



# Firestore para Aplicações Móveis

Dispositivos Móveis

Prof. Edson Mota, PhD, MSc, PMP

# SQL **vs** NoSQL



# Diferenças

- Os bancos de dados convencionais são relacionais, preocupam-se em estabelecer relações entre temas, respeitando critérios rígidos de integridade.
- Na última década, ficou claro que nem todo problema de gerenciamento e análise de dados precisa ser solucionado por um SGBD convencional.
- Uma alternativa para isso são os bancos de dados NoSQL

(NoSQL → Não usa SGBD)

# Motivações NoSQL

- Há duas razões principais que motivam a troca de um SGBDR por um sistema NoSQL (Michael Stonebraker)
  - Necessidade de um desempenho melhor
  - Necessidade de mais flexibilidade na organização dos dados

# Características

- Esquema completamente flexível
- Nem sempre consistente
- Falta de padronização
- Via de regra, não suporta JOINS, embora, as ferramentas forneçam soluções de contorno.
- **Bancos de dados NoSQL são projetados com foco na escalabilidade horizontal e trabalham de forma independente do hardware.**

# Tipos de NoSQL

- **Key/Value Stores**

- Armazena os dados por meio de uma estrutura de mapeamento chave/valor

- **Famílias de Colunas**

- Consistem em uma estratégia de organização que agrupa colunas que armazenam o mesmo tipo de dados

- **Baseado em Grafos**

- Composto de nós, arestas e propriedades, estes bancos de dados estruturam suas informações em um formato de grafo

- **Baseado em Documentos (mais popular)**

- Consiste em um sistema de organização no qual um documento oferece uma visão geral da coleção, com seus atributos e valores
- Este atributo pode ser composto por um único campo ou um conjunto de campos que podem ter novas referências e assim por diante.

# Representação dos Dados

- Os bancos de dados **NoSQL** podem utilizar diferentes formatos para representação de seus dados, isso varia em função do seu tipo
  - Alguns formatos são: **XML**, **BSON** (Binary JSON), **CQL** (Cassandra Query Language)
- Entre os **bancos de dados de documento**, uma representação comum é o formato:
  - JavaScript Object Notation (**JSON**)

```
{
  "nome": "Maria",
  "idade": 30,
  "email": "maria@gmail.com",
  "endereco": {
    "rua": "Rua A",
    "numero": 123,
    "cidade": "São Paulo"
  },
  "telefones": [
    "(11) 1111-1111",
    "(11) 2222-2222"
  ],
  "ativo": true
}
```

# Alguns Bancos de Dados Relacionais

ORACLE





# Alguns Bancos de Dados NOSQL



firebase



elasticsearch

Já ouviu falar em **Firestore**?



# O que é isso?

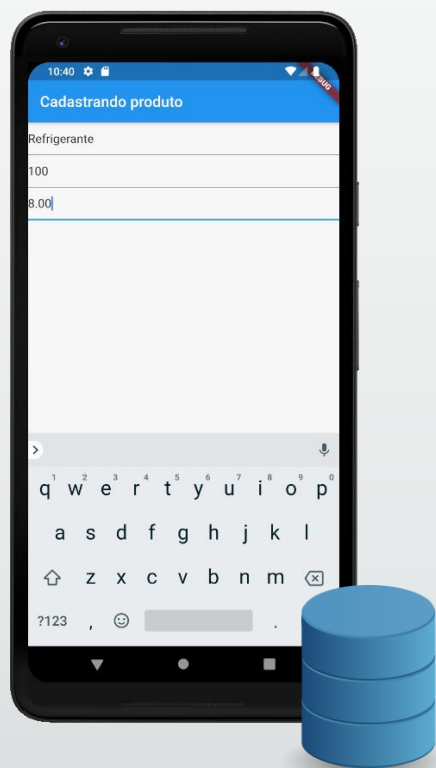
- Firebase é uma plataforma poderosa para sincronismo de dados
- Baseada em NoSQL
- Foco na atualização de dados em tempo real

The Google logo, consisting of the word "Google" in its characteristic multi-colored font.

firebase

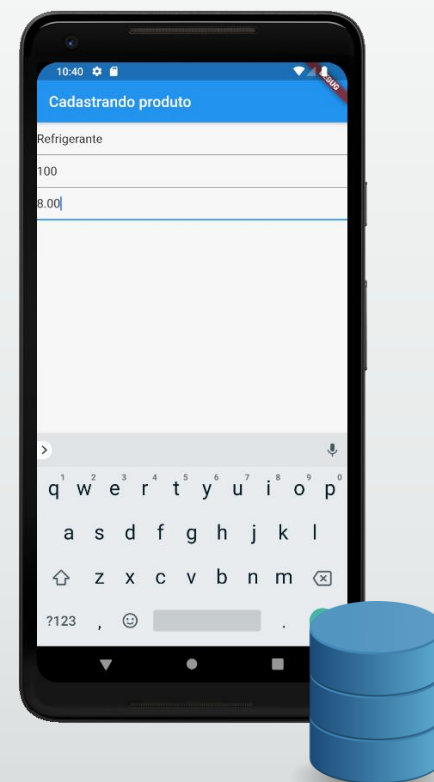
# Persistência de Dados

Cadastro: Produto A



Existe um  
problema nessa  
abordagem?

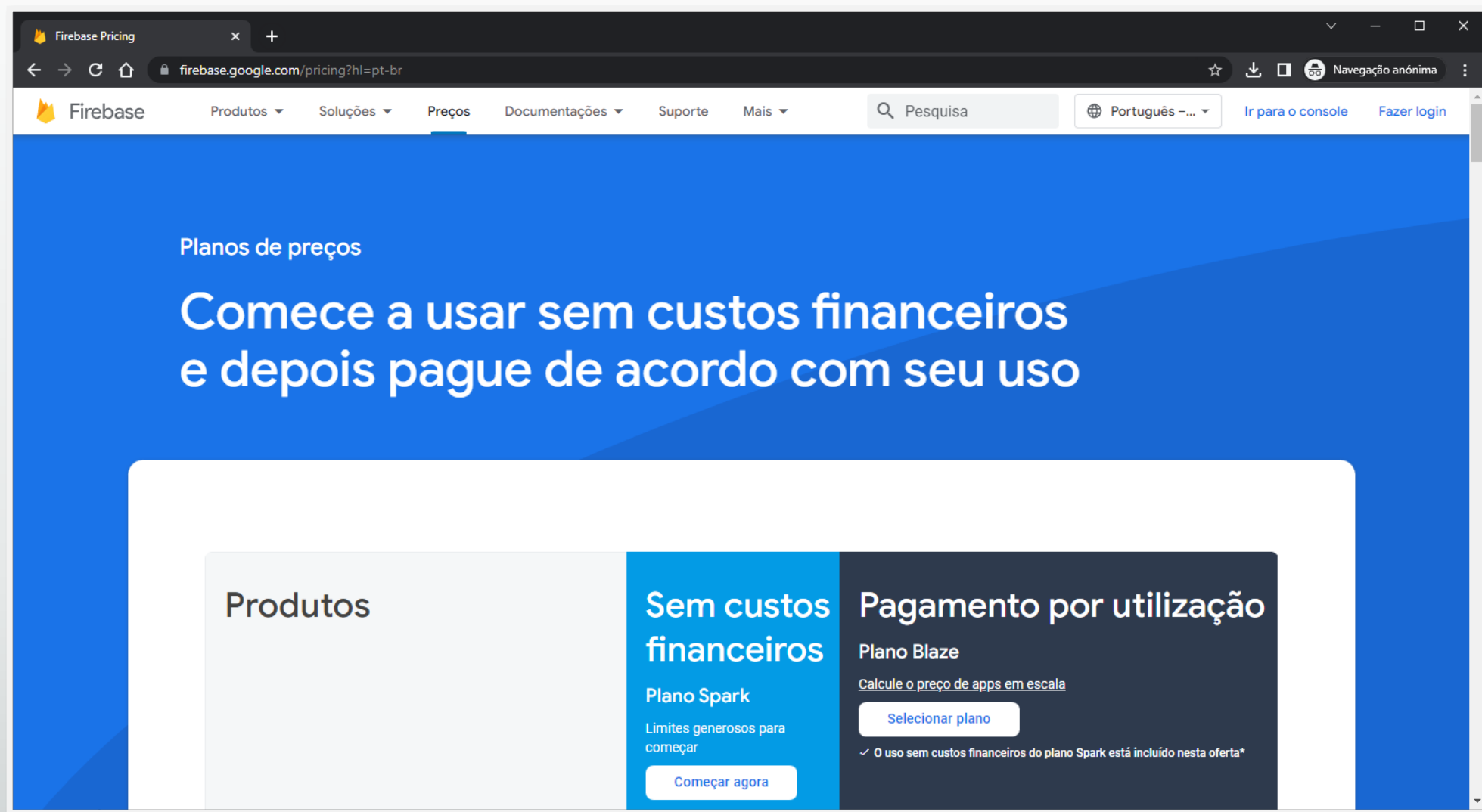
Cadastro: Produto B



# Onde entra o Firebase?

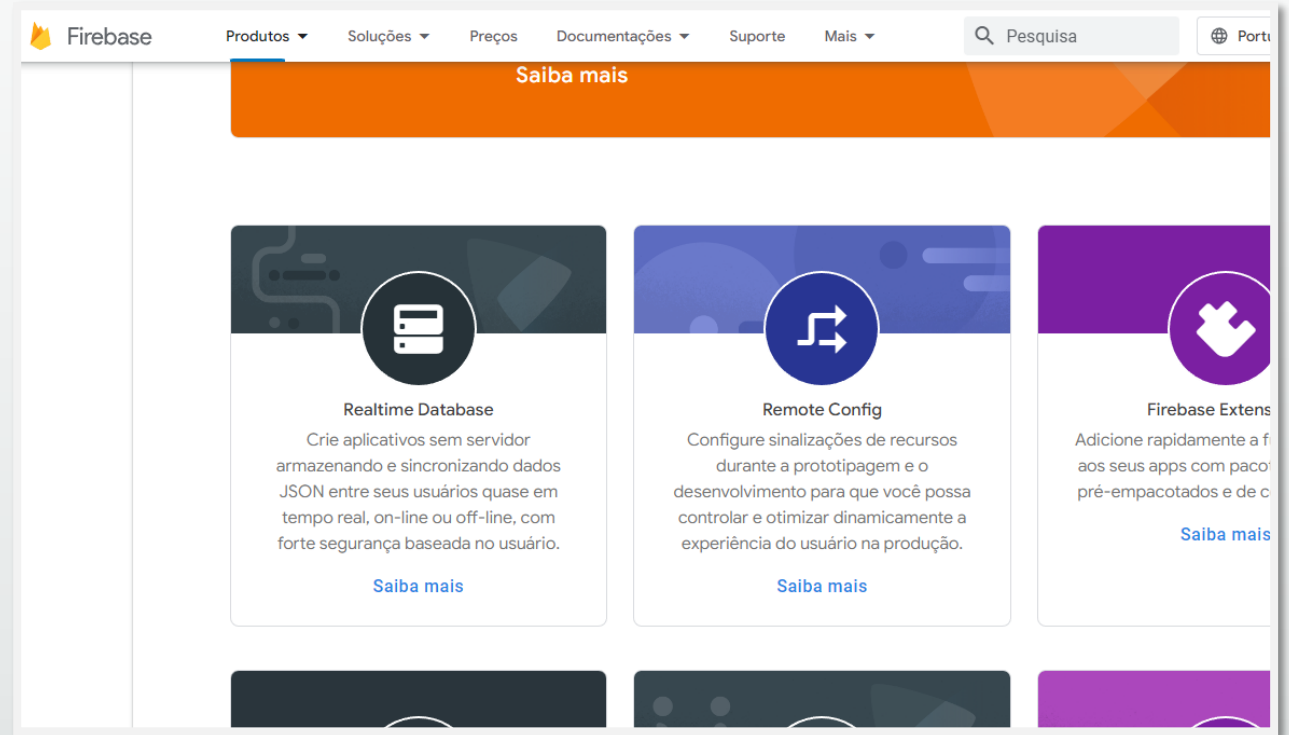


# É uma Solução Comercial (free/paga)



# Firestore oferece vários produtos

- RealTime Database
- Serviços de Autenticação
- Serviços de Armazenamento
- Serviços de Monitoramento
- Cloud Message / Notificações
- E muitos outros.
- Além de uma ampla documentação adaptada para iOS ou Android



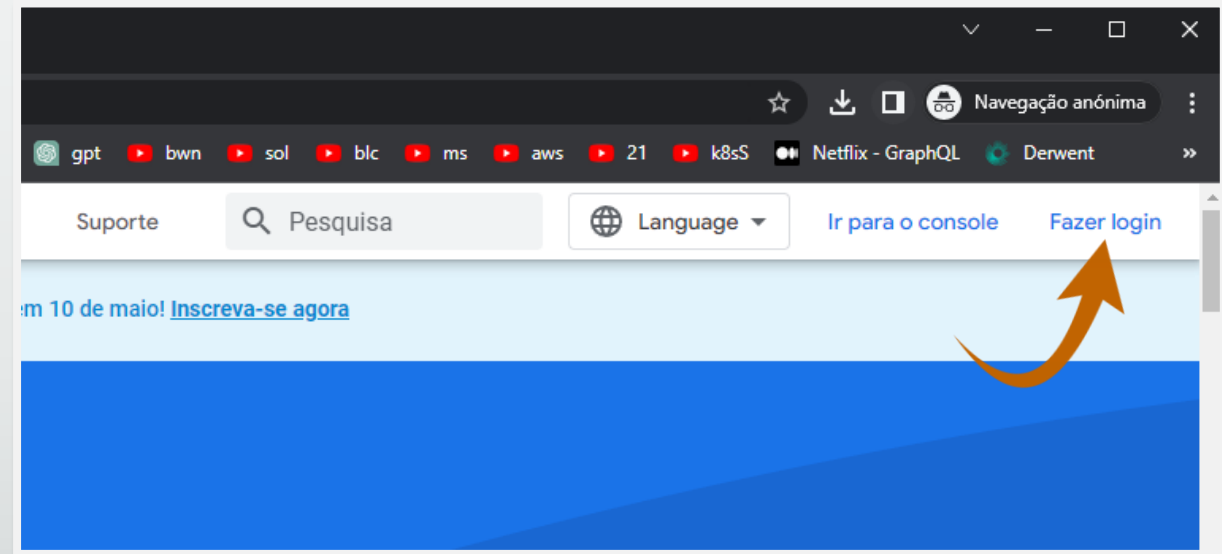
# Conta **Firestore**.





