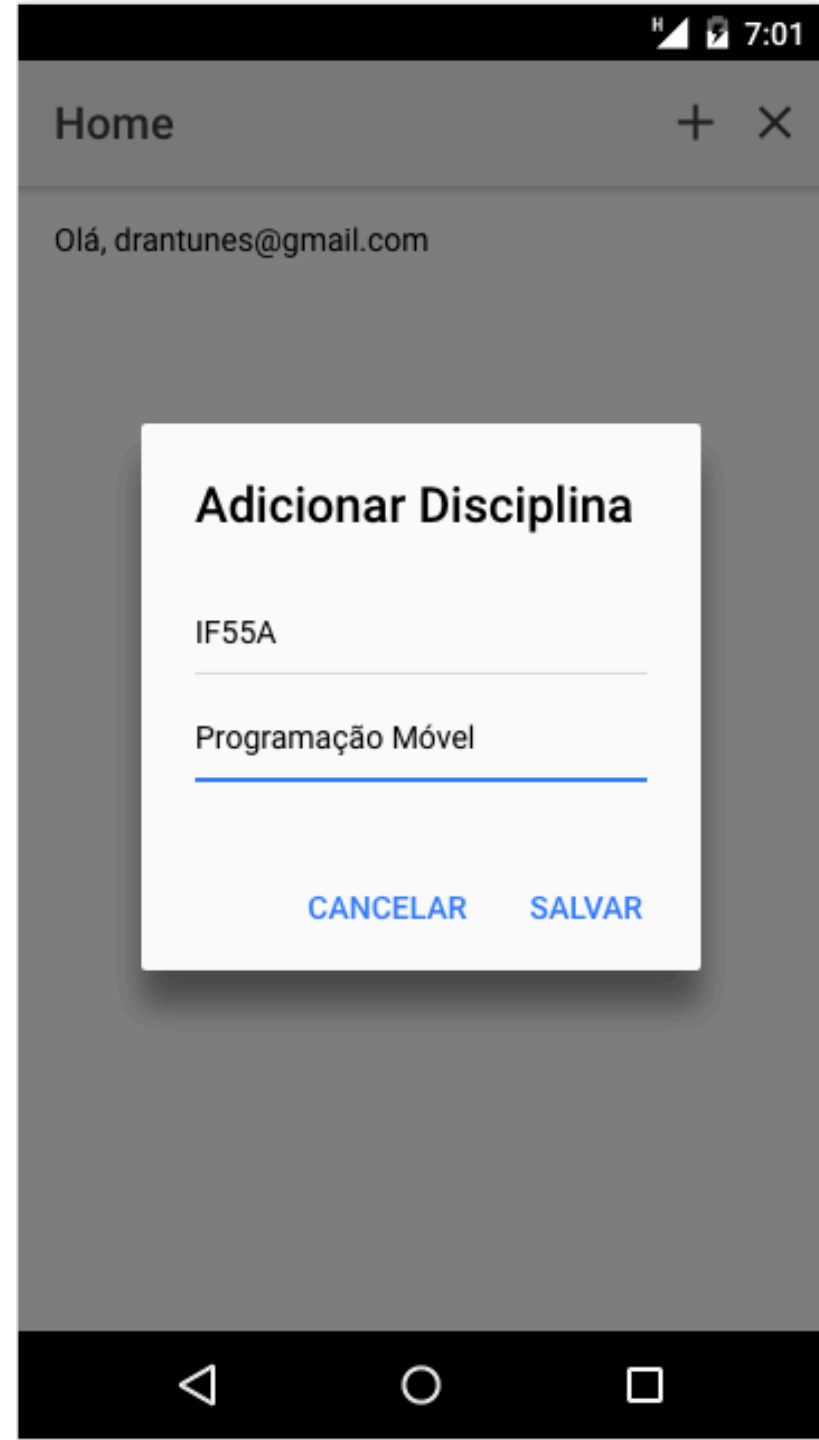
Adicionando uma disciplina

*Usando nosso exemplo, vamos inserir no Firebase uma nova Disciplina usando o componente de Alert. Vamos inserir na seguinte referência: **id_usuario/disciplinas/codigo***

Primeiro, vamos criar um Alert para solicitar as informações da disciplina do usuário.

```
21  adicionar() {  
22      let add = this.alert.create({  
23          title: 'Adicionar Disciplina',  
24          inputs: [  
25              { type: 'text', name: 'codigo', placeholder: 'Codigo' },  
26              { type: 'text', name: 'nome', placeholder: 'Nome' }  
27          ],  
28          buttons: [  
29              { text: 'Cancelar' },  
30              { text: 'Salvar', handler: (dados) => this.add(dados) }  
31          ]  
32      });  
33      add.present();  
34  }
```


