



## Firebase Database

Permite sincronizar as informações das aplicações no banco de dados NOSQL na nuvem do Firebase sincronizando automaticamente para todos os dispositivos conectados.

exemploautenticacao-c2c24

 mensagens

 8IYsMh7cRbeaTxE8NmJ3uarpiPp2

 -L500C81UF18H0Mz8XFU

id: "-L500C81UF18H0Mz8XFU"

mensagem: "Mensagem inicial"

 -L500DyxXYnFNJUUVLJVg

id: "-L500DyxXYnFNJUUVLJVg"



mensagem: "Bem vindo a todos"

 -L500H1lbYeuJjWaj\_Uw

id: "-L500H1lbYeuJjWaj\_Uw"











mensagem: "Bons Projetos no Code XP"

# Configurando o Ambiente

Assistant  



Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)



- ▶  **Analytics**  
Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)
- ▶  **Cloud Messaging**  
Deliver and receive messages and notifications reliably across cloud and device. [More info](#)
- ▶  **Authentication**  
Sign in and manage users with Firebase, accepting emails, Google Sign-In, Facebook and other providers. [More info](#)
- ▶  **Realtime Database**  
Store and sync data in realtime across all connected clients. [More info](#)
- ▶  **Storage**  
Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)
- ▶  **Remote Config**  
Customize and experiment with app behavior using cloud-based configuration parameters. [More info](#)
- ▶  **Test Lab**  
Test your apps against a wide range of physical devices hosted in Google's cloud. [More info](#)
- ▶  **Crash Reporting**  
Get actionable insights and reports on app crashes, ANRs or other errors. [More info](#)
- ▶  **App Indexing**  
Get your app content into Google Search. [More info](#)
- ▶  **Dynamic Links**  
Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. [More info](#)

No painel lateral do *Assistente do Firebase*, selecione **Realtime Database**

Assistant  



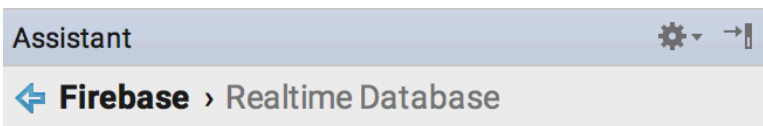
Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

- ▼  **Realtime Database**  
Store and sync data in realtime across all connected clients. [More info](#)  
 [Save and retrieve data](#)