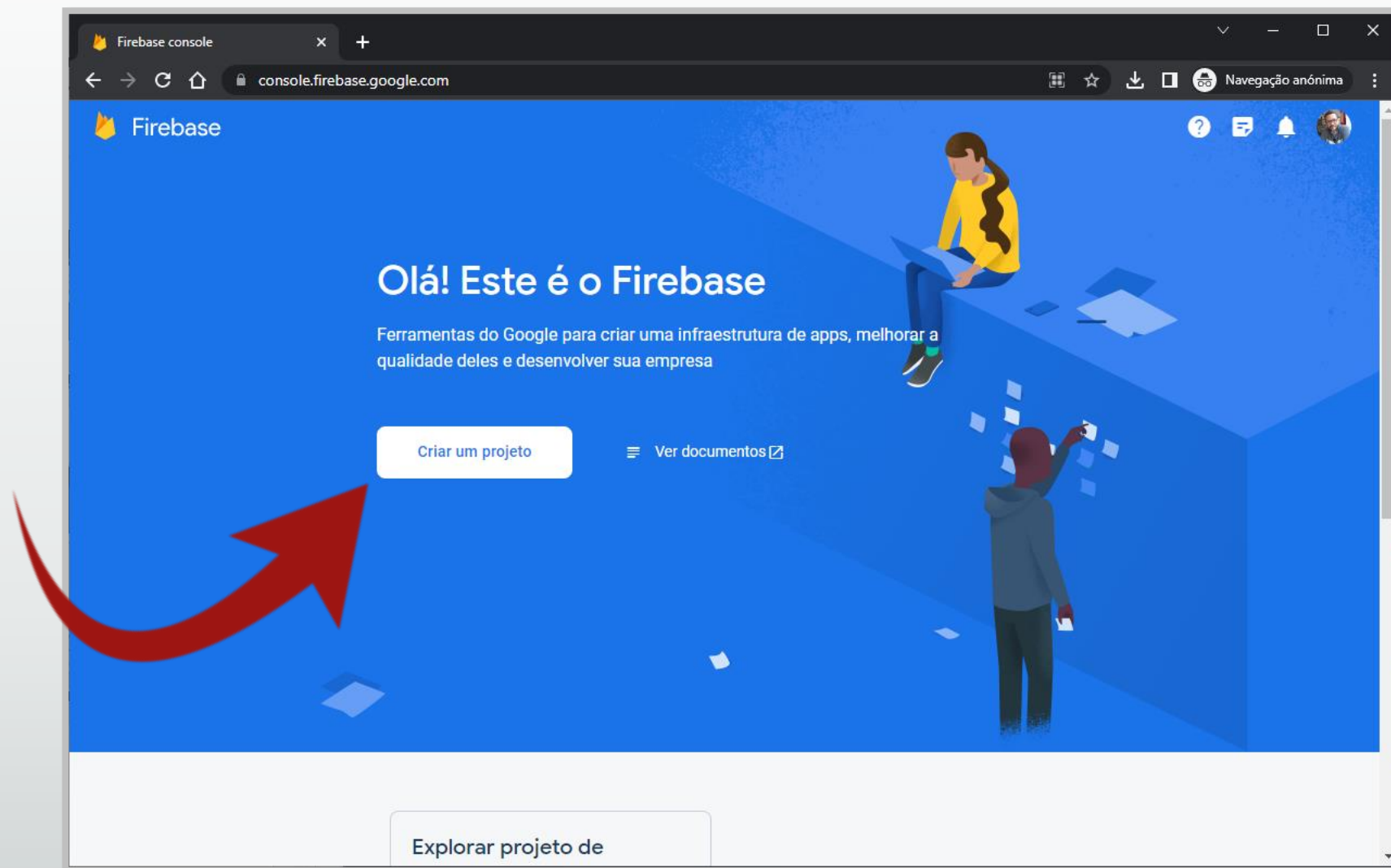
# Criar uma Conta

- A primeira coisa a fazer é criar uma conta no Firebase.
- Essa conta garantirá acesso aos serviços de maneira que seja possível testar os recursos dessa plataforma
- É preciso ter uma conta no Google
- Em seguida, acesse o **console**

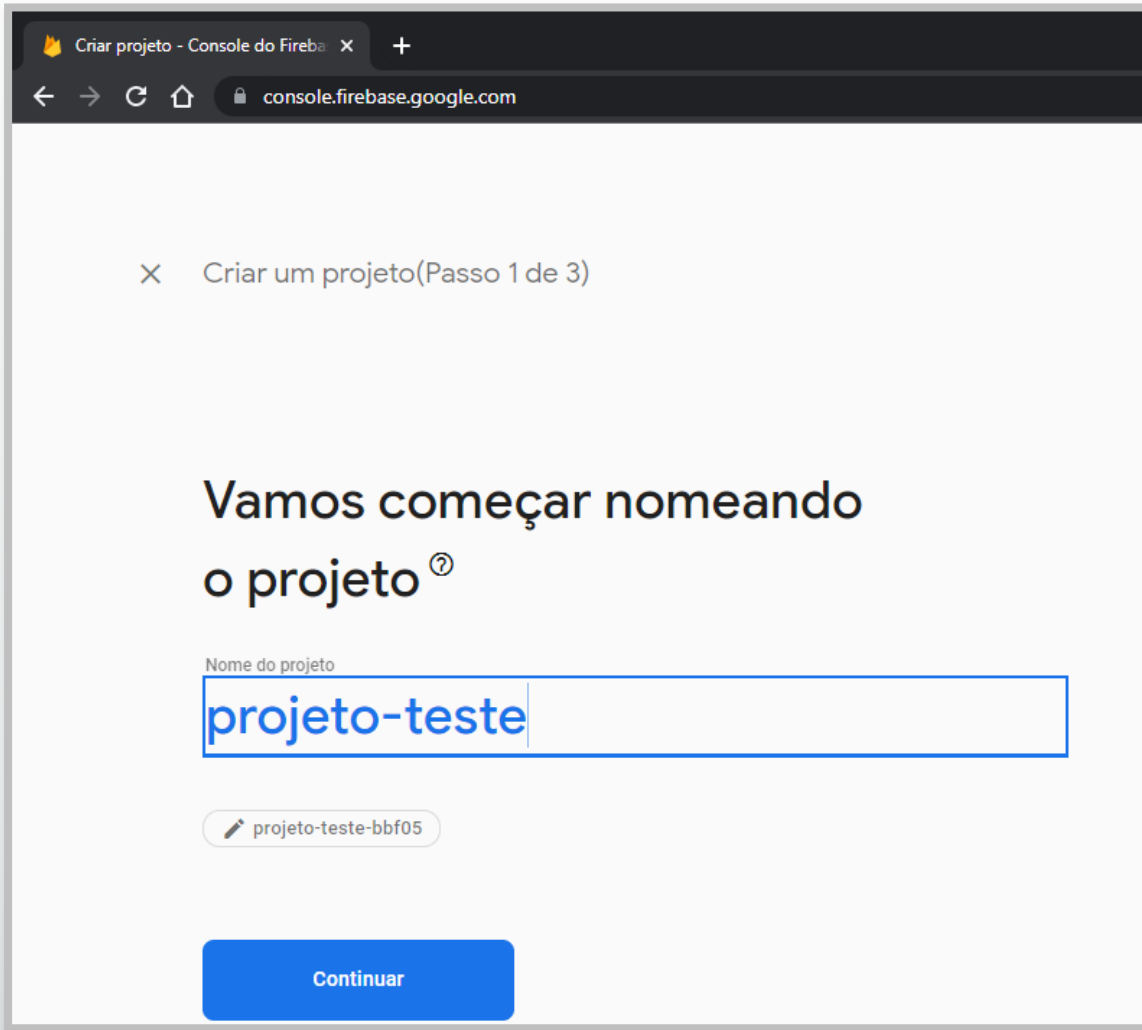


# Criando um Projeto

- Um projeto no Firebase é usado para agrupar um conjunto de recursos para o seu APP, além de fornecer definições gerais de acesso e outras configurações




# Criando um Projeto



The screenshot shows the 'Criar projeto' (Create project) page in the Firebase console. The browser tab is 'Criar projeto - Console do Firebase' and the address bar shows 'console.firebase.google.com'. The page title is 'Criar um projeto (Passo 1 de 3)'. The main heading is 'Vamos começar nomeando o projeto'. Below this is a text input field labeled 'Nome do projeto' containing the text 'projeto-teste'. Below the input field is a preview of the project name 'projeto-teste-bbf05' with an edit icon. At the bottom is a blue button labeled 'Continuar'.

- **Desmarque** a opção **Google Analytics**, caso contrário, você precisará primeiro criar uma conta naquela plataforma também
- Uma vez configurado, crie o projeto. O processo pode demorar alguns minutos.

 **Firestore**

 **Visão geral do projeto** ⚙️

Categorias dos produtos

Criação ▾

Liberar e monitorar ▾

Analytics ▾

Engajamento ▾

🗪 Todos os produtos

**Spark** Faz...  
Sem custos financeiros US\$ 0/mês

projeto-teste ▾

? 📧 🔔 

✉️ Receber atualizações por e-mail sobre novos recursos, pesquisas e eventos do Firebase

Inscreva-se ✕

**projeto-teste** Plano Spark

**Comece adicionando o  
Firebase ao seu  
aplicativo**



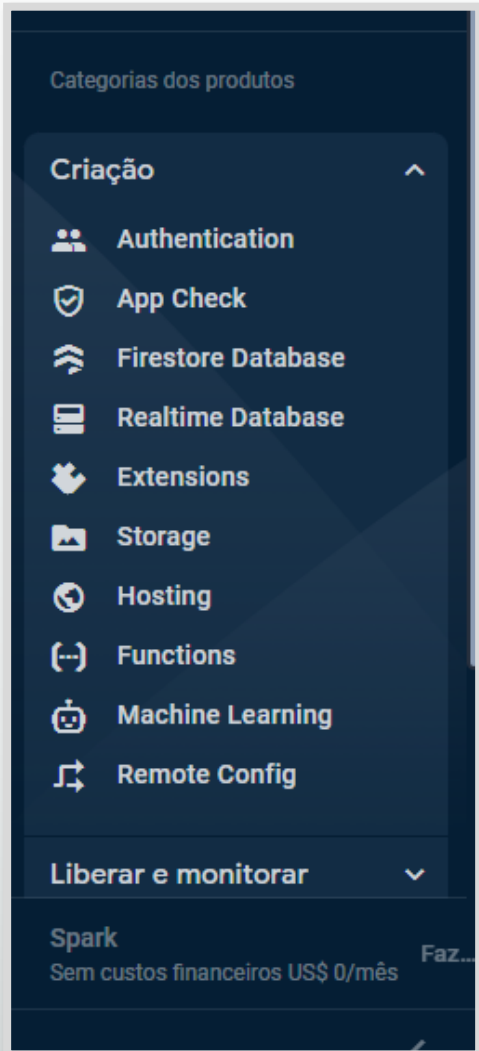
Adicione um app para começar



Armazene e sincronize dados de app em milissegundos ✕



# Recursos



- A barra lateral oferece recursos gerais do Firebase, além de acesso a serviços e produtos
- **Authentication**, por exemplo, nos permite criar um sistema de autenticação baseado em email, telefone, senha, facebook, twitter, etc.
- **Realtime Database**: Banco de dados estruturado em formato JSON
- **Firestore Database**: Banco de dados NoSQL baseado em documentos
- **Storage**: Permite armazenamento de arquivos
- E muitos outros recursos.
  - Vale a pena estudar, testar e entender como funcionam
  - Use a documentação!

