

Firestore - kotlin



Prof. Me. Hélio Esperidião

O que é NoSQL?



NoSQL é um termo usado para uma estrutura de banco de dados diferente, na qual estamos acostumados a utilizar, os não relacionais



maior diferencial entre esse tipo de banco e os tradicionais são a velocidade e alta escalabilidade.



Hoje, temos 5 modelos diferentes, sendo eles:

Modelos



CHAVE-VALOR



DOCUMENTO

Chave-Valor

- Esse modelo é o mais simples entre os bancos de dados NoSQL. Sua estrutura é muito semelhante ao Map do Java.
 - Redis
 - DyanamoDB
 - Scalaris
 - Project Voldemort





Os objetos “Map” confiam seus dados em um algoritmo hash (hash code).



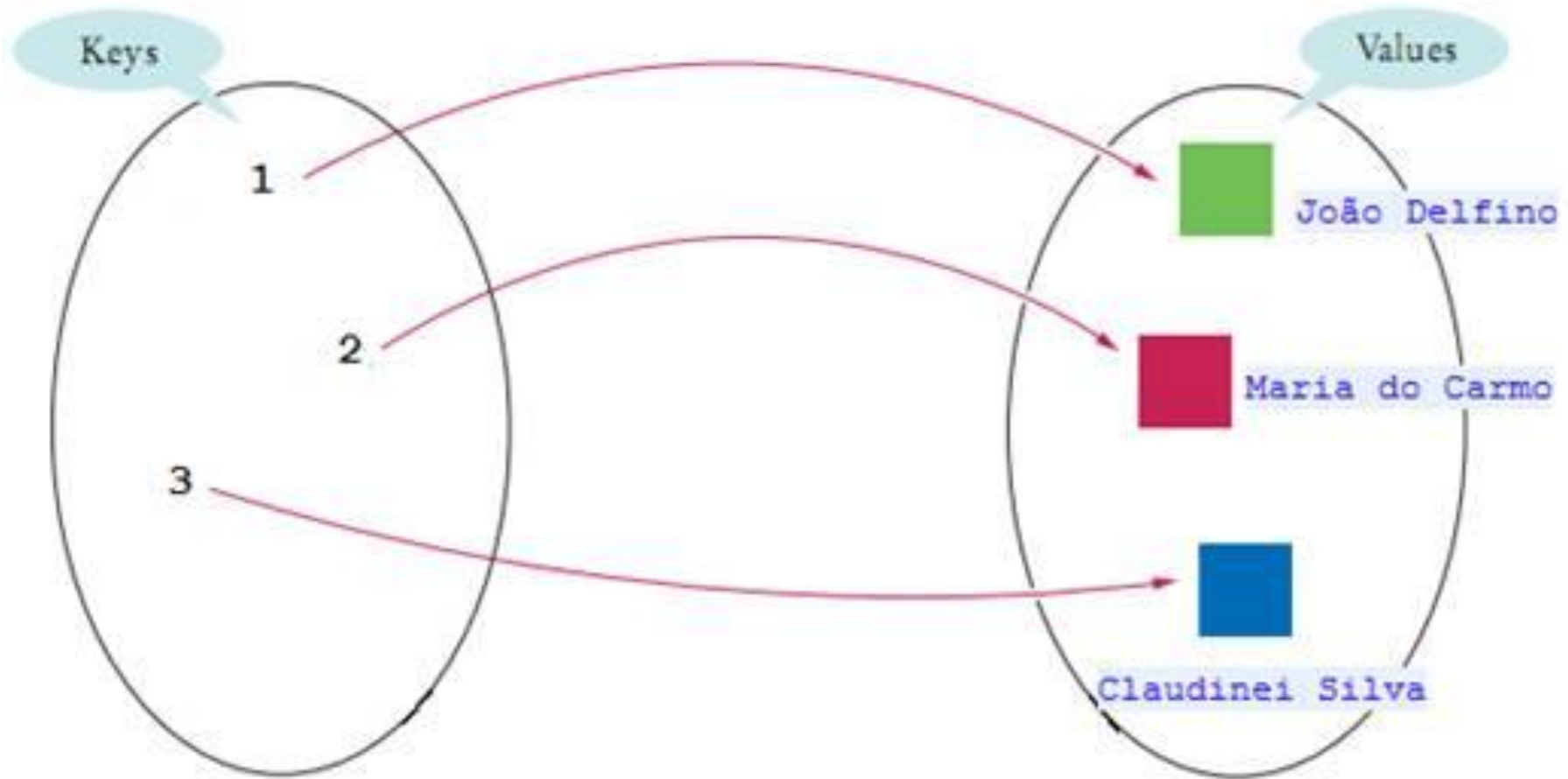
Esse algoritmo transforma uma grande quantidade de dados em uma pequena quantidade de informações, sendo que o mecanismo de busca se baseia na construção de índices.



