
UNITAU
Universidade de Taubaté

Firebase

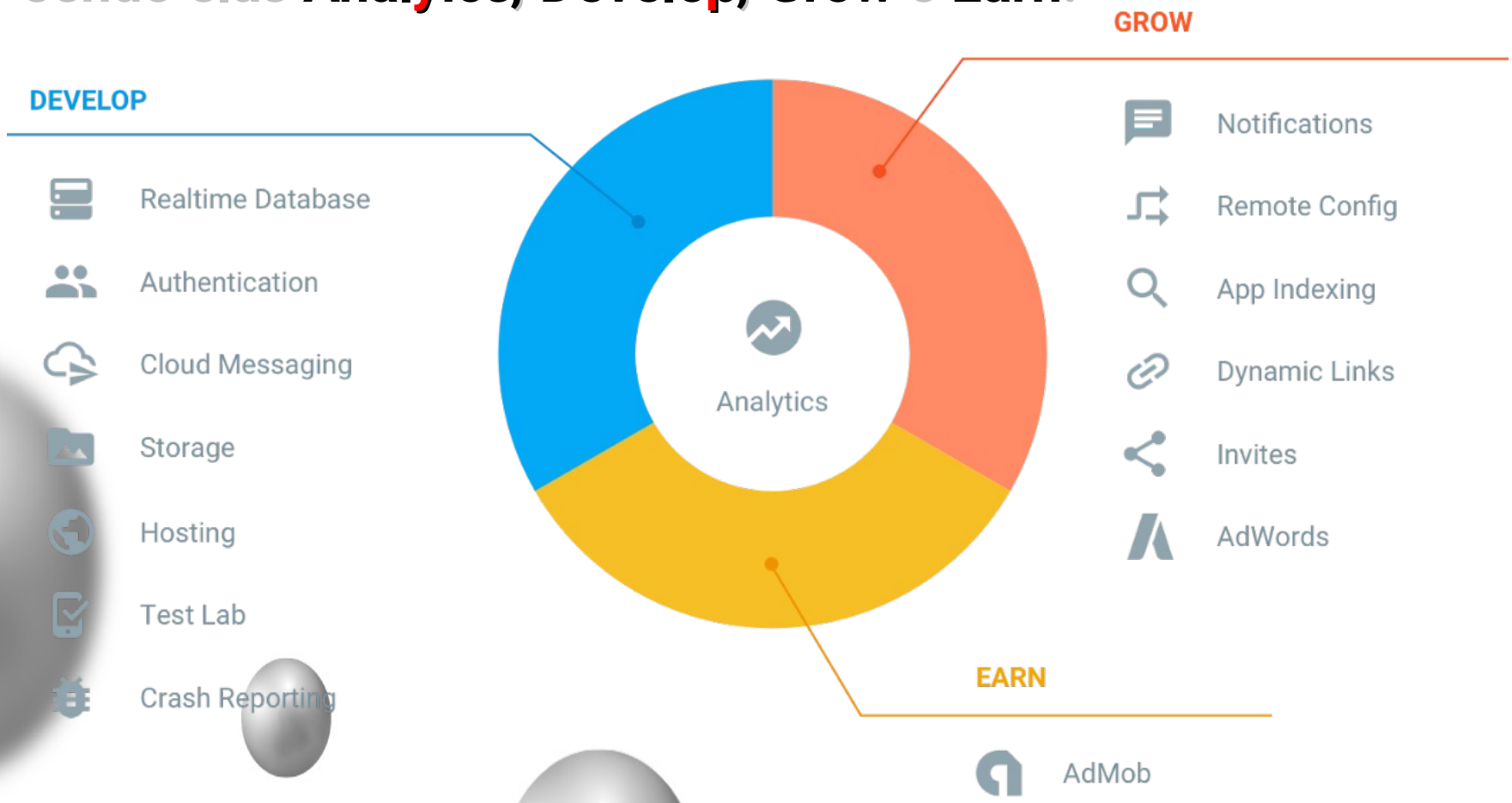
Firebase

“Firebase é uma plataforma de desenvolvimento mobile (e web) adquirida pela Google em 2014. Com foco em ser um back-end completo e de fácil usabilidade, essa ferramenta disponibiliza diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicativos.”

http://micreiros.com/firebase-o-que-e-e-como-funciona/

Firebase

O Firebase oferece uma gama de serviços que podem ser utilizados, eles são separados por 4 grandes categorias, sendo elas Analyics, Develop, Grow e Earn.



https://blog.rocketseat.com.br/firebase/

Firebase

- **Realtime Database**: Banco de dados que sincroniza os dados com os dispositivos em tempo real. Regras de segurança podem ser configuradas para definir quem tem acesso a quais dados.
- **Authentication**: Possibilita autenticação através de contas do Google, Facebook, Twitter, Github ou um sistema de contas próprio.
- **Cloud Messaging**: Permite enviar mensagens para os usuários através do aplicativo. É possível definir para quais grupos de usuários a mensagem será enviada.
- **Storage**: Armazena dados do usuário através de uma conexão segura e permite o compartilhamento dos mesmos.

http://micreiros.com/firebase-o-que-e-e-como-funciona/

Firebase

- **Hosting**: Serviço para hospedagem de sites com certificado SSL
- **Test Lab**: Serviço para testar o aplicativo em diferentes tipos de dispositivos. Possui uma ferramenta de testes automatizada (Robo Test), porém permite que o desenvolvedor crie seus próprios scripts de teste.
- **Crash Reporting**: Coleta informações de falhas que os usuários estão experienciando no aplicativo.
- **Notifications**: Envia notificações personalizadas para o usuário.
- **Remote Config**: Permite que versões diferentes do aplicativo sejam publicadas para diferentes usuários. Pode ser usado para testar mudanças com um grupo pequeno de usuários antes de aplicá-las definitivamente.

http://micreiros.com/firebase-o-que-e-e-como-funciona/

Firebase

- **App Indexing**: Permite que o aplicativo seja encontrado e pesquisas no Google Search, caso o assunto que o usuário procura seja relacionado com o app.
- **Dynamic Links**: Usado para criar links que executam determinadas ações no aplicativo. Também é possível definir diferentes ações para diferentes dispositivos e para casos em que o usuário ainda não tenha o aplicativo instalado.
- **Invites**: Utiliza o Dynamic Links para criar convites personalizados para o aplicativo, que podem ser enviados pelo usuário para outras pessoas.
- **AdWords**: Ferramenta para publicar anúncios do aplicativo no Google, YouTube ou Play Store.

http://micreiros.com/firebase-o-que-e-e-como-funciona/

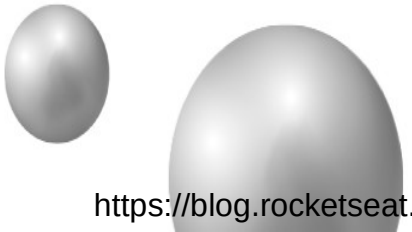
Firebase

- **AdMob: Facilita a monetização do aplicativo, colocando anúncios que encaixem no design do mesmo. O serviço prioriza automaticamente as fontes que retornam um maior lucro.**

http://micreiros.com/firebase-o-que-e-e-como-funciona/

Firebase

O Analytics possibilita a análise em tempo real sobre o comportamento dos usuários nos seus Apps, além de disponibilizar dados sobre falhas, compras no aplicativo, desempenho de links diretos e muito mais.



https://blog.rocketseat.com.br/firebase/

Firebase

Crie uma nova aplicação Flutter com o nome **firebaseapp**.

O nome do pacote do pacote pode ser visto no arquivo **android/app/src/main/AndroidManifest.xml**. Esse nome será usado na configuração do Firebase.

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firebaseapp">
```

Copie toda a estrutura de subdiretórios e arquivos do subdiretório **lib** da aplicação **persisitencia_sqlite**.

Crie o subdiretórios **assets** e **assets/images** e copie o arquivo **google_logo.jpg** para o subdiretório **assets/images**.

Altere o arquivo **pubspec.yaml** para incluir a referência ao subdiretório **assets/images**:

```
  assets:
    - assets/images/
```

Firebase

Altere o arquivo pubspec.yaml para incluir as dependências de firebase_core, firebase_auth, google_sign_in e cloud_firestore:

```
dependencies:
  firebase_core: ^1.1.0
  firebase_auth: ^0.15.2
  google_sign_in: ^4.1.0
  cloud_firestore: ^0.12.5
  flutter:
    sdk: flutter
```

Instale os pacotes adicionados nas dependências.

Firebase

Altere o arquivo main.dart no subdiretório lib:

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'pages/signin.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(primarySwatch: Colors.blue,),
      home: SignIn(),
    );
  }}
```

Firebase

No arquivo **student.dart** no subdiretório **lib/models**:

Altere a linha:

```
  int id;
```

para:

```
  String id;
```

Altere a linha:

```
    id = 0;
```

para:

```
    id = "";
```

Firebase

Crie o arquivo user.dart no subdiretório lib/models:

```
//
// Modelo para dados do usuario
//
class User {
  String email;
  String password;

  // Construtor padrao
  User({this.email, this.password});

  // Construtor de uma sessao vazia
  User.empty() {
    email = "";
    password = "";
  }
```

Firebase

```
  // Construtor baseando em JSON
  factory User.fromJson(Map<String, dynamic> json) {
    return User(
      email: json['email'],
      password: json['password']
    );
  }

  // Converte um usuario para JSON
  Map<String, dynamic> toJson() {
    var map = {
      'email': email,
      'password': password
    };
    return map;
  }
}
```

Firebase

Altere o arquivo database.dart no subdiretório lib/database:

```
import 'package:cloud_firestore/cloud_firestore.dart';

class DatabaseHelper {
  static final DatabaseHelper _instance =
DatabaseHelper.internal();
  factory DatabaseHelper() => _instance;
  DatabaseHelper.internal();

  static final _db = FirebaseFirestore.instance;
```

Firebase

```
  static Future<List<Map>> getAll(String _table) async {
    print("Database getAll");
    List<Map> ret;
    String _errorMsg = "";
    QuerySnapshot _query = await
_db.collection(_table).get()
      .catchError((_error) {
        _errorMsg = "${_error}"; });
    if (_query != null) {
      ret = List<Map>();
      List<DocumentSnapshot> _documents = _query.docs;
      for (int i = 0; i < _documents.length; i++) {
        Map _map = _documents[i].data();
        _map['id'] = _documents[i].id;
        ret.add(_map);
      }
    }
    if (ret != null)
      return ret;
    else
      return Future<List<Map>>.error(_errorMsg);
  }
```

Firebase

```
  static Future<Map> getByID(String _table, String _id)
async {
    print("Database getByID");
    Map ret;
    String _errorMsg = "";
    DocumentSnapshot _document = await
_db.collection(_table).doc(_id)
      .get()
      .catchError((_error) {
        _errorMsg = "${_error}";
      });
    if (_document != null) {
      ret = _document.data();
      ret['id'] = _document.id;
    }
    if (ret != null)
      return ret;
    else
      return Future<Map>.error(_errorMsg);
  }
```

Firebase

```
  static Future<int> insert(String _table, Map _map) async {
    print("Database insert");
    int ret;
    String _errorMsg = "";
    _map.remove('id');
    DocumentReference _documentReference = await
_db.collection(
      _table
    ).add(_map)
     .catchError((_error) {
       _errorMsg = "${_error}";
    });
    if(_documentReference.id != null)
      ret = 1;
    if (ret != null)
      return ret;
    else
      return Future<int>.error(_errorMsg);
  }
```

Firebase

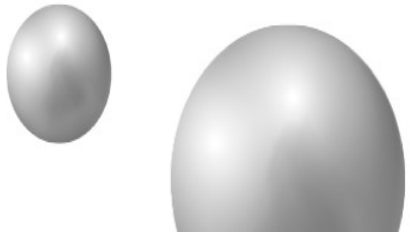
```
static Future<int> update(String _table, Map _map) async {
  print("Database update");
  int ret;
  String _errorMsg = "";
  String _id = _map['id'];
  _map.remove('id');
  await _db.collection(_table).doc(_id)
    .update(_map)
    .catchError((_error) {
      _errorMsg = "${_error}";
    });
  if (_errorMsg == "")
    ret = 1;
  if (ret != null)
    return ret;
  else
    return Future<int>.error(_errorMsg);
}
```

Firebase

```
  static Future<int> delete(String _table, String _id) async
{
    print("Database delete");
    int ret;
    String _errorMsg = "";
    await _db.collection(_table).doc(_id).delete()
      .catchError((_error) {
        _errorMsg = "${_error}";
      });
    if (_errorMsg == "")
      ret = 1;
    if (ret != null)
      return ret;
    else
      return Future<int>.error(_errorMsg);
  }
```

Firebase

```
  void close() {
    print("CloseDB");
  }
}
```



Firebase

Crie o subdiretório **lib/services**. Crie o arquivo **authentication.dart** no subdiretório **lib/services**.

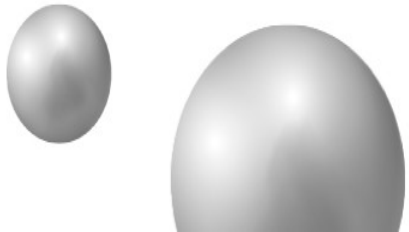
```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:google_sign_in/google_sign_in.dart';

class AuthHelper {
  static final AuthHelper _instance = AuthHelper.internal();
  factory AuthHelper() => _instance;
  AuthHelper.internal();

  static final GoogleSignIn _googleSignIn = GoogleSignIn();
  static final FirebaseAuth _auth = FirebaseAuth.instance;
```

Firebase

```
  static Future<bool> signInWithGoogle() async {
    print("signInWithGoogle");
    String _errorMsg = "";
    final GoogleSignInAccount _googleUser = await
GoogleSignIn().signIn();
    final GoogleSignInAuthentication _googleAuth = await
_googleUser.authentication;
    final _credential = GoogleAuthProvider.credential(
      accessToken: _googleAuth.accessToken,
      idToken: _googleAuth.idToken, );
    UserCredential _userCredential;
    _userCredential = await
FirebaseAuth.instance.signInWithCredential(_credential)
      .catchError((_error) {
        _errorMsg = "${_error}";  });
    if (_userCredential != null) {
      User _user = _userCredential.user;
      print("signed in " + _user.email);
      return true;
    }
    else return Future<bool>.error(_errorMsg);
  }
```

Firebase

```
    static  Future<bool>  createUser(String  _email,  String
_password) async {
    print("createUser");
    String _errorMsg = "";
    final _authResult = await
_auth.createUserWithEmailAndPassword(
      email: _email,
      password: _password,
    ).catchError((_error) {
      _errorMsg = "${_error}";
    });
    if ( _authResult != null ) {
      print("signed in ");
      return true;
    }
    else
      return Future<bool>.error(_errorMsg);
  }
```



Firebase

```
  static Future<bool> signInWithEmail(String _email, String
_password) async {
    print("signInWithEmail");
    String _errorMsg = "";
                      final     _authResult     =     await
_auth.signInWithEmailAndPassword(
      email: _email,
      password: _password,
    ).catchError((_error) {
      _errorMsg = "${_error}";
    });
    if ( _authResult != null ) {
      print("signed in ");
      return true;
    }
    else
      return Future<bool>.error(_errorMsg);
  }
}
```

Firebase

Copie o arquivo **lib/dao/student_dao.dart** da aplicação **rest_jwt** para o subdiretório **lib/dao**.

No arquivo **student_dao.dart** no subdiretório **lib/dao**:

Altere a linha:

```
Future<Student> getByID(int _id) async {
```

para:

```
Future<Student> getByID(String _id) async {
```

Altere a linha:

```
Future<int> delete(int _id) async {
```

para:

```
Future<int> delete(String _id) async {
```

Firebase

Copie o arquivo **lib/bloc/student_bloc.dart** da aplicação **rest_jwt** para o subdiretório **lib/bloc**.