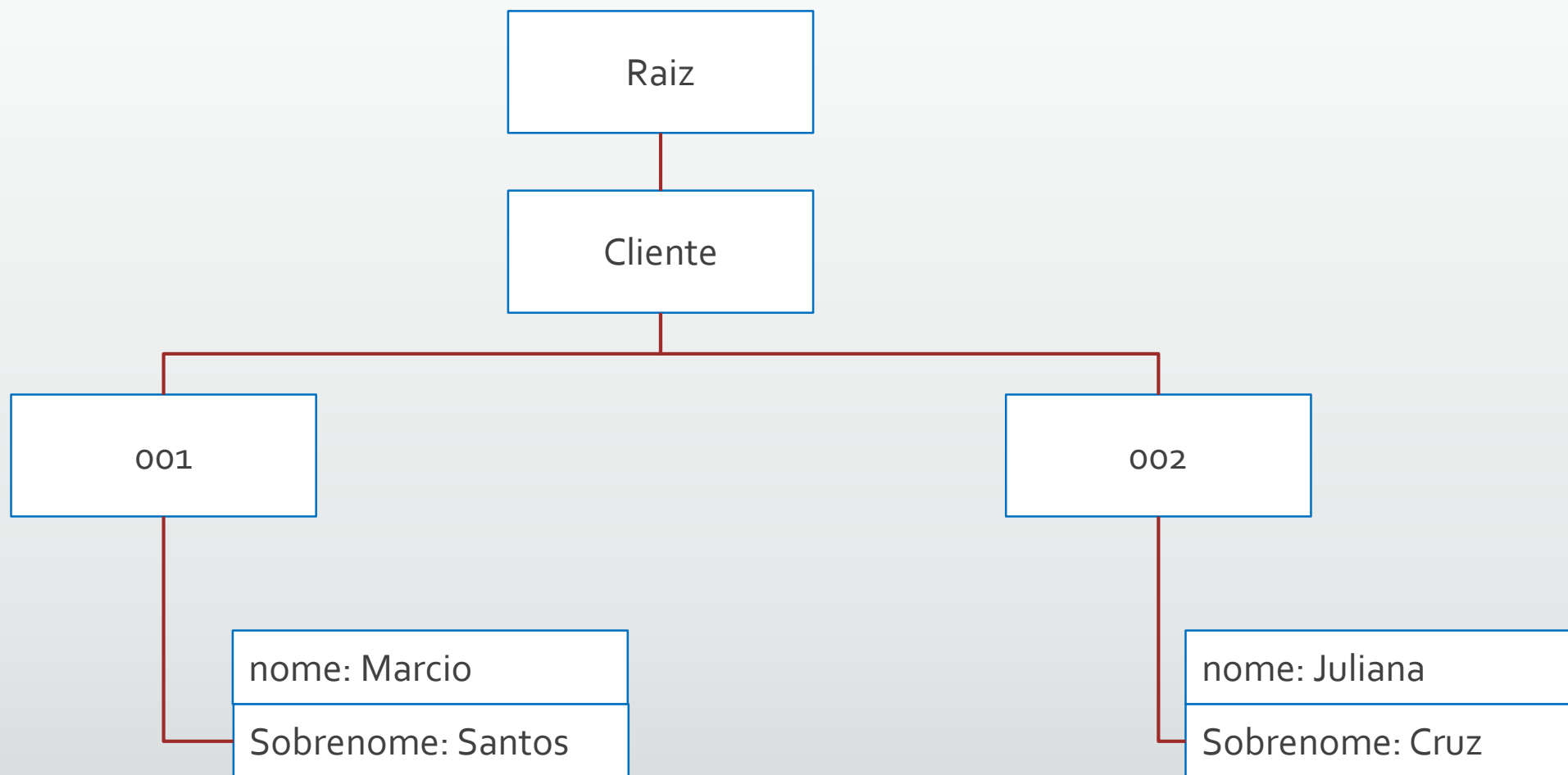
# Estrutura de Dados

Cliente		
Código	Nome	Sobrenome
001	Marcio	Santos
002	Juliana	Cruz



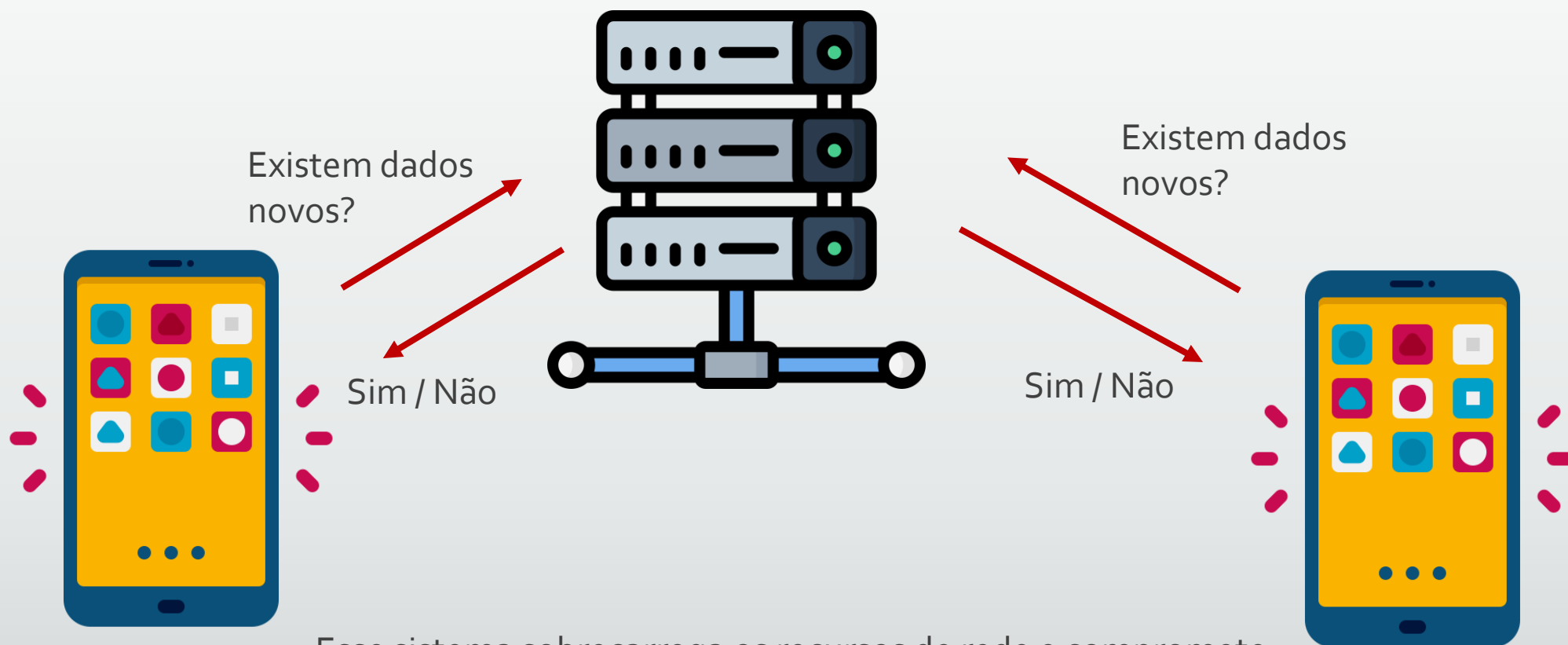
```
{
  "clientes": [
    {
      "001": {
        "nome": "Marcio",
        "sobrenome": "Santos"
      },
      "002": {
        "nome": "Juliana",
        "sobrenome": "Cruz"
      }
    }
  ]
}
```

# Armazenamento de Dados



# Sistemas de Requisições Convencionais

Em serviços de nuvens convencionais, os sistemas precisam manter uma comunicação constante com o banco de dados, a fim de identificar quando ocorreram as modificações.

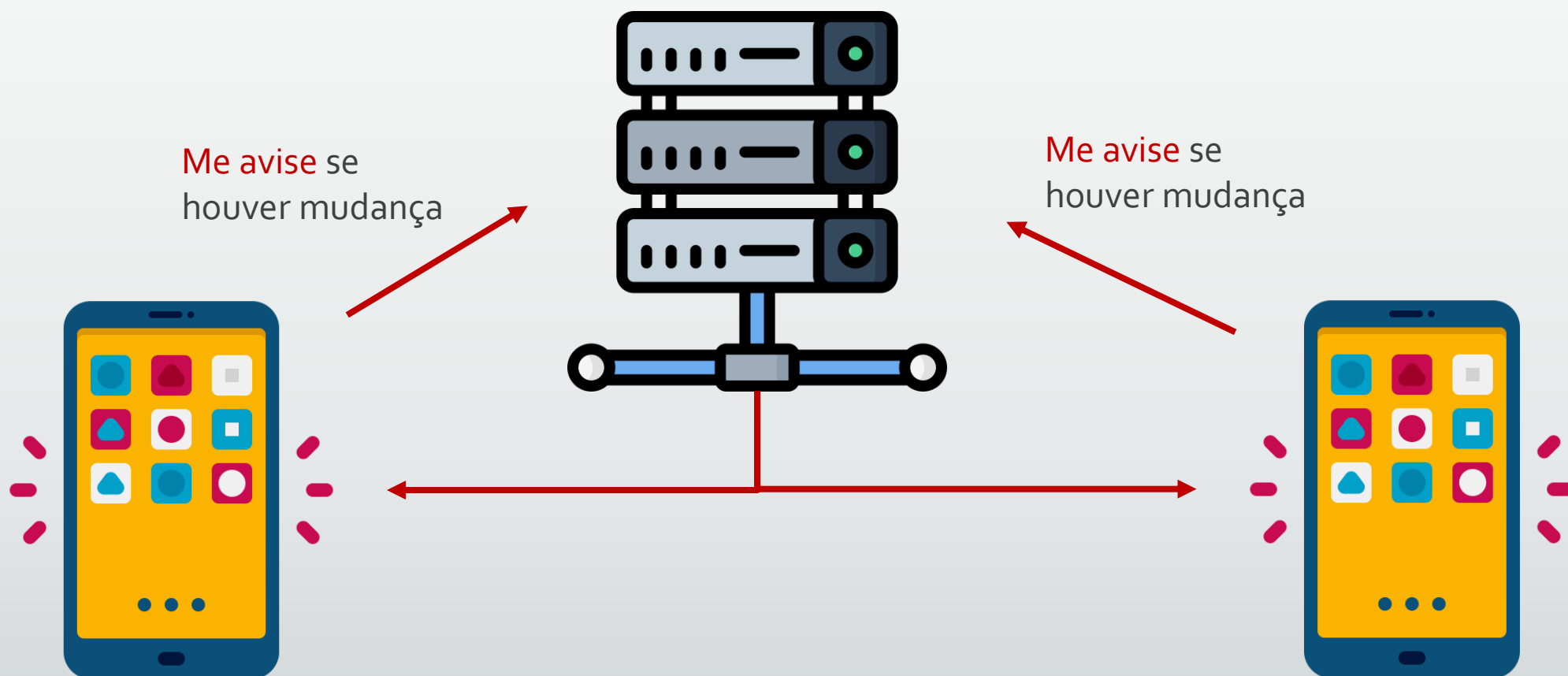


Esse sistema sobrecarrega os recursos de rede e compromete o processamento do dispositivo.

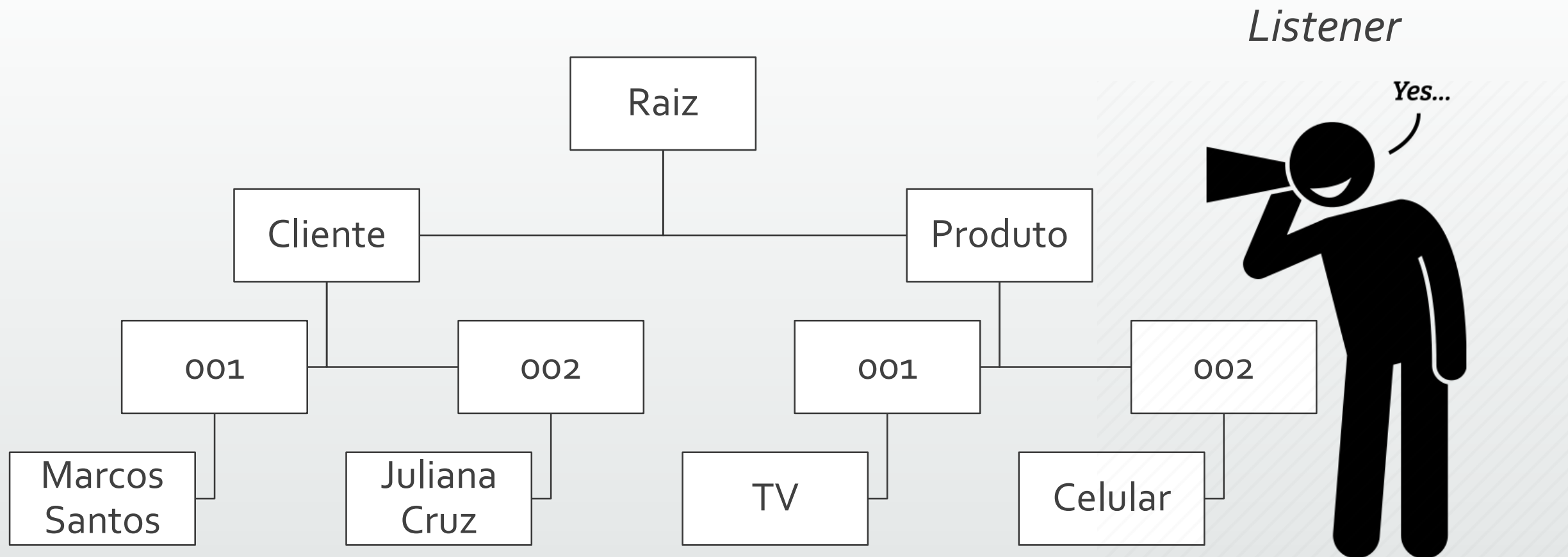


# Sistema de Requisições **Firestore**

Trata-se de um sistema baseado em notificações. O serviço Firestore fica responsável por informar a aplicação quando ocorrerem mudanças na estrutura de dados. Ao fazer isso, os dados são sincronizados.



# Como funciona?

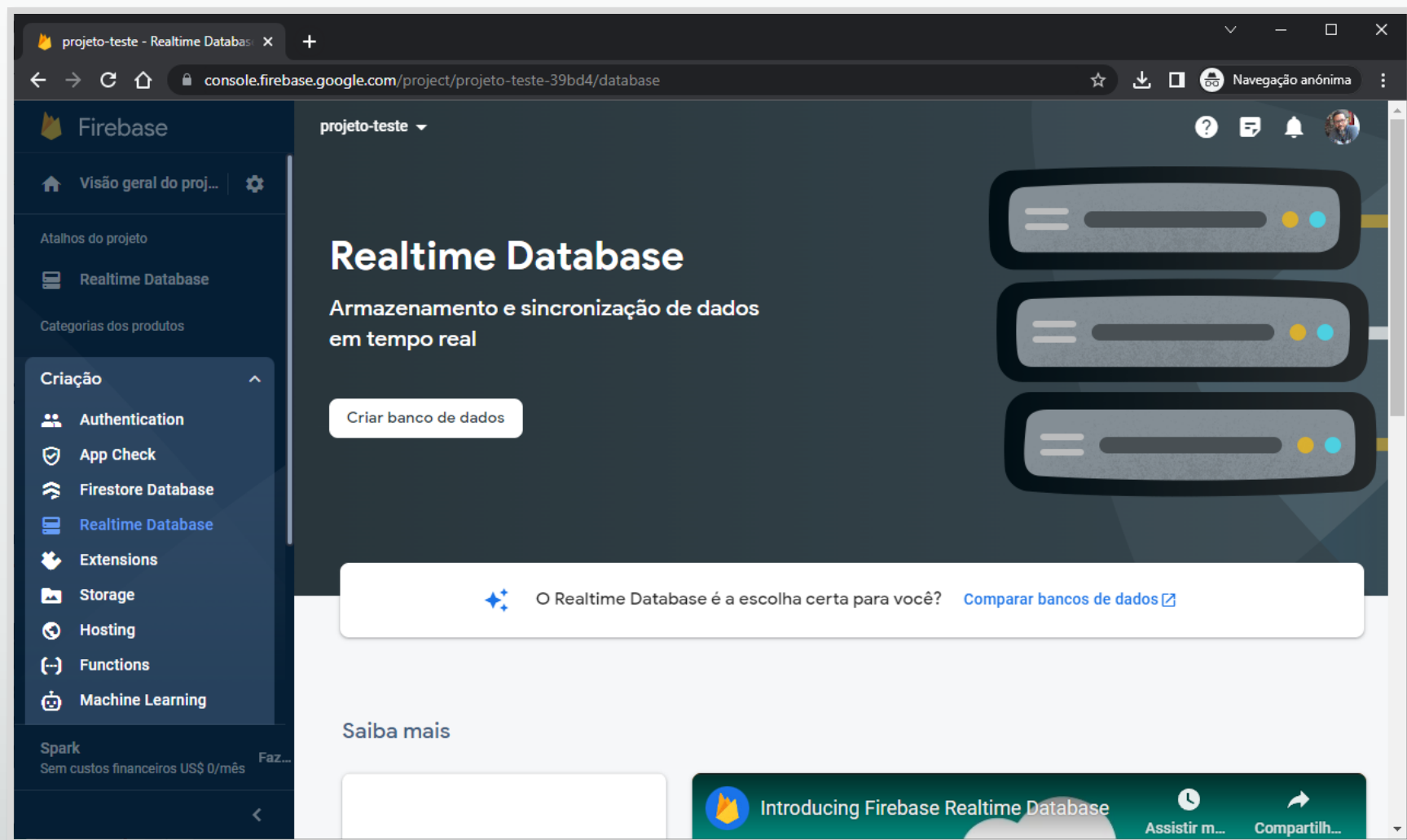


Um Listener deve ser programado na implementação do APP de modo que as notificações de mudanças sejam comunicadas tão logo elas ocorram na estrutura de dados.