Um exemplo prático pode ser usado como uma lista telefônica onde a letra seria o índice a ser procurado, para conseguir achar mais fácil o nome desejado.

Mapa java

mapa



Exemplo

```
public static void main(String[] args) {  
    Map map = new HashMap();  
    //Adding elements to map  
    map.put(1, "ana");  
    map.put(5, "Maria");  
    map.put(2, "Paula");  
    map.put(6, "Patricia");  
    //Converting to Set so that we can traverse  
    Set set = map.entrySet();  
    Iterator itr = set.iterator();  
    //Converting to Map.Entry so that we can get key and value separately  
    while (itr.hasNext()) {  
        Map.Entry entry = (Map.Entry) itr.next();  
        System.out.println(entry.getKey() + " " + entry.getValue());  
    }  
}
```



Talvez seja o mais conhecido por conta da popularidade do MongoDB e, como consequência, acaba sendo porta de entrada para quem quer fugir dos modelos tradicionais.



Seu diferencial é a possibilidade de armazenar dados semi-estruturados, ou seja, você não precisa ter um schema pré-definido

Documento



Tem como característica conter todas as informações importantes em um único documento.



É livre de esquemas, possuir identificadores únicos universais (UUID).



Possibilitar a consulta de documentos através de métodos avançados de agrupamento e filtragem (MapReduce).

Banco de Dados Orientado a Documentos



Também são chamados de Bancos NoSQL (Not Only SQL).



NoSQL é devido à ausência do SQL.



Alguns chegaram a defender o termo NoREL (Not Relational), mas diferente do anterior este não foi muito aceito.



De forma resumida esse tipo de Banco de Dados não traz consigo as ideias do modelo relacional e nem a linguagem SQL.

NoSQL



É uma formatação leve de troca de dados.



Para seres humanos, é fácil de ler e escrever.



Para máquinas, é fácil de interpretar e gerar.



É baseado em um subconjunto da linguagem de programação JavaScript, Standard ECMA-262 3a Edição -Dezembro - 1999.

JSON - JavaScript Object Notation

JSON

JSON é em formato texto e completamente independente de linguagem

Formato ideal de troca de dados

é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida (parsing) entre sistemas

ESTRUTURA

Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *object*, record, struct, dicionário, hash table, keyed list, ou arrays associativas.

Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *array*, vetor, lista ou sequência.

Exemplo de json MongoDB



```
{ "_id" : { "$oid" : "5dc1d7b8c971ab0a28326098" }, "cpf" : "33333333390",  
"rg" : "405558592", "nome" : "Hélio", "Curso" : "Administração" }
```



```
{ "_id" : { "$oid" : "5dc1e7124f01312cd4b4e166" }, "cpf" : "1234567891111",  
"rg" : "456456", "nome" : "Andreia", "Curso" : "Engenharia" }
```

EXEMPLO - JavaScript

```
var myObj = { "name": "John", "age": 31, "city": "New York" };  
var myJSON = JSON.stringify(myObj);  
window.location = "demo_json.php?x=" + myJSON;
```

EXEMPLO - JavaScript

```
var myJSON = '{ "name": "John", "age": 31, "city": "New York" }';  
var myObj = JSON.parse(myJSON);  
document.getElementById("demo").innerHTML = myObj.name;
```