No arquivo **student_bloc.dart** no subdiretório **lib/bloc**:

Altere a linha:

```
void getByID(int _id) async {
```

para:

```
void getByID(String _id) async {
```

Altere a linha:

```
Future <int> delete(int _id) async {
```

para:

```
Future <int> delete(String _id) async {
```

Firebase

Copie o arquivo **lib/pages/home.dart** da aplicação **rest_jwt** para o subdiretório **lib/pages**.

No arquivo **home.dart** no subdiretório **lib/pages**:

Altere o nome da aplicação:

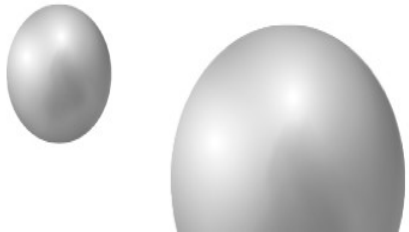
```
appBar: AppBar(
  title: Text("Firebase"),
```

Altere a linha:

```
_detail(BuildContext context, int id) async {
```

para:

```
_detail(BuildContext context, String id) async {
```



Firebase

No arquivo **detail.dart** no subdiretório **lib/pages**:

Altere a linha:

```
final int id;
```

para:

```
final String id;
```

Firebase

Crie o arquivo signin.dart no subdiretório lib/pages:

```
import 'package:flutter/material.dart';
import '../models/user.dart';
import '../services/authentication.dart';
import '../utils/showerror.dart';
import 'home.dart';

//
// Classe para sigin no Web Service
//
class SignIn extends StatelessWidget {
  final GlobalKey<FormState> _formStateKey =
GlobalKey<FormState>();
  User _user = User.empty();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Sign In"),
      ),
```

Firebase

```
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: Column(
          children: <Widget>[
            Form(
              key: _formStateKey,
              autovalidate: true,
              child: Padding(
                padding: EdgeInsets.all(16.0),
                child: Column(
                  children: <Widget>[
                    TextFormField(
                      initialValue: _user.email,
                      decoration: InputDecoration(labelText:
'Email',),
                      validator: (value) =>
_validateEmail(value),
                      onSaved: (value) => _user.email =
value,
                    ),
```

Firebase

```
                    TextFormField(
                      initialValue: _user.password,
                      decoration: InputDecoration(labelText:
'Password',),
                      obscureText: true,
                      validator: (value) =>
_validatePassword(value),
                      onSaved: (value) => _user.password =
value,
                    ),
                  ],
                ),
              ),
            ),
            SizedBox(
              width: double.maxFinite,
              child: RaisedButton(
                onPressed: () => _signinEmail(context),
                color: Colors.blue[300],
                child: Text("Log In")
              ),
            ),
```

Firebase

```
            FlatButton(
              onPressed: () => _register(context),
              child: Text("Register")
            ),
            OutlineButton(
              splashColor: Colors.grey,
              onPressed: () => _signinGoogle(context),
              shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(40)),
              highlightElevation: 0,
              borderSide: BorderSide(color: Colors.grey),
              child: Padding(
                padding: const EdgeInsets.fromLTRB(0, 10, 0,
10),
                child: Row(
                  mainAxisSize: MainAxisSize.min,
                  mainAxisAlignment:
MainAxisAlignment.center,
```

Firebase

```
                    children: <Widget>[
                      Image(
                        image:
AssetImage("assets/images/google_logo.jpg"),
                      height: 35.0 ),
                      Padding(
                        padding: const EdgeInsets.only(left:
10),
                        child: Text('Sign in with Google',
                          style: TextStyle(fontSize: 20,color:
Colors.grey),
                        ),
                      )
                    ],
                  ),
                ),
              )
            ],
          ),
        )
      );
  }
```

Firebase

```
  //
  // Valida o email
  //
  String _validateEmail(String value) {
    String ret = null;
    value = value.trim();
    if (value.isEmpty)
      ret = "Email e obrigatorio";
    else if (!RegExp(r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?
^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+").hasMatch(value))
      ret = "Email inválido";
    return ret;
  }
```

Firebase

```
//
// Valida a password
//
String _validatePassword(String value) {
  String ret = null;
  value = value.trim();
  if (value.isEmpty)
    ret = "Pasword e obrigatoria";
  else if(value.length < 4)
    ret = "Minimo de 4 caracteres";
  return ret;
}
```

Firebase

```
  //
  // Autenticar email/password
  //
  void _signinEmail(BuildContext context) async {
    print("Email signin");
    String _errorMsg;
    if (_formStateKey.currentState.validate()) {
      _formStateKey.currentState.save();
      var _signed = await AuthHelper.signInWithEmail(
          _user.email,
          _user.password
      ).catchError((_error) {
        _errorMsg = "${_error}"; });
      if(_signed != null) {
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (BuildContext
context) => Home())
        );
      } else
        showError(context, _errorMsg);
    }
  }
```

Firebase

```
  //
  // Registrar email/password
  //
  void _register(BuildContext context) async {
    print("Register");
    String _errorMsg;
    if (_formStateKey.currentState.validate()) {
      _formStateKey.currentState.save();
      var _signed = await AuthHelper.createUser(
        _user.email,
        _user.password
      ).catchError((_error) {
        _errorMsg = "${_error}"; });
      if(_signed != null) {
        Navigator.pushReplacement(context,
            MaterialPageRoute(builder: (BuildContext
context) => Home())
        );
      } else
        showError(context, _errorMsg);
    }
  }
```

Firebase

```
  //
  // Autenticar conta do Google
  //
  void _signinGoogle(BuildContext context) async {
    print("Google signin");
    String _errorMsg = "";
    var _signed = await
AuthHelper.signInWithGoogle().catchError((_error) {
      _errorMsg = _error;
    });
    if(_signed != null) {
      Navigator.pushReplacement(context,
          MaterialPageRoute(builder: (BuildContext context)
=> Home())
      );
    } else
      showError(context, _errorMsg);
  }
}
```

Firebase

Para configuração da aplicação no Firebase é necessário um certificado SHA1.

Para obter o certificado de debug, execute o comando:

```
keytool -list -v -alias androiddebugkey -keystore
~/.android/debug.keystore
```

A senha padrão para o armazenamento de chaves de debug é **android**.

Firebase

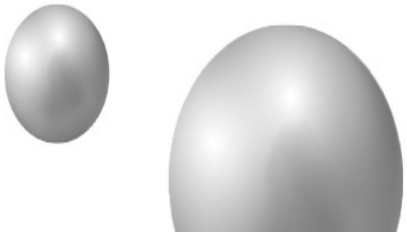
Para utilizar o Firebase em uma aplicação, será necessário uma conta Google para criar um projeto no Firebase.

O Firebase possui um plano gratuito com limitações em alguns serviços e um plano pago com custos por utilização de cada serviço.

Criaremos um novo projeto Firebase usando o plano gratuito.

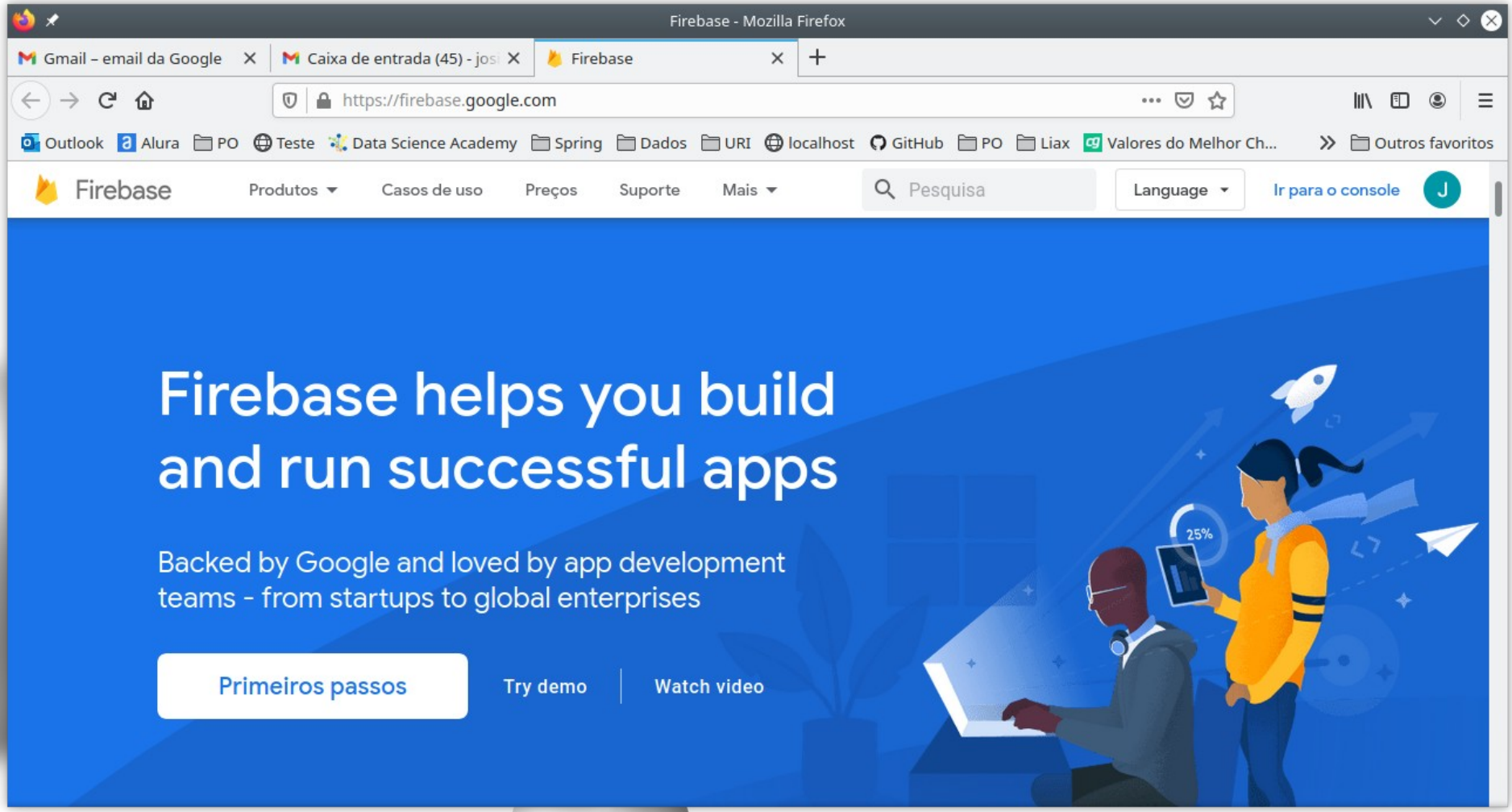
Acesse a página inicial do Firebase em:

https://firebase.google.com/



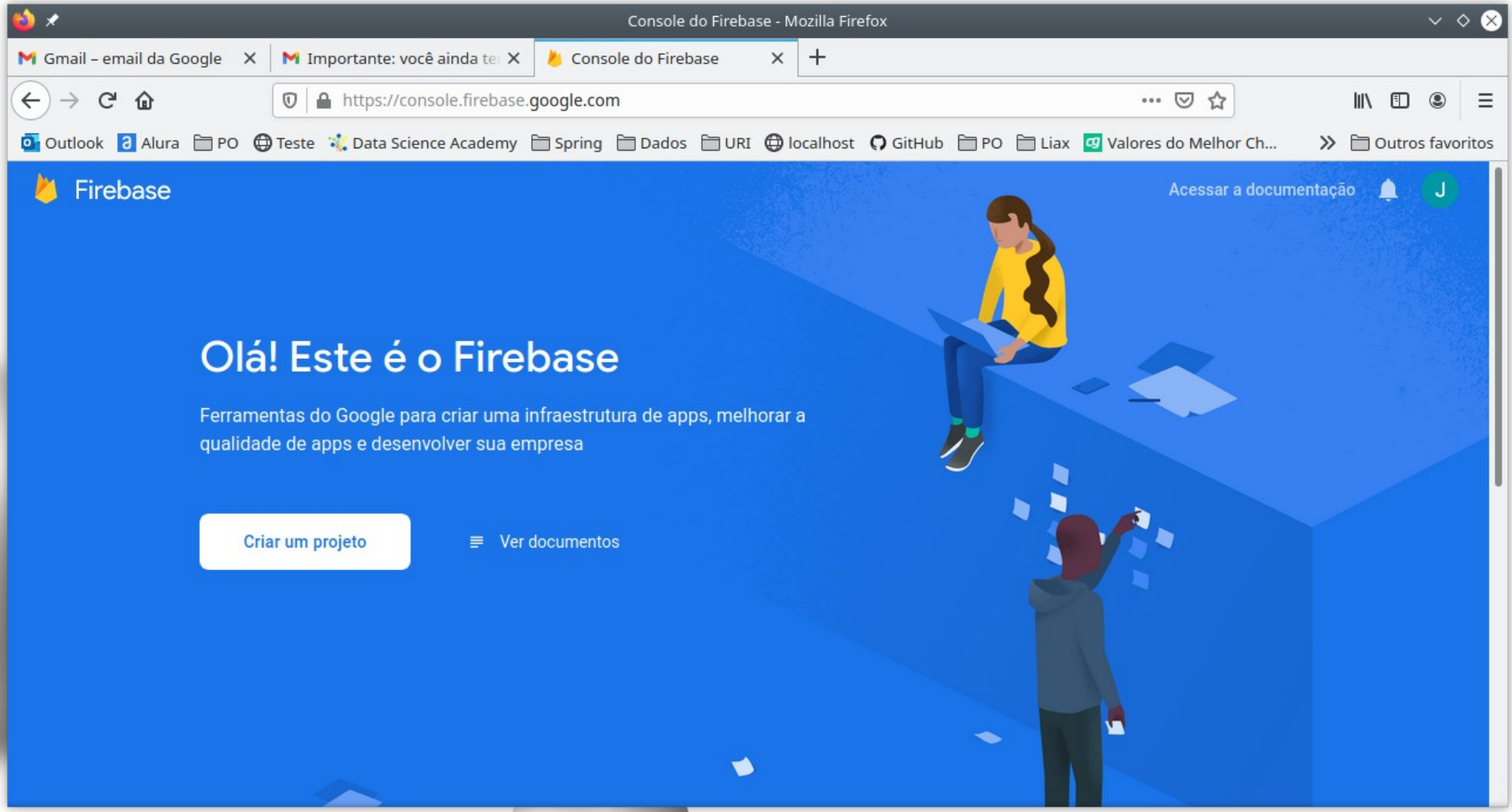
Firebase

Clique em Primeiros Passos e faça login.