# Criando o Banco de Dados

# Realtime Database

- Vamos criar um serviço de banco de dados em tempo real.
- A página do Realtime Database oferece uma ampla documentação para integração destes recursos com Android, iOS, entre outras plataformas.
- Para começar, clique em Criar Banco de Dados



# Proximidade do Servidor

**Configurar banco de dados** ×

**1** Opções de banco de dados — **2** Regras de segurança

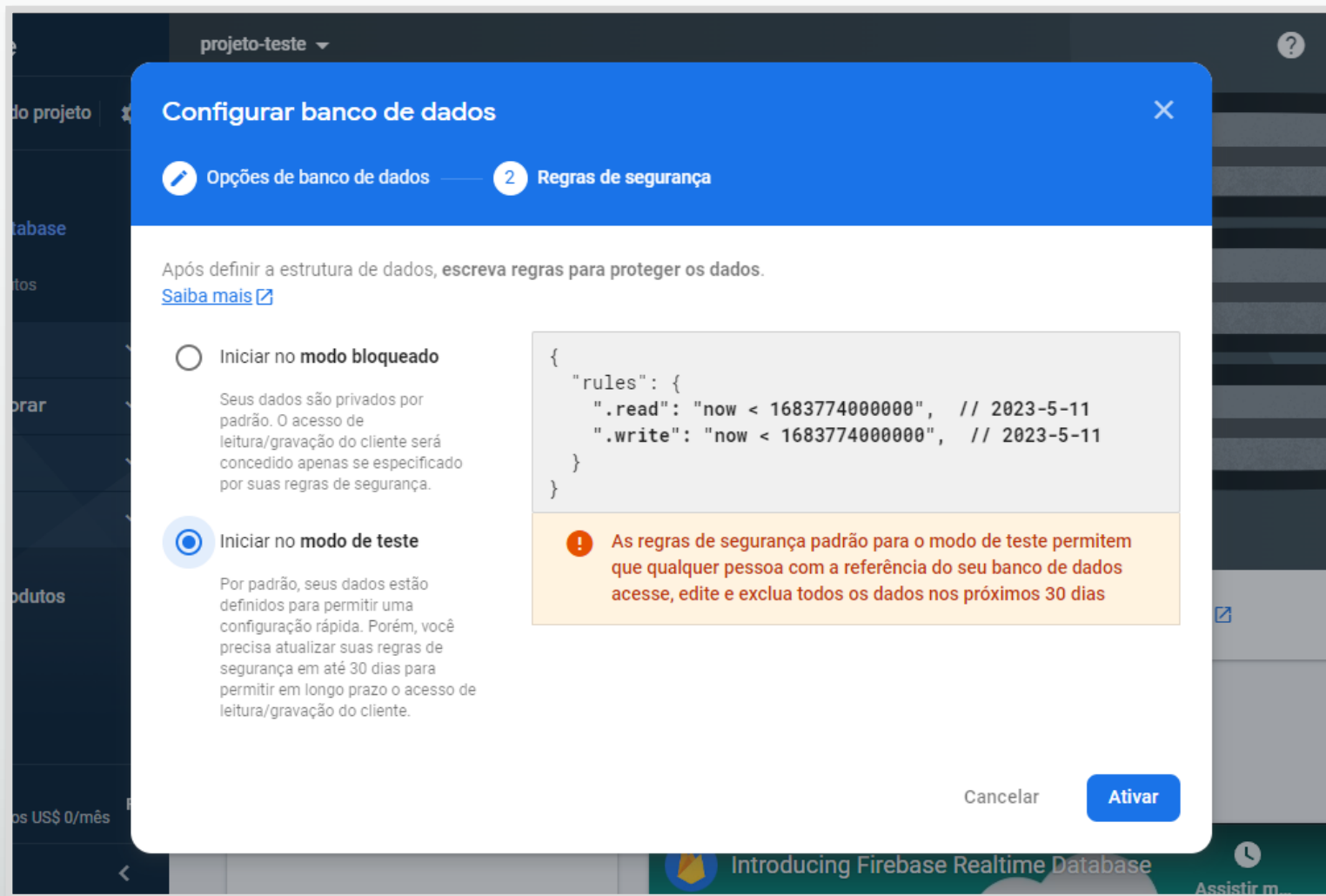
Sua configuração de local é onde os dados do Realtime Database serão armazenados.

Local do Realtime Database

Estados Unidos (us-central1) ▼

Cancelar **Próxima**

# Permitir Acesso Anônimo



- Essa configuração vai permitir que qualquer pessoa com o link possa acessar e modificar a sua estrutura de dados
- **Naturalmente, essa não é uma boa prática**, mas vamos resolver isso quando estudarmos os métodos de autenticação.
- Clique em **Ativar** para carregar o novo banco

projeto-teste - Realtime Database

console.firebase.google.com/project/projeto-teste-39bd4/database/projeto-teste-39bd4-default-rtdb/data

Navegação anônima

Realtime Database

DadosRegrasBackupsUsoExtensões

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

Configurar o App Check

Criação

Liberar e monitorar

Analytics

Engajamento

Todos os produtos

Spark

Configurações Gerais

Endereço do banco de dados

Nó Raiz

Adiciona e deleta objetos

https://projeto-teste-39bd4-default-rtdb.firebaseio.com

https://projeto-teste-39bd4-default-rtdb.firebaseio.com/: null

Local do banco de dados: Estados Unidos (us-central1)



# Realtime Database

Proteja os recursos do Realtime Database de abusos, como fraude de faturamento ou phishing [Configurar o App Check](#)

https://projeto-teste-39bd4-default-rtdb.firebaseio.com

https://projeto-teste-39bd4-default-rtdb.firebaseio.com/

```
└─ Clientes
  └─ 001
    └─ nome: "Marcio"
    └─ sobrenome: "Santos"
  └─ 002
    └─ nome: "Juliana"
    └─ sobrenome: "Cruz"
└─ Produtos
```

Utilize o montador para gerar  
essa estrutura de dados

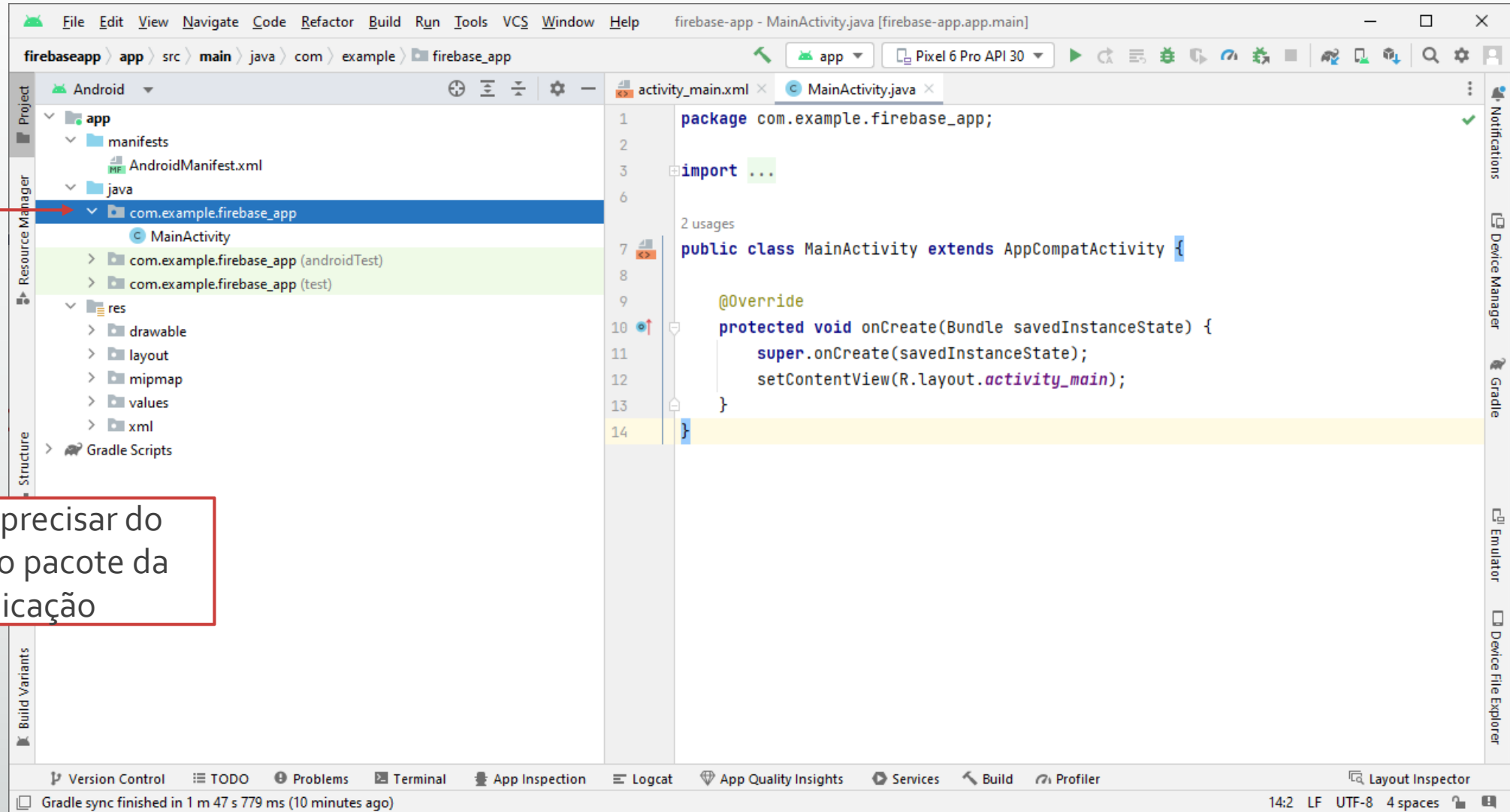


# Revisão

- Crie uma estrutura de dados no Realtime Database do Firebase para armazenar informações sobre alunos de uma escola. Sua estrutura deve ter os seguintes campos:
  - Nome Completo
  - Idade do Aluno
  - Telefone e email do aluno
  - Semestre que o aluno está cursando
  - Disciplinas Cursadas
  - Histórico de Pagamentos
- Certifique-se de que cada campo esteja armazenado em um local apropriado na árvore do banco de dados e que os dados sejam salvos em tempo real, conforme as informações são adicionadas ou atualizadas.

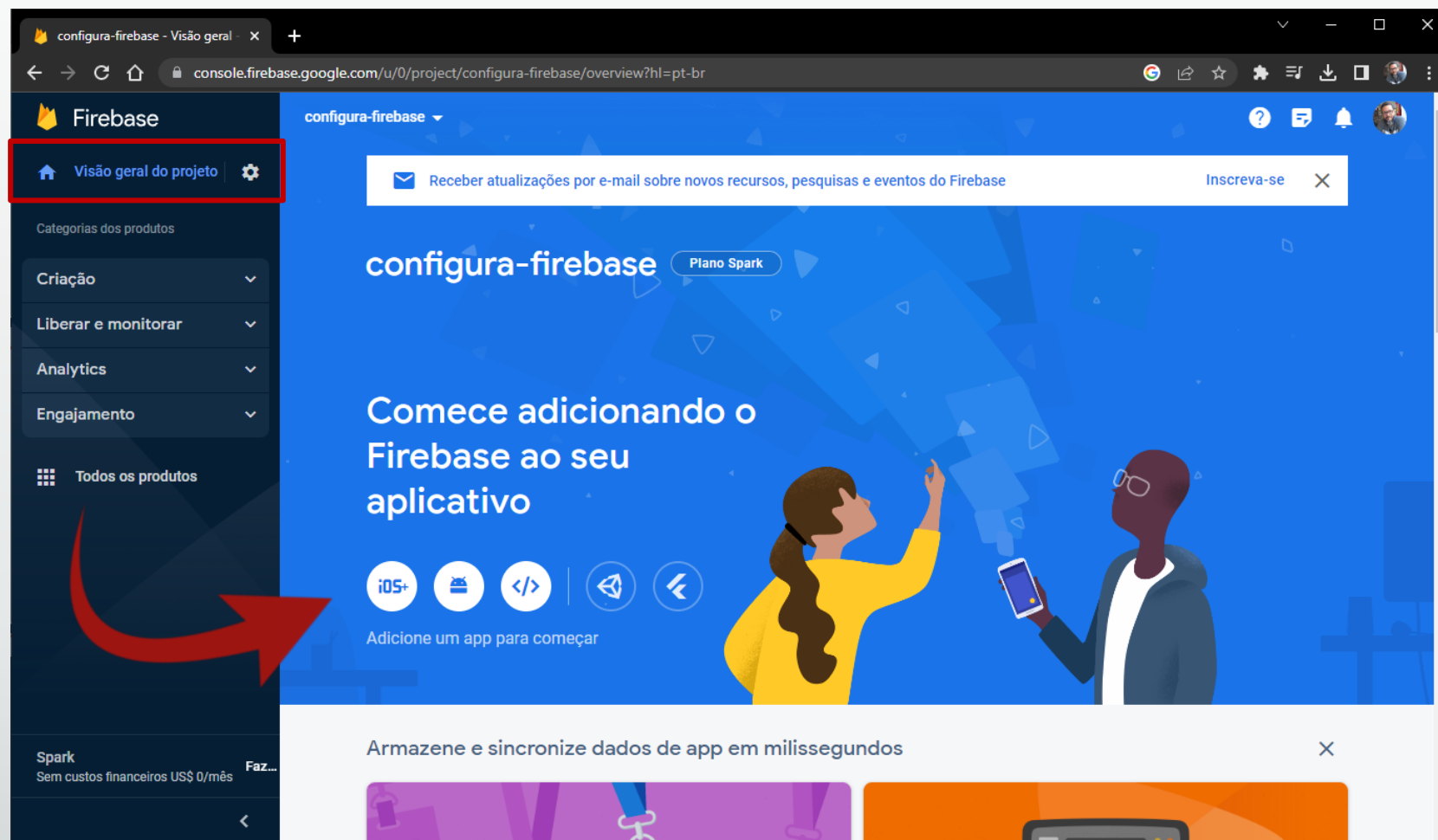
# Comunicação e Bibliotecas

# Registrando o APP no Android Studio



# Registrando o APP no Android Studio

Na visão geral do Firebase, identifique o ícone do Android Studio para iniciar a configuração



## Registrando o APP no Android Studio

Clicando no ícone do Android, ele irá abrir a tela na qual podemos estabelecer e integrar as configurações do APP junto ao Firebase.