Exemplo Array

```
<script>
```

```
var myObj, x,y;
```

```
myObj = {
```

```
    "name": "John",
```

```
    "age": 30,
```

```
    "cars": [ "Ford", "BMW", "Fiat" ]
```

```
};
```

```
x = myObj.name;
```

```
y=myObj.cars[0];
```

```
document.getElementById("demo").innerHTML = x + " - " +y;
```

```
</script>
```

```
<script>

var myObj, i, j, x = "";
myObj = {
  "name": "John",
  "age": 30,
  "cars": [
    { "name": "Ford", "models": [ "Fiesta", "Focus", "Mustang" ] },
    { "name": "BMW", "models": [ "320", "X3", "X5" ] },
    { "name": "Fiat", "models": [ "500", "Panda" ] }
  ]
}

for (i in myObj.cars) {
  x += "<h2>" + myObj.cars[i].name + "</h2>";
  for (j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j] + "<br>";
  }
}

document.getElementById("demo").innerHTML = x;

</script>
```

Varrendo
todos as
posições

Exemplo php

```
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```

Retorna um object ou um array associativo se o parâmetro opcional assoc é TRUE.

```
<?php
$myArr = array("John", "Mary", "Peter", "Sally");

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

O Google tem o compromisso de promover a igualdade racial para as

Firebase helps you build and run successful apps

Backed by Google and loved by app development
teams - from startups to global enterprises

[Primeiros passos](#)[Try demo](#)[Watch video](#)

Firebase

Seus projetos do Firebase



Adicionar projeto


Criar um
novo
projeto

× Criar um projeto(Passo 1 de 3)

Vamos começar com um
nome para o projeto[?]

Nome do projeto

pdmFirebase

 pdmfirebase-3c53c

Continuar

Nome do
projeto

× Criar um projeto(Passo 2 de 2)

Google Analytics para seu projeto do Firebase

O Google Analytics é uma solução de análise gratuita e ilimitada. Com ele, é possível segmentar, gerar relatórios e muito mais nos seguintes produtos: Firebase Crashlytics, Cloud Messaging, Mensagens no app, Configuração remota, Teste A/B, Previsões e Cloud Functions.

O Google Analytics ativa:

- × Teste A/B ⓘ
- × Segmentação de usuários em produtos do Firebase ⓘ
- × Previsão do comportamento de usuários ⓘ
- × Usuários sem falhas ⓘ
- × Gatilhos do Cloud Functions com base em eventos ⓘ
- × Geração de relatórios ilimitada gratuita ⓘ

Ativar o Google Analytics neste projeto
Recomendado

[Anterior](#)

[Criar projeto](#)