E clique no link **Save and retrieve data**



# Configurando o Ambiente



## Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

### 1 Connect your app to Firebase

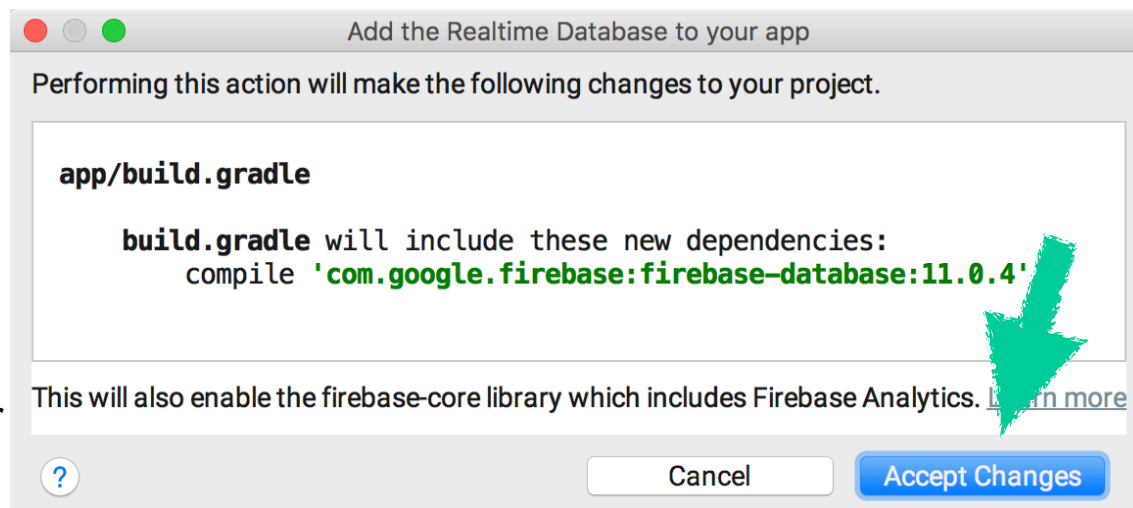
OK Connected

### 2 Add the Realtime Database to your app

Add the Realtime Database to your app

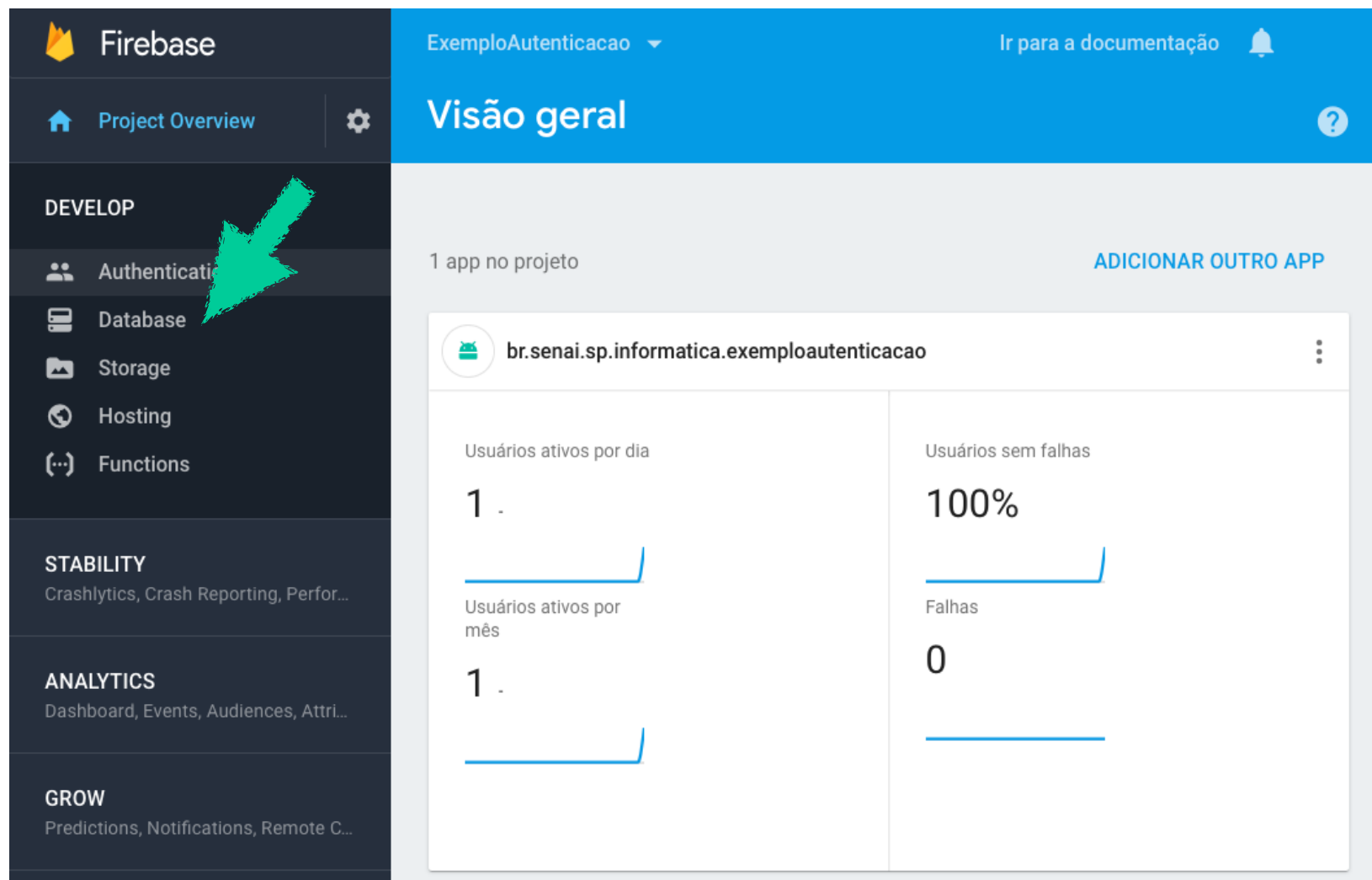
E em seguida clique no botão **Accept Changes** deste diálogo para configurar a aplicação