Nome do pacote Android, consiste no nome dado ao projeto, por exemplo, com.exemplo.teste.com

O pacote pode ser achado também no arquivo build.gradle (Module.App) ou ainda no AndroidManifest.xml

- × Adicionar o Firebase ao seu app Android

1 Registrar app

Nome do pacote do Android 

```
com.example.firebase_app
```

## Pacote do App

Apelido do app (opcional) (?)

MyAppFirebase

Nome do APP

Certificado de assinatura de depuração SHA-1 (opcional) ②

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:!

**i** Necessário para o Dynamic Links, o Login do Google ou o suporte por telefone no Auth. Edite os certificados SHA-1 nas configurações.

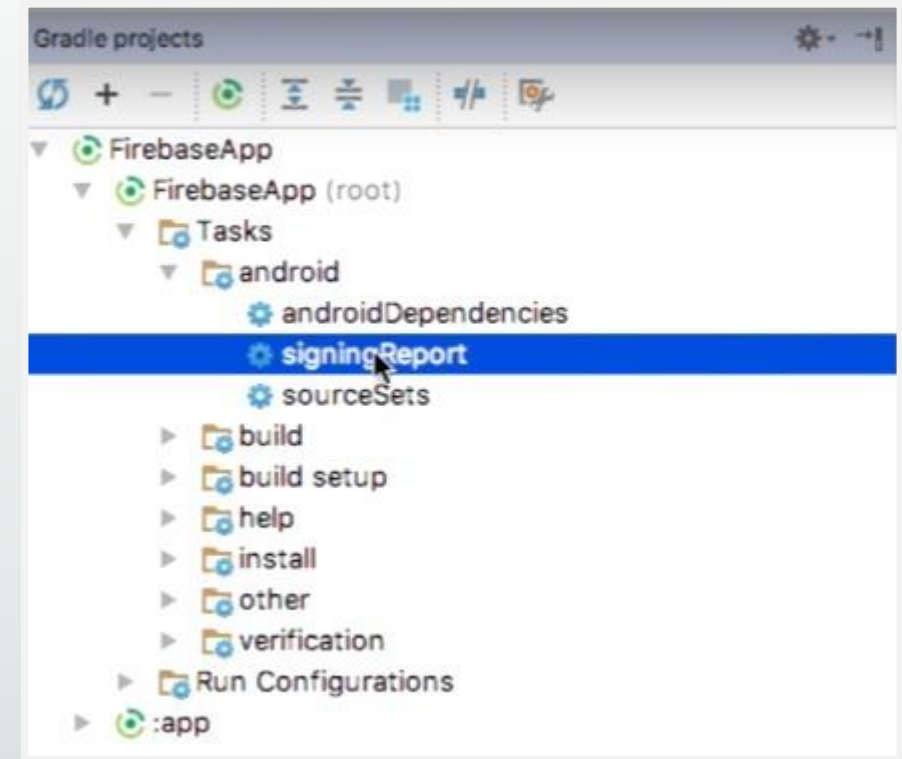
**Registrar app**

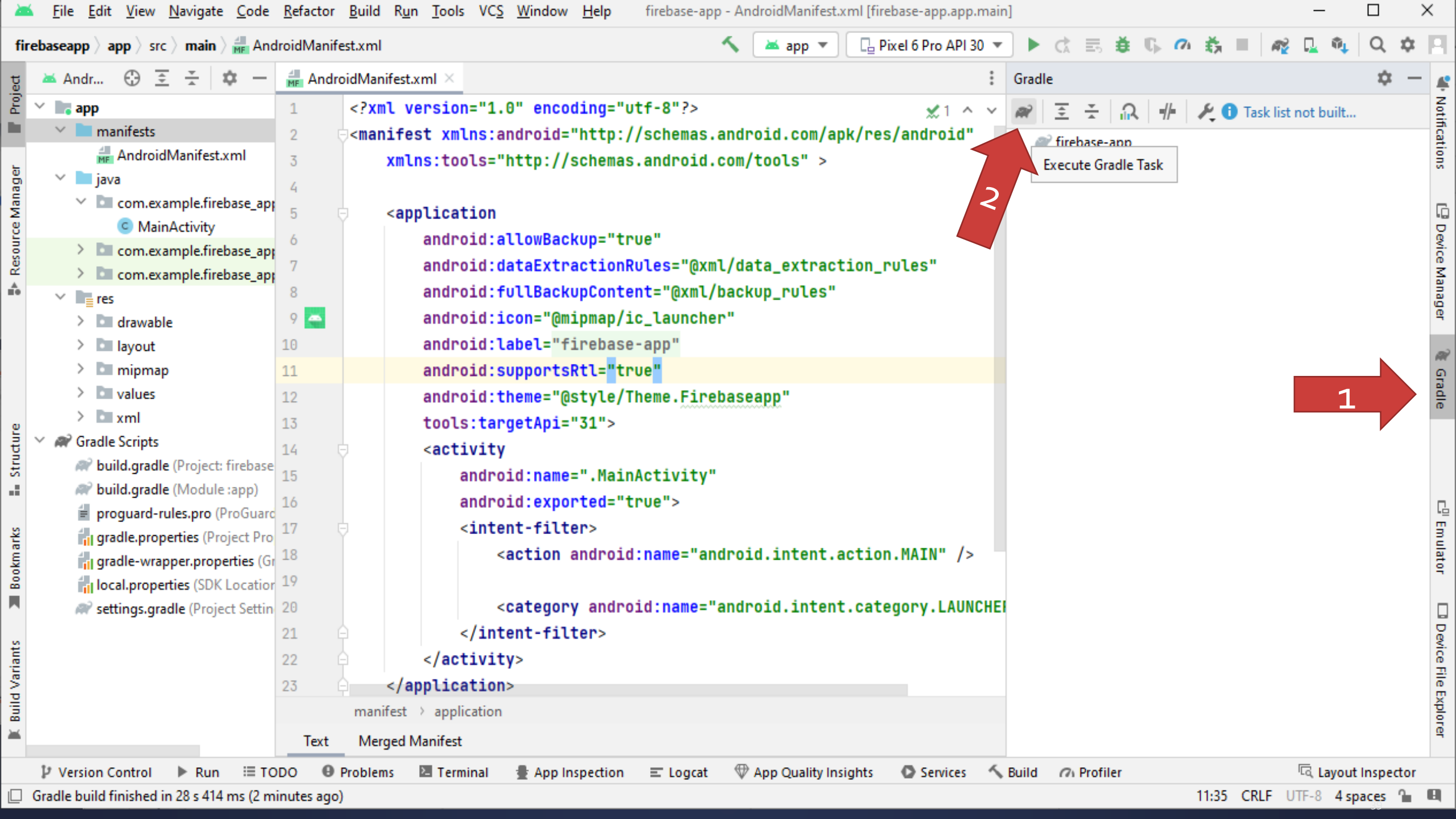
Assinatura de Depuração

2 Faça o download e adicione o arquivo de configuração

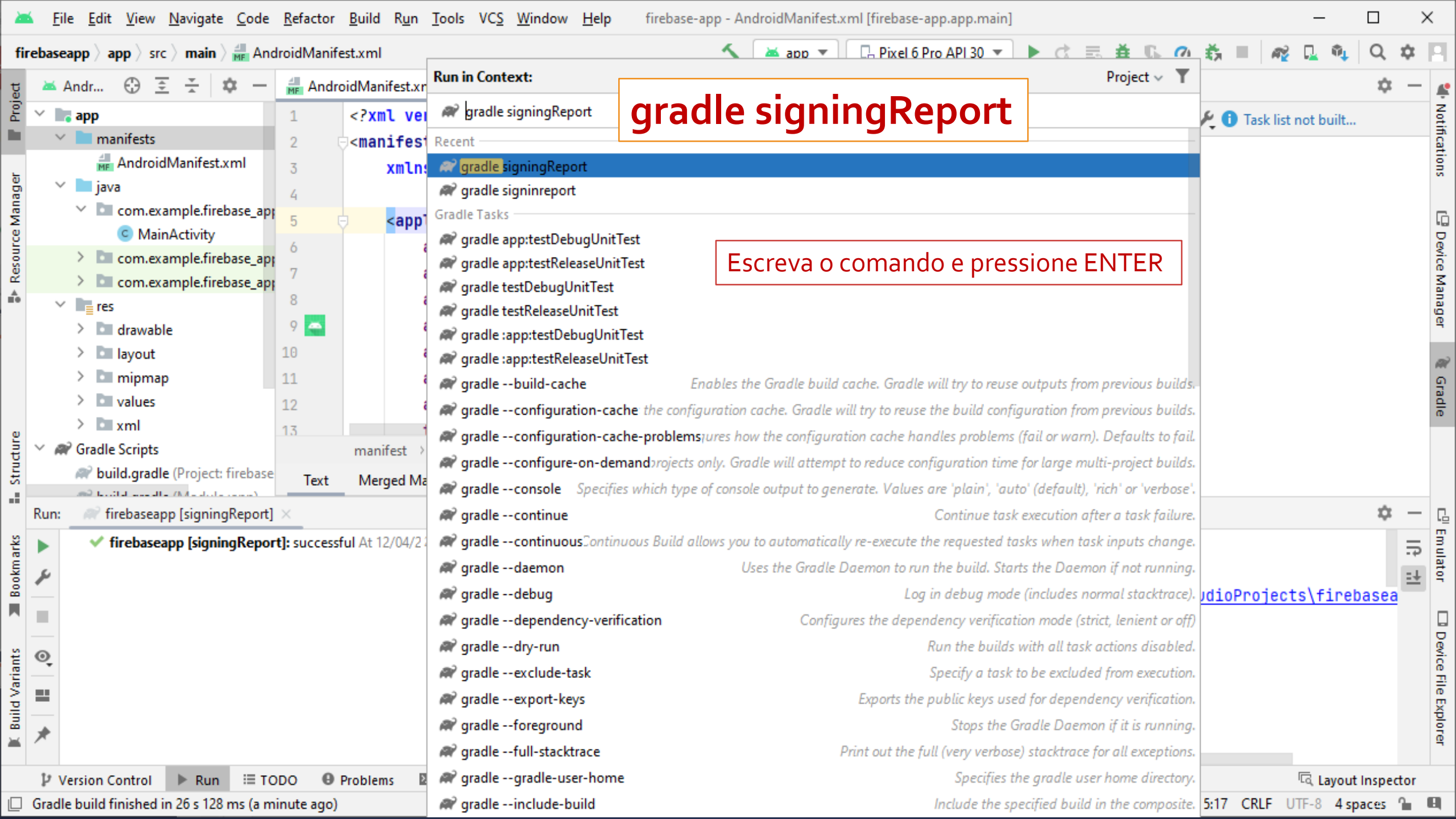
# Obtendo a Assinatura – Versões Anteriores

- Para fazer isso, utilizaremos o *Signing Report* do Gradle
- O processo consiste em:
  - Acessar a aba Gradle Projects
  - Localizar o arquivo **signingReport**
  - Duplo clique para executar
  - A assinatura será mostrada no console ao clicar no ícone “A/B”
  - Copie o SHA1 para o Firebase.





