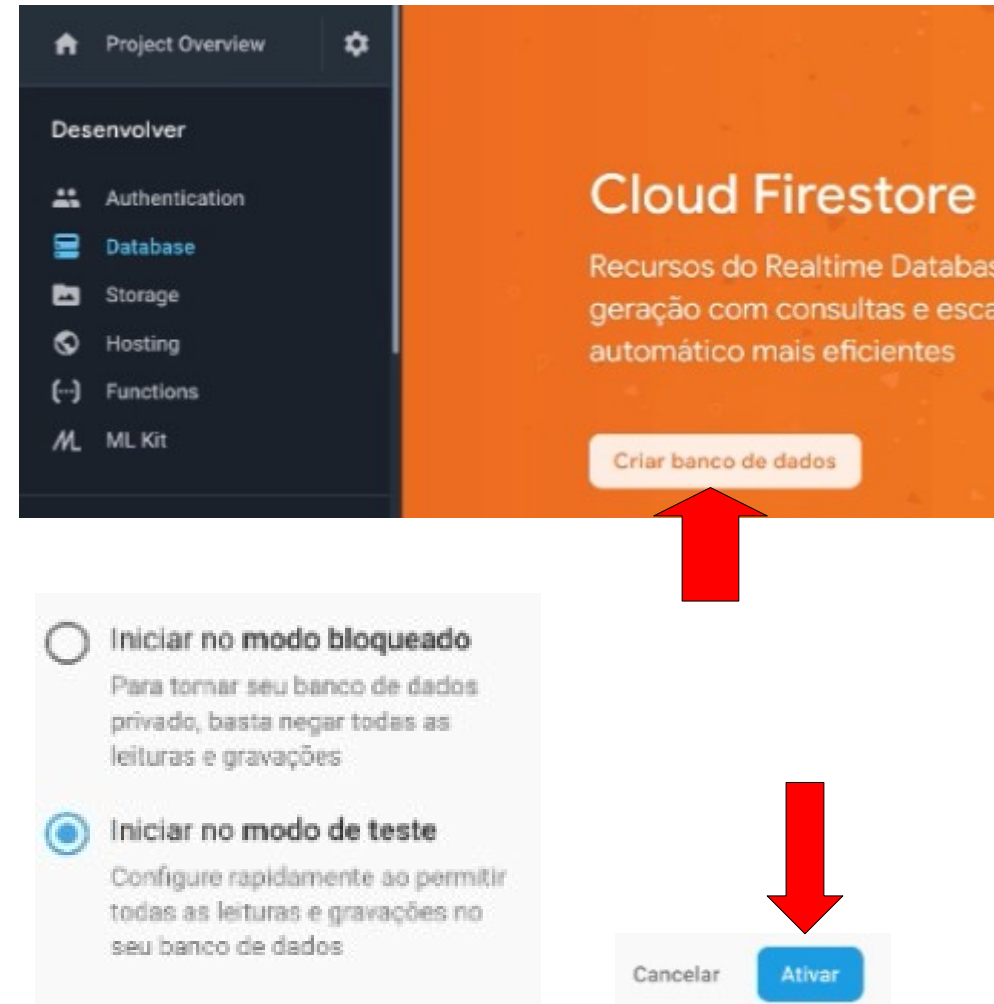


Firestore e FloatingActionButton



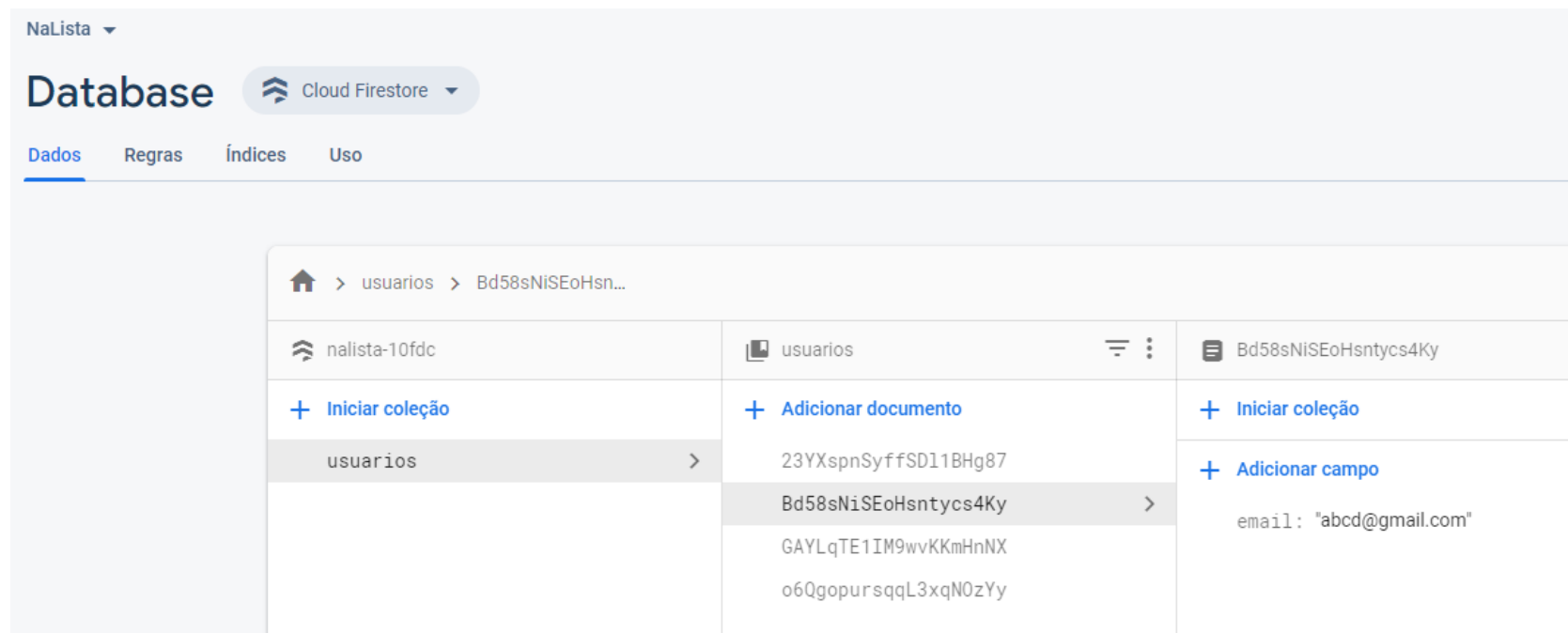
Firestore Cloud Firestore

- O Cloud Firestore ou simplesmente Firestore é um banco de dados não relacional e real time.
- Para criar um banco de dados basta ir na opção **Database** do console do Firebase e selecionar **Criar banco de dados**.
- Iniciar em **modo teste** é interessante para fazer as implementações e testes iniciais. É possível, posteriormente, mudar as regras de acesso para deixar o banco privado.



Firestore console

- Nesse console é possível inserir coleções, documentos e campos em documentos.
- Também é possível definir regras de acesso dentre outras configurações.