Desative google Analytics

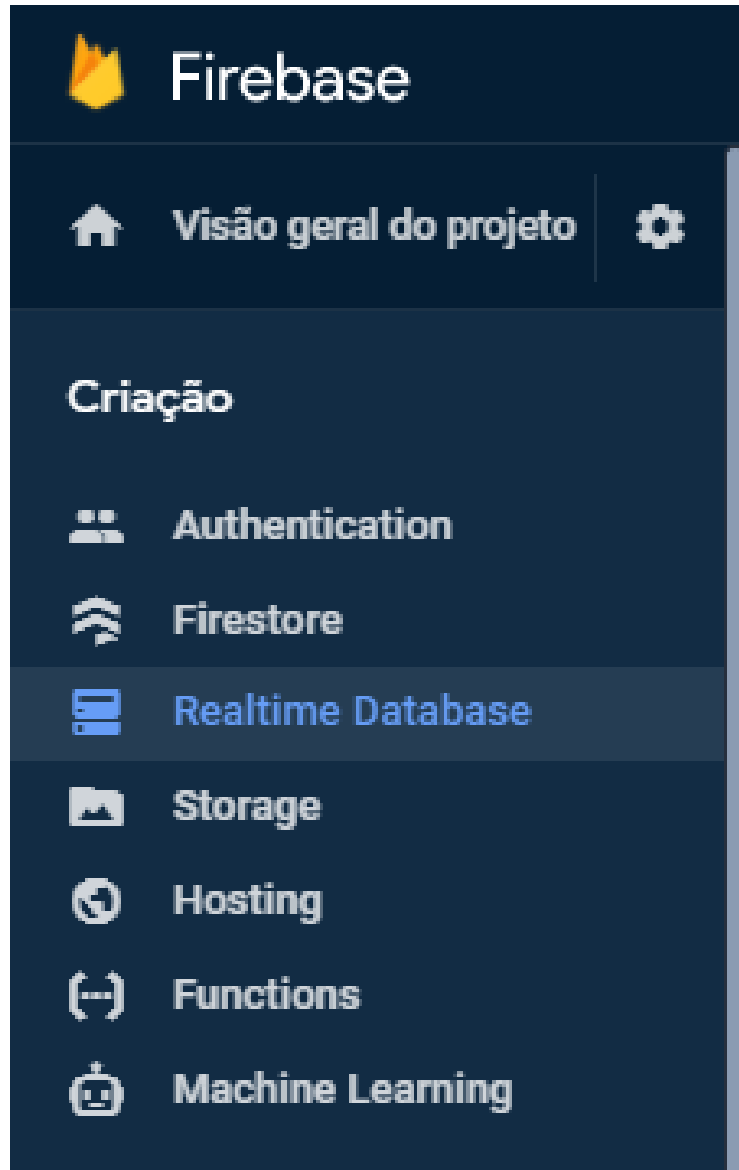


pdmFirebase

✓ Seu novo projeto está pronto

[Continuar](#)

Projeto Criado



RealTime Database

Realtime Database

Armazenamento e sincronização de dados em tempo real

Criar banco de dados

⚡ O Realtime Database é a escolha certa para você? [Comparar bancos de dados](#)

Criar um
realtime
Database

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 **Avançar**

Selecione o
servidor

Configurar banco de dados



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

Após definir a estrutura de dados, será necessário criar regras para proteger seus dados.

[Saiba mais](#)

- Iniciar no modo bloqueado**
Seus dados são privados por padrão. O acesso de leitura/gravação do cliente será concedido apenas se especificado por suas regras de segurança.
- Iniciar no modo de teste**
Por padrão, seus dados estão definidos para permitir uma configuração rápida. Porém, você precisa atualizar suas regras de segurança em até 30 dias para permitir em longo prazo o acesso de leitura/gravação do cliente.

```
{  
  "rules": {  
    ".read": "now < 1619665200000", // 2021-4-29  
    ".write": "now < 1619665200000", // 2021-4-29  
  }  
}
```

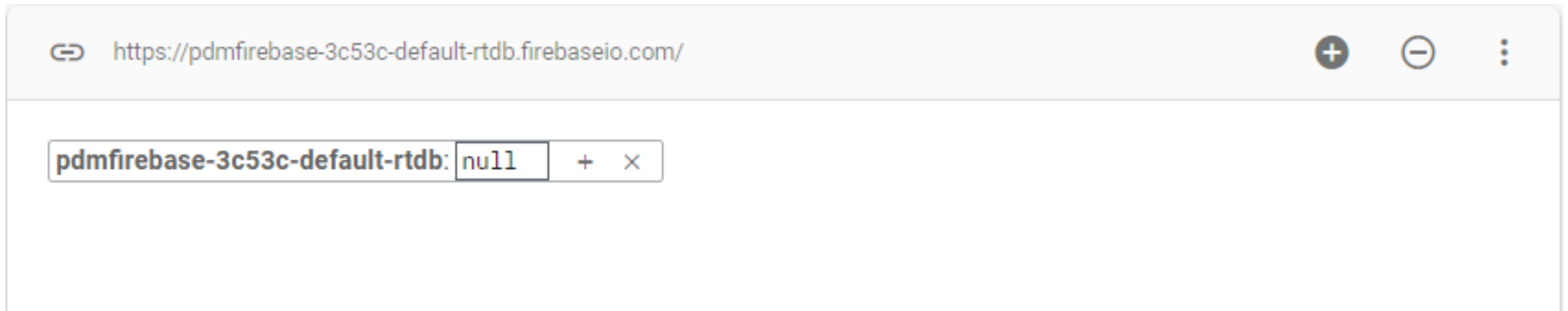
! As regras de segurança padrão para o modo de teste permitem que qualquer pessoa com a referência do seu banco de dados acesse, edite e exclua todos os dados nele por 30 dias