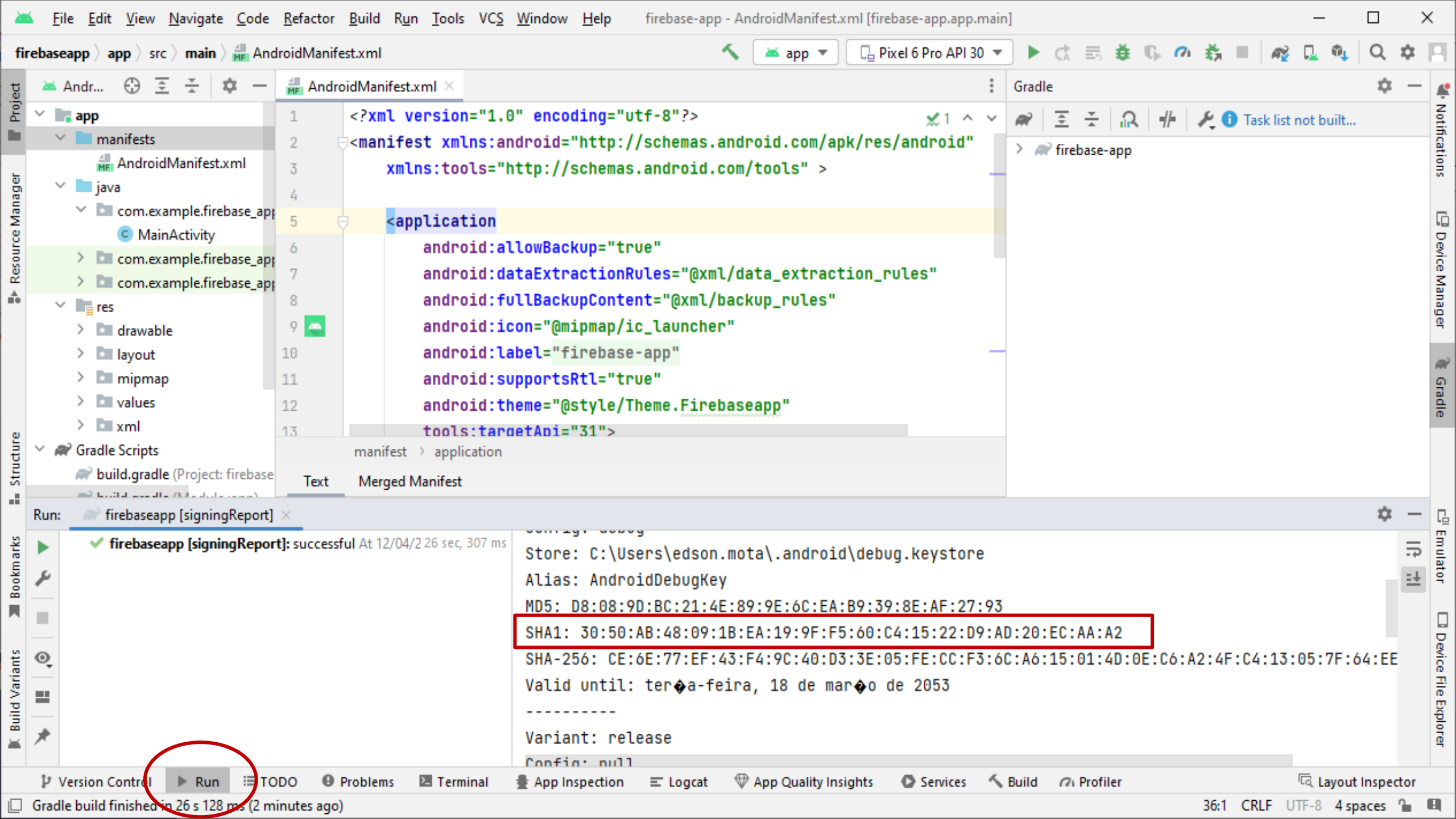




gradle signingReport

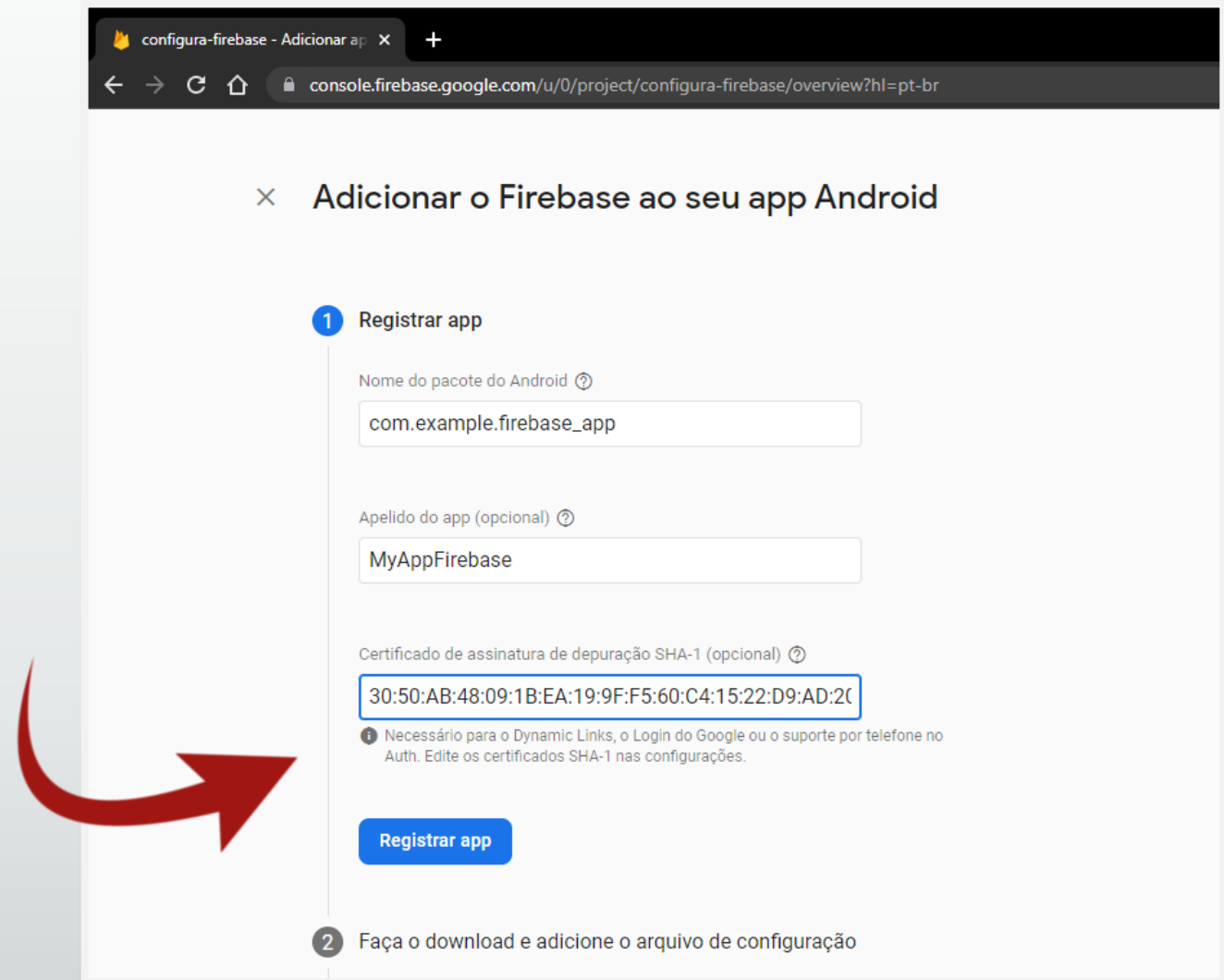
Escreva o comando e pressione ENTER





# Registrando o APP no Android Studio

- Agora, pode-se fazer o registro do APP



configura-firebase - Adicionar ap x +

← → ↻ 🏠 console.firebase.google.com/u/0/project/configura-firebase/overview?hl=pt-br

× Adicionar o Firebase ao seu app Android

1 Registrar app

Nome do pacote do Android ⓘ

com.example.firebase\_app

Apelido do app (opcional) ⓘ

MyAppFirebase

Certificado de assinatura de depuração SHA-1 (opcional) ⓘ

30:50:AB:48:09:1B:EA:19:9F:F5:60:C4:15:22:D9:AD:2C

ⓘ Necessário para o Dynamic Links, o Login do Google ou o suporte por telefone no Auth. Edite os certificados SHA-1 nas configurações.

Registrar app

2 Faça o download e adicione o arquivo de configuração

# Registrando o APP no Android Studio

- Nesse ponto, precisamos fazer o download de um arquivo JSON com as configurações de acesso ao Firebase.
- Esse arquivo deverá ser inserido na pasta app do seu Android Studio

## ✕ Adicionar o Firebase ao seu app Android

✓ Registrar app

Nome do pacote Android: com.example.firebase\_app, apelido do app: MyAppFirebase

2 Faça o download e adicione o arquivo de configuração para o Android Studio abaixo | [Unity](#) [C++](#)

↓ Fazer o download de google-services.json

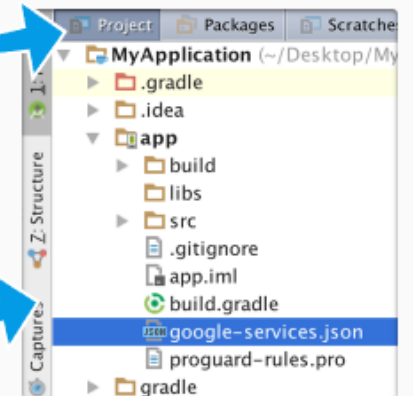
Mude para a visualização **Projeto** no Android Studio para ver o diretório raiz.

Mova o arquivo `google-services.json` salvo para o diretório raiz do módulo (nível do app).

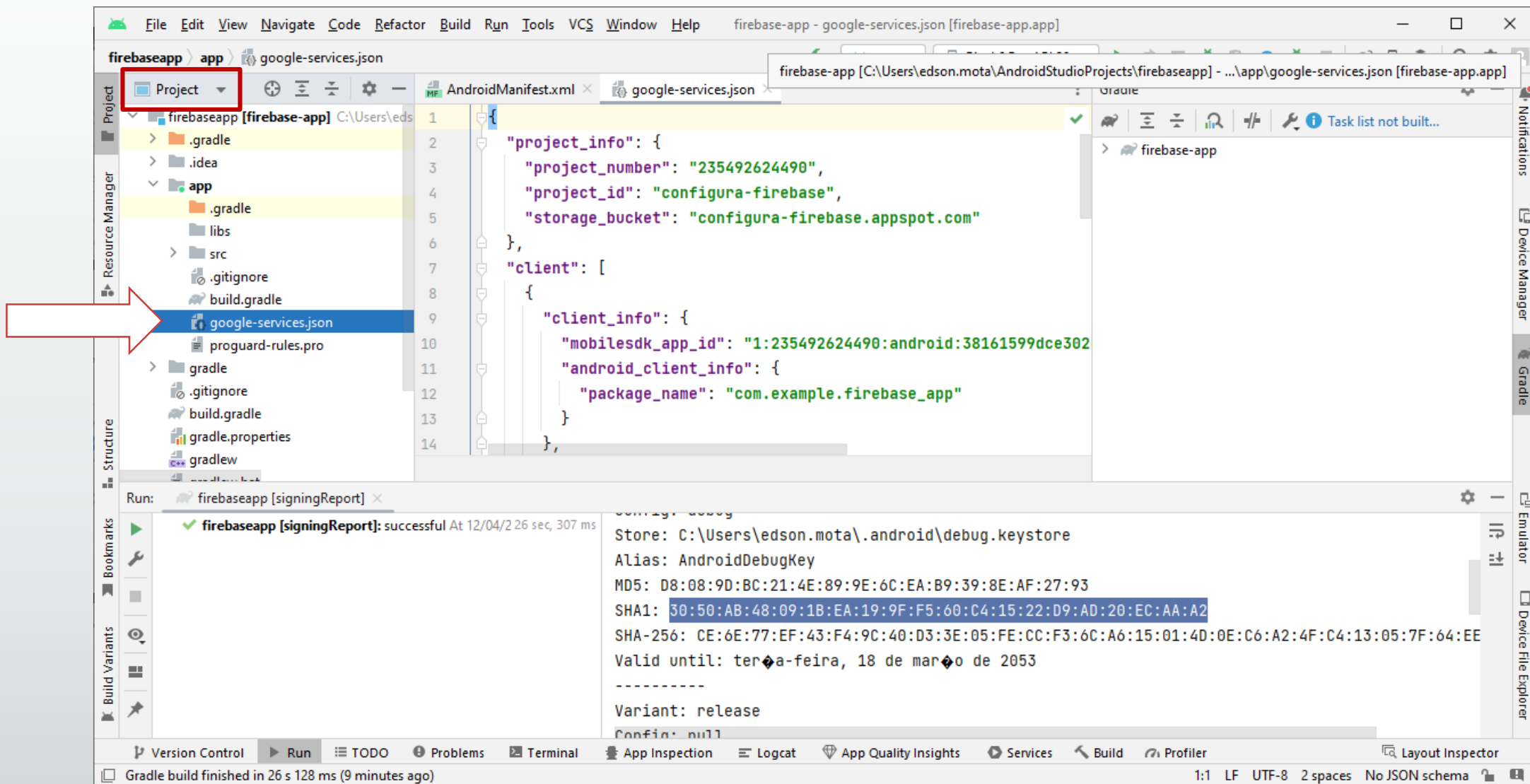


google-services.json

Próxima



# Registrando o APP no Android Studio



# Registrando o APP no Android Studio - Configuração

Depois de toda a configuração realizada, o Firebase irá mostrar um conjunto de configurações que deve constar nas bibliotecas do seu Android Studio, mais especificamente no Gradle

## × Adicionar o Firebase ao seu app Android



Registrar app

Nome do pacote Android: com.example.firebase\_app, apelido do app: MyAppFirebase



Faça o download e adicione o arquivo de configuração



Adicionar o SDK do Firebase

Instruções para Gradle | [Unity](#) [C++](#)

1. Para tornar os valores de configuração do arquivo `google-services.json` acessíveis aos SDKs do Firebase, você precisa do plug-in do Gradle para os Serviços do Google.

Adicione o plug-in como uma dependência do buildscript ao arquivo `build.gradle` no nível do projeto:

Arquivo do Gradle no nível raiz do projeto (`<project>/build.gradle`):

```
buildscript {
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }
    dependencies {
        ...
        // Add the dependency for the Google services Gradle plugin
        classpath 'com.google.gms:google-services:4.3.15'
    }
}
```



# Modificando o Build.Gradle\Project

## 3 Adicionar o SDK do Firebase

Instruções para Gradle | [Unity](#) [C++](#)

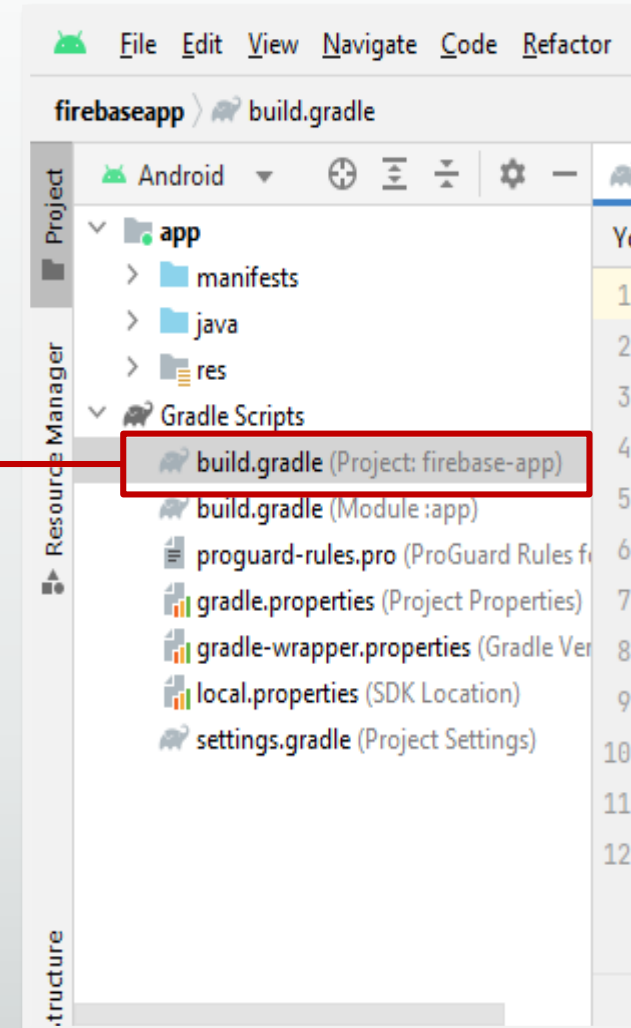
1. Para tornar os valores de configuração do arquivo `google-services.json` acessíveis aos SDKs do Firebase, você precisa do plug-in do Gradle para os Serviços do Google.

