Pesquisar valores, vamos imaginar que queremos todos os usuários com nome de Nogueira.

Output: alleph nogueira | Fernanda nogueira | crixus nogueira....

Vamos criar o método pesquisar dados.

Aqui estamos trazendo todos os dados

private fun pesquisarDados() {  
 // Acessando a lista de usuarios  
 val referenciaUsuario = bancoDeDados  
 .collection("usuarios")  
  
 // Acessando a lista de documento  
  
 referenciaUsuario.addSnapshotListener **{** querySnapshopt, error **->** val listaDeDocumentos = querySnapshopt?.*documents* // Onde vamos armazenar o resultado dos nomes  
 var listaResultadoDocumento = ""  
  
 listaDeDocumentos?.*forEach* **{** documentSnapshot **->** // Percorrendo a lista de documentos.  
 val dados = documentSnapshot?.*data* if (dados != null) {  
 val nome = dados["nome"] // Estamos acessando a chave nome, para pegar o valor dela  
 val anoNascimento = dados["anoNascimento"]  
  
  
 listaResultadoDocumento += "$nome - $anoNascimento\n" // Aqui estamos acumulando todos os usuarios.  
  
 binding.textResultado.*text* = listaResultadoDocumento  
 }  
 **}  
  
  
 }**}

# Agora vamos trazer somente os dados que queremos.

Usamos o whereEqualTo para pegar os dados que são iguais.

private fun pesquisarDados() {  
 // Acessando a lista de usuarios  
 val referenciaUsuarios = bancoDeDados  
 .collection("usuarios")  
 **.whereEqualTo("nome", "Alleph N") // Onde nome for igual a Alleph (Atenção ele diferencia maiuscula de minuscula)**  
 // Acessando a lista de documento  
  
 referenciaUsuarios.addSnapshotListener **{** querySnapshopt, error **->** val listaDeDocumentos = querySnapshopt?.*documents* // Onde vamos armazenar o resultado dos nomes  
 var listaResultadoDocumento = ""  
  
 listaDeDocumentos?.*forEach* **{** documentSnapshot **->** // Percorrendo a lista de documentos.  
 val dados = documentSnapshot?.*data* if (dados != null) {  
 val nome = dados["nome"] // Estamos acessando a chave nome, para pegar o valor dela  
 val anoNascimento = dados["anoNascimento"]  
  
  
 listaResultadoDocumento += "$nome - $anoNascimento\n" // Aqui estamos acumulando todos os usuarios.  
  
  
 }  
 **}** binding.textResultado.*text* = listaResultadoDocumento  
 **}**}

# Agora pegando os dados que não são iguais

private fun pesquisarDados() {  
 // Acessando a lista de usuarios  
 val referenciaUsuarios = bancoDeDados  
 .collection("usuarios")  
 //.whereEqualTo("nome", "Alleph N") // Onde nome for igual a Alleph (Atenção ele diferencia maiuscula de minuscula)  
 .whereNotEqualTo("nome", "Alleph N") // Agora ele vai pegar todos que nao for igual a Alleph N  
  
 // Acessando a lista de documento  
  
 referenciaUsuarios.addSnapshotListener **{** querySnapshopt, error **->** val listaDeDocumentos = querySnapshopt?.*documents* // Onde vamos armazenar o resultado dos nomes  
 var listaResultadoDocumento = ""  
  
 listaDeDocumentos?.*forEach* **{** documentSnapshot **->** // Percorrendo a lista de documentos.  
 val dados = documentSnapshot?.*data* if (dados != null) {  
 val nome = dados["nome"] // Estamos acessando a chave nome, para pegar o valor dela  
 val anoNascimento = dados["anoNascimento"]  
  
  
 listaResultadoDocumento += "$nome - $anoNascimento\n" // Aqui estamos acumulando todos os usuarios.  
  
  
 }  
 **}** binding.textResultado.*text* = listaResultadoDocumento  
 **}**}

# Pesquisar uma lista

Nesse exemplo vamos pegar apenas 2 itens dessa lista.

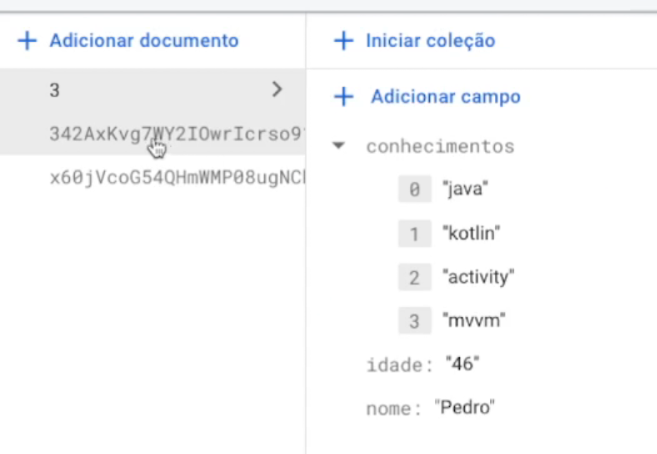
Mas imagina no alta pressão GNV, onde o usuário pode escolher mais de 2 classificação, poderia escolher ver os postos 3 estrelas e 4 estrelas.

\*\*\* Todo código a baixo segue igual, por isso vou colocar somente a parte do filtro. \*\*\*

private fun pesquisarDados() {  
 // Acessando a lista de usuarios  
 val referenciaUsuarios = bancoDeDados  
 .collection("usuarios")  
 //.whereEqualTo("nome", "Alleph N") // Onde nome for igual a Alleph (Atenção ele diferencia maiuscula de minuscula)  
 //.whereNotEqualTo("nome", "Alleph N") // Agora ele vai pegar todos que nao for igual a Alleph N  
 .whereIn("nome", *listOf*("Alleph N", "Crixus")) // Pesquisar apenas o valores que estao dentro da lista

Tambem temos um notIn que seria ao contrário, iria exibir todos que não são AllephN e Crixus

# Usuarios com uma lista de array



Também podemos ter usuários com uma lista de array.

Imagina que seja um aplicativo tipo linkedin, onde os usuários tem seus conhecimentos.

Agora como já temos esses array, podemos pesquisar somente os usuários que tem conhecimento ex: Kotlin

private fun pesquisarDados() {  
 // Acessando a lista de usuarios  
 val referenciaUsuarios = bancoDeDados  
 .collection("usuarios")  
 //.whereEqualTo("nome", "Alleph N") // Onde nome for igual a Alleph (Atenção ele diferencia maiuscula de minuscula)  
 //.whereNotEqualTo("nome", "Alleph N") // Agora ele vai pegar todos que nao for igual a Alleph N  
 //.whereIn("nome", listOf("Alleph N", "Crixus")) // Pesquisar apenas o valores que estao dentro da lista  
 //.whereNotIn("nome", listOf("Alleph N", "Crixus")) // Aqui vamos trazer todos que nao esteja na lista.  
 .whereArrayContains("conhecimentos", "kotlin") // Criamos um array no banco de dados e trouxemos apenas os usuarios que conhecem kotlin

Dessa forma vai trazer todos os usuários que tem Kotlin como conhecimento.

Isso é muito útil, podemos criar um aplicativo de finanças e trazer apenas as compras acima de 100 reais por exemplo.