*Esta tela irá abrir o alert e solicitar as informações ao usuário. Ao clicar em **salvar**, chamamos a função **add** e passamos os dados do formulário como parâmetro.*



Primeiro, vamos definir qual o caminho que desejamos gravar os dados. Em nosso caso será:

id_usuario/disciplinas/código

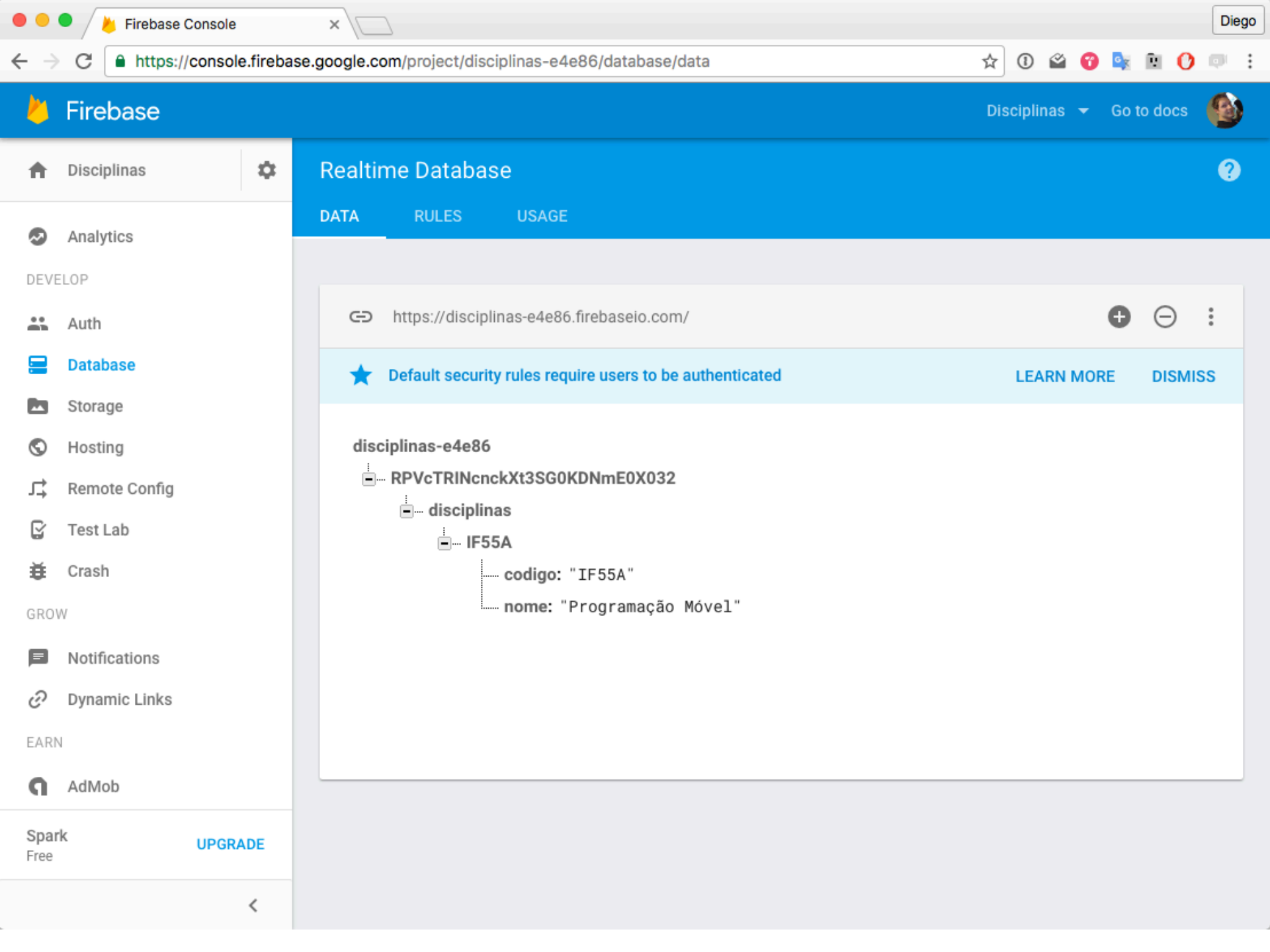
```
36 add(dados) {  
37   // Caminho ou Referencia de onde queremos gravar os dados  
38   let caminho = this.usuario.uid + '/disciplinas/' + dados.codigo;  
39   // O objeto de dados que desejamos gravar  
40   let objeto = {  
41     codigo: dados.codigo,  
42     nome: dados.nome  
43   };  
44   // Chamamos a referência e usamos o método set para gravar  
45   this.firebase.database().ref(caminho).set(objeto);  
46 }
```