Adicione o plug-in como uma dependência do buildscript ao arquivo `build.gradle` no nível do projeto:

Arquivo do Gradle no nível raiz do projeto (`<project>/build.gradle`):

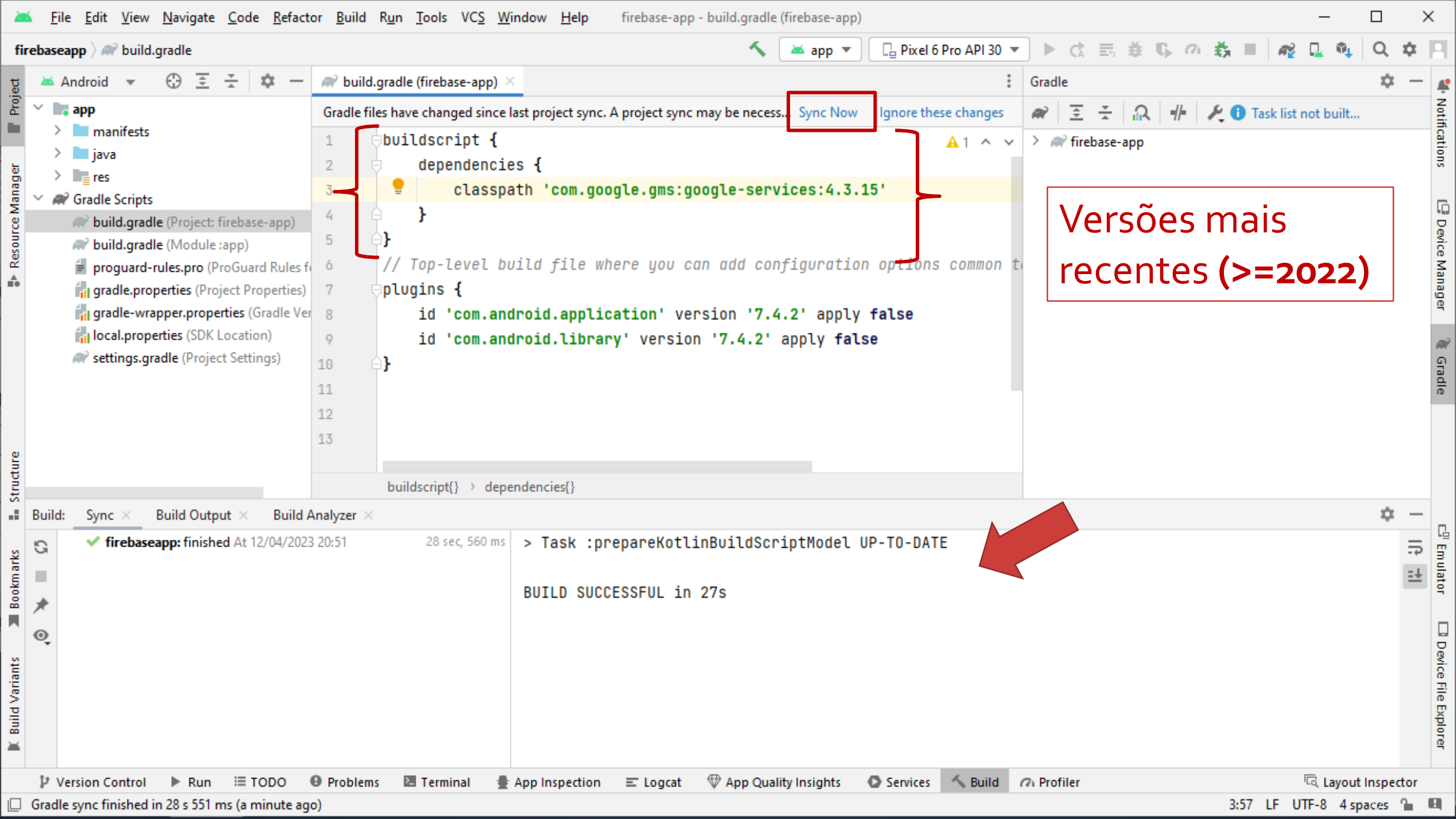
```
buildscript {
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }
    dependencies {
        ...
        // Add the dependency for the Google services Gradle plugin
        classpath 'com.google.gms:google-services:4.3.15'
    }
}

allprojects {
    ...
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }
}
```



Acesse o arquivo e verifique se as configurações são iguais as apresentadas no Firebase

**Caso não sejam,** você pode estar usando uma versão mais recente. Nesse caso, basta inserir o código completo





# Segunda Configuração

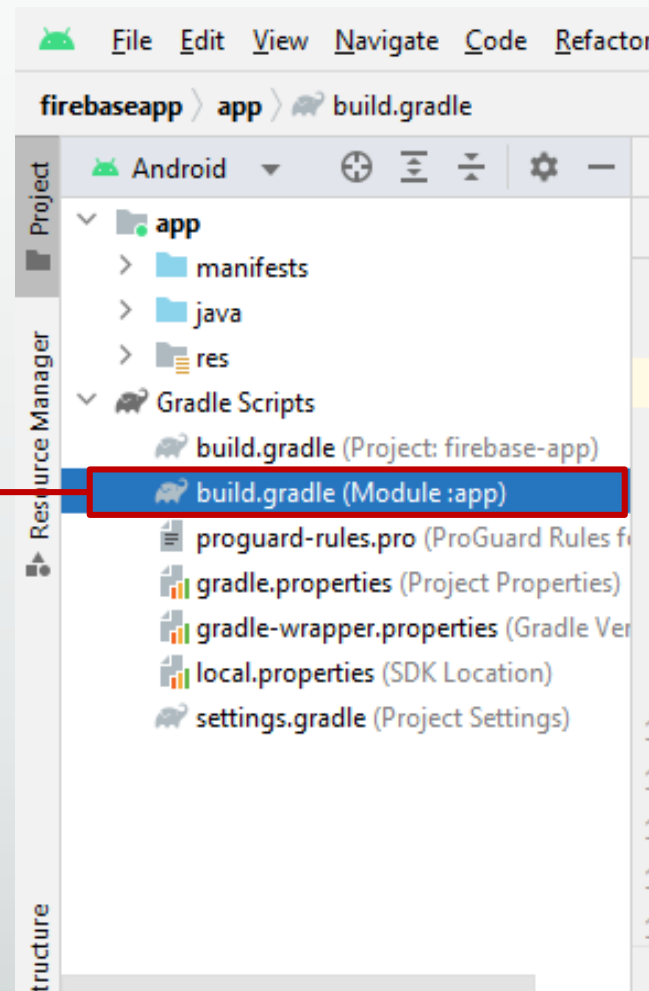
2. Em seguida, no arquivo `build.gradle` do módulo (nível do app), adicione o plug-in `google-services` e todos os SDKs do Firebase que você quer usar no app:

Arquivo do Gradle do módulo (nível do app) `<project>/<app-module>/build.gradle`:

```
plugins {  
    id 'com.android.application'  
    // Add the Google services Gradle plugin  
    id 'com.google.gms.google-services'  
    ...  
}  
  
dependencies {  
    // Import the Firebase BoM  
    implementation platform('com.google.firebase:firebase-bom:31.3.0')  
  
    // TODO: Add the dependencies for Firebase products you want to use  
    // When using the BoM, don't specify versions in Firebase dependencies  
    // https://firebase.google.com/docs/android/setup#available-libraries  
}
```

Ao usar a BoM do Firebase para Android, o app sempre vai utilizar versões da biblioteca do Firebase compatíveis.

[Saiba mais](#)



Agora, a configuração deverá ocorrer no arquivo *build.gradle* do módulo.

Identifique os pontos em que são solicitadas as alterações e realize as modificações



# Configuração Finalizada

× Adicionar o Firebase ao seu app Android

✓

 Registrar app  
Nome do pacote Android: com.example.firebase\_app, apelido do app: MyAppFirebase

✎

 Faça o download e adicione o arquivo de configuração

✎

 Adicionar o SDK do Firebase

4

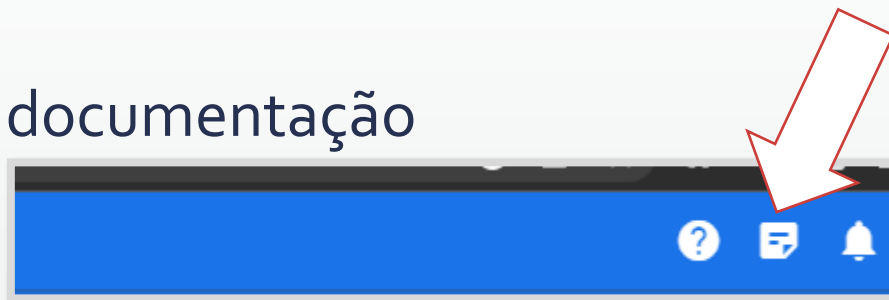
 Próximas etapas

Anterior

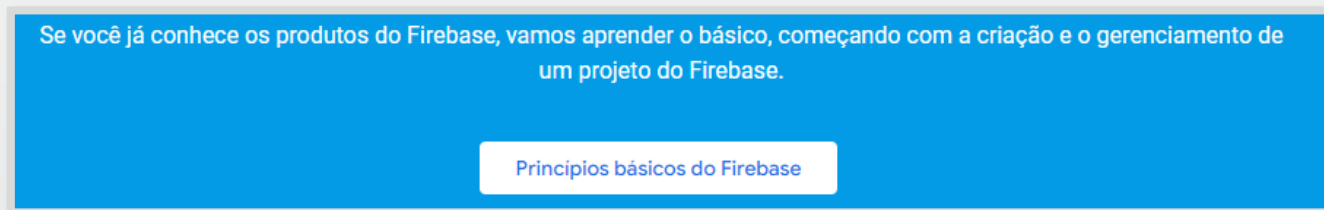
Continuar no console

# Registrando Biblioteca

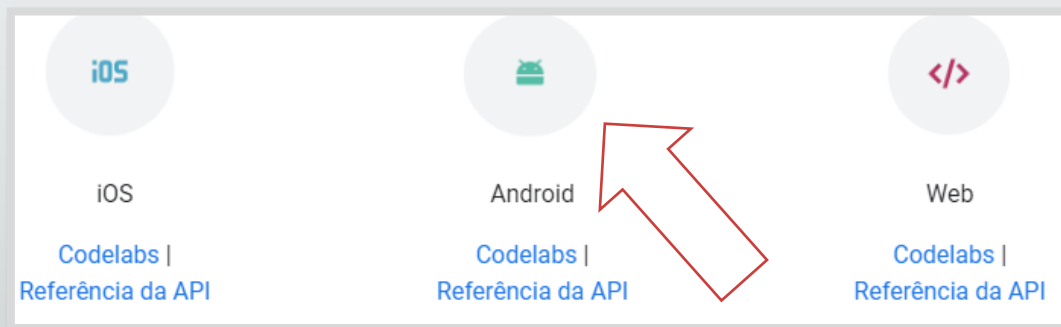
- Na tela do console, acesse a documentação



- Acesse o link princípios básicos do Firebase



- Configurações para integração entre o Firebase e o Android Studio



# Adicionando Bibliotecas de Serviço

- Identifique a seção bibliotecas disponíveis e vamos incluir uma nova biblioteca no projeto
- Essa seção fornece acesso ao conjunto de bibliotecas que são necessárias para utilização dos serviços Firebase.
- Pode-se utilizar recursos de autenticação, storage, notificações, entre outros.
- **Vamos utilizar o “Realtime Database”**

Realtime Database	com.google.firebase:firebase-database	20.1.0
-------------------	---------------------------------------	--------

# Adicionando Bibliotecas de Serviço

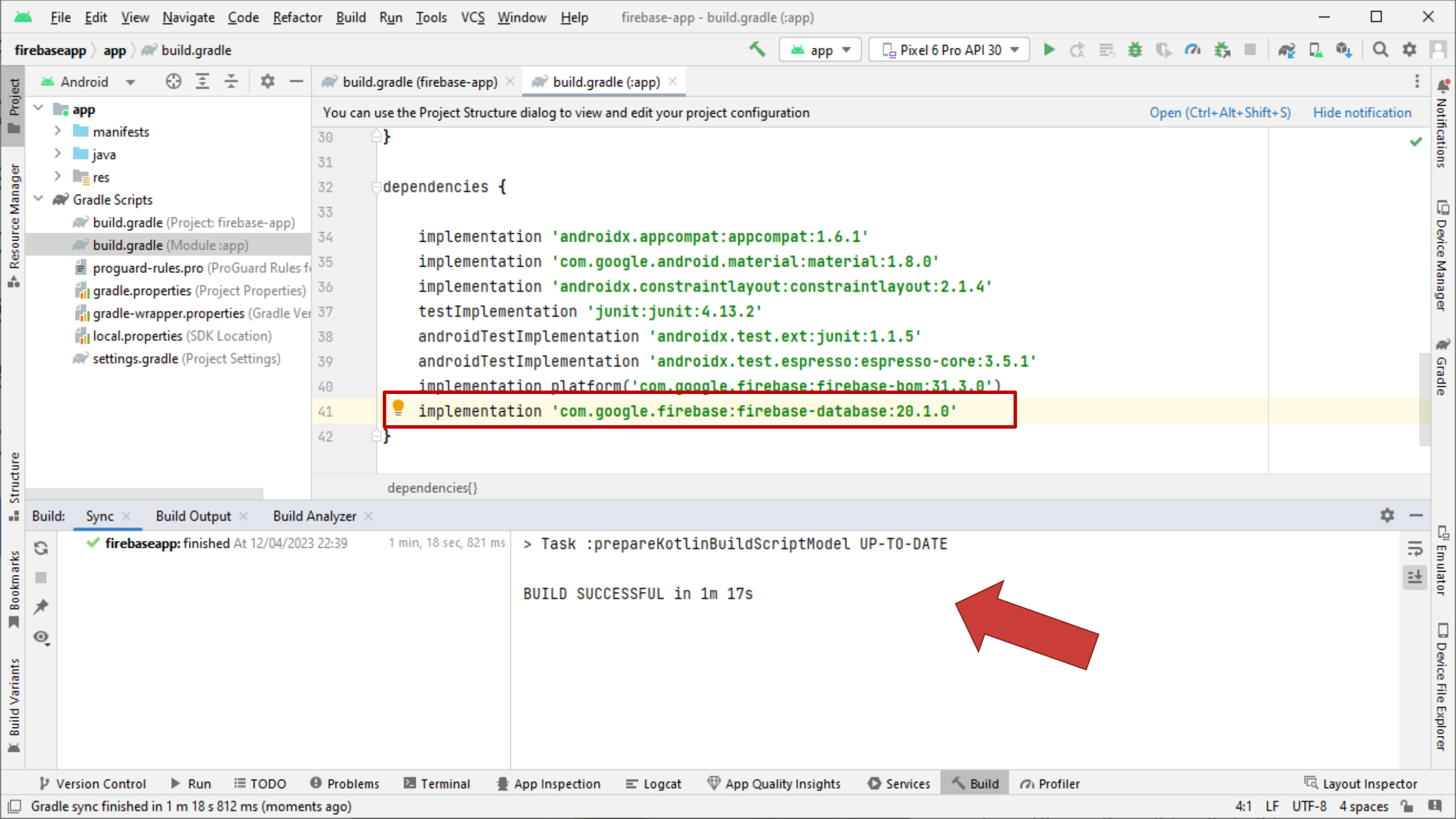
- O processo é simples, apenas copie a dependência

Realtime Database	com.google.firebase:firebase-database	20.1.0
-------------------	---------------------------------------	--------

- Volte no Build.Gradle <Module>, na seção dependências e cole a nova dependência

```
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.8.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
    implementation platform('com.google.firebase:firebase-bom:31.3.0')  
    implementation 'com.google.firebase:firebase-database:20.1.0'  
}
```

- Utilize o botão **Sync Now** para adicionar a nova dependência



# Acessando o Firebase

# Banco de Dados

- Estrutura do banco de dados para a implementação do nosso primeiro acesso a dados utilizando Firebase.