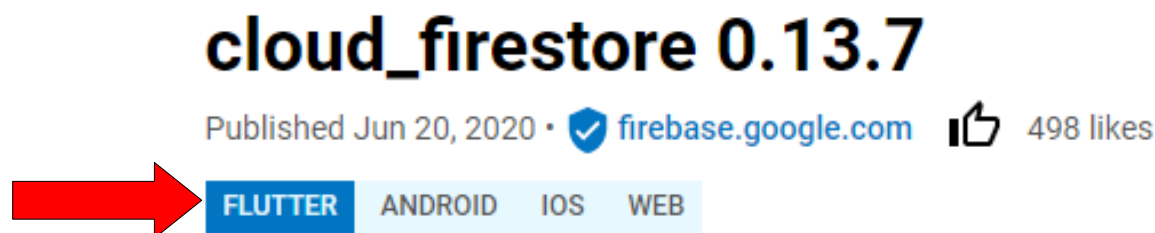
Firestore

- Coleções são conjuntos de documentos e documentos possuem campos.
- Coleções seriam o mais próximo de uma tabela no modelo relacional, mas aqui não existe o conceito de coluna pois os campos são dinâmicos.
- Documentos seriam como registros ou linhas de uma tabela, mas novamente os campos são dinâmicos, ou seja, um documento pode ter o campo “X” e outro o campo “Y” e ambos podem ter um campo “Z”.
- Todo documento possui um **documentID** que identifica esse documento de forma única. O documentID é similar a uma chave primária artificial, ou seja, criada fora da lógica de negócio.
- O uso de bancos de dados não relacionais tem crescido muito nos últimos anos. Dentre as vantagens estão a capacidade de armazenar grandes quantidades de dados. Por outro lado, não existem as restrições de integridade, cabendo ao programador garantir que os dados permanecerão consistentes. Um bom exemplo é a não existência de chaves estrangeiras, nesse contexto, cabe ao programador não corromper a consistência dos dados.



