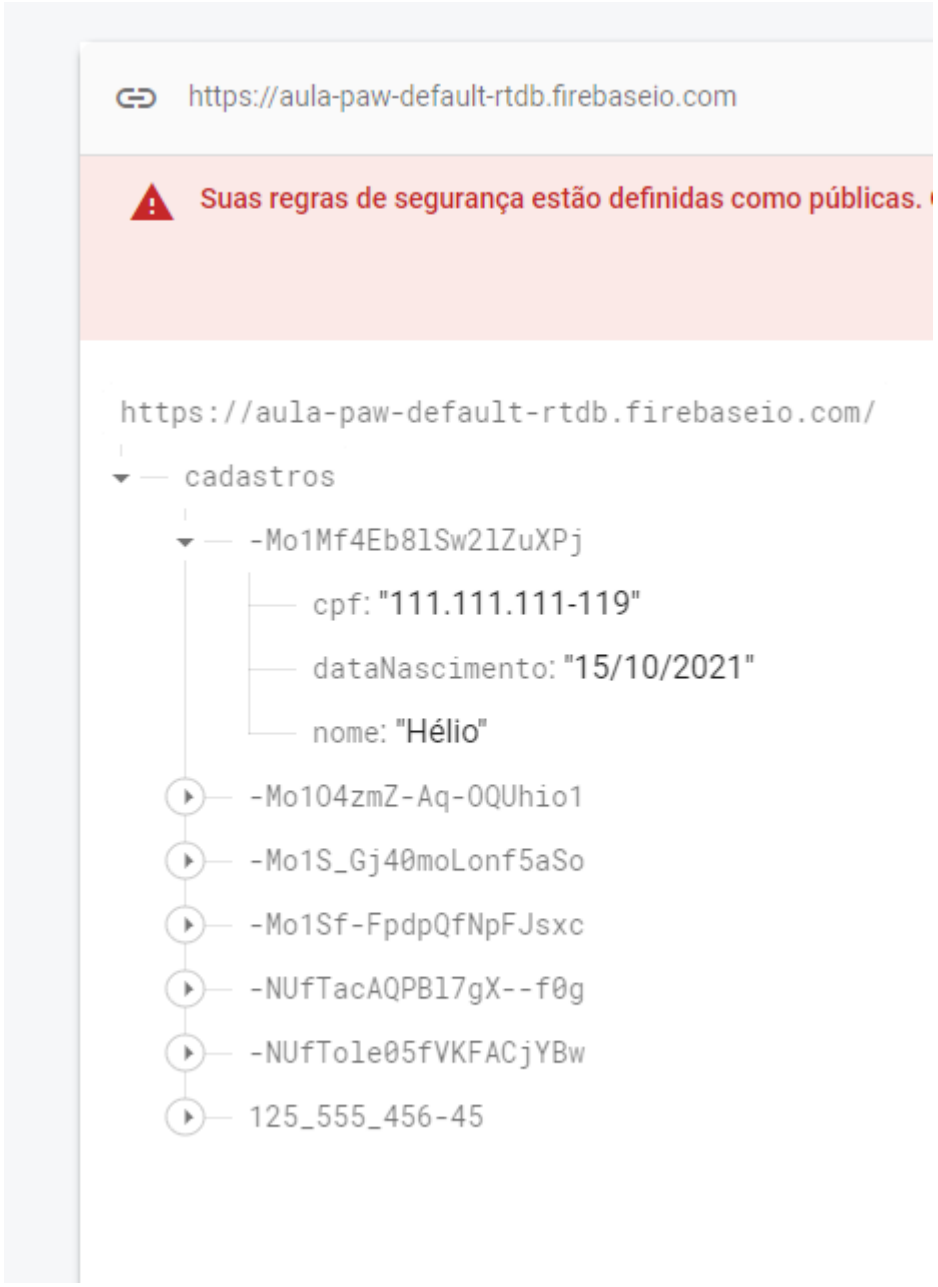
Operações básicas - Firebase realtime database

Prof. Me. Hélio Esperidião

Estrutura dos dados



Cadastrar um novo

```
fun cadastrarDados(){
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference("/cadastros/")
    val novoCadastro = mapOf(
        "cpf" to "123.456.678-99",
        "dataNascimento" to "15/10/2000",
        "nome" to "Benedita"
    )
    myRef.push().setValue(novoCadastro).addOnCompleteListener { task ->
        if (task.isSuccessful) {
            // Registro adicionado com sucesso
        } else {
            // Trate o erro aqui
        }
    }
}
```

Listar Todos

```
fun listarTodos(){
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference("/cadastros/")
    myRef.get().addOnCompleteListener { task ->
        if (task.isSuccessful) {
            val clientes = task.result
            if (clientes.exists()) {
                for (cliente in clientes.children) {
                    val idDoCadastro = cliente.key
                    val nome = cliente.child("nome").value as String
                    val cpf = cliente.child("cpf").value as String
                    val dataNascimento = cliente.child("dataNascimento").value as String
                }
            }
        }
    }
}
```

Buscar cpf

```
fun buscarDados(){
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference("/cadastros/")
    val cpf = "222.222.222-22"
    val query = myRef.orderByChild("cpf").equalTo(cpf)
    query.addListenerForSingleValueEvent(object : ValueEventListener {
        override fun onDataChange(clientes: DataSnapshot) {
            if (clientes.exists()) {
                for (cliente in clientes.children) {
                    val idDoCadastro = cliente.key
                    val nome = cliente.child("nome").value as String
                    val dataNascimento = cliente.child("dataNascimento").value as String
                }
            }
        }
        override fun onCancelled(databaseError: DatabaseError) {
            // Trate o erro aqui
        }
    })
}
```

Atualizar dados

```
fun atualizarDados(){
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference("/cadastros/")
    val buscarCpf = "222.222.222-22"
    val query = myRef.orderByChild("cpf").equalTo(buscarCpf)
    query.addListenerForSingleValueEvent(object : ValueEventListener {
      override fun onDataChange(clientes: DataSnapshot) {
        if (clientes.exists()) {
          for (cliente in clientes.children) {
            val idCliente = cliente.key.toString()
            val nome = cliente.child("nome").value as String
            val dataNascimento = cliente.child("dataNascimento").value as String
            val atualizar = mapOf(
              "cpf" to "222.222.222-22",
              "dataNascimento" to "04/10/2021",
              "nome" to "Maria Aparecida")
            myRef.child(idCliente).updateChildren(atualizar).addOnCompleteListener { task -> }
          }
        }
      }
      override fun onCancelled(databaseError: DatabaseError) {
      }
    })
}
```

Ouvir Modificações

```
fun ouvirModificacoes(){
    val idDoCadastr= "-Mo1Mf4Eb8lSw2lZuXPj"
    val database = FirebaseDatabase.getInstance()
    val myRef = database.getReference("/cadastros/"+idDoCadastr+"/")
    val valueEventListener = object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {
            if (snapshot.exists()) {
                val nome = snapshot.child("nome").value as String
                val cpf = snapshot.child("cpf").value as String
                val dataNascimento = snapshot.child("dataNascimento").value as String
            }
        }
        override fun onCancelled(error: DatabaseError) {        }
    }
    myRef.addValueEventListener(valueEventListener)
}
```