



Desenvolvimento para Dispositivos Móveis



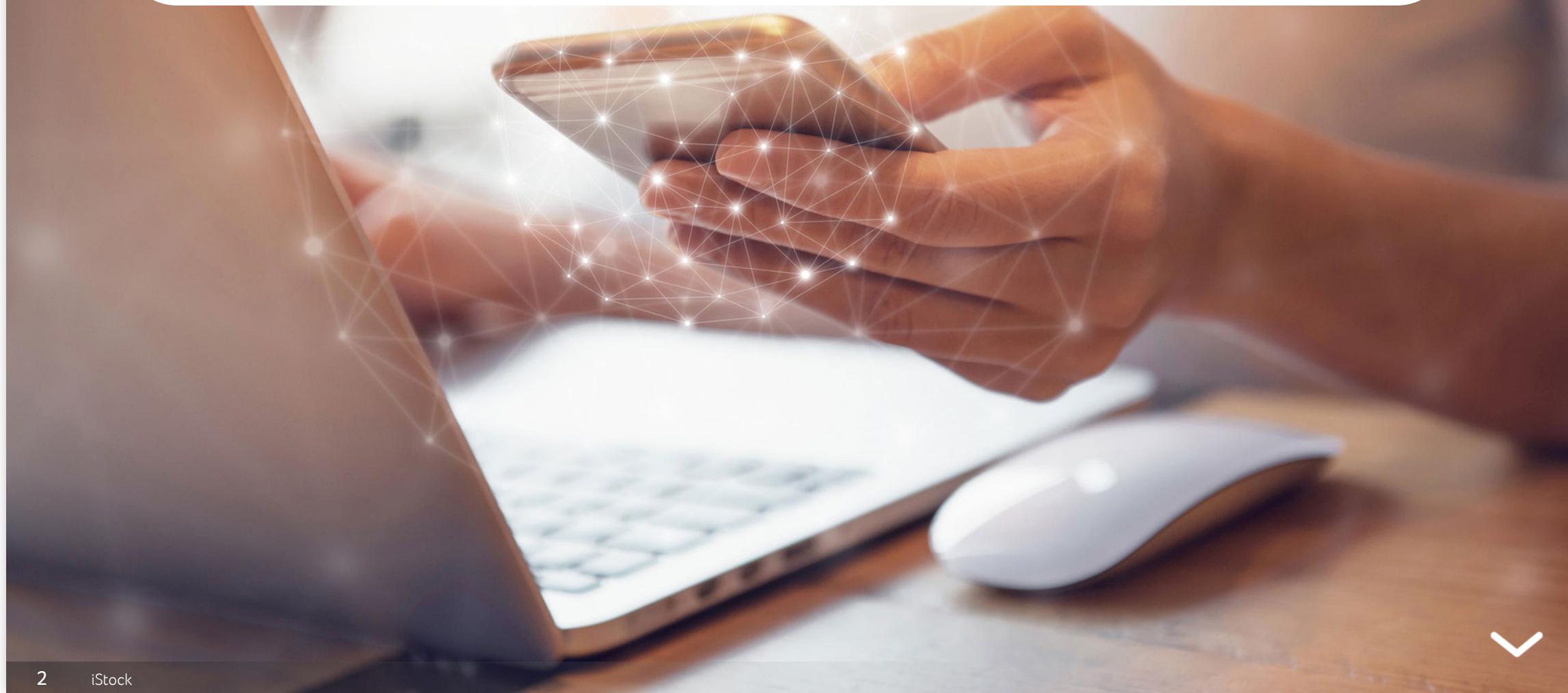
Weaula 3

Trabalhando com banco de dados na nuvem





Nesta webaula você vai ver a manipulação de informações armazenadas no Realtime Database.



Ao utilizar um aplicativo para troca de mensagens instantâneas instalado no seu dispositivo móvel, você é capaz de enviar e receber mensagens em tempo real. Você deve abrir a janela de conversação com algum contato disponível no aplicativo e, a partir dessa tela, as mensagens poderão ser enviadas e recebidas.

Desta forma, vamos ver como gravar e ler informações de um banco de dados na nuvem.

Relembrando

Como vimos anteriormente, o Realtime Database é um banco de dados NoSQL que armazena dados na nuvem. Eles são sincronizados em tempo real com todos os usuários conectados ao banco de dados. Também vimos que o Realtime Database armazena os dados em uma estrutura JSON por meio de chave-valor (key-value).

Clique na imagem para compreender melhor a estrutura de como os dados estão armazenados.

Informações armazenadas – estrutura Realtime Database

raizDoProjeto:

+ produtos:

+ -LFUXkJLOgdPKYw7JINs

descricao: "Smartphone"

preco: 1800

+ -LFUXkb5xcNyAISw3k2d

descricao: "Tablet"

preco: 2200

Fonte: elaborada pelo autor.



Acessar o banco de dados Realtime Database

Para que possamos gravar ou ler dados no Realtime Database, devemos recuperar uma instância do banco de dados.

FirebaseDatabase é o objeto que nos fornece o ponto de entrada para acessarmos o Realtime Database. Portanto, declare um objeto *FirebaseDatabase* e chame pelo método *getInstance ()*.

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```





Por meio da instância do *FirebaseDatabase*, podemos acessar uma instância de *DatabaseReference*.

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

Neste momento, a instância de *DatabaseReference* nos retorna a raiz do projeto da estrutura do Realtime Database.





Inserir dados no Realtime Database

A partir da instância do `DatabaseReference` podemos inserir dados no Realtime Database. Vejo três métodos essenciais para implementarmos esta funcionalidade.

O método `child()` é responsável por criar ou acessar uma referência específica dentro da raiz do projeto do Realtime Database.

O método `push()` também é responsável por criar uma referência específica dentro da raiz do projeto do Realtime Database, contudo, esse método gera uma chave exclusiva para o objeto que será armazenado.

O método `setValue()` armazena o objeto na referência especificada. O objeto enviado para o Realtime Database é convertido automaticamente para a estrutura do JSON.



Clique na imagem para ver o código abaixo responsável por armazenar um objeto Produto no Realtime Database.

```
// Declaramos um objeto produto para ser armazenado no banco de dados
Produto produto = new Produto("Smartphone", 1800f);

// Recuperamos uma instância do Realtime Database
FirebaseDatabase database = FirebaseDatabase.getInstance();

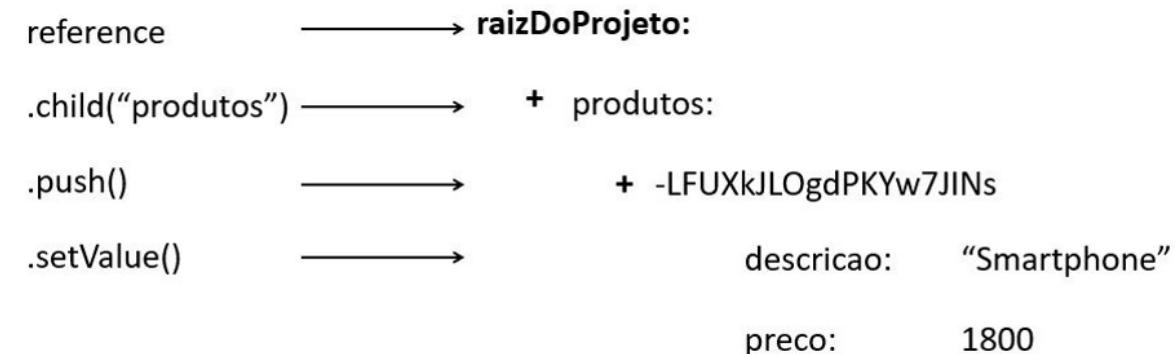
// Recuperamos a raiz do projeto
DatabaseReference reference = database.getReference();

// Através do método child(), acessamos a referência "produtos".
// Em seguida, através do método push(), inserimos uma referência
// gerada automaticamente para identificar o produto
// Finalizamos inserindo o objeto Produto no banco de dados
reference.child("produtos").push().setValue(produto);
```

Fonte: elaborada pelo autor.



A imagem a seguir ilustra o retorno de cada método dentro do Realtime Database (de acordo com sua estrutura).



Fonte: elaborada pelo autor.





Ler dados do Realtime Database

A partir da instância do `DatabaseReference` podemos definir uma referência para lermos dados do Realtime Database. Explore a galeria e conheça os métodos que auxiliam o acesso a referências específicas no Realtime Database.

O método `orderByChild()` recebe como parâmetro uma String e ordena os resultados pelo valor informado no parâmetro. O código abaixo retorna todas as referências de “produtos” ordenadas pelo atributo “descricao”.



```
reference.child("produtos").  
orderByChild("descricao");
```





Acessar os objetos recuperados do banco de

O *ChildEventListener* é responsável por monitorar alterações em uma determinada referência do banco de dados, ou seja, é possível sempre detectar quando um objeto é incluído, atualizado, excluído ou movido.

Clique na imagem para visualizar o código que declara um objeto *ChildEventListener*.

```
// Declaramos o ChildEventListener
private ChildEventListener childEventListener = new
ChildEventListener() {

    @Override
    public void onChildAdded(@NonNull DataSnapshot dataSnapshot,
    @Nullable String s) {
        // Método responsável por detectar sempre que um objeto é
        incluído
    }

    @Override
    public void onChildChanged(@NonNull DataSnapshot dataSnapshot,
    @Nullable String s) {
        // Método responsável por detectar sempre que um objeto é
        atualizado
    }

    @Override
    public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
        // Método responsável por detectar sempre que um objeto é
        excluído
    }

    @Override
    public void onChildMoved(@NonNull DataSnapshot dataSnapshot,
    @Nullable String s) {
        // Método responsável por detectar sempre que um objeto é
        movido
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        // Método chamado quando ocorre um erro com o banco de dados
    }
};
```

Fonte: elaborada pelo autor





Para utilizar o *ChildEventListener* devemos definir uma referência do banco de dados por meio do objeto *DatabaseReference*. Em seguida devemos chamar pelo método *addChildEventListener ()* e passar como parâmetro o *ChildEventListener* criado. O código a seguir é capaz de detectar todas as inclusões, exclusões ou modificações que ocorrem na referência de “produtos”.

```
@Override  
protected void onStart() {  
    super.onStart();  
  
    // Adicionamos o ChildEventListener para a referência do  
    // Realtime Database  
    reference.child("produtos")  
        .addChildEventListener(childEventListener);  
}
```

Fonte: elaborada pelo autor.



Cada método de retorno recebe como parâmetro um objeto DataSnapshot. Para acessar os dados contidos no objeto DataSnapshot, chame pelo método `getValue()` e informe como parâmetro a classe Java que representa o objeto que se deseja recuperar.

Clique na imagem para visualizar melhor o código para recuperar o objeto Produto.

```
@Override  
public void onChildAdded(@NonNull DataSnapshot dataSnapshot,  
@Nullable String s) {  
    // Método responsável por detectar sempre que um objeto é  
    // incluído  
    Produto produto = dataSnapshot.getValue(Produto.class);  
}
```

Fonte: elaborada pelo autor



Remover Listener

Ao adicionar um *ChildEventListener*, o aplicativo receberá os objetos sempre que houver modificações nas referências do Realtime Database. Contudo, o usuário poderá sair do aplicativo e navegar para outras telas. Portanto, é importante informar ao aplicativo que pare de “ouvir” as modificações que ocorrem no banco de dados.

Essa técnica deve ser implementada para economizar dados de internet do usuário e para evitar que o aplicativo continue recebendo atualizações dos dados, principalmente quando o usuário não está com a tela do aplicativo em primeiro plano.





Os passos a seguir, indicam como remover um *listener* de uma referência do Realtime Database.

- 1) Sobreponha o método *onStop()* correspondente ao ciclo de vida da *Activity*.
- 2) Chame pela mesma referência na qual o listener foi adicionado.
- 3) Chame pelo método *removeEventListener()* e passe como parâmetro o listener que será removido.

```
@Override  
protected void onStop() {  
    super.onStop();  
    reference.child("produtos")  
        .removeEventListener(childEventListener);  
}
```

Fonte: elaborada pelo autor.





Excluir dados no Realtime Database

A partir da instância do DatabaseReference, podemos excluir dados do banco de dados. Defina qual a referência no Realtime Database que deseja remover e chame pelo método `removeValue()`. O código abaixo exclui todos os produtos do banco de dados.

```
reference.child("produtos").removeValue();
```



Atualizar dados no Realtime Database

A partir da instância do DatabaseReference, podemos atualizar dados do banco de dados. Defina qual a referência no Realtime Database que deseja atualizar e chame pelo método `updateChildren()`.

Clique na imagem e veja o código que atualiza o produto “Smartphone” para “Smartphone Android” e o seu valor para “1500”.

```
reference.child("produtos").orderByChild("descricao").equalTo("Smartphone")
    .addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            // Verificamos todas as referências existentes através
            // do método getChildren()
            for (DataSnapshot data : dataSnapshot.getChildren()) {
                // Objeto com a estrutura chave-valor
                // String é a chave
                // Object é o valor
                Map<String, Object> valoresAtualizados = new
                HashMap<>();
                // Informamos qual atributo desejamos atualizar e
                // o valor atualizado
                valoresAtualizados.put("descricao", "Smartphone
                Android");
                // Atualizamos o produto
                data.getRef().updateChildren(valoresAtualizados);
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
        }
    });
}
```

Fonte: elaborada pelo autor.

Nesta webaula você viu a implementação de funcionalidades para manipulação de dados no Realtime Database. Seja sempre curioso e pesquise por mais funcionalidades disponíveis na plataforma Firebase.





Vídeo de encerramento







Bons estudos!

