# Unidade 2

## Aula 1 - Especializando nossos Models (ResponseModel)

Já sabemos um pouco da linguagem Swift. Ela parece ser mais fácil do que Java, pois temos menos restrições e mais possibilidades. Como ainda não temos conhecimento sobre a criação de telas iOS começaremos a implementar a camada de Models do nosso aplicativo de filmes SearTV.

Para seguir o padrão adotado no app Android, iremos criar um modelo para a resposta do serviço que chamaremos de ResponseModel e um modelo dos dados que serão exibidos nas Views, que chamaremos de ViewModel.

Como ambos os tipos de models estão diretamente relacionados entre si, deixaremos ambos em uma única struct. Ela terá a seguinte estrutura:

struct MyModel {  
 struct ResponseModel {  
 // propriedades que representarão a resposta do serviço  
 }  
  
 struct ViewModel {  
 // propriedades que serão exibidas na tela  
 }  
}

Como o ResponseModel espelha o que receberemos em um JSON, ou seja, a resposta de um serviço web, todas as suas propriedades deverão espelhar as do JSON recebido. Para isto mostraremos alguns links de referência destes serviços.

### Atividade

Crie todos os ResponseModels do nosso app. Teremos três deles: um para Gêneros, outro para um item da lista de filmes e outro para os detalhes de um filme. Tome como base a documentação do serviço web TMDB (<https://developers.themoviedb.org/3>) utilizando alguns serviços que implementamos no curso de Android:

* Busca de filmes:
  + Populares: <https://developers.themoviedb.org/3/movies/get-popular-movies>
  + Melhores: <https://developers.themoviedb.org/3/movies/get-top-rated-movies>
  + Em breve: <https://developers.themoviedb.org/3/movies/get-upcoming>
* Listar Gêneros: <https://developers.themoviedb.org/3/genres>
* Mostrar detalhes do filme: <https://developers.themoviedb.org/3/movies>

### Solução

Vamos começar com o ResponseModel de Gêneros.

### Models para Gênero

Vamos criar um Response Model para o serviço de **Gêneros**.

Primeiro devemos tomar como referência o JSON que esperamos receber do serviço web. Você pode consultar nesta página:

<https://developers.themoviedb.org/3/genres>

1. Na aba **Try it out** e use a seguinte chave para efetuar o teste:

* 0d2d0307fd89b460e176ba0033dc5c46

1. Clique em **SEND REQUEST** e você obterá um JSON:

{  
 "genres": [  
 {  
 "id": 28,  
 "name": "Ação"  
 },  
 {  
 "id": 12,  
 "name": "Aventura"  
 },  
 {  
 "id": 16,  
 "name": "Animação"  
 },  
 ...  
 ]  
}

1. Para identificar o modelo, basta pegarmos apenas um item do array do JSON acima:

{  
 "id": 28,  
 "name": "Ação"  
 }

Assim podemos identificar que o nosso Response Model de Gênero terá os seguintes atributos:

* id: Um identificador do tipo **Int**
* name: o nome do gênero do tipo **String**

Obs.: O tipo dos campos estão descritos na aba **Definition** do site fornecido.

Agora o que precisamos é implementar nosso ViewModel

1. Crie um novo arquivo com o nome **GenreModels.swift**. Se preferir crie um group com nome **Model** para organizar o projeto.
2. Neste arquivo crie uma nova struct e dê o nome **Genre**:

struct Genre {  
  
}

1. Dentro desta struct crie mais duas structs, uma com o nome ResponseModel e outra com o nome ViewModel (lembrando que ainda não iremos implementar os ViewModels):

struct Genre {  
 struct ResponseModel {  
 }  
  
 struct ViewModel {  
 }  
}

1. Tendo como referência o JSON visto em passos anteriores crie, na struct ResponseModel, os atributos com nomes idênticos aos obtidos no JSON. Seus tipos devem ser compatíveis com os tipos do JSON também:

struct Genre {  
 struct ResponseModel {  
 let id: Int?  
 let name: String?  
 }  
  
 struct ViewModel {  
 }  
}

DICA: Como boa prática, deixamos todos os atributos dos nossos models como **Optional**. Isto nos força a tratar a nulidade em um uso posterior e evitar crashes no aplicativo.

Temos o nosso modelo.