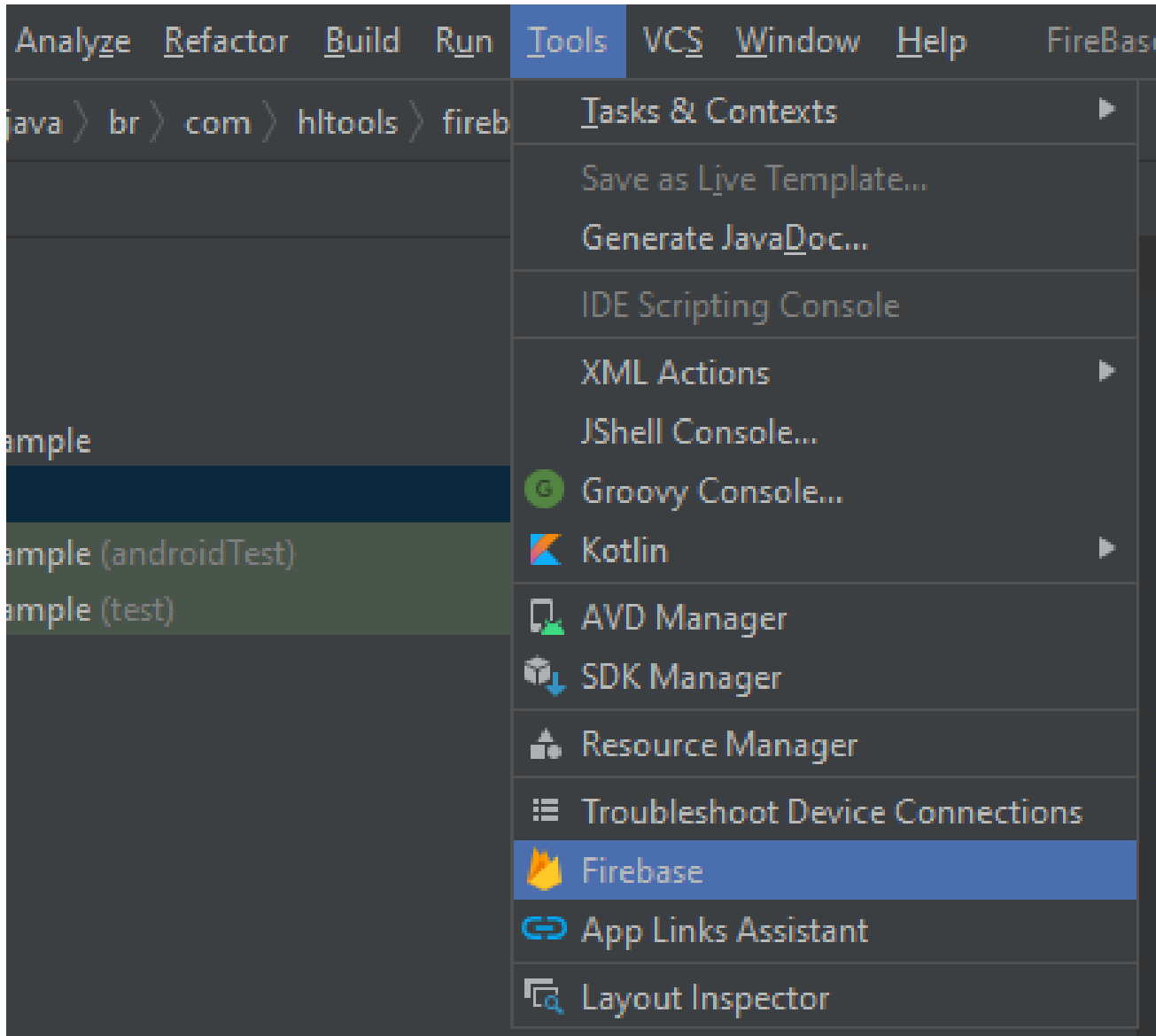
Cancelar

Ativar

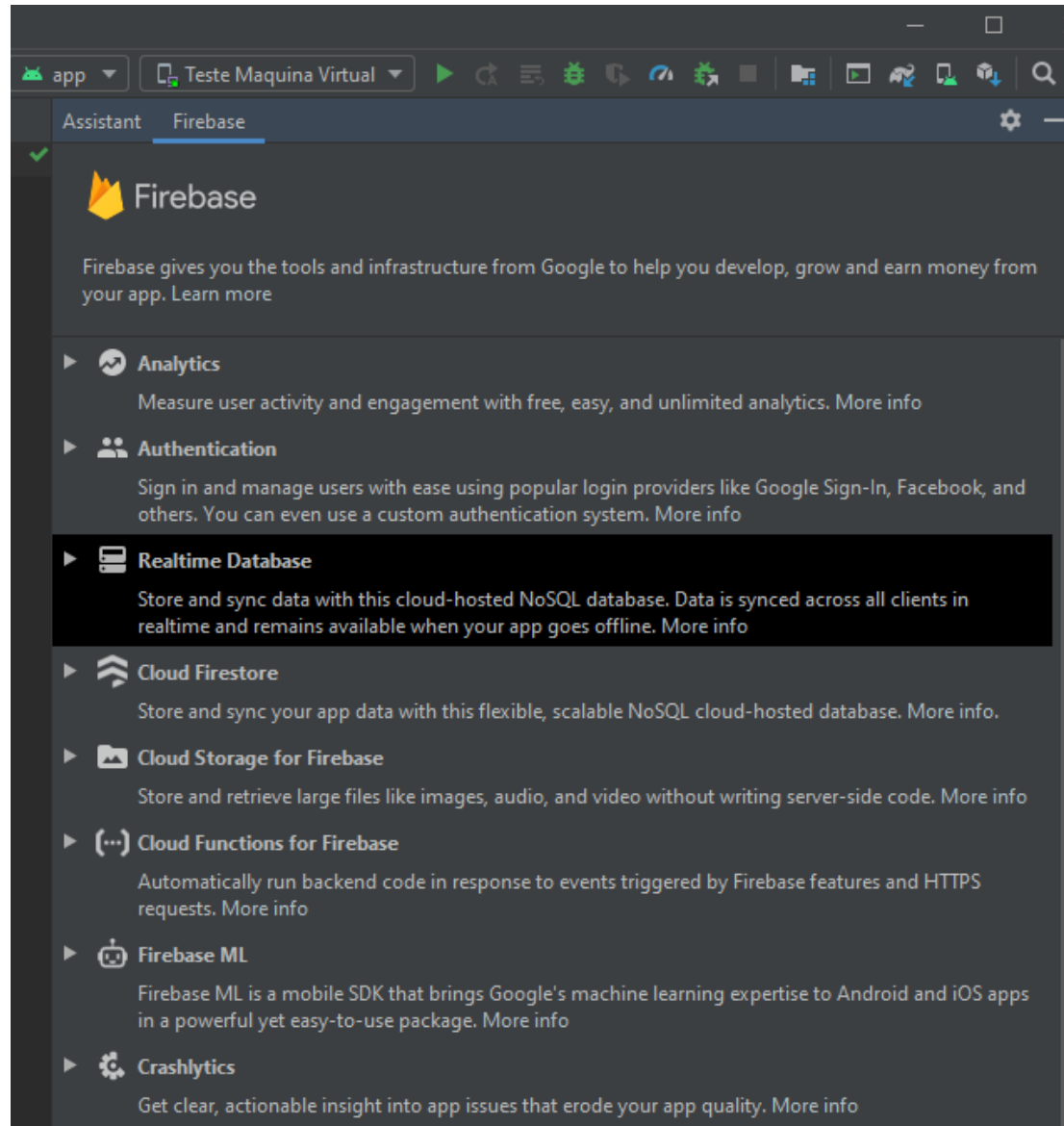
Regras de Segurança

Banco de dados criado





Configurações do android Studio



Selecione Realtime dataBase

Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.

Launch in browser

1 Connect your app to Firebase

Connect to Firebase

2 Add the Realtime Database to your app

Add the Realtime Database SDK to your app

NOTE: After adding the SDK, here are some other helpful configurations to consider:

- **Are you using Kotlin?**
You can use the Firebase KTX libraries to write idiomatic Kotlin. Change the `firebase-database` library dependency line in `build.gradle` to `firebase-database-ktx`.
- **Do you want an easier way to manage library versions?**
You can use the Firebase Android BoM to manage your Firebase library versions and ensure that your app is always using compatible library versions.

To use the Realtime Database, you need to create the database in the Firebase console.

3 Configure Realtime Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to.

By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication, you can configure your rules for public access. This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

4 Write to your database

Retrieve an instance of your database using `getInstance()` and reference the location you want to write to.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Selecione Connect to Firebase

- Faça login no google

Comece adicionando o
Firebase ao seu



Faça login
e escolha o
projeto



Seu app Android foi criado no
Firebase.

Ele está pronto para ser conectado ao projeto do Android Studio.