# Gravando Dados no Firebase

2 usages

```
public class MainActivity extends AppCompatActivity {
```

```
// Criando uma instancia do banco de dados utilizando o getReference() para identificar o nó raiz
```

1 usage

```
private DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
//Criando um novo nó no firebase
```

```
reference.child( pathString: "Permissoes").setValue("all");
```

```
}
```

```
}
```





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

[Configu](#)



<https://configura-firebase-default-rtdb.firebaseio.com>



<https://configura-firebase-default-rtdb.firebaseio.com/>

— Permissoes: "all"

▼ — Usuario

▼ — 001

— login: "teste"

— nome: "Jose Santos"

— senha: "teste"

▼ — 002

— login: "juli"

— nome: "Juliana Cruz"

— senha: 1234

Faz...

# DatabaseReference

- O objeto DatabaseReference tem a função de posicionar a leitura de dados em um ponto do documento a partir de suas chaves

```
private DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
```

O objeto reference tem  
acesso a todo o  
conjunto de dados

Esse comando obtém uma instancia do  
banco e em seguida referencia uma  
posição no documento, nesse caso, a  
referência é a do nó Raiz


# DatabaseReference

- Quando desejamos restringir os dados retornados a uma entrada, ou uma chave no documento, uma estratégia pode ser a seguinte;

```
private DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
```

- Obtendo um subconjunto de dados

```
DatabaseReference usuarios = reference.child(pathString: "Usuario");
```



Nesse caso, a referência é filtrada pela chave usuário, assim, mesmo que existam diversas outras chaves no documento, o resultado será o “ramo” usuário

# Adicionando dados

- O que ocorre se rodarmos estes conjuntos de comandos?

```
reference.child( pathString: "Usuario").child( pathString: "003").child( pathString: "nome").setValue("Antônio Costa");  
reference.child( pathString: "Usuario").child( pathString: "003").child( pathString: "login").setValue("antonio");  
reference.child( pathString: "Usuario").child( pathString: "003").child( pathString: "senha").setValue("12345");
```

▼ — 003

- login: "antonio"
- nome: "Antônio Costa"
- senha: "12345"



# Editando Dados

- Se a chave não existir será uma inclusão, caso contrário, será uma alteração

```
reference.child(pathString: "Usuario").child(pathString: "003").child(pathString: "nome").setValue("Antônio Costa de Souza");  
reference.child(pathString: "Usuario").child(pathString: "003").child(pathString: "login").setValue("Tony");  
reference.child(pathString: "Usuario").child(pathString: "003").child(pathString: "senha").setValue("54321");
```

▼ 003 + 🗑

- login: "Tony"
- nome: "Antônio Costa de Souza"
- senha: "54321"



# Podemos Passar **Objetos**?



# Gravando Objetos

```
public class Usuario {  
    2 usages  
    private String nome;  
    2 usages  
    private String login;  
    2 usages  
    private String senha;  
  
    1 usage  
    public Usuario() {  
    }  
  
    public String getName() { return nome; }  
    1 usage  
    public void setName(String nome) { this.nome = nome; }  
  
    public String getLogin() { return login; }  
    1 usage  
    public void setLogin(String login) { this.login = login; }  
  
    public String getSenha() { return senha; }  
    1 usage  
    public void setSenha(String senha) { this.senha = senha; }  
}
```

```
// Fazendo a referência a um ponto do documento
```

```
DatabaseReference usuarios = reference.child( pathString: "Usuario");
```

```
// Trabalhando com objetos
```

```
Usuario usu = new Usuario();
```

```
usu.setName("Marcelo Cardoso");
```

```
usu.setLogin("marcelo");
```

```
usu.setSenha("teste");
```

```
usuarios.child( pathString: "004").setValue(usu);
```

Utilizando como referência a entrada usuário, basta passar o objeto no método setValue que a estrutura do objeto será persistida;

▼ — Usuario

▼ — 001

login: "teste"

nome: "Jose Santos"

senha: "teste"

▼ — 002

login: "juli"

nome: "Juliana Cruz"

senha: 1234

▼ — 003

login: "Tony"

nome: "Antônio Costa de Souza"

senha: "54321"

▼ — 004

login: "marcelo"

nome: "Marcelo Cardoso"

senha: "teste"



## Recuperando dados

- Para recuperar dados do Firebase, precisamos utilizar um *listener* que irá “ouvir” as mudanças e manter seus dados sincronizados.
- O listener funciona como um sistema de notificações que informa em um método a ocorrência de uma eventual mudança

# Anatomia de um Listener

```
DatabaseReference usuarios = reference.child( pathString: "Usuario");

usuarios.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {

    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Esse método será notificado quando qualquer mudança ocorrer no nó usuários

Implementa um tratamento de erros que é disparado quando a requisição for cancelada por algum motivo

# Testando o RealTime Database

- Para testar, insira o código abaixo no seu método onDataChange e modifique o seu banco de dados.

```
DatabaseReference usuarios = reference.child( pathString: "Usuario");
```

```
usuarios.addValueEventListener(new ValueEventListener() {
```

```
    2 usages
```

```
    @Override
```

```
    public void onDataChange(@NonNull DataSnapshot snapshot) {
```

```
        System.out.println(snapshot.getValue().toString());
```

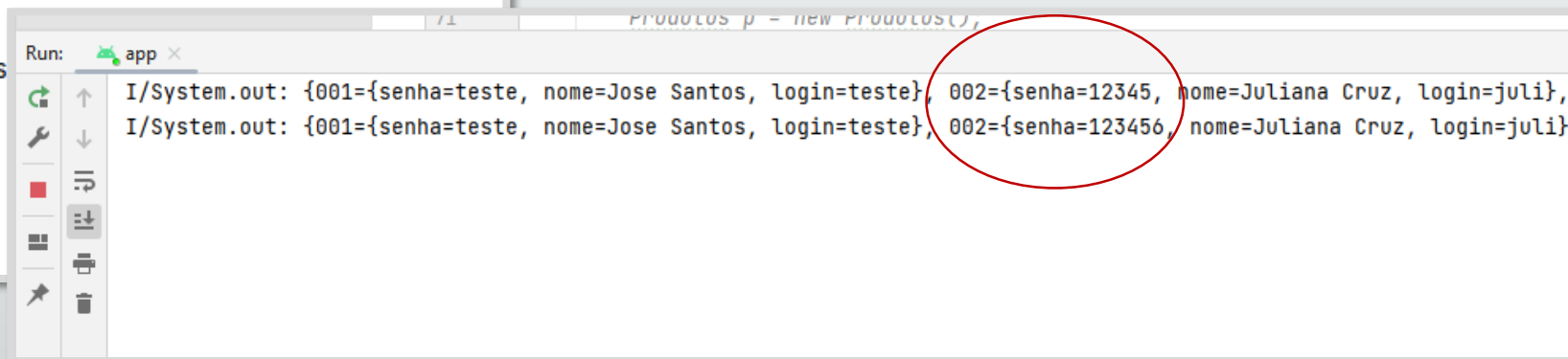
```
    }
```

```
    @Override
```

```
    public void onCancelled(@NonNull DatabaseError error) {
```

```
    }
```

```
});
```



```
Run: app x
I/System.out: {001={senha=teste, nome=Jose Santos, login=teste}, 002={senha=12345, nome=Juliana Cruz, login=juli},
I/System.out: {001={senha=teste, nome=Jose Santos, login=teste}, 002={senha=123456, nome=Juliana Cruz, login=juli}}
```

# Segmentando por Objetos

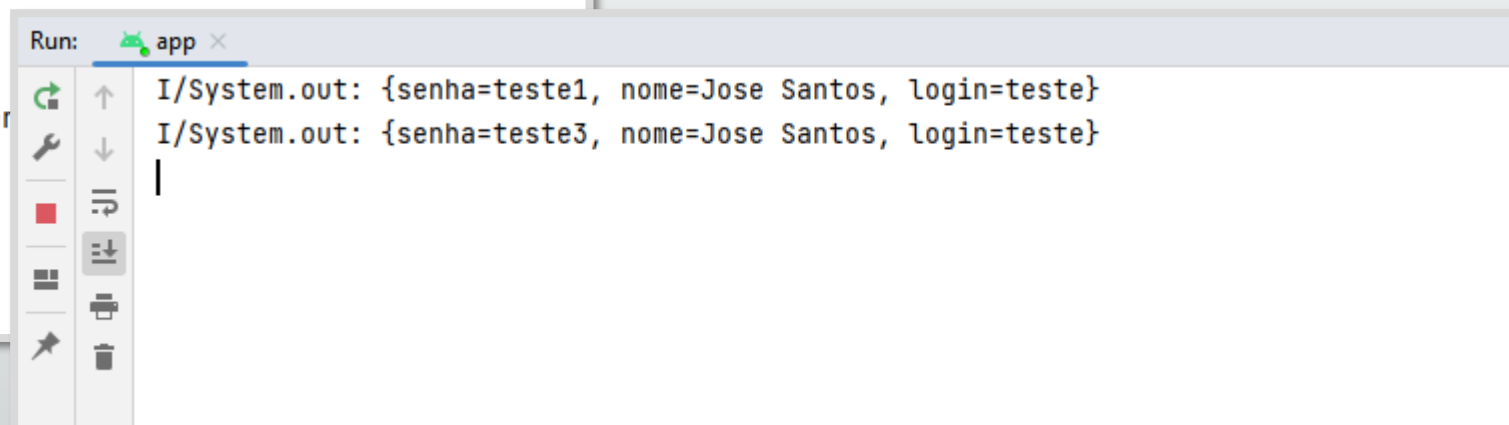
- Escutando apenas as mudanças ocorridas no registro de usuário "001"

```
DatabaseReference usuarios = reference.child( pathString: "Usuario").child( pathString: "001");

usuarios.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        System.out.println(snapshot.getValue().toString());
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {}
});
```

- Nesse caso mudanças em outros usuários não tem efeito para este listener.



```
Run: app x
I/System.out: {senha=teste1, nome=Jose Santos, login=teste}
I/System.out: {senha=teste3, nome=Jose Santos, login=teste}
```

# Criando uma Lista

- Uma opção para listar dados é incluir todos os elementos em uma lista tipada (ArrayList).
- Um objeto criando no contexto pode ser inserido na lista
- Os dados da lista serão atualizados a cada mudança na base de dados

```
DatabaseReference usuarios = reference.child( pathString: "Usuario");

listUsers = new ArrayList<>();

usuarios.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        listUsers.clear();

        for(DataSnapshot current_user: snapshot.getChildren()){
            Usuario u = new Usuario();
            u.setNome(current_user.child( path: "nome").getValue().toString());
            u.setLogin(current_user.child( path: "login").getValue().toString());
            u.setSenha(current_user.child( path: "senha").getValue().toString());
            listUsers.add(u);
        }
        updateList();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

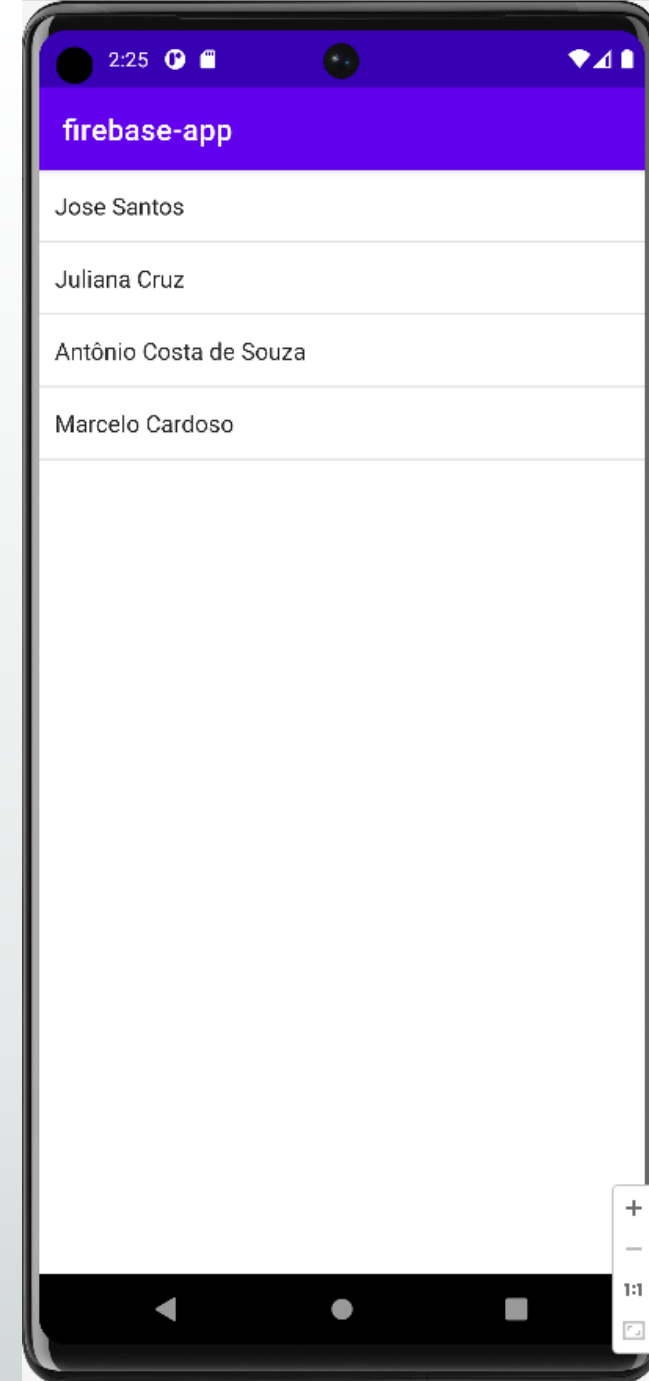
    }

});
```

# ListView RealTime

1 usage

```
public void updateList(){  
    ArrayAdapter<Usuario> adaptador = new ArrayAdapter<>(  
        getApplicationContext(),  
        android.R.layout.simple_list_item_1,  
        android.R.id.text1,  
        listUsers  
    );  
    ListViewUsuarios.setAdapter(adaptador);  
}
```



# Atividade

- Utilizando o mesmo projeto que já criamos no Firebase, elabore uma interface gráfica no Android capaz de permitir o cadastramento de livros.
- Este cadastro deve conter:
  - Título, Editora, Ano, Avaliações do usuário (1 a 5)
- Uma listagem deve mostrar todos os livros cadastrados e suas respectivas avaliações.

**Bons Estudos!**