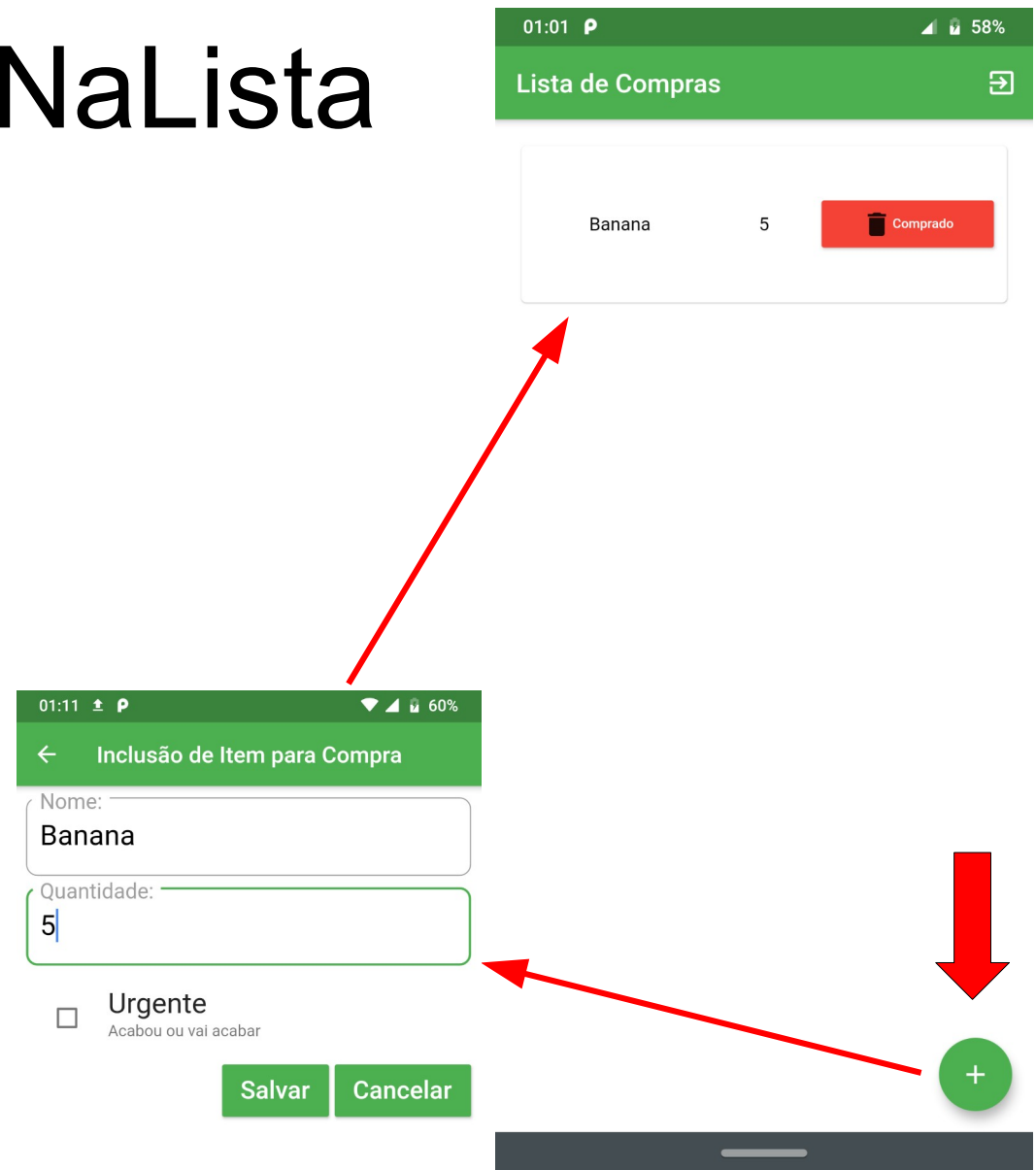
Instalação do plugin

- Mais uma vez precisaremos ir ao site **pub.dev**.
- Dessa vez procuraremos pelo plugin **cloud_firestore**.
- O procedimento de instalação é similar ao de outros plugins que já instalamos e se encontra em:
https://pub.dev/packages/cloud_firestore#-installing-tab-
- Esse plugin funciona em Android, iOS e Web.