Agora adicione a configuração do Firebase na aplicação clicando no botão **Add Realtime Database to your app**



# Configurando o Ambiente

Na console do Firebase seleciona **Database** na coluna a esquerda da tela.



The screenshot shows the Firebase console interface. On the left sidebar, the 'Database' option is highlighted with a red arrow. The main area displays the 'Visão geral' (Overview) for the project 'ExemploAutenticacao'. The interface includes a header with the project name and a link to documentation. The main content area shows '1 app no projeto' and a button to 'ADICIONAR OUTRO APP'. Below this, there are two columns of metrics:

Metric	Value
Usuários ativos por dia	1
Usuários ativos por mês	1
Usuários sem falhas	100%
Falhas	0

# Configurando o Ambiente

## Database



### Realtime Database

Armazenar e sincronizar dados em tempo real em todos os clientes conectados

[Saiba mais](#)

PRIMEIROS PASSOS

Após a configuração da aplicação acesse o **Firestore console** e em Database clique no botão **Primeiros Passos**

Na aba de *Dados* é possível encontrar a URL de referência ao Database e os dados, que são atualizados em tempo real

## Database

Realtime Database

DADOS

REGRAS

BACKUPS

USO

<https://exemploautenticacao-c2c24.firebaseio.com/>

★ As regras padrão de segurança exigem que os usuários se autentiquem

[SAIBA MAIS](#)


[DISPENSAR](#)

exemploautenticacao-c2c24: null

# Configurando o Ambiente

Por fim configure o **Build.gradle** da aplicação acrescentando a reverência do **Firestore** **UI** muito útil na construção de *ListViews* e *RecyclerViews*

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:27.0.2'  
    implementation 'com.android.support:design:27.0.2'  
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'  
  
    implementation 'com.google.firebase:firebase-auth:11.8.0'  
    implementation 'com.google.firebase:firebase-database:11.8.0'  
  