Então, vamos criar um objeto com os *dados* enviados por parâmetro, que contém o *código* e o *nome* da disciplina.

```
36  add(dados) {  
37      // Caminho ou Referencia de onde queremos gravar os dados  
38      let caminho = this.usuario.uid + '/disciplinas/' + dados.codigo;  
39      // O objeto de dados que desejamos gravar  
40      let objeto = {  
41          codigo: dados.codigo,  
42          nome: dados.nome  
43      };  
44      // Chamamos a referencia e usamos o metodo set para gravar  
45      this.firebase.database().ref(caminho).set(objeto);  
46  }
```

Em seguida, vamos chamar o método `database()`, passar a referência do caminho de gravação e então chamar o método `set` passando o `objeto` como parâmetro.

```
36  add(dados) {  
37      // Caminho ou Referencia de onde queremos gravar os dados  
38      let caminho = this.usuario.uid + '/disciplinas/' + dados.codigo;  
39      // O objeto de dados que desejamos gravar  
40      let objeto = {  
41          codigo: dados.codigo,  
42          nome: dados.nome  
43      };  
44      // Chamamos a referência e usamos o método set para gravar  
45      this.firebase.database().ref(caminho).set(objeto);  
46  }
```



Disciplinas

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Crash

GROW

Notifications

Dynamic Links

EARN

AdMob

Spark
Free

UPGRADE

Realtime Database

DATA

RULES

USAGE

https://disciplinas-e4e86.firebaseio.com/

★ Default security rules require users to be authenticated

LEARN MORE

DISMISS

disciplinas-e4e86

RPVcTRINcnckXt3SG0KDNmE0X032

disciplinas

IF55A

codigo: "IF55A"

nome: "Programação Móvel"

Recuperando dados

*Como o Firebase é um banco de dados em tempo real, a recuperação dos dados é um pouco diferente. Vamos utilizar um evento **on** que fica aguardando mudanças no banco e então retorna um **snapshot** do banco a cada mudança.*

Recuperando dados

Este método é chamado no construtor da classe. Então, a cada mudança no banco, o Firebase irá executar esta função automaticamente.

```
24 checkDisciplinas() {  
25     // Caminho ou Referencia de onde queremos recuperar os dados  
26     let caminho = this.usuario.uid + '/disciplinas';  
27     // Na referência usamos o método ON para esperar mudanças de valor  
28     this.firebaseio().ref(caminho).on('child_added', (snapshot) => {  
29         // Então atribuímos o valor do snapshot à lista de disciplinas  
30         this.disciplinas.push(snapshot.val());  
31     });  
32 }
```

Recuperando dados

No exemplo, quando um item for adicionado (*child_added*) em *id_usuario/disciplinas*, a função é disparada e podemos atualizar nossa lista de disciplinas.

```
24 checkDisciplinas() {  
25     // Caminho ou Referencia de onde queremos recuperar os dados  
26     let caminho = this.usuario.uid + '/disciplinas';  
27     // Na referência usamos o método ON para esperar mudanças de valor  
28     this.firebase.database().ref(caminho).on('child_added', (snapshot) => {  
29         // Então atribuímos o valor do snapshot à lista de disciplinas  
30         this.disciplinas.push(snapshot.val());  
31     });  
32 }
```


Recuperando dados

Então usamos o retorno da função, *snapshot.val()*, e realizamos um *push* em nossa lista de disciplinas.

```
24 checkDisciplinas() {  
25     // Caminho ou Referencia de onde queremos recuperar os dados  
26     let caminho = this.usuario.uid + '/disciplinas';  
27     // Na referência usamos o método ON para esperar mudanças de valor  
28     this.firebase.database().ref(caminho).on('child_added', (snapshot) => {  
29         // Então atribuímos o valor do snapshot à lista de disciplinas  
30         this.disciplinas.push(snapshot.val());  
31     });  
32 }
```