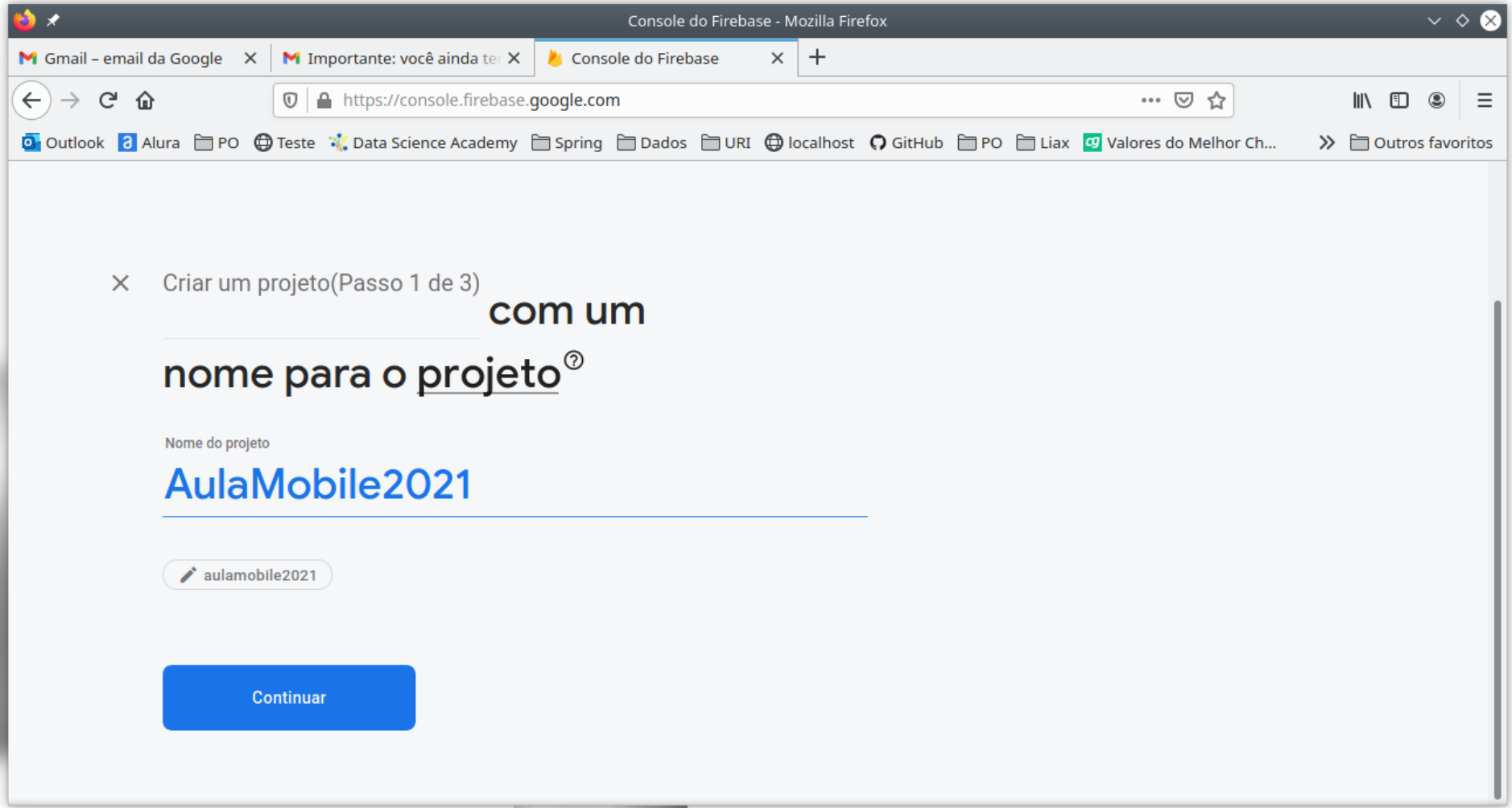
Firebase

Clique em Criar um projeto.



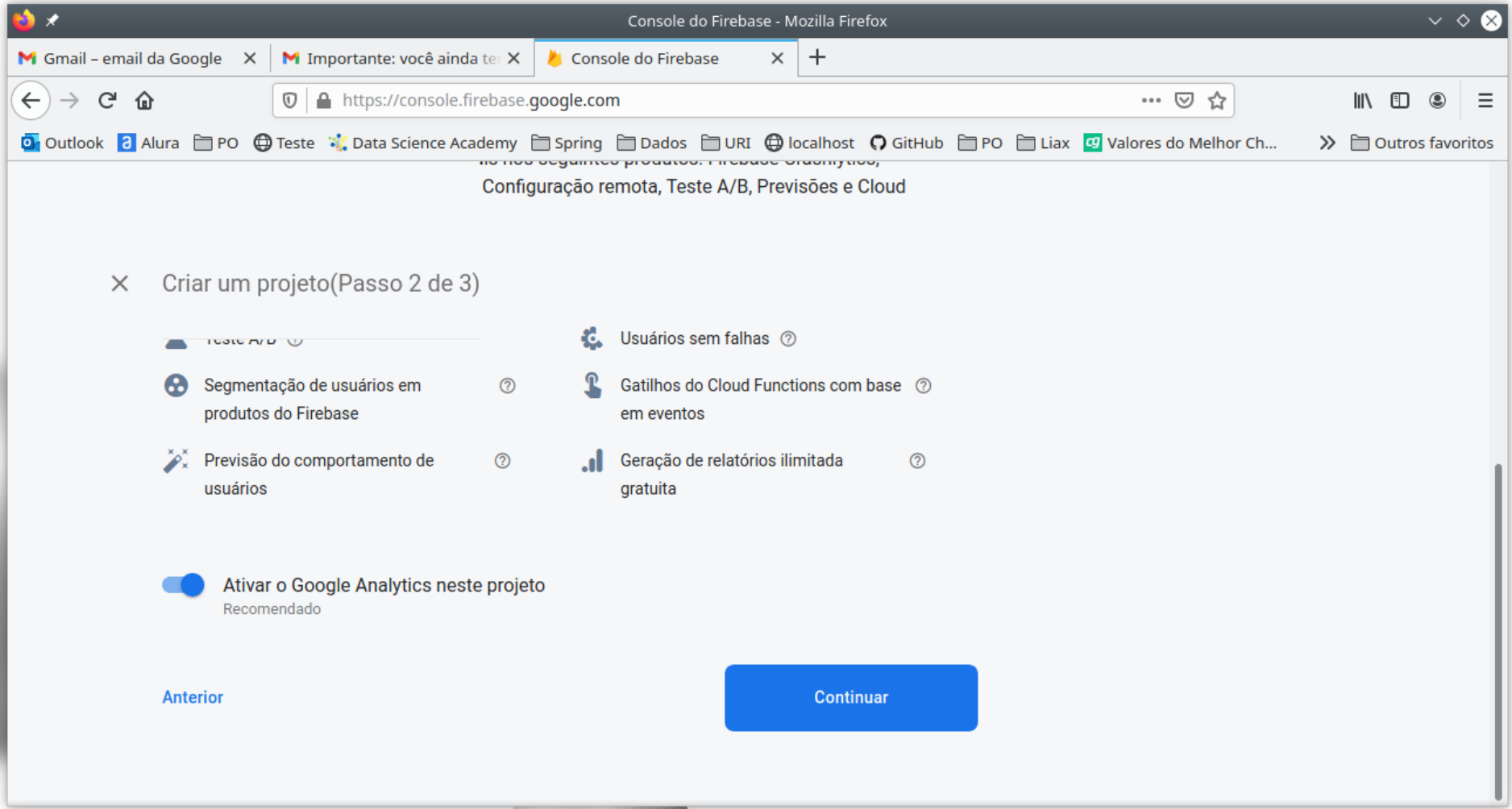
Firebase

Escolha o nome do projeto.



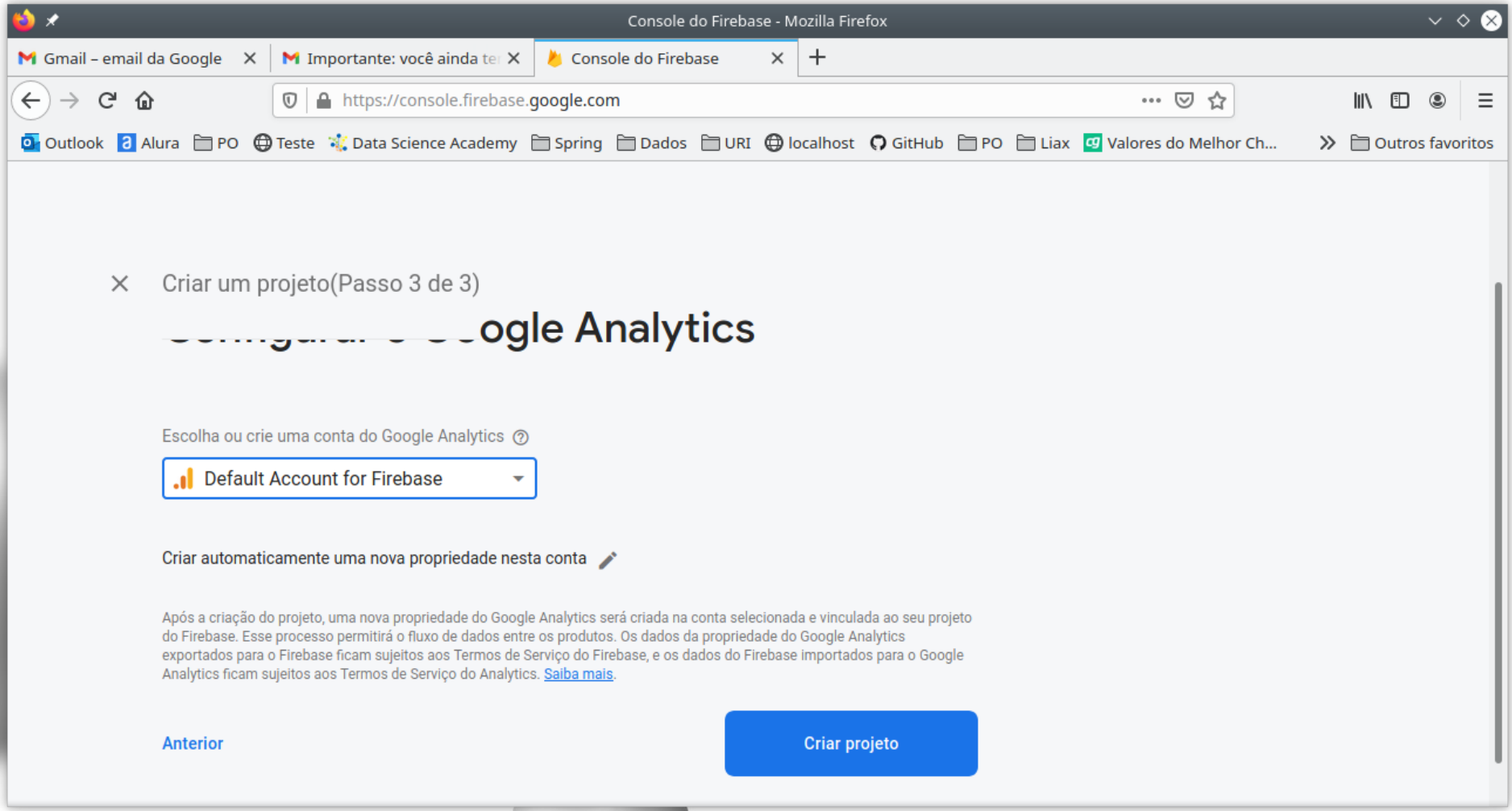
Firebase

Aceite a ativação do Google Analytics para o projeto.



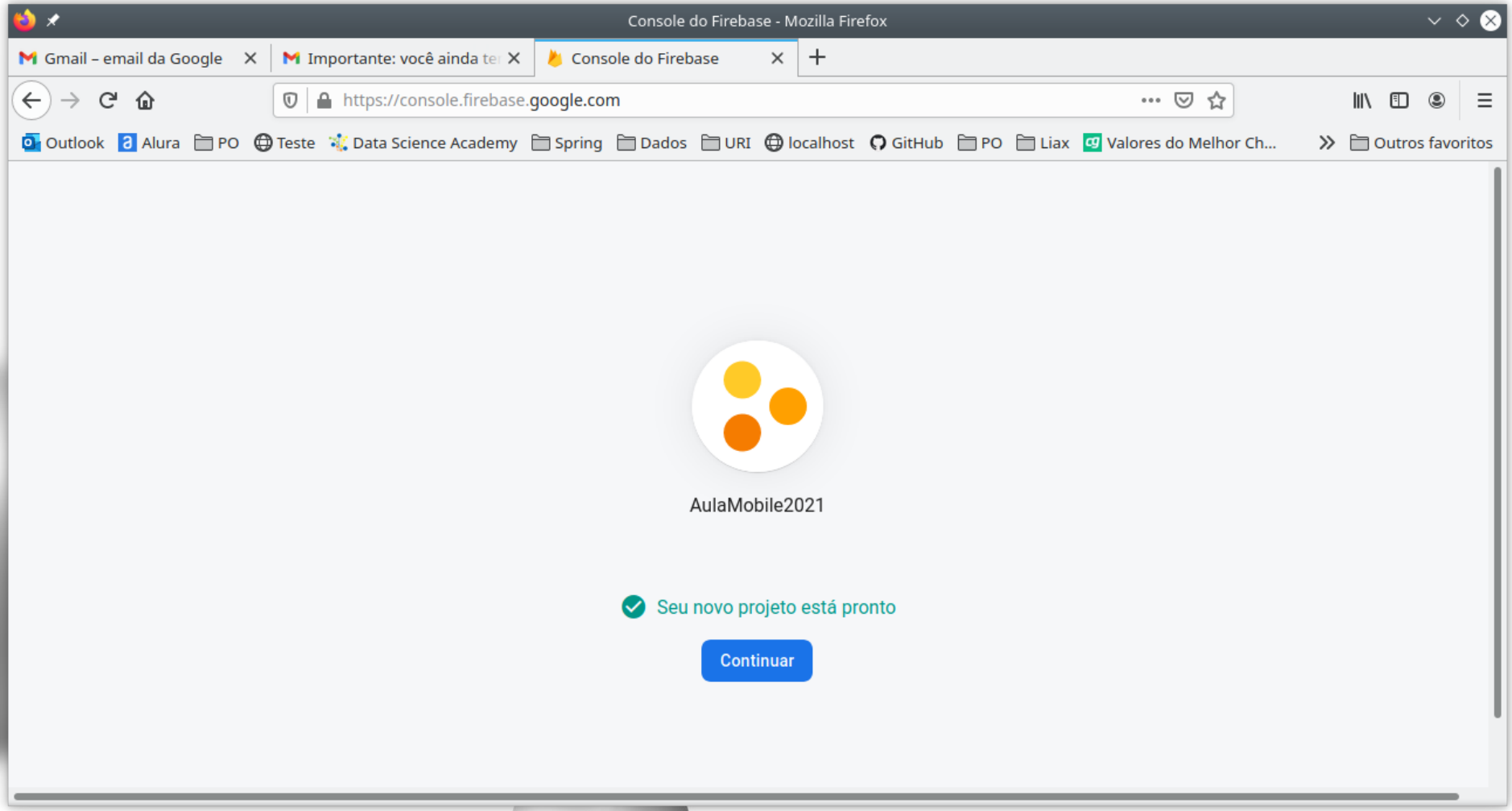
Firebase

Utilize a conta padrão para o Firebase.