    implementation 'com.firebaseui:firebase-ui-database:3.1.1'  
}
```

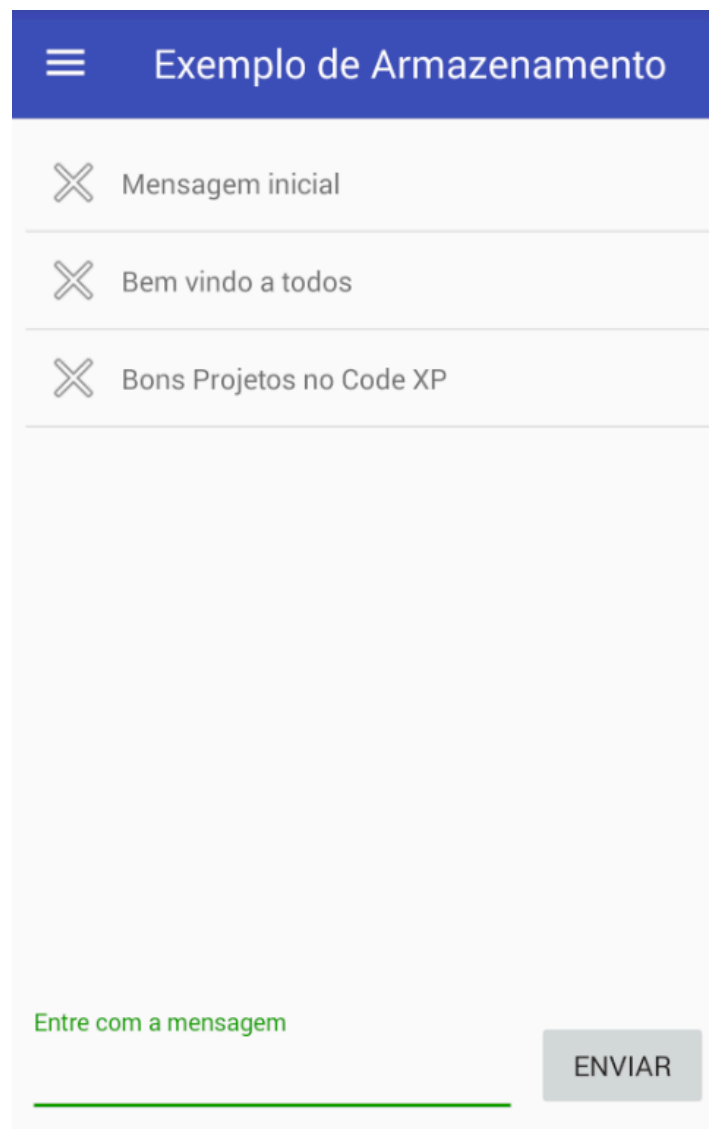


# A Aplicação

Para demonstrar como armazenar dados no *Firebase* foi a aplicação de autenticação foi alterada para poder salvar mensagens simples.

Com este aplicativo podemos criar uma lista de mensagens e recuperá-las a qualquer momento.

Caso queira excluir uma ou mais mensagens, basta clicar no **X** que aparece a frente de cada mensagem.



☰ Exemplo de Armazenamento

- ✕ Mensagem inicial
- ✕ Bem vindo a todos
- ✕ Bons Projetos no Code XP

Entre com a mensagem

ENVIAR



A configuração do **Adapter** para o *ListView* é implementado com a utilização uma classe do *Firebase UI*, a ***FirebaseListOptions***<>

```
@Override
protected void onStart() {
    super.onStart();

    if(adapter == null) {
        FirebaseListOptions<Msgagem> options = new FirebaseListOptions.Builder<Msgagem>()
            .setQuery(dao.getReference(), Msgagem.class)
            .setLayout(R.layout.layout_msgagem)
            .setLifecycleOwner(this)
            .build();

        adapter = new MsgagemAdapter(options);

        listView.setAdapter(adapter);
    }
}
```

```
public class MensagemAdapter extends FirebaseListAdapter<Mensagem> {
    public MensagemAdapter(@NonNull FirebaseListOptions<Mensagem> options) {
        super(options);
        dao.verificaMensagens();
    }

    @Override
    protected void populateView(View view, Mensagem model, int position) {
        hideProgressDialog();

        TextView tvMsg = view.findViewById(R.id.tvMsg);
        ImageView imgDel = view.findViewById(R.id.imgDel);

        tvMsg.setText(model.getMensagem());
        imgDel.setTag(model.getId());
        imgDel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View view) {
                AlertDialog.Builder alerta = new AlertDialog.Builder(context: MainActivity.this);
                alerta.setMessage("Confirma a exclusão desta Mensagem?");
                alerta.setNegativeButton(text: "Não", listener: null);
                alerta.setPositiveButton(text: "Sim", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        dao.remover((String)view.getTag(),
                            new CallbackMessage(msg: "Falha ao excluir a mensagem!"));
                    }
                });
                alerta.create();
                alerta.show();
            }
        });
    }
}
```

A classe **MensagemAdapter** deve estender outra classe do *Firebase UI*, a **FirebaseListAdapter<>**

Para salvar os dados de um Objeto Java no Firebase deveremos construir um mapa de objetos definindo a posição na estrutura de dados que desejamos utilizar.

A alteração de alguma informação ocorre da mesma maneira.

