

## Descrição

A quem este app se destina

## Features

## Mocks de UI

Tela 1 - MainActivity

Tela 2 - DetailsActivity

Tela 3 - Widget

## Principais considerações

Esquema de persistência de dados.

Cenário básico de UX.

Bibliotecas utilizadas.

Utilização dos serviços Firebase.

## Próximos Passos: Tasks necessárias para concluir o projeto

Task 1: Setup do Projeto

Task 2: Mapear o modelo

Task 3: Criação do esquema de persistência de dados

Task 4: Configurações da classe Repository

Task 5: Implementar o layout da MainActivity

Task 6: Criação da ViewModel

Task 7: Data Binding na UI

Task 8: Criar layout e ViewModel para a DetailsActivity

Task 9: Transição de MainActivity e DetailsActivity

Task 10: Serviços do Firebase

Task 11: Free e Paid Flavors

Task 12: Acessibilidade

Task 13: Widget de personagens favoritos

**GitHub Username:** J-Henrique (<https://github.com/J-Henrique>)

# GoT Collection

## Descrição

A saga Game of Thrones possui uma infinidade de personagens, culturas e informações; sendo um universo fictício extremamente rico. Entretanto, nem sempre é simples lembrar informações básicas de cada personagem, o que pode deixar os fãs meio perdidos no decorrer da história. Este app visa facilitar a obtenção de informações específicas de cada personagem, a fim de prover um guia para consultas rápidas, de forma organizada; e assim gerar uma melhor experiência aos fãs.

## A quem este app se destina

Este app destina-se a leitores e fãs da saga Game of Thrones.

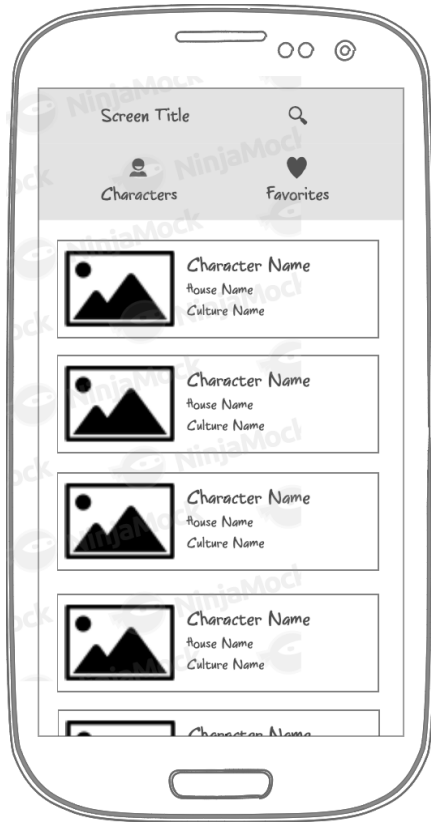
## Features

A aplicação tem como principais recursos:

- Procurar um determinado personagem
- Exibir detalhes, como livros em que ele aparece, casa que pertence ou cultura
- Salvar personagem na lista de favoritos
- Listar os personagens favoritados por meio de um widget na Home Screen
- Permitir acessar rapidamente informações de personagens, ao clicar na listagem do widget

## Mocks de UI

### Tela 1 - MainActivity



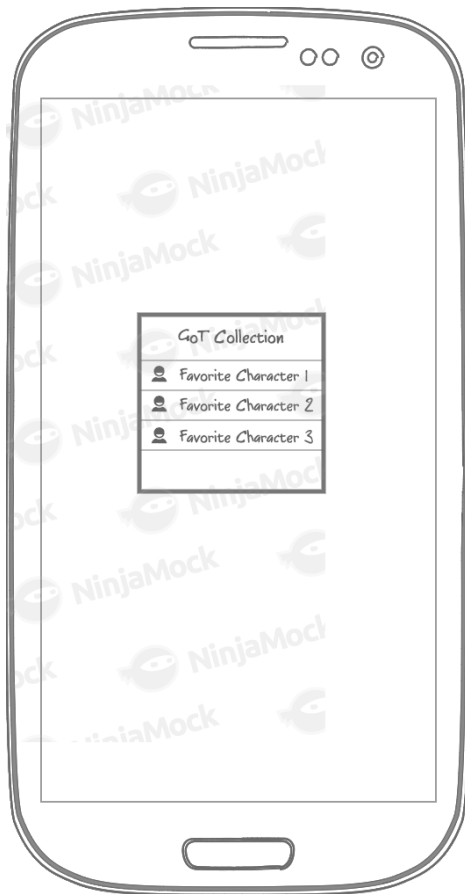
Tela responsável por exibir todos os cards de personagens, o personagem filtrado ou favoritos.

## Tela 2 - DetailsActivity



Tela com os detalhes do personagem selecionado.

### Tela 3 - Widget



Widget na Home do dispositivo, contendo os personagens já favoritados.

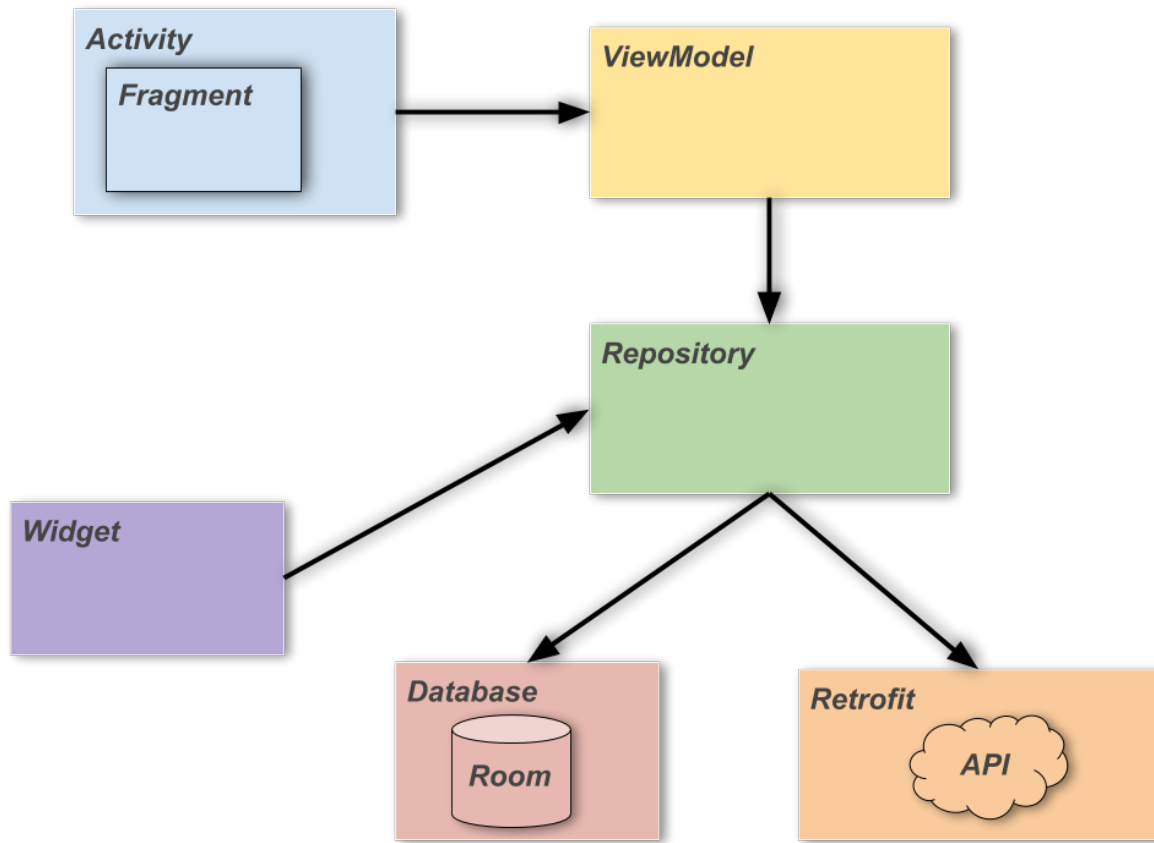
## Principais considerações

### Esquema de persistência de dados.

O acesso às informações dos personagens será de duas formas: consumindo uma API ou consultado um banco de dados local.

Somente os personagens favoritados serão persistidos em banco de dados, por meio da biblioteca Room e utilizando o padrão MVVM. Dessa forma, quando o usuário clicar no botão "favoritar" o objeto será gravado em banco, estando disponível tanto pelo App quanto pelo Widget.

O diagrama abaixo demonstra a relação entre as classes e o método de acesso ao banco:



### Cenário básico de UX.

O cenário básico de utilização do aplicativo se resume em:

- O usuário abre o aplicativo
- É realizada uma busca por um determinado personagem e exibida na tela
- O toque no card do personagem filtrado realiza uma transição para a tela contendo mais detalhes dele
- Clicando no Floating Button da tela de detalhes, o personagem é marcado como favorito
- Se o usuário apertar o botão Home, e na Home ele tiver adicionado o widget do app, o personagem favoritado estará listado junto com os demais itens previamente favoritados
- Ao tocar sobre um destes itens, a tela de detalhes do personagem selecionado é exibida

### Bibliotecas utilizadas.

As bibliotecas utilizadas serão:

- **Picasso**, para carregar imagens da web e caching

- **Retrofit**, consumo de APIs e gerenciamento de requisições
- **Room**, persistência de dados em banco de dados local
- **Parceler**, para serializar e deserializar objetos compartilhados entre as activities

## Utilização dos serviços Firebase.

Os serviços do Google Play Services utilizados serão 2. São eles:

- **Google Analytics para Firebase**, para identificar informações básicas do público que está utilizando a aplicação e quais as buscas mais realizadas
- **Google Admob para Firebase**, para exibir anúncios na versão free do app

## Próximos Passos: Tasks necessárias para concluir o projeto

### Task 1: Setup do Projeto

Criar as dependências das bibliotecas utilizadas no projeto (Retrofit, Picasso, Firebase, Material Design) e habilitar o Data Binding.

### Task 2: Mapear o modelo

Criar o modelo do objeto que será utilizado para mapear os personagens retornados do serviço (<https://api.got.show/doc/>) . O atual objeto retornado da API segue o seguinte modelo: [https://github.com/Rostlab/JS16\\_ProjectA/blob/master/app/models/character.js](https://github.com/Rostlab/JS16_ProjectA/blob/master/app/models/character.js)

### Task 3: Criação do esquema de persistência de dados

Criar o modelo de persistência:

- Classe de banco de dados
- Classe repository que irá consultar o banco
- Interface DAO com as operações necessárias (Listar dados e inserir)
- Modificar o modelo criado anteriormente com a notação @Entity

### Task 4: Configurações da classe Repository

Mapear endpoints da API:

- Retorna todos characters (<https://api.got.show/api/characters/>)
- Retorna character baseado em um nome passado por parâmetro (<https://api.got.show/api/characters/:name>)

Consulta em banco de dados:

- Retorna todos registros salvos na tabela de personagens

- Insere registro

**Obs:** Todas essas funções deverão estar expostas para consumo por meio de uma classe ViewModel

## Task 5: Implementar o layout da MainActivity

Na MainActivity adicionar os componentes:

- TopBar com o botão de busca
- TabLayout com duas opções (Characters e Favorites)
- RecyclerView para listagem dos cards de personagens (criar o layout para o card)

## Task 6: Criação da ViewModel

Adicionar a ViewModel para a MainActivity. Esta classe deverá:

- Consultar o Repository para obter dados da API e banco de dados, que serão utilizados para popular a RecyclerView
- Expor um método para inserir objetos no banco de dados, que será invocado pela Activity

## Task 7: Data Binding na UI

Implementar funções da Activity:

- Popular a RecyclerView de personagens com os objetos retornados da API
- Popular a RecyclerView de personagens favoritos com os objetos salvos no banco de dados
- Possibilitar a busca por um determinado personagem

**Obs:** As chamadas deverão sempre ser realizadas de forma assíncrona, para não bloquear a Thread principal

## Task 8: Criar layout e ViewModel para a DetailsActivity

Criar o layout da activity que irá exibir detalhes do personagem selecionado:

- Permitir que os dados sejam carregados por Data Binding
- Exibir a imagem do personagem selecionado
- Exibir seus detalhes
- Permitir que o clique no botão de favorito salve o objeto em questão no banco de dados (chamar a função da ViewModel)



### **Task 9: Transição de MainActivity e DetailsActivity**

Permitir que as telas realize transições entre si:

- Ao clicar em um card, deverá ser realizado a navegação para os detalhes deste personagem (o objeto transferido deve ser serializado)
- Implementar uma animação básica entre as activities

### **Task 10: Serviços do Firebase**

- Configurar o serviço de Analytics para salvar dados básicos do usuário que está acessando a aplicação
- Configurar o serviço Admob para exibir uma propaganda na tela de detalhes do personagem

### **Task 11: Free e Paid Flavors**

Habilitar um flavor para a versão paga e um para a versão free da app.

A versão paga não deverá exibir propagandas ao usuário (remover dependências desnecessárias).

### **Task 12: Acessibilidade**

Implementar descrições de conteúdo para que o app seja acessível à usuários com deficiência visual

### **Task 13: Widget de personagens favoritos**

Implementação do widget:

- O widget deverá conter uma lista onde irá exibir os personagens marcados como favoritos
- Para popular a lista deverá ser realizada uma consulta ao Repository
- O clique em um item da lista irá exibir a activity com os detalhes do personagem
- Atualizar o botão da activity de detalhes para que quando clicado seja feito um broadcast, para que os itens do widget sejam atualizados