Conectar

Click em
conectar

Assistant | Firebase

← Firebase > Realtime Database

Get started with Realtime Database


The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.

Launch in browser

- 1 Connect your app to Firebase
 - ✓ Connected
- 2 Add the Realtime Database to your app
 - Add the Realtime Database SDK to your app
 - NOTE:** After adding the SDK, here are some other helpful configurations to consider:
 - **Are you using Kotlin?**
You can use the Firebase KTX libraries to write idiomatic Kotlin. Change the `firebase-database` library dependency line in `build.gradle` to `firebase-database-ktx`.
 - **Do you want an easier way to manage library versions?**
You can use the Firebase Android BoM to manage your Firebase library versions and ensure that your app is always using compatible library versions.
 - To use the Realtime Database, you need to create the database in the Firebase console.
- 3 Configure Realtime Database Rules
 - The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to.
 - By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up Authentication, you can configure your rules for public access. This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

SDK

Faça as configurações para adicionar o realtime database no app

 Add the Realtime Database SDK to your app ✕

Performing this action will make the following changes to your project.

`build.gradle (project-level)`

Add rules to include the Google Services Gradle plugin:

```
classpath 'com.google.gms:google-services:4.3.5'
```

`app/build.gradle`

Apply the Google Services Gradle plugin:

```
apply plugin: 'com.google.gms.google-services'
```

Add the library dependency:

```
implementation 'com.google.firebase:firebase-database:19.7.0'
```

Accept Changes

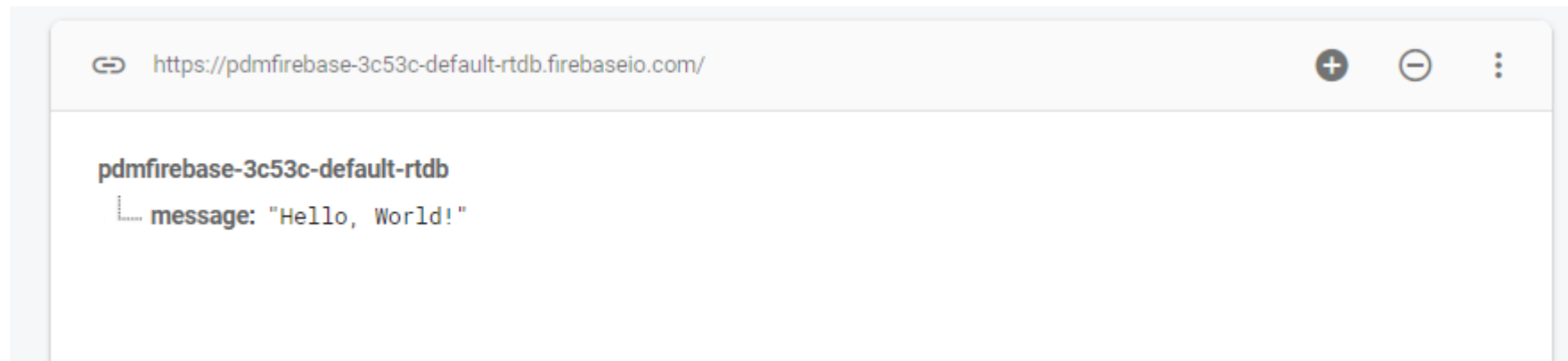
Cancel

Aplique as
configurações
de importar
bibliotecas

Olá Mundo

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val database = FirebaseDatabase.getInstance()  
        val myRef = database.getReference("message")  
  
        myRef.setValue("Hello, World!")  
    }  
}
```

Olá Mundo



```
fun cadastrar(){
    var btnCadastrar: Button = findViewById(R.id.btnCadastrar)

    btnCadastrar.setOnClickListener{
        var txtEmail: EditText = findViewById(R.id.txtEmail)
        var txtsenha: EditText = findViewById(R.id.txtSenha)
        var email:String = txtEmail.text.toString()
        var senha:String = txtsenha.text.toString()

        val dados = HashMap<String, Any>()

        dados["email"] = email
        dados["senha"] = senha

        var tempCaminho = email.replace(".", "")
        val database = FirebaseDatabase.getInstance()
        val myRef = database.getReference("usuario/"+tempCaminho+"/")
        myRef.setValue(dados)
    }
}
```

Cadastrando usuarios