```
public void salvar(Mensagem obj, DatabaseReference.CompletionListener callback) {  
    if(obj.getId() == null) {  
        obj.setId(reference.push().getKey());  
    }  
  
    Map<String, Object> map = new HashMap<>();  
    map.put("id", obj.getId());  
    map.put("mensagem", obj.getMensagem());  
  
    Map<String, Object> updates = new HashMap<>();  
    updates.put("/mensagens/" + user.getUid() + "/" + obj.getId(), map);  
  
    base.updateChildren(updates, callback);  
}
```

Para excluirmos um item na estrutura de dados, deveremos criar um mapa de objetos e atribuir **NULL**, assim o Firebase entenderá de quererá excluir a informação.

```
public void remover(String id, DatabaseReference.CompletionListener callback) {  
    Map<String, Object> updates = new HashMap<>();  
    updates.put("/mensagens/" + user.getId() + "/" + id, null);  
  
    base.updateChildren(updates, callback);  
}
```

Talvez tenha notado que todo o método utiliza **DatabaseReference.CompletionListener** como argumento.

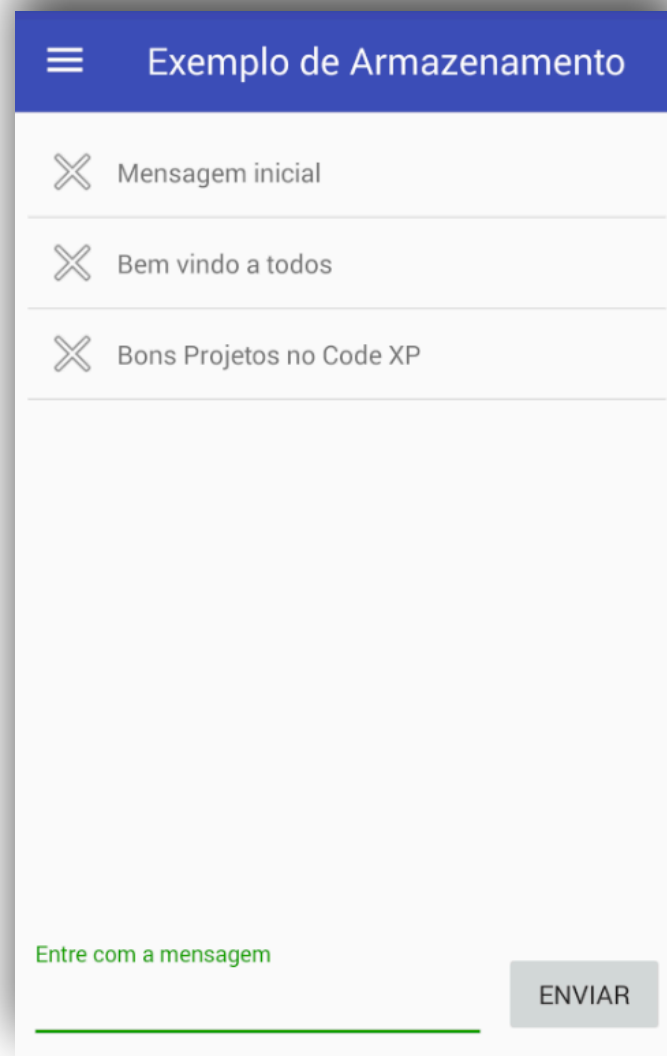
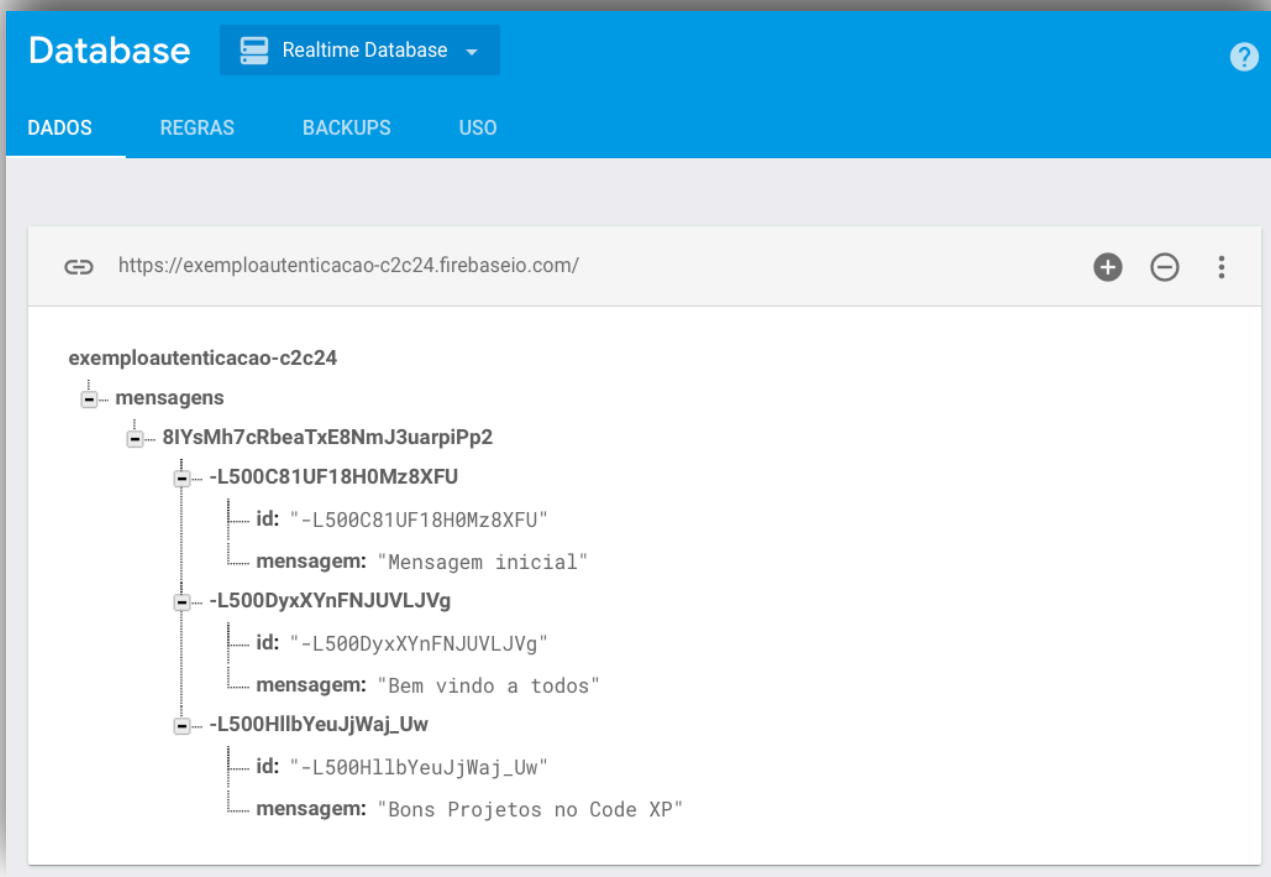
Isto é necessário porque toda a transação no Firebase é assíncrona, assim sendo é necessário a utilização de um mecanismo de *Call Back* para possibilitar um retorno da informação referente a execução solicitada.

Para simplificar a utilização e tornar mais prático, criei uma classe, a **CallBackMessage** que foi utilizada para transformar a recepção dos erros em mensagens ao usuário.

```
public class CallBackMessage implements DatabaseReference.CompletionListener {  
    private String msg;  
  
    public CallBackMessage(String msg) { this.msg = msg; }  
  
    @Override  
    public void onComplete(DatabaseError error, DatabaseReference reference) {  
        if (error != null) {  
            Toast.makeText(context: MainActivity.this, msg,  
                Toast.LENGTH_LONG).show();  
        }  
    }  
}
```

# A Aplicação

Ao utilizar a aplicação é possível notar a atualização dos dados na console do Firebase em tempo real.



FIM