Recuperando dados

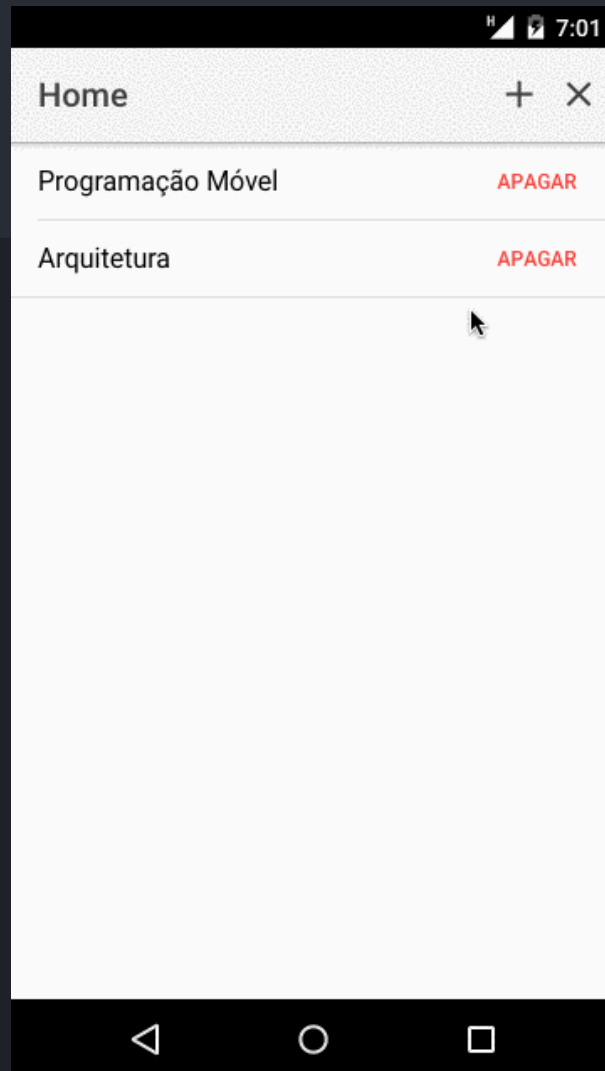
Portanto, cada vez que adicionarmos uma nova disciplina, a lista de disciplinas será atualizada automaticamente.

```
24 checkDisciplinas() {  
25     // Caminho ou Referencia de onde queremos recuperar os dados  
26     let caminho = this.usuario.uid + '/disciplinas';  
27     // Na referência usamos o método ON para esperar mudanças de valor  
28     this.firebase.database().ref(caminho).on('child_added', (snapshot) => {  
29         // Então atribuímos o valor do snapshot à lista de disciplinas  
30         this.disciplinas.push(snapshot.val());  
31     });  
32 }
```

```

19 <ion-list>
20   <ion-item *ngFor="let disciplina of disciplinas; let i = index">
21     {{ disciplina.nome }}
22     <button clear item-right danger (click)="remover(disciplina.codigo, i)">
23       Apagar
24     </button>
25   </ion-item>
26 </ion-list>

```



Removendo dados

Basta chamar a função *remove* em uma referência. Além disso, precisamos remover da lista local usando o *splice*.

```
35  remover(codigo, index) {  
36      // Caminho ou Referencia de onde queremos recuperar os dados  
37      let caminho = this.usuario.uid + '/disciplinas/' + codigo;  
38      // Na referência usamos o método ON para esperar mudanças de valor  
39      this.firebase.database().ref(caminho).remove();  
40      // Deletamos da lista local  
41      this.disciplinas.splice(index, 1);  
42  }
```

Atualizando dados

Basta chamar a função *update* em uma referência e passar o objeto com os dados que serão atualizados.

```
35  update(i) {  
36      // Caminho ou Referencia de onde queremos recuperar os dados  
37      let caminho = this.usuario.uid + '/disciplinas/' + this.disciplinas[i].codigo;  
38      // Na referência usamos o método ON para esperar mudanças de valor  
39      this.firebase.database().ref(caminho).update({  
40          codigo: this.disciplinas[i].codigo,  
41          nome: this.disciplinas[i].nome  
42      });  
43  }
```

Atualizando dados

Então, basta você alterar os dados usando o `[(ngModel)]` e então chamar a função atualizar passando o `index` do item a ser atualizado ou mesmo todo o objeto.

```
35 update(i) {  
36     // Caminho ou Referencia de onde queremos recuperar os dados  
37     let caminho = this.usuario.uid + '/disciplinas/' + this.disciplinas[i].codigo;  
38     // Na referência usamos o método ON para esperar mudanças de valor  
39     this.firebase.database().ref(caminho).update({  
40         codigo: this.disciplinas[i].codigo,  
41         nome: this.disciplinas[i].nome  
42     });  
43 }
```

Mais exemples

<https://firebase.google.com/docs/database/web/save-data>

<https://firebase.google.com/docs/database/web/retrieve-data>