Faça isto para todos os possíveis models do nosso projeto.

### Lista de Filmes

Para a lista de filmes teremos três filtros: POPULAR, MELHORES e EM BREVE

A referência para estes serviços estão nos links abaixo:  
-**Lista de filmes populares:** <https://developers.themoviedb.org/3/movies/get-popular-movies>  
-**Lista dos melhores filmes:** <https://developers.themoviedb.org/3/movies/get-top-rated-movies>  
-**Lista com lançamento em breve:** <https://developers.themoviedb.org/3/movies/get-upcoming>

Veja que o JSON dos três serviços atendem à mesma estrutura então precisamos apenas de um Response Model.

1. Crie um arquivo com o nome MovieModels.swift
2. Insira a struct nele:

struct Movie {  
 struct ResponseModel {  
 let page: Int?  
 let results: [Result]?  
 let total\_results: Int?  
 let total\_pages: Int?  
   
 struct Result {  
 let poster\_path: String?  
 let adult: Bool?  
 let overview: String?  
 let release\_date: String?  
 let genre\_ids: [Int]?  
 let id: Int  
 let original\_title: String?  
 let original\_language: String?  
 let title: String?  
 let backdrop\_path: String?  
 let popularity: Double?  
 let vote\_count: Int?  
 let video: Bool?  
 let vote\_average: Double?  
 }   
 }  
   
 struct ViewModel {  
 }  
}

### Detalhe do Filme

Faça o mesmo procedimento para o Response Model de detalhes do filme:

**Link para referência:** <https://developers.themoviedb.org/3/movies/get-movie-details>

1. Crie um arquivo chamado **MovieDetailsModels.swift**
2. Insira a seguinte struct nele:

struct MovieDetails {  
 struct ResponseModel {  
 typealias GenreModel = Genre.ResponseModel  
   
 let backdrop\_path: String?  
 let budget: Int?  
 let genres: [GenreModel]?  
 let id: Int  
 let original\_title: String?  
 let overview: String?  
 let vote\_average: Double?  
 let revenue: Int?  
 let title: String?  
 let release\_date: Date?  
 let runtime: Int?  
 }  
   
 struct ViewModel {  
 }  
}

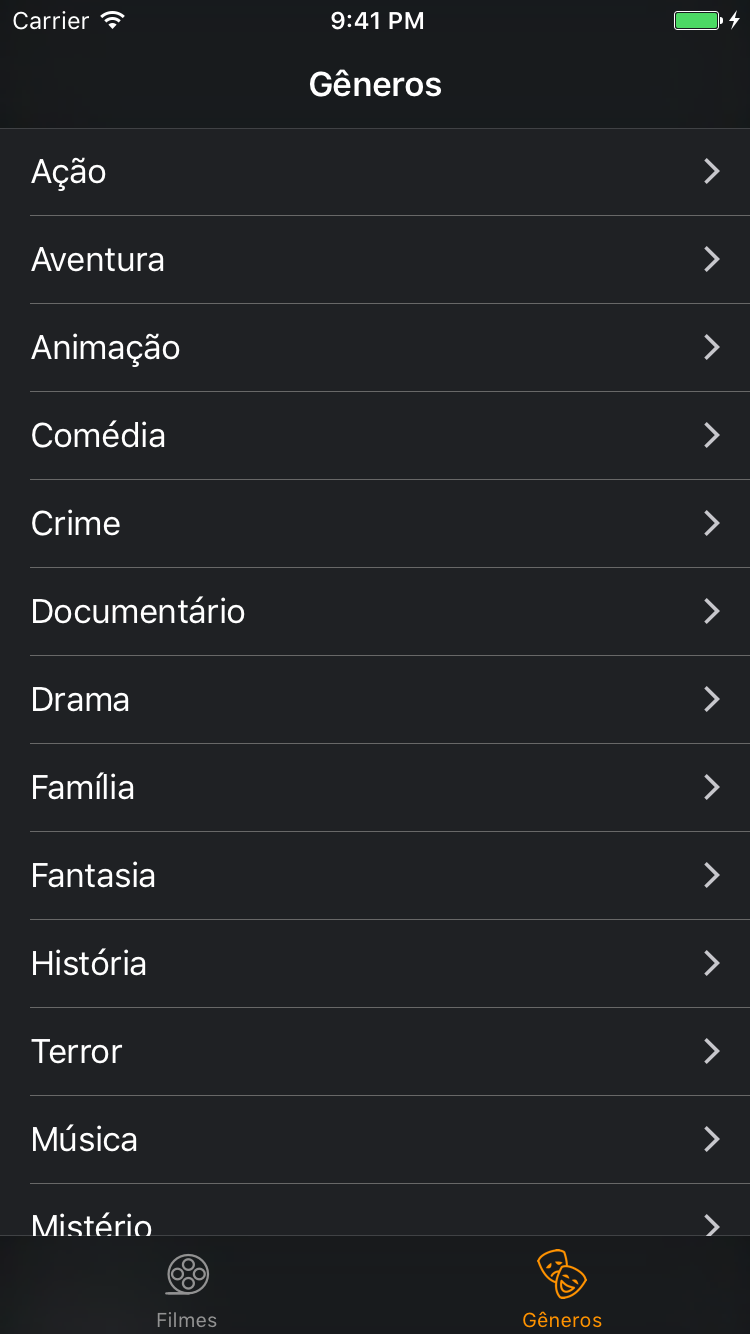
Repare que aqui estamos reutilizando o Response Model de Gênero. Apenas demos um apelido a ele.