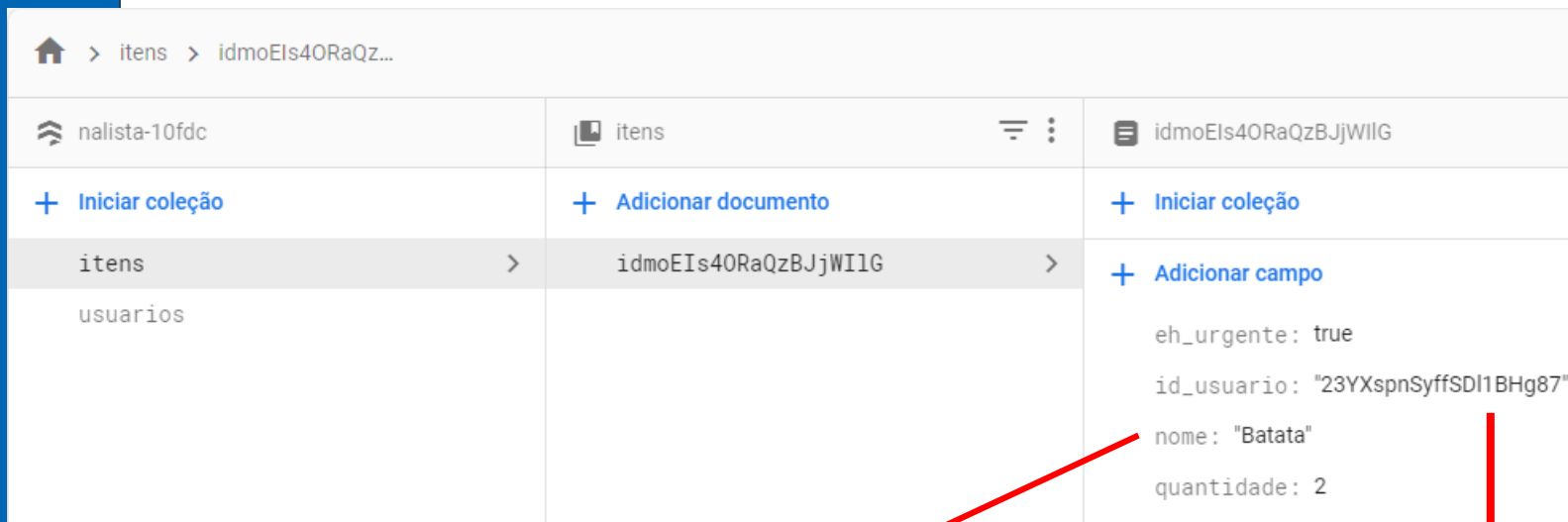


TelaPrincipal – NaLista

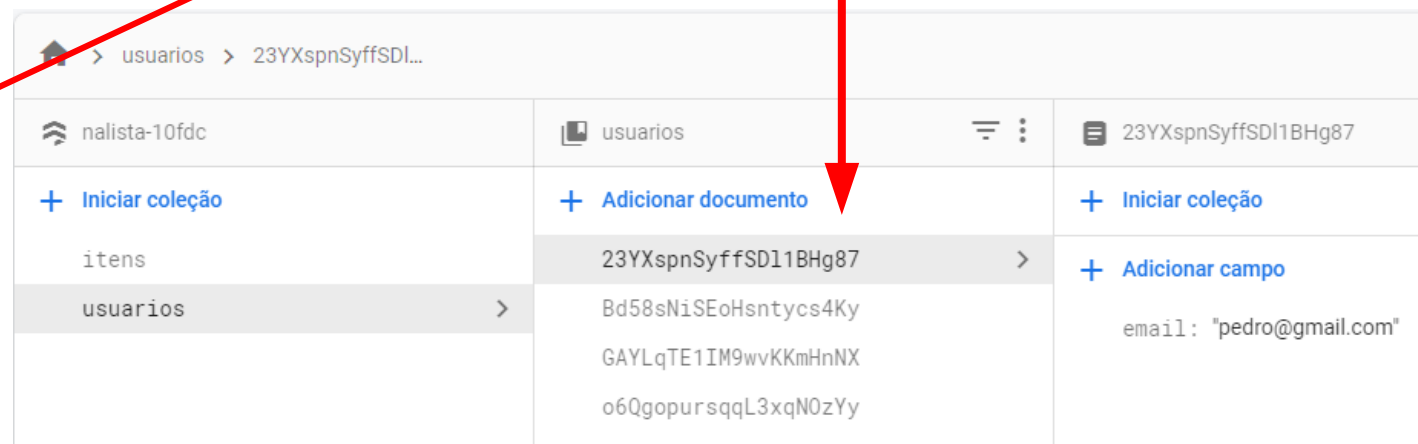
- Trabalharemos a inclusão, busca e exclusão de dados a partir da **TelaPrincipal** do app **NaLista**.
- Nessa tela temos um cadastro de itens de compra e esses itens devem estar vinculados ao usuário que logou no sistema.
- Basicamente, pessoas que quiserem compartilhar uma lista de compras devem usar o mesmo usuário, assim se alguém cadastrar um item (ou fizer uma compra – equivalente a apagar um item) dará ciência dessa situação a todos que compartilham o mesmo usuário.
- O armazenamento desses itens será feito pelo **Firestore**.
- Ao lado temos a **TelaPrincipal** e a **TelaEdicaoItem**.



Um item no console do Firestore



This is a screenshot of a mobile application's 'Inclusão de Item para Compra' (Item Addition for Purchase) screen. It features a green header with a back arrow and the title. Below the header, there are two input fields: 'Nome:' with the value 'Batata' and 'Quantidade:' with the value '2'. A checkbox labeled 'Urgente' is checked, with the text 'Acabou ou vai acabar' (Finished or about to finish) below it. At the bottom, there are two green buttons: 'Salvar' (Save) and 'Cancelar' (Cancel).