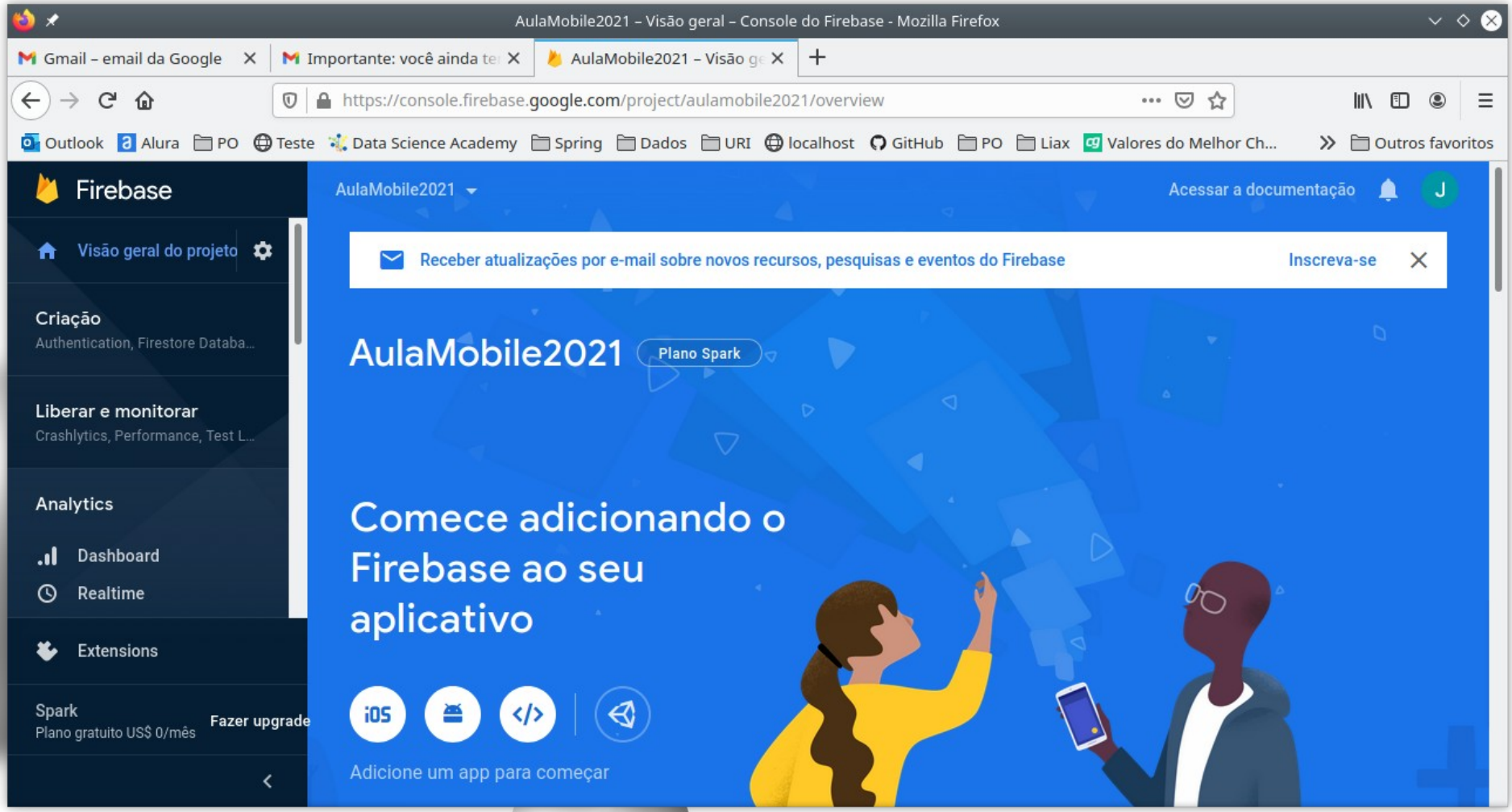
Firebase

Clique em Continuar para ver o console do projeto.



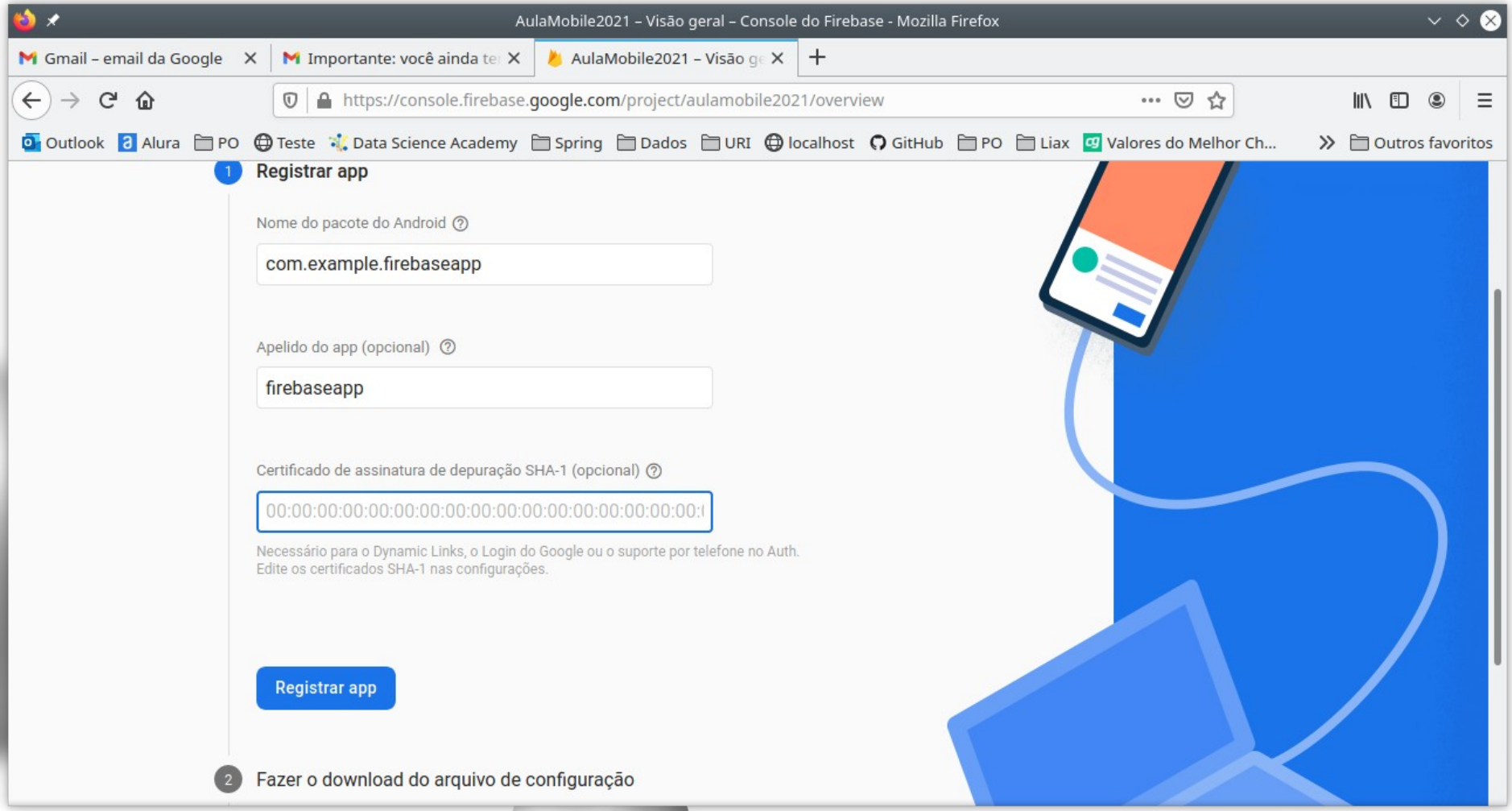
Firebase

Clique no botão para adicionar uma aplicação Android.



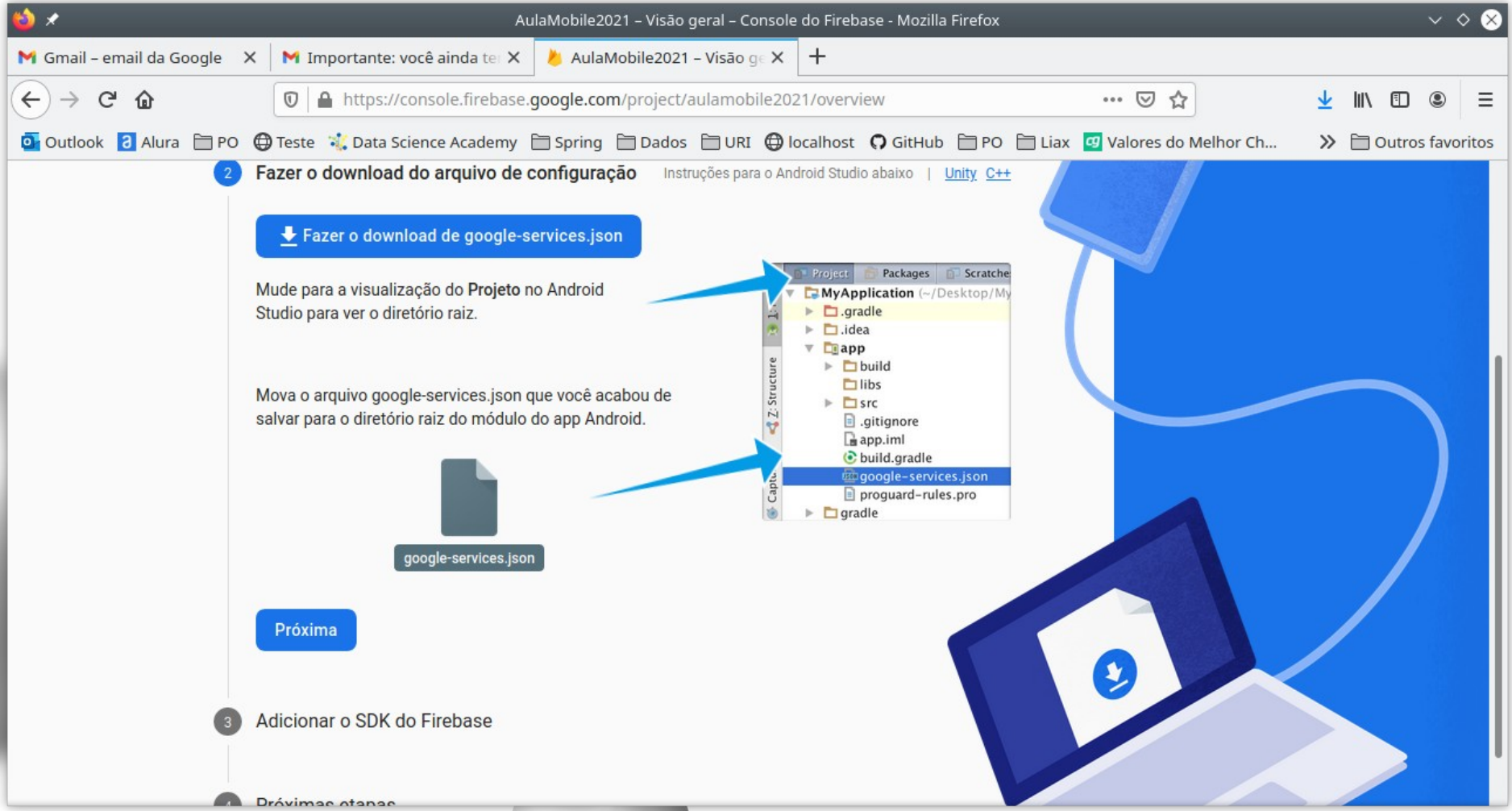
Firebase

Coloque os dados do projeto do Flutter e o certificado SHA1.



Firebase

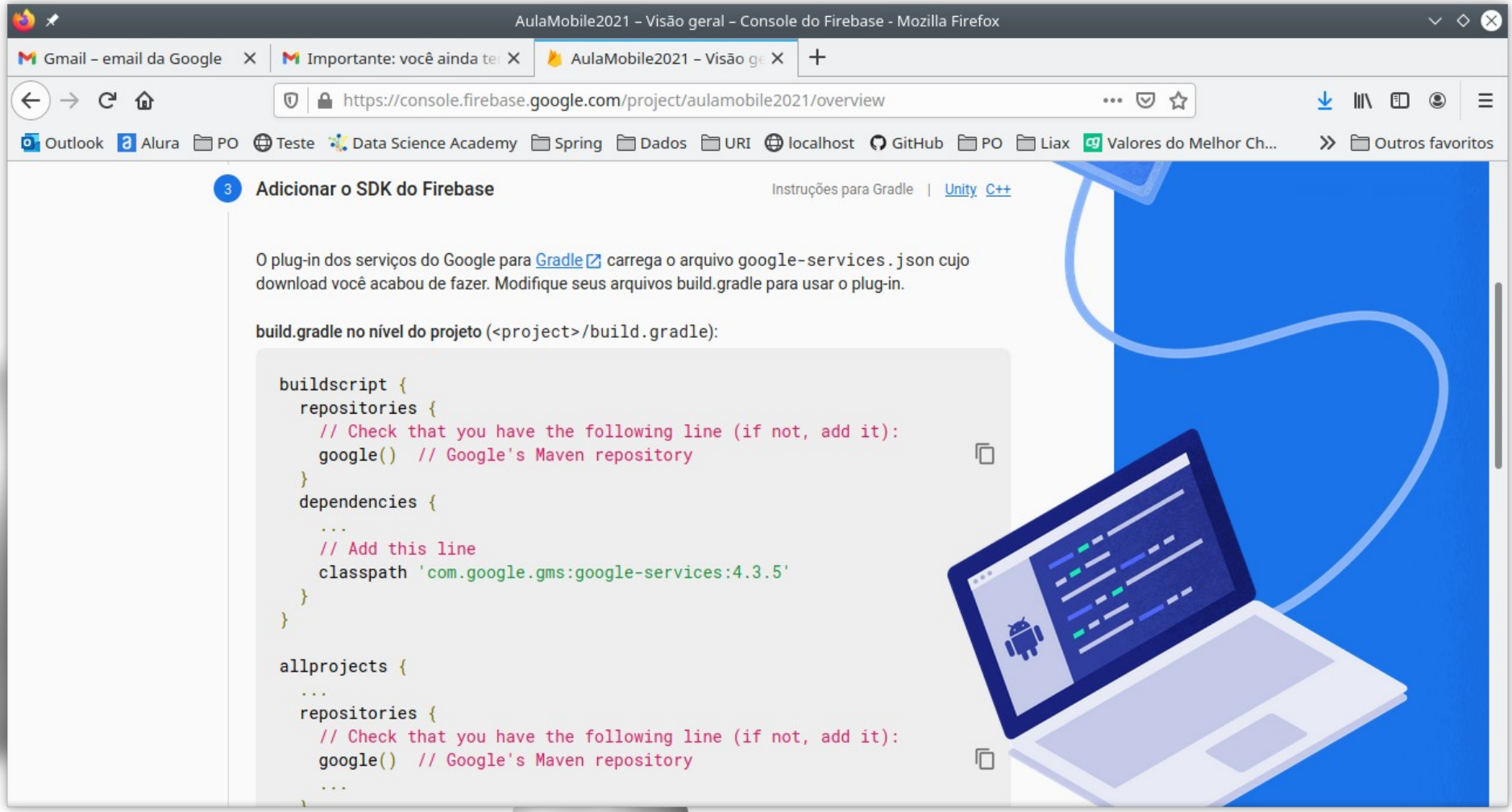
Faça o download do arquivo para o subdiretório android/app.



Firebase

Siga as instruções para adicionar o SDK do Firebase.

Execute o comando flutter packages get.



Firebase

- **build.gradle** no nível do projeto: **android/build.gradle**
- **build.gradle** no nível do app: **android/app/build.gradle**

Se, ao compilar a aplicação, ocorrer o erro:

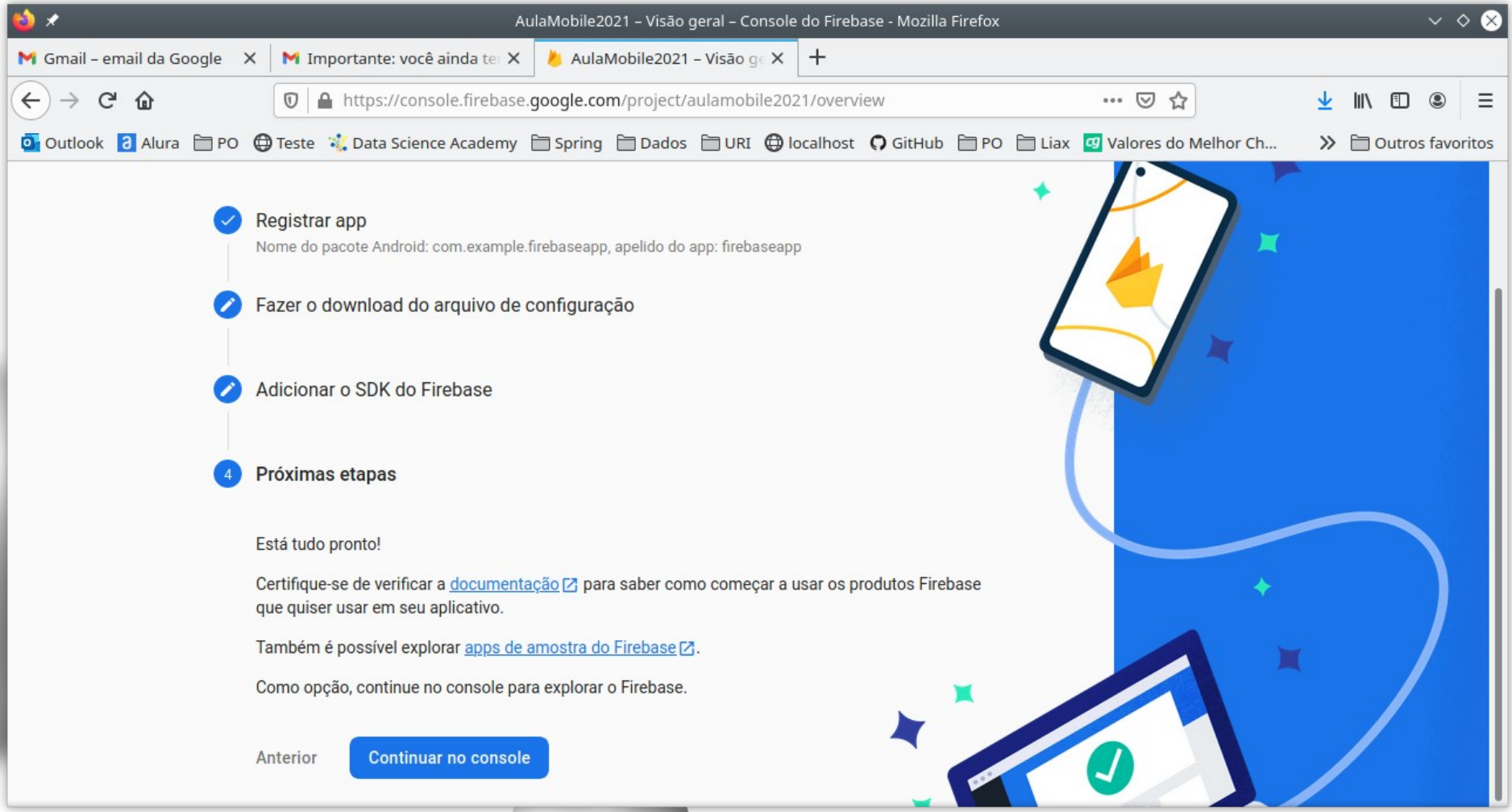
```
The number of method references in a .dex file cannot exceed 64K.
```

Será necessário incluir no build.gradle no nível do app, a seguinte linha:

```
    defaultConfig {
        // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
        applicationId "com.example.firebaseapp"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode flutterVersionCode.toInteger()
        versionName flutterVersionName
        multiDexEnabled true
    }
```

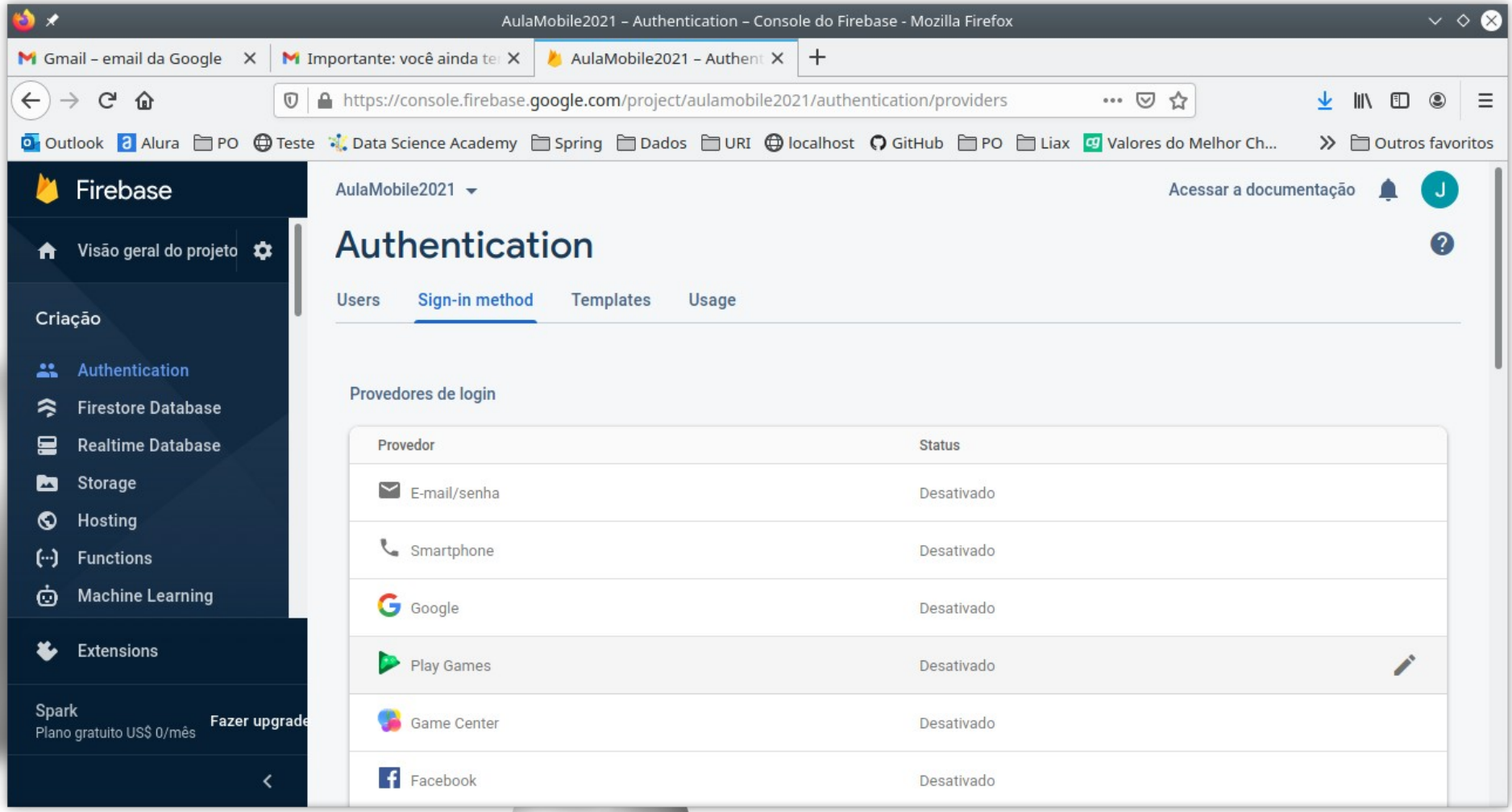
Firebase

Volte para o console do projeto.



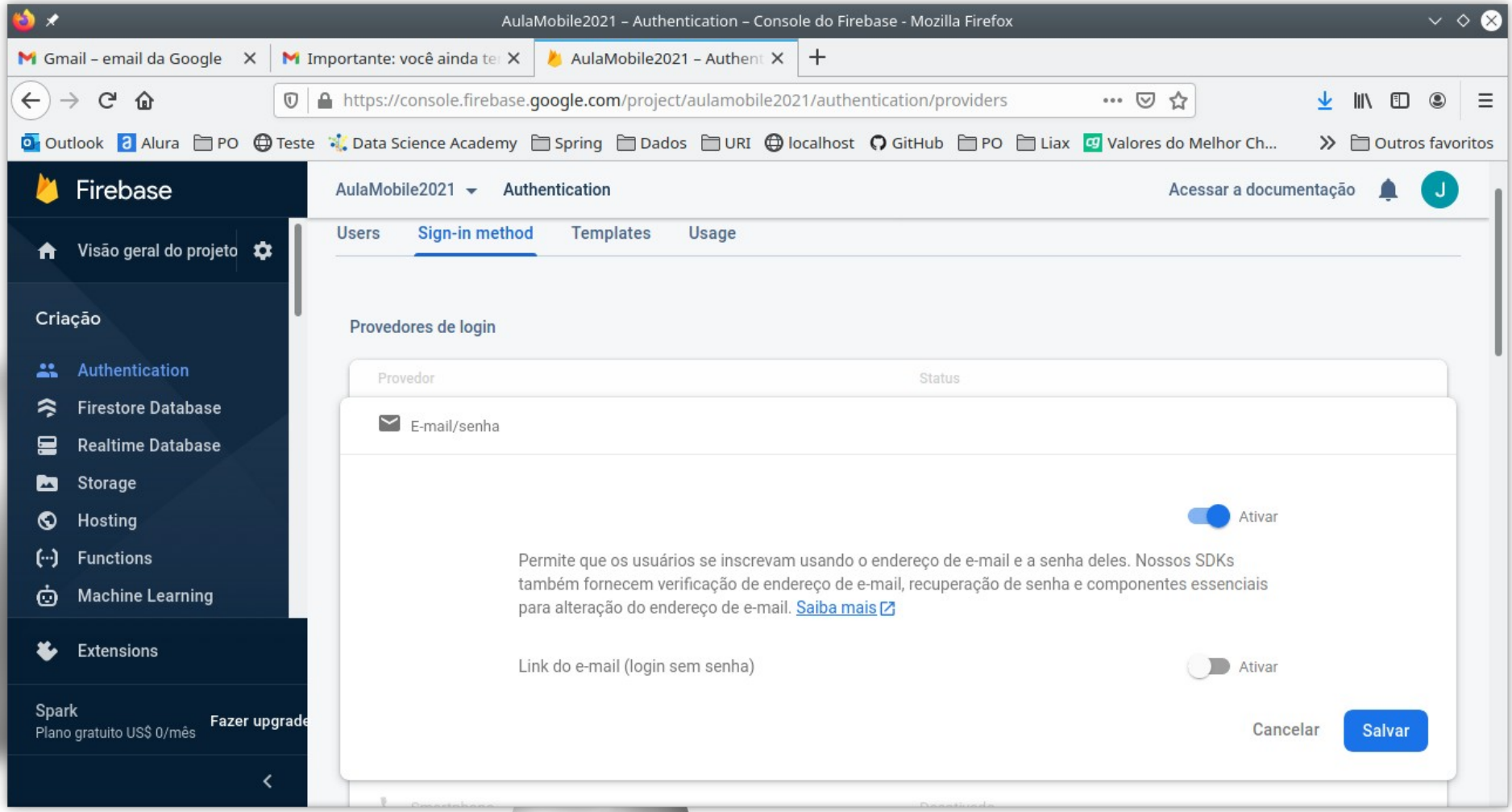
Firebase

Selecione a aba de Sign-in method da autenticação.



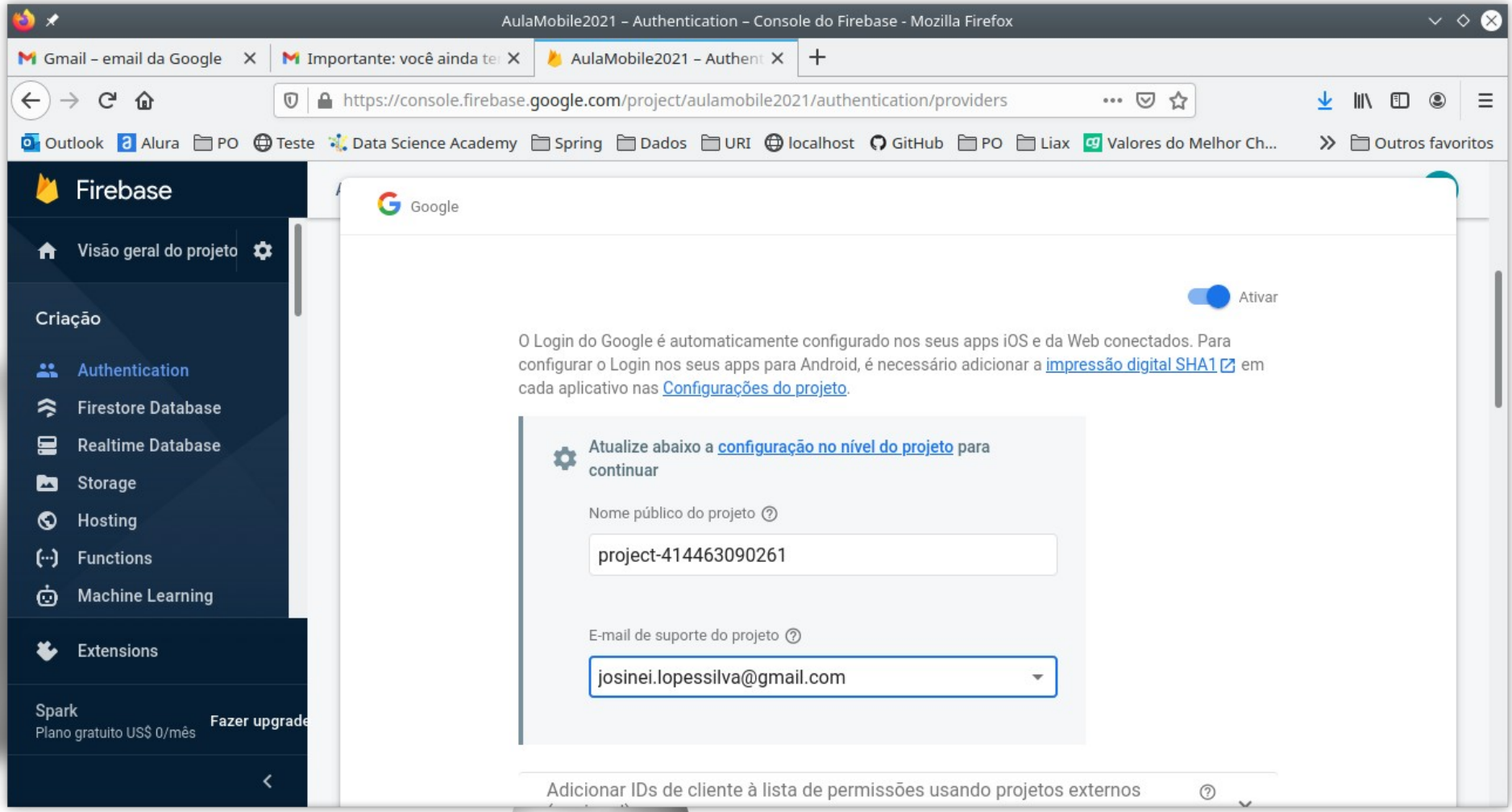
Firebase

Habilite o sign-in com o e-mail.



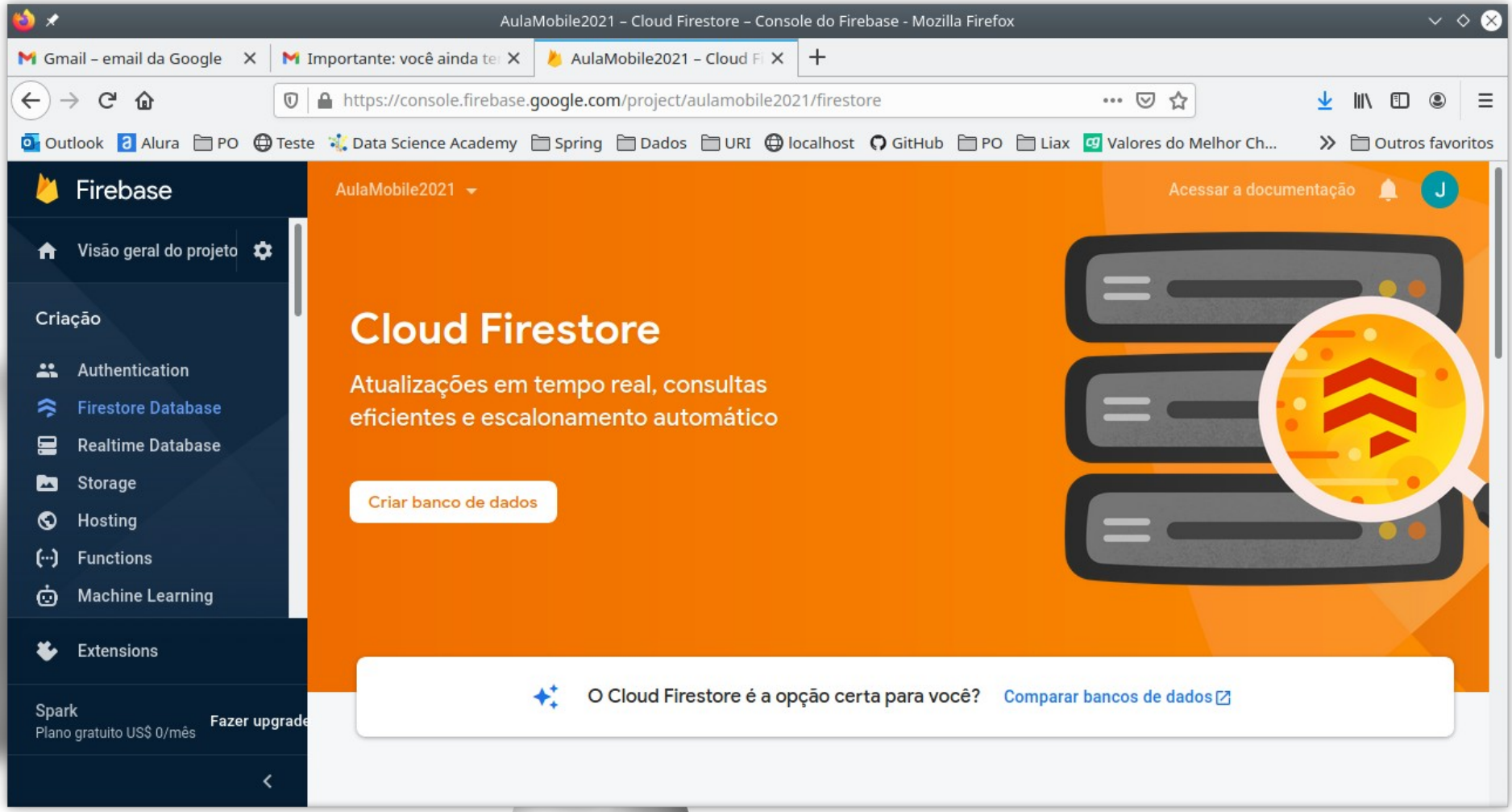
Firebase

Habilite o sign-in com conta do Google.



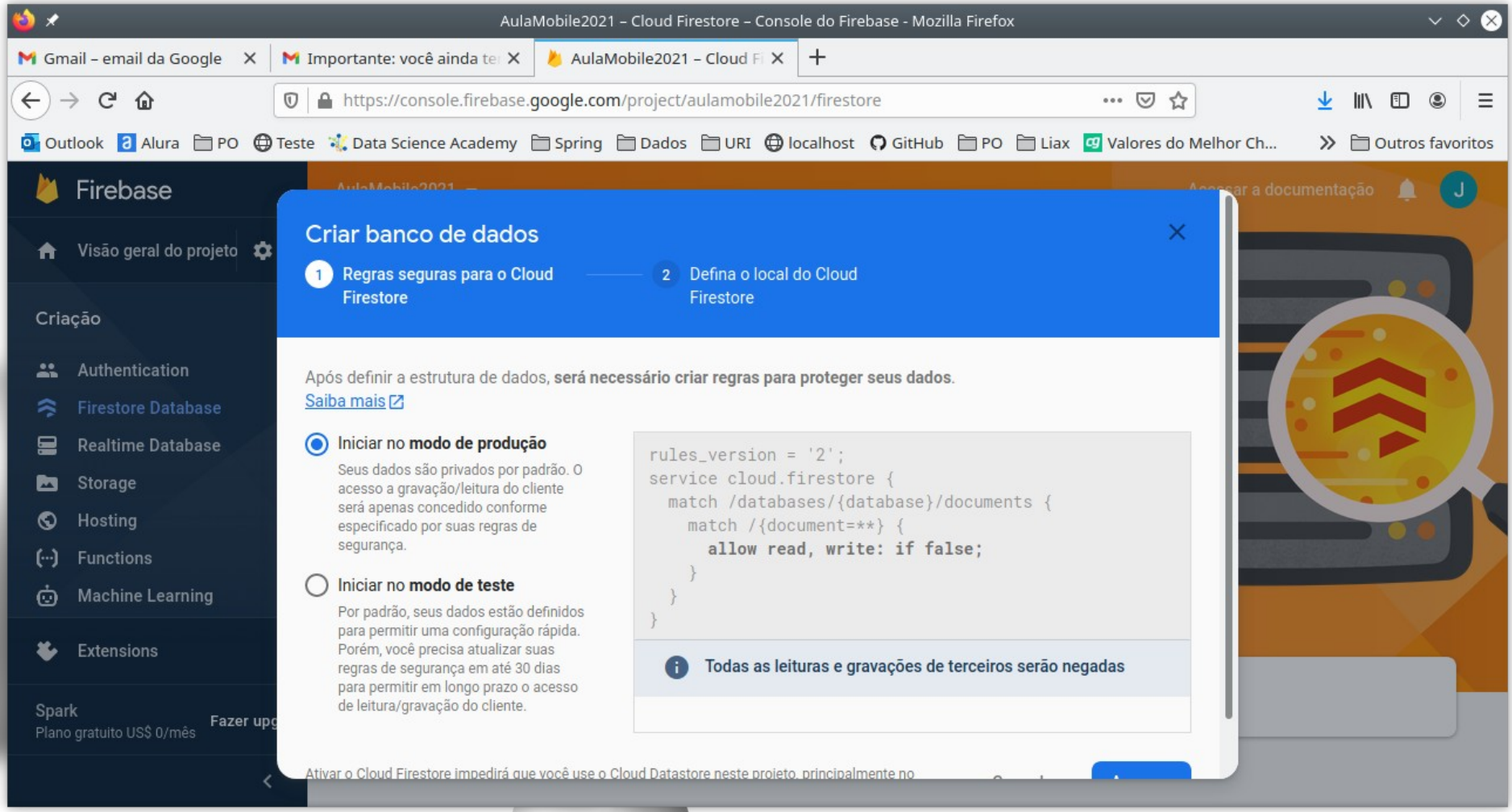
Firebase

Selecione a configuração do database.



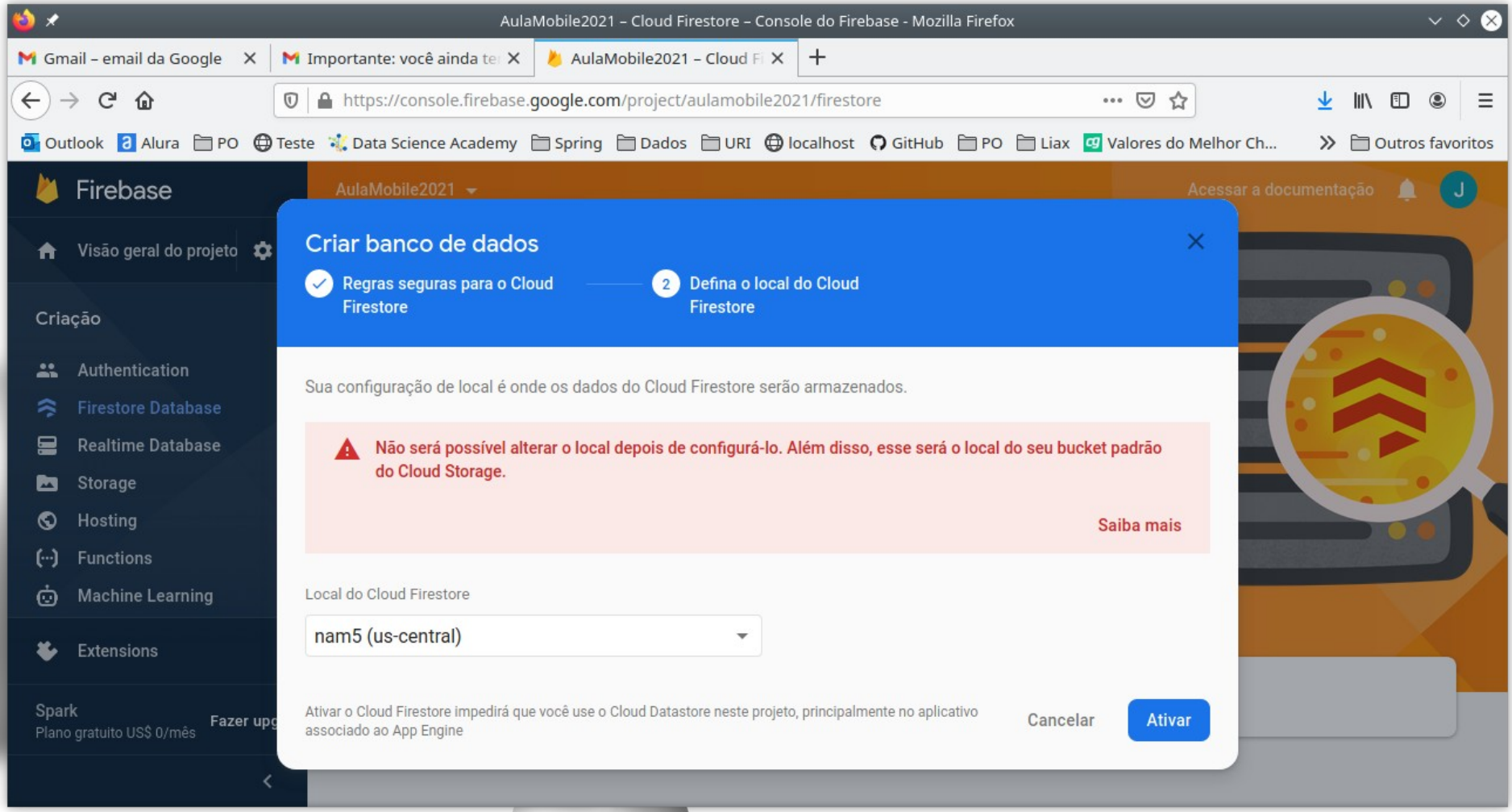
Firebase

Inicie no modo de produção.



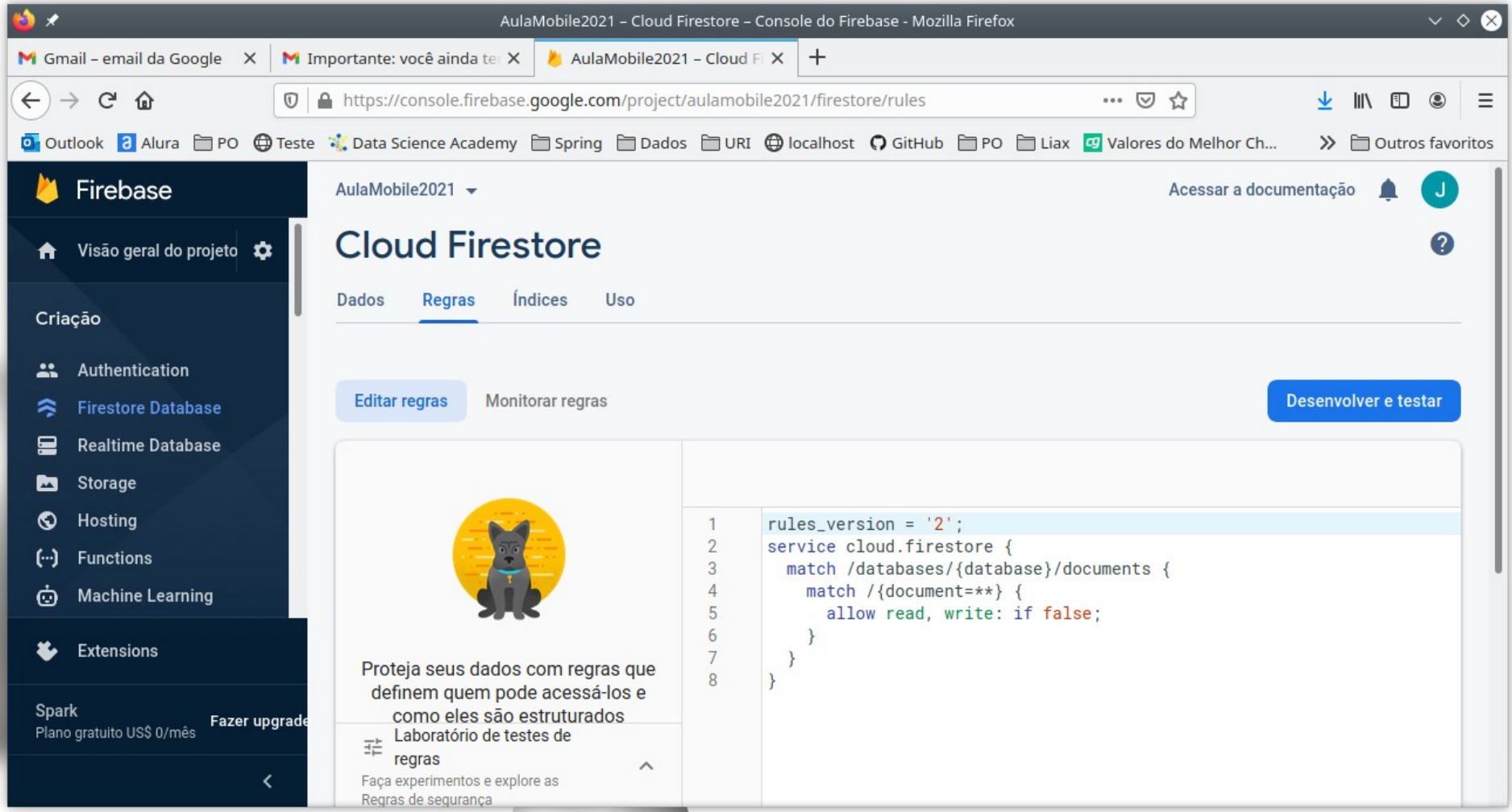
Firebase

Selecione o local do Cloud Firestore.



Firebase

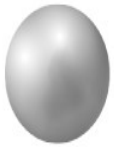
Selecione a aba de Regras.



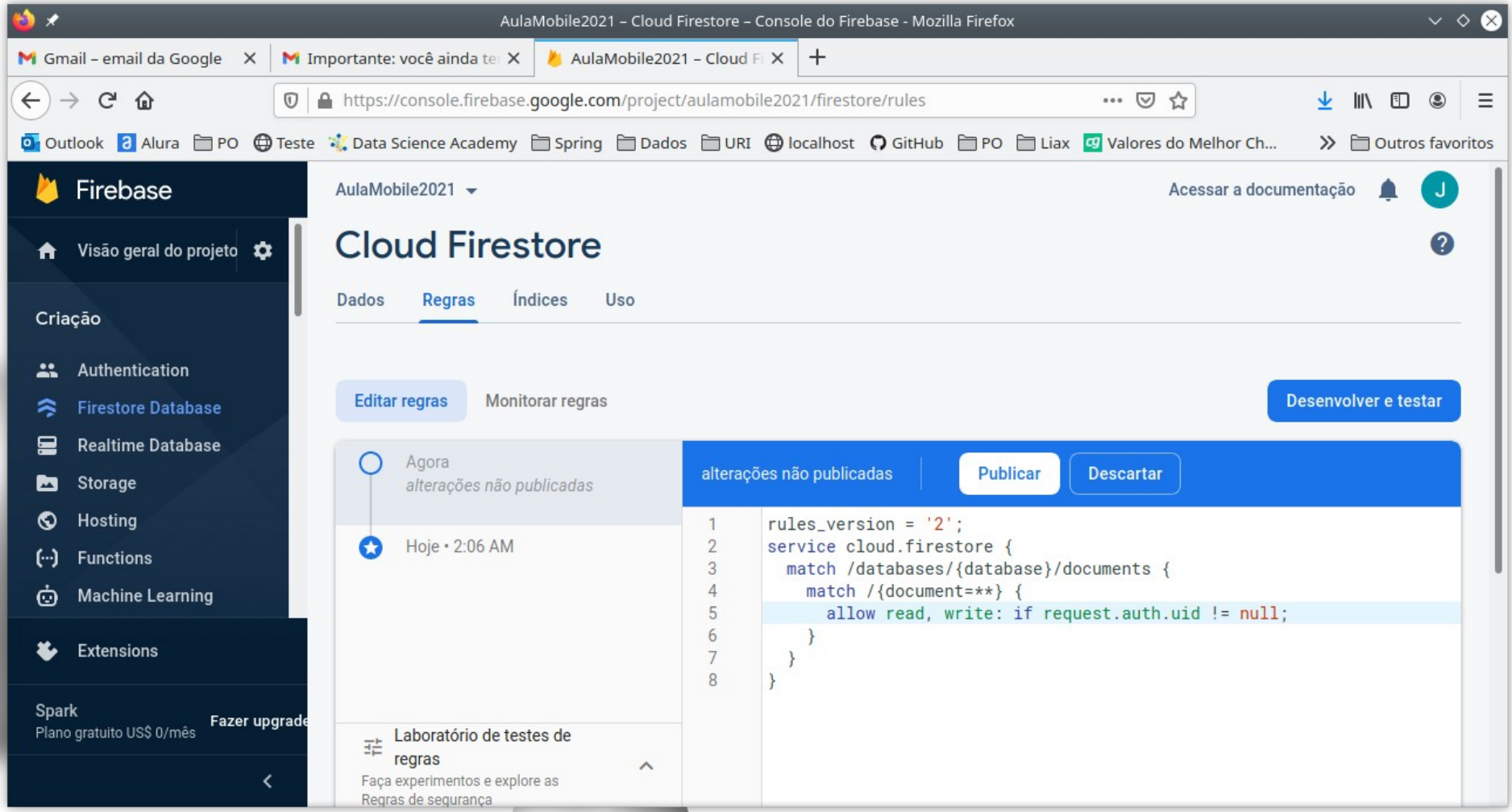
Firebase

Altere a permissão para permitir que qualquer usuário autenticado tenha acesso aos dados:

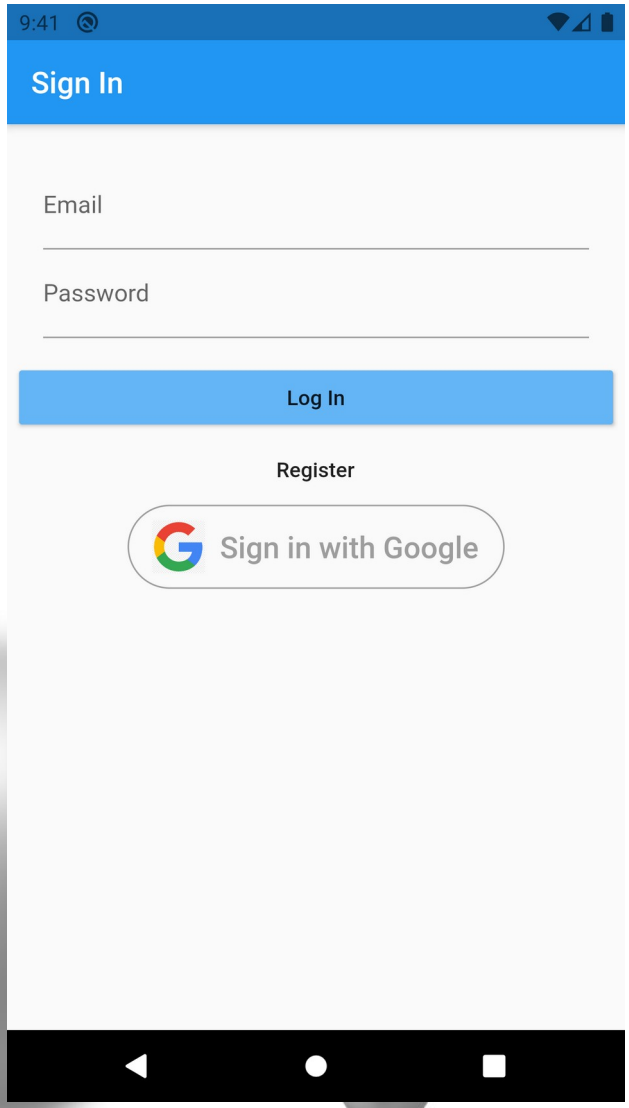
```
allow read, write: if request.auth.uid != null;
```

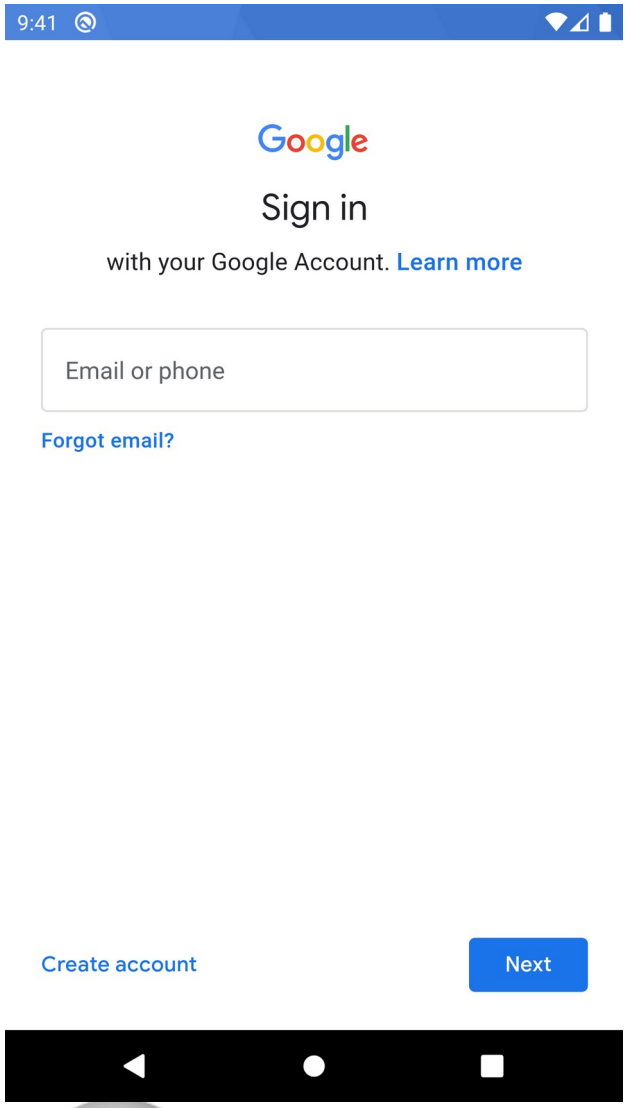


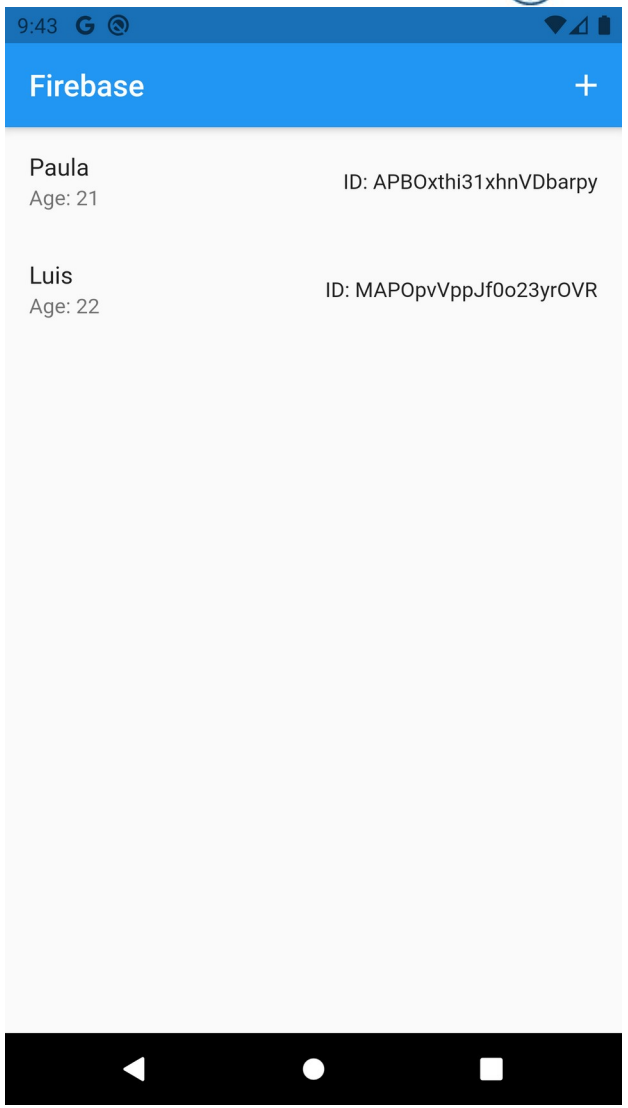
Firebase


AulaMobile2021 – Cloud Firestore – Console do Firebase - Mozilla Firefox
Gmail – email da Google
Importante: você ainda te
AulaMobile2021 – Cloud Fi
https://console.firebase.google.com/project/aulamobile2021/firestore/rules
Outlook
Alura
PO
Teste
Data Science Academy
Spring
Dados
URI
localhost
GitHub
PO
Liax
Valores do Melhor Ch...
Outros favoritos
Firebase
AulaMobile2021
Acessar a documentação
Visão geral do projeto
Cloud Firestore
Criação
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
Extensions
Spark
Plano gratuito US$ 0/mês
Fazer upgrade
Dados
Regras
Índices
Uso
Editar regras
Monitorar regras
Desenvolver e testar
Agora
alterações não publicadas
Hoje • 2:06 AM
alterações não publicadas
Publicar
Descartar
1 rules_version = '2';
2 service cloud.firestore {
3 match /databases/{database}/documents {
4 match /{document=**} {
5 allow read, write: if request.auth.uid != null;
6 }
7 }
8 }
Laboratório de testes de regras
Faça experimentos e explore as Regras de segurança

Firebase


9:41
Sign In
Email
Password
Log In
Register
Sign in with Google


9:41
Google
Sign in
with your Google Account. Learn more
Email or phone
Forgot email?
Create account
Next


9:43
Firebase
Paula
Age: 21
ID: APBOxthi31xhnVDbarpy
Luis
Age: 22
ID: MAPOpvVppJf0o23yrOVR

Broadcasting

A aplicação pode receber os documentos da coleção sempre que houver alguma alteração na coleção usando a stream:

```
FirebaseFirestore.instance.collection(_table).snapshots()
```

No arquivo **database.dart** no subdiretório **lib/database**:

Acrescente a importação:

```
import 'dart:async';
```

Acrescente o método para transformar a stream:

```
  static void handleSnapshot(_snapshot, EventSink _sink) {
    List<Map> ret = List<Map>();
    _snapshot.docs.forEach((_document) {
      Map _map = _document.data();
      _map['id'] = _document.id;
      ret.add(_map);
    });
    _sink.add(ret);
  }
```

Broadcasting

Acrescente o método para obter a stream com os documentos da coleção:

```
  static Stream<List<Map>> snapshots(String _table) {
    StreamTransformer<QuerySnapshot,List<Map>>
snapshotTransformer =
    StreamTransformer.fromHandlers(handleData:
handleSnapshot);
    return
FirebaseFirestore.instance.collection(_table).snapshots()
      .transform(snapshotTransformer);
  }
```

Broadcasting

No arquivo **student_dao.dart** no subdiretório **lib/dao**:

Acrescente o método para transformar a stream:

```
  void handleSnapshot(_snapshot, EventSink _sink) {
    List<Student> ret = List<Student>();
    _snapshot.forEach((_map) {
      Student _student = Student.fromJson(_map);
      ret.add(_student); });
    _sink.add(ret);
  }
```

Acrescente o método para obter a stream com os documentos da coleção:

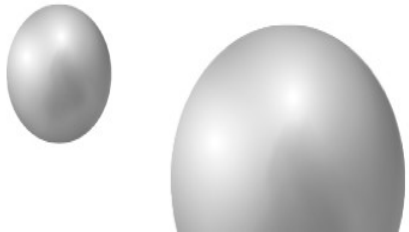
```
  Stream<List<Student>> snapshots() {
    StreamTransformer<List<Map>,List<Student>>
snapshotTransformer =
    StreamTransformer.fromHandlers(handleData:
handleSnapshot);
    return DatabaseHelper.snapshots('student')
      .transform(snapshotTransformer);
  }
```

Broadcasting

No arquivo **student_bloc.dart** no subdiretório **lib/bloc**:

Acrescente o método para obter a stream com os documentos da coleção:

```
Stream<List<Student>> snapshots() {
  return _dao.snapshots();
}
```



Broadcasting

No arquivo home.dart no subdiretório lib/pages:

Altere a linha:

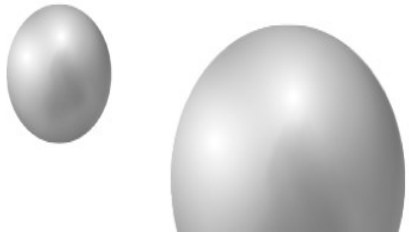
```
stream: _bloc.stream,
```

para:

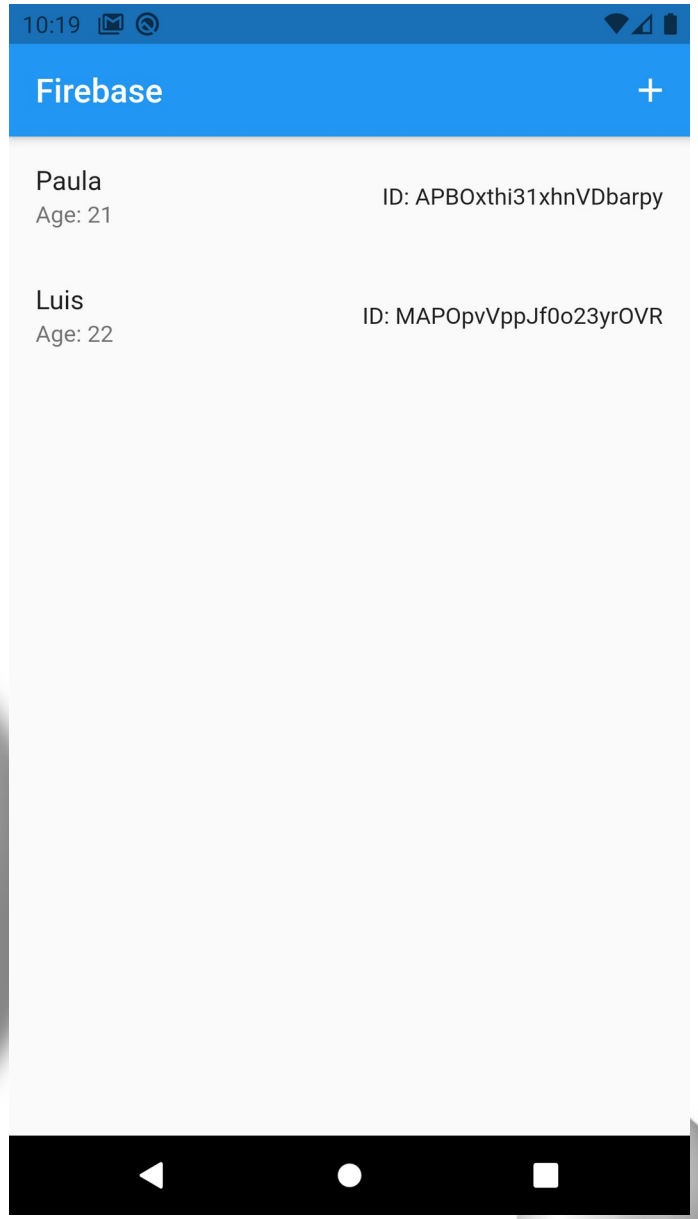
```
stream: _bloc.snapshots(),
```

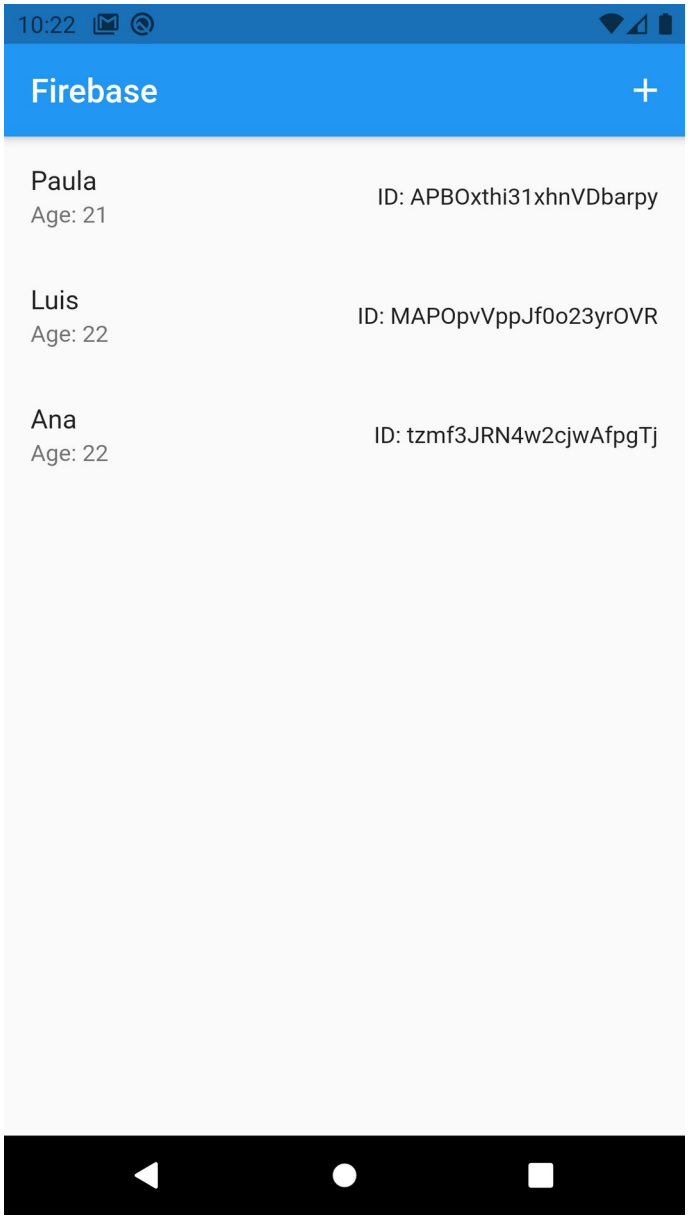
Remova a linha:

```
_bloc.getAll();
return _waitWidget;
```



Broadcasting

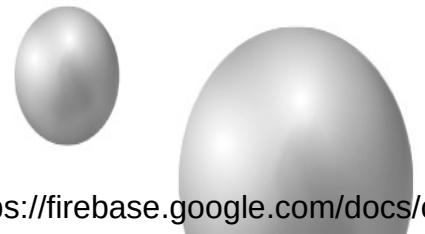




Messaging

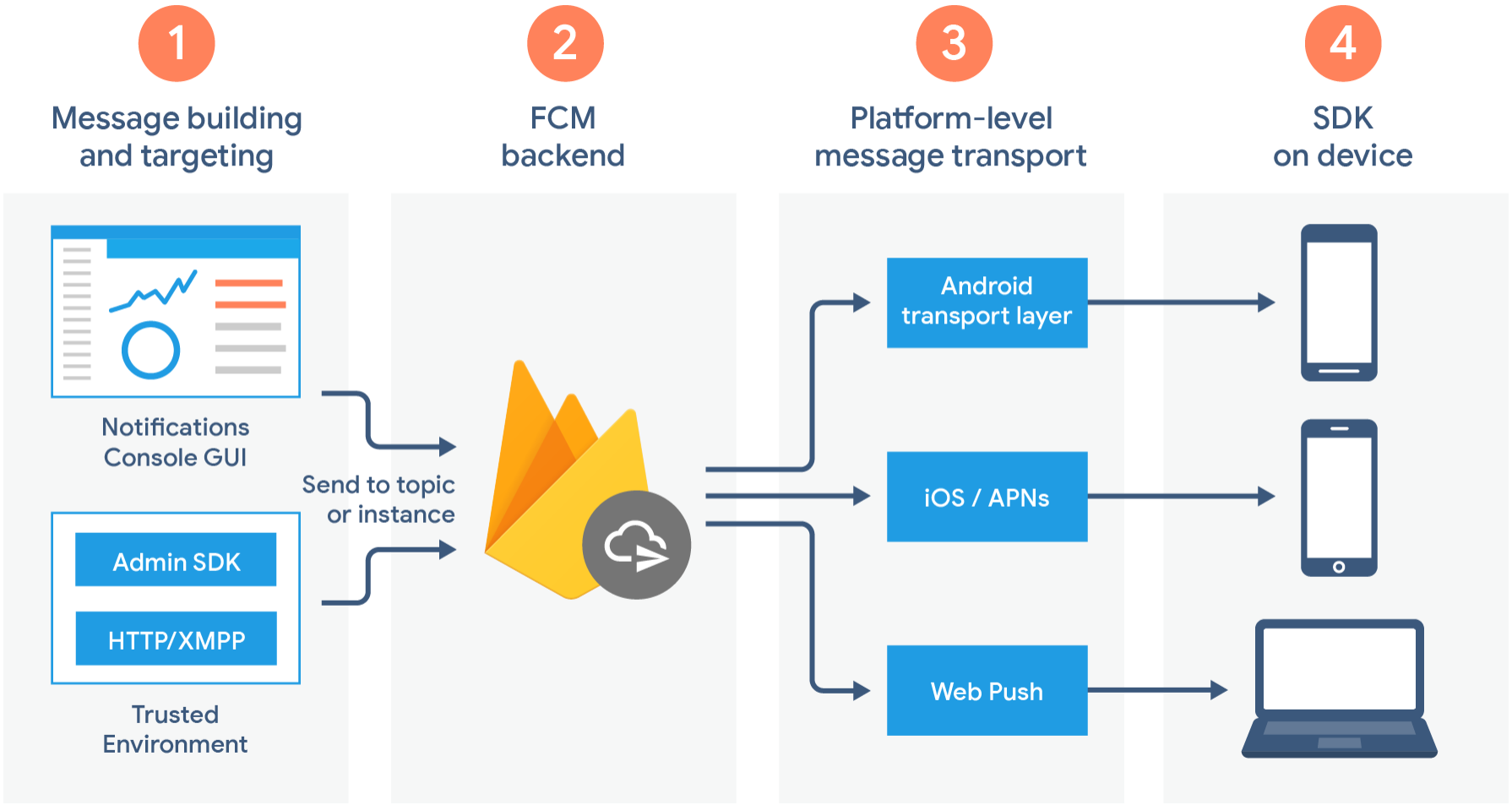
“O Firebase Cloud Messaging (FCM) é uma solução de mensagens entre plataformas que permite o envio confiável de notificações sem custo.

Usando o FCM, você pode notificar um app cliente de que novos e-mails ou outros dados estão disponíveis para sincronização. Você pode enviar mensagens de notificação para promover novas interações e a retenção de usuários. Para casos de uso como mensagens instantâneas, uma mensagem pode transferir um payload de até 4 KB para um app cliente.”



https://firebase.google.com/docs/cloud-messaging?hl=pt-br

Messaging



https://firebase.google.com/docs/cloud-messaging/fcm-architecture?hl=pt-br

Messaging

O Cloud Messaging permite o envio de mensagens ao usuários de uma aplicação utilizando diferentes critérios para selecionar a quais usuários a mensagem será direcionada.

É possível utilizar o console do Firebase ou outra aplicação para enviar as mensagens.

Quando o device receber uma mensagem para a aplicação será disparado um evento:

- **onMessage** disparado se a aplicação está executando em primeiro plano.
- **onResume** disparado se a aplicação está executando em segundo plano.
- **onLaunch** disparado se a aplicação não está em execução.

Messaging

	App in Foreground	App in Background	App Terminated
Notification on Android	`onMessage`	Notification is delivered to system tray. When the user clicks on it to open app `onResume` fires if `click_action: FLUTTER_NOTIFICATION_CLICK` is set (see below).	Notification is delivered to system tray. When the user clicks on it to open app `onLaunch` fires if `click_action: FLUTTER_NOTIFICATION_CLICK` is set (see below).
Notification on iOS	`onMessage`	Notification is delivered to system tray. When the user clicks on it to open app `onResume` fires.	Notification is delivered to system tray. When the user clicks on it to open app `onLaunch` fires.
Data Message on Android	`onMessage`	`onMessage` while app stays in the background.	*not supported by plugin, message is lost*
Data Message on iOS	`onMessage`	Message is stored by FCM and delivered to app via `onMessage` when the app is brought back to foreground.	Message is stored by FCM and delivered to app via `onMessage` when the app is brought back to foreground.

https://pub.dev/packages/firebase_messaging

Messaging

A mensagem será exibida no system tray se a aplicação não estiver em primeiro plano quando o device receber a mensagem.

Para que os eventos onResume e onLaunch sejam disparados ao clicar na mensagem e abrir a aplicação, será necessário o envio do click_action na mensagem.

Messaging

Altere o arquivo **pubspec.yaml** para incluir a dependência de **firebase_messaging**:

```
dependencies:
  firebase_core: ^1.1.0
  firebase_auth: ^1.1.2
  google_sign_in: ^5.0.2
  cloud_firestore: ^1.0.7
  firebase_messaging: ^6.0.13
  flutter:
    sdk: flutter
```

Instale o pacote adicionado nas dependências.

Messaging

Altere o arquivo **AndroidManifest.xml** no subdiretório **android/app/src/main**:

```
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|
keyboardHidden|keyboard|screenSize|smallestScreenSize|
locale|layoutDirection|fontScale|screenLayout|density|
uiMode"
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <intent-filter>
                <action
android:name="FLUTTER_NOTIFICATION_CLICK"/>
                <category
android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
```

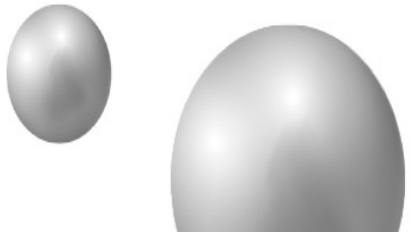
Messaging

Crie o arquivo showmessage.dart no subdiretório lib/utils:

```
import 'package:flutter/material.dart';

void showMessage(BuildContext context, String _title, String
_body) async {
  return await showDialog<void>(
    context: context,
    barrierDismissible: false,
    builder: (BuildContext context) {
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20.0),
        ),
```

Messaging

```
        title: Row(
            children: [
              Icon(Icons.cloud,color:
Colors.deepOrange,size: 48.0,),
              Expanded(
                child: Text(
                  _title,
                  style: TextStyle(fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center,
                ),
              ),
            ]
        ),
```



Messaging

```
        content: Text(_body),
        actions: <Widget>[
          FlatButton(
            child: Text('Ok'),
            onPressed: () => Navigator.of(context).pop(),
          ),
        ],
      );
    },
  );
}
```

Messaging

No arquivo home.dart no subdiretório lib/pages:

Inclua as importações:

```
import 'package:firebase_messaging/firebase_messaging.dart';
import '../utils/showmessage.dart';
```

Inclua as linhas:

```
  // BLoC para estudantes
  StudentBLoC _bloc = StudentBLoC();
  // Gerenciador das mensagens
  final FirebaseMessaging _firebaseMessaging =
FirebaseMessaging();
```

Messaging

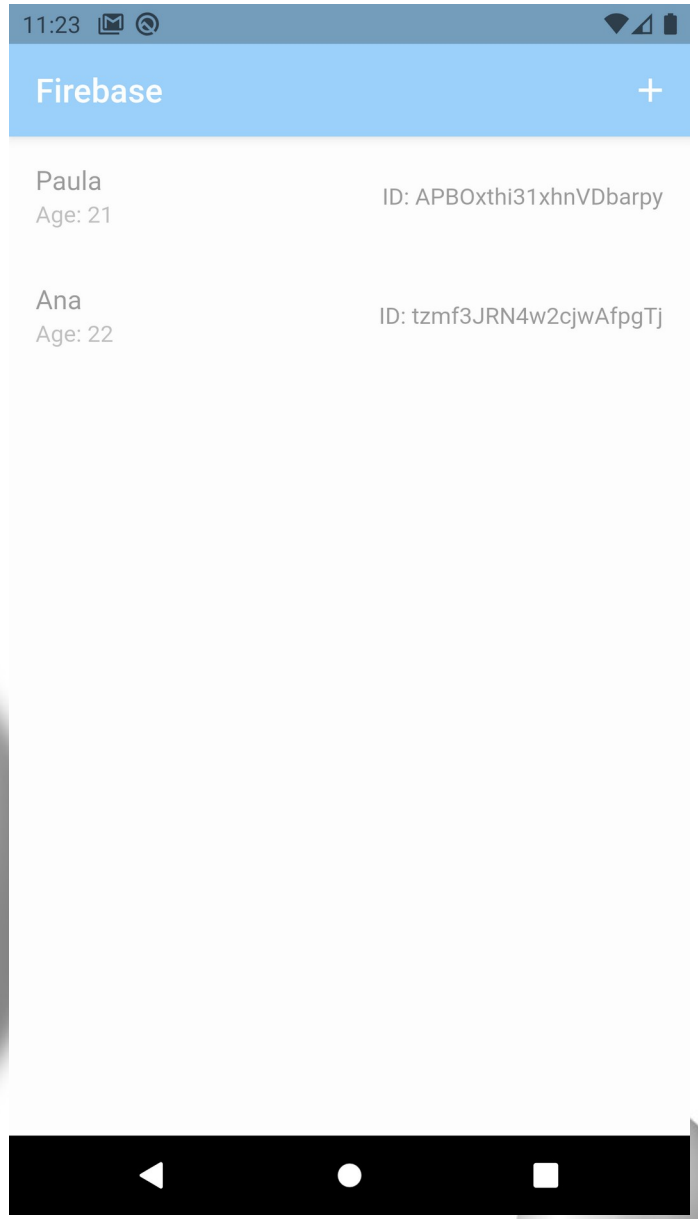
Inclua o gerenciamento das mensagens:

```
@override
void initState() {
  super.initState();
  _firebaseMessaging.configure(
    onMessage: (Map<String, dynamic> message) async {
      print('on message $message');
      await showMessage(context,
        message["notification"]["title"],
        message["notification"]["body"]
      );
    },
    onResume: (Map<String, dynamic> message) async {
      print('on resume $message');
      await showMessage(context,
          message["notification"]["title"],
          message["notification"]["body"]
      );
    },
```

Messaging

```
    onLaunch: (Map<String, dynamic> message) async {
      print('on launch $message');
      await showMessage(context,
          message["notification"]["title"],
          message["notification"]["body"]
      );
    }
  );
}
```

Messaging


11:23
Firebase
Paula
Age: 21
ID: APBOxthi31xhnVDbarpy
Ana
Age: 22
ID: tzmf3JRN4w2cjwAfpgTj


11:22
Firebase
Paula
Age: 21
ID: APBOxthi31xhnVDbarpy
Ana
Age: 22
ID: tzmf3JRN4w2cjwAfpgTj
Teste
Exemplo de mensagem do Cloud Messaging
Ok