## Aula 2: Especializando nossos Models (ViewModel)

Agora que aprendemos o básico sobre classes e structs no Swift iremos criar models que representarão os dados a serem apresentados na tela. Isto diminuirá a carga dos nossos Controllers.

Esta representação de dados chamaremos de ViewModel (pois se trata do modelo a ser apresentado na camada View). Geralmente as propriedades de um ViewModel são do tipo String, pois irão ser apresentadas por Labels, Text Fields podem ser utilizados como URL de imagens.

A maneira de identificar as propriedades do ViewModel é analisando o design das telas. Veja a tela de Gêneros:

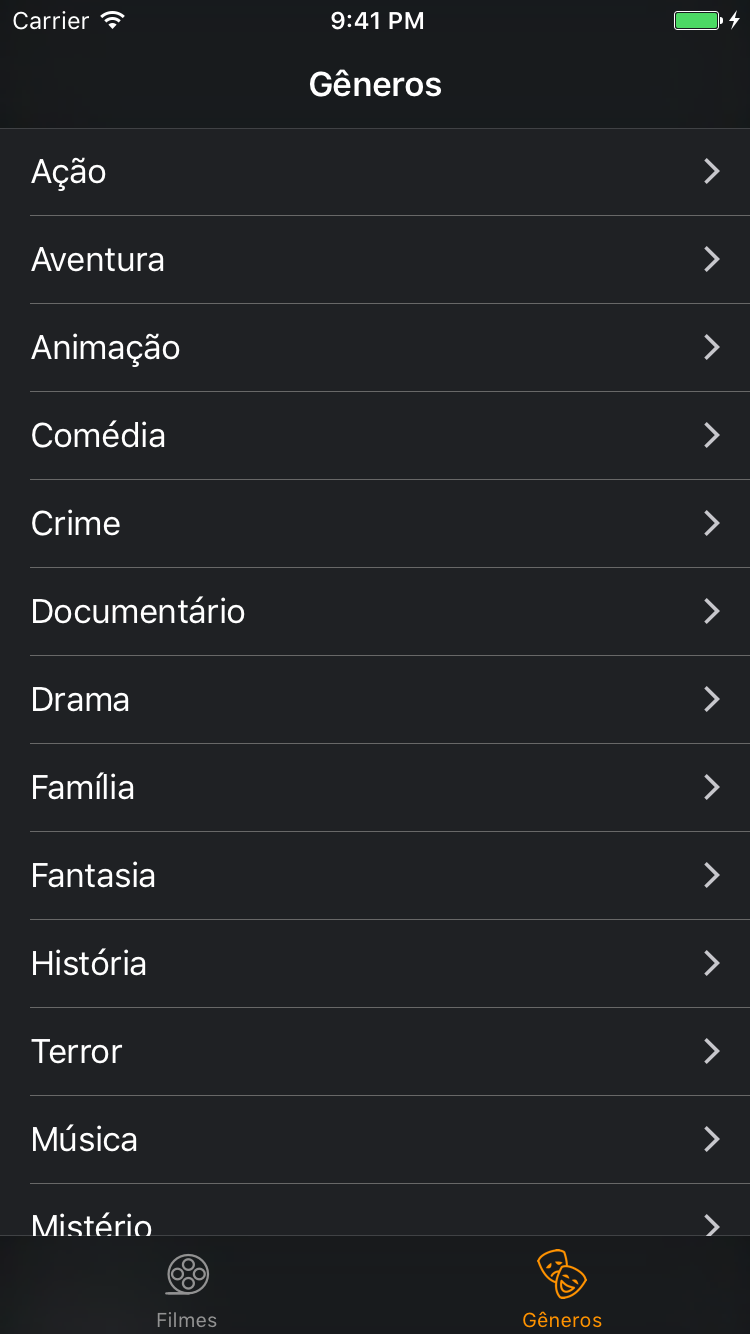


Veja que a unica informação que exibimos é o **nome do gênero** e ele é teoricamente uma Label. Então nosso View Model deve conter o atributo **name** do tipo **String**.

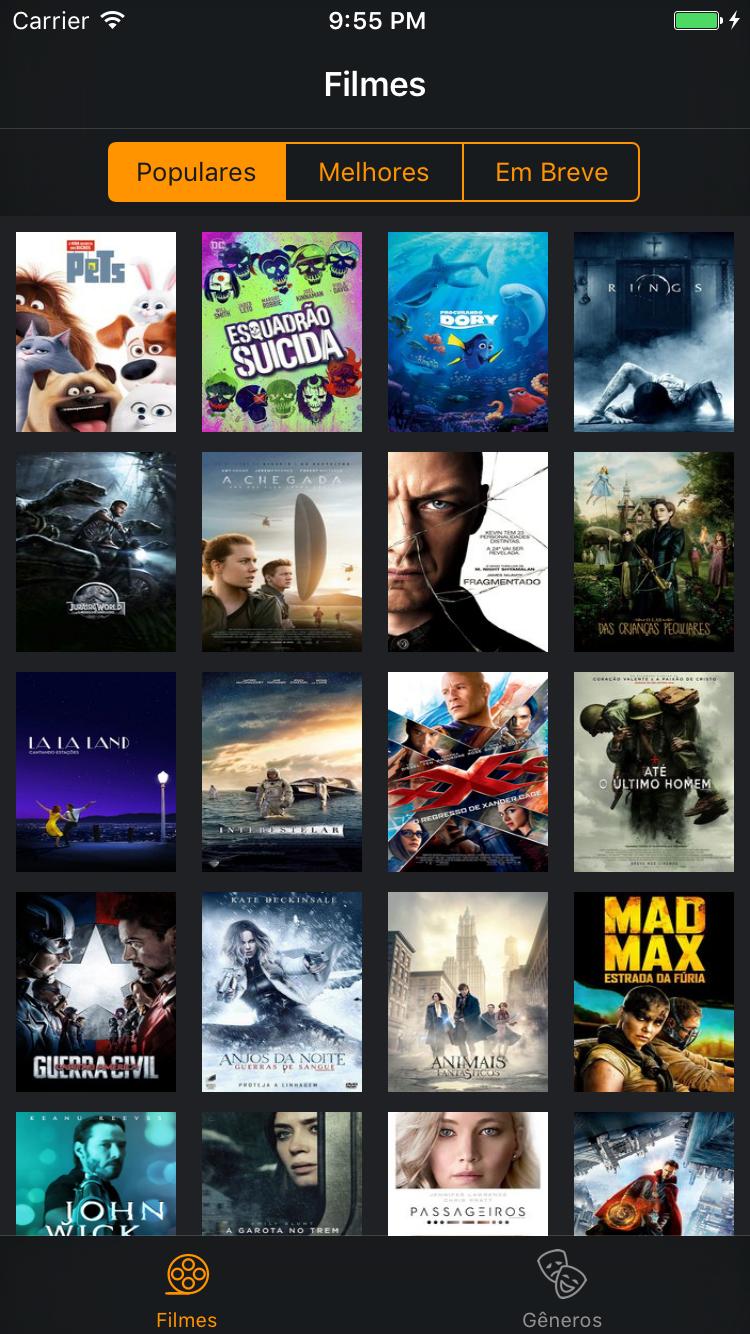
### Atividade

Crie os ViewModels das três do nosso app.

* Gêneros



* Filmes



* Detalhes do Filme



### Solução

Na aula passada implementamos os Response Models, que se tratam da representação dos dados oriundos do serviço web. Agora criaremos a representação dos dados que serão exibidos na tela.

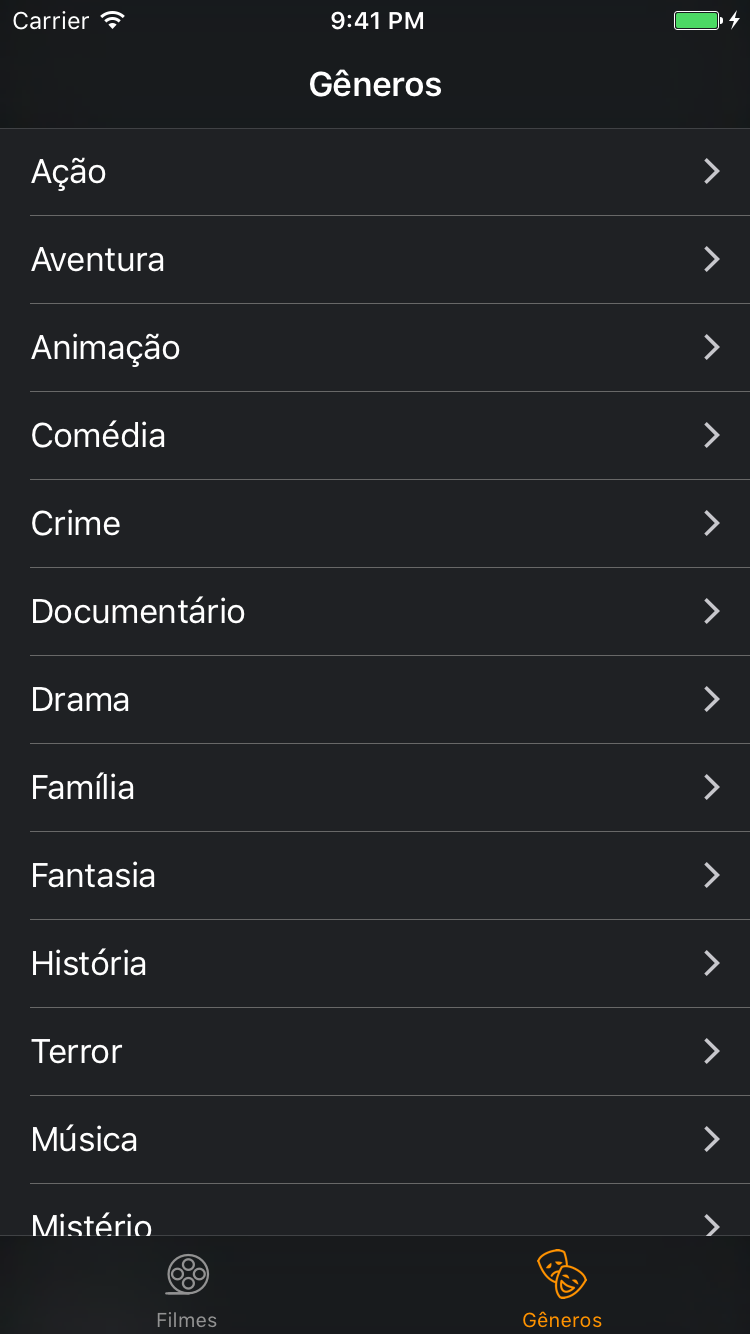
A maioria dos valores de View Model serão do tipo String, e o intuito de fazermos isso é livrar a carga de conversão da ViewController.

Teoricamente temos que obter os dados do serviço e convertê-los para uma versão apresentável na nossa tela. Estes dados apresentáveis serão os View Models e a ViewController terá apenas o trabalho de exibí-lo. Quem fará a conversão dos valores serão outras entidades que criaremos no projeto futuramente.

Para criarmos nossos View Models seguiremos a mesma lógica dos Response Models, porém nossa fonte será o design da tela. Vejamos com a tela Gênero.