FloatingActionButton

- É o botão que acionar a funcionalidade principal ou mais comum de uma tela.
- No nosso caso utilizamos um **FloatingActionButton** para adicionar novos itens na lista de compras.
- Há um parâmetro padrão para tratar isso num **Scaffold** (tela). A imagem abaixo mostra esse parâmetro (floatingActionButton) e o uso do **FloatingActionButton**.

```
floatingActionButton: FloatingActionButton(  
  child: Icon(Icons.add),  
  onPressed: () {  
    _controle.irParaTelaEdicaoItem(context);  
  },  
)
```

01:11 60%

< Inclusão de Item para Compra

Nome: Banana

Quantidade: 5


☐ Urgente
Acabou ou vai acabar

Salvar Cancelar



TelaPrincipal

```
- appBar: AppBar(  
  title: Text("Lista de Compras"),  
  actions: <Widget>[  
    IconButton(  
      icon: Icon(Icons.exit_to_app),  
      onPressed: () {  
        _controle.sair(context);  
      },  
    ), // IconButton  
  ], // <Widget>[]  
), // AppBar
```



```
void sair(BuildContext context){  
  FirebaseAuth.instance.signOut();  
  push(context, TelaLogin(), replace: true);  
}
```



- Além do FloatingActionButton na **TelaPrincipal** temos o **AppBar**. Nessa **AppBar** são colocadas os **actions** que são widgets colocados na **AppBar** (nesse caso um **IconButton**) para funcionar como menus.
- No app **NaLista** quando o único **IconButton** existente é clicado ele aciona o método **sair** do controlador associado a essa tela.
- O método **sair** promove o logout (**FirebaseAuth.instance.signOut()**), ou seja, não haverá mais usuário logado, e retornará a para a **TelaLogin**.



StreamBuilder para o Firestore

```
Stream<QuerySnapshot> get stream => _collection_itens.where("id_usuario", isEqualTo: usuario.id).snapshots();
```

```
Container _stream_builder() {  
  return Container(  
    padding: EdgeInsets.all(16),  
    child: StreamBuilder<QuerySnapshot>(  
      stream: _controle.stream,   
      builder: (context, snapshot) {  
        if(!snapshot.hasData){  
          return Center(  
            child: CircularProgressIndicator(),  
          ); // Center  
        }  
        _controle.obterItens(snapshot.data);  
        return _listView();  
      }  
    ), // StreamBuilder  
  ); // Container  
}
```

```
void obterItens(QuerySnapshot data){  
  document_itens = data.documents;  
  itens = document_itens.map((DocumentSnapshot document) {  
    return Item.fromMap(document.data);  
  }).toList();  
}
```



- Na linha acima temos a busca dos itens do usuário logado (**ControladorTelaPrincipal**).
- O **StreamBuilder** então vai se encarregar de promover a atualização da **TelaPrincipal** quando os itens do usuário forem obtidos.
- Enquanto não tivermos dados será apresentado um **CircularProgressIndicator**.
- Quando os dados chegarem estarão dentro de um objeto do tipo **QuerySnapshot**.
- Na sequência fazemos a conversão de Map para Item através do construtor nomeado **fromMap**, para cada um dos documentos existentes em **data**. Na prática estamos convertendo uma lista de **documents** numa lista de **itens** através do método **map**.
- Essa lista de itens será utilizada para desenhar os itens no ListView criado em **_listView()**.



Flutter 2.0

```
void obterItens(QuerySnapshot data){  
  document_itens = data.documents;  
  itens = document_itens.map((DocumentSnapshot document) {  
    return Item.fromMap(document.data);  
  }).toList();  
}
```

```
void obterItens(QuerySnapshot data){  
  document_itens = data.docs;  
  itens = document_itens!.map((DocumentSnapshot document) {  
    Map<String, dynamic> data = document.data()! as Map<String, dynamic>;  
    return Item.fromMap(data);  
  }).toList();  
}
```



ListView de Itens

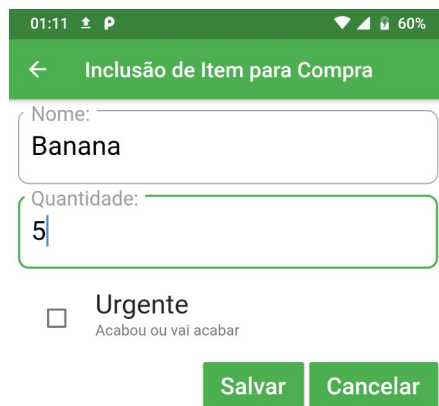
- Aqui tem-se a criação do **ListView** através da lista de itens (**_controle.itens**) gerada anteriormente.
- **CardItem** é um widget que representa o desenho de uma linha de item, ou seja, uma linha que apresenta os dados do item e o botão “Comprado”.

```
ListView _listView() {  
  return ListView.builder(  
    itemCount: _controle.itens != null ? _controle.itens.length : 0,  
    itemBuilder: (context, index) {  
      Item item = _controle.itens[index];  
      return CardItem(item, _controle, index);  
    },  
  ); // ListView.builder  
}
```



Inserção de um item usando Firestore

```
void _inserir_item() {  
  Item item = Item(  
    nome: controlador_nome.text,  
    quantidade: int.parse(controlador_quantidade.text),  
    id_usuario: usuario.id,  
    eh_urgente: eh_urgente,  
  );  
  // Salvando no serviço de armazenamento  
  DocumentReference docRef = _collection_itens.document();  
  docRef.setData(item.toMap());  
}
```



The screenshot shows a mobile application interface with a green header bar containing a back arrow and the text "Inclusão de Item para Compra". Below the header, there are two text input fields: "Nome:" with the value "Banana" and "Quantidade:" with the value "5". Below these fields is a checkbox labeled "Urgente" with the subtitle "Acabou ou vai acabar". At the bottom of the form are two green buttons: "Salvar" and "Cancelar". The status bar at the top shows the time "01:11", signal strength, and battery level "60%".

- A inserção de um item é feita pelo **ControladorTelaEdicao**.
- Basicamente ele cria um item novo com os dados preenchidos no formulário, cria um documento novo e atualizada os dados desse documento fazendo a conversão de um item num Map através do método **toMap()**.



Flutter 2.0

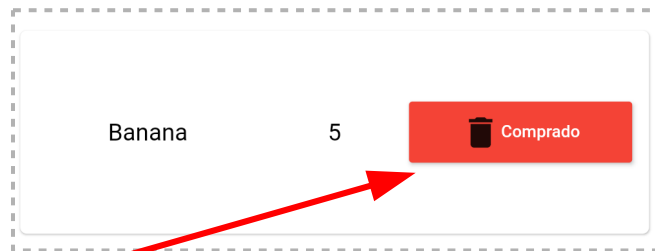
```
void _inserir_item() {  
  Item item = Item(  
    nome: controlador_nome.text,  
    quantidade: int.parse(controlador_quantidade.text),  
    id_usuario: usuario.id,  
    eh_urgente: eh_urgente,  
  );  
  // Salvando no serviço de armazenamento  
  DocumentReference docRef = _collection_itens.document();  
  docRef.setData(item.toMap());  
}
```

```
void _inserir_item() {  
  Item item = Item(  
    nome: controlador_nome.text,  
    quantidade: int.parse(controlador_quantidade.text),  
    id_usuario: usuario.id,  
    eh_urgente: eh_urgente,  
  );  
  
  // Salvando no serviço de armazenamento  
  DocumentReference docRef = _collection_itens.doc();  
  docRef.set(item.toMap());  
}
```



Exclusão de um item usando Firestore

```
void excluirItem(int index){  
  DocumentSnapshot documentSnapshot = document_itens[index];  
  documentSnapshot.reference.delete();  
}
```



```
Expanded(  
  flex: 3,  
  child: BotaoIcone(  
    texto: "Comprado",  
    ao_clicar: () async {  
      controle.excluirItem(index);  
    },  
    cor: Colors.red,  
    icone: Icons.delete,  
  ), // BotaoIcone  
), // Expanded
```

- Basicamente quando o usuário, na **TelaPrincipal**, clica no botão **“Comprado”** de uma linha, é acionado o método **excluirItem** do controlador passando o índice da linha clicada.
- O método **excluirItem**, a partir do índice do item que se deseja excluir, apaga o item selecionado.
- Para fins de uso desse app, comprar algo significa que posso retirá-lo da lista de compras.



Dúvidas?