#### ViewModels de Gênero

Primeiramente devemos analisar o design da tela:



Veja que a unica informação que exibimos é o **nome do gênero** e ele é teoricamente uma Label. Então nosso View Model deve conter o atributo **name** do tipo **String**.

Então o ViewModel do nosso arquivo **GenreModels** ficaria assim:

struct ViewModel {  
 let name: String  
 }

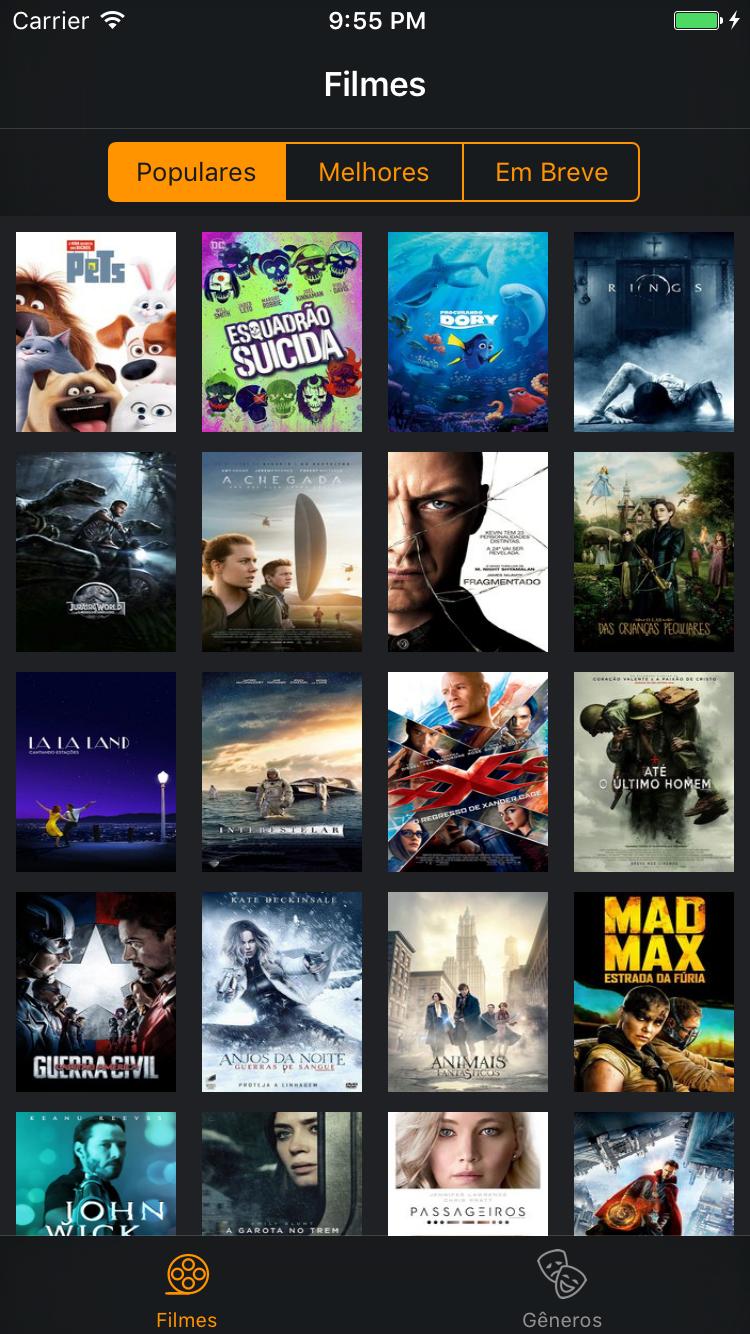
e a struct completa ficará assim:

struct Genre {  
 struct ResponseModel {  
 let name: String?  
 let id: Int?  
 }  
   
 struct ViewModel {  
 let name: String  
 }  
}

Faça isto para todas as telas

#### Lista de Filmes

Veja o design:



Como fizemos no JSON de Gêneros, precisamos reduzir as informações. Nesta tela vemos uma coleção de imagens, então o View Model terá apenas a URL da imagem que queremos exibir.

Nas aulas seguintes aprenderemos como obter as imagens do servidor.

A implementação do View Model ficará assim:

struct ViewModel {  
 let id: Int  
 let posterPath: String  
 }

Veja que há um **id** na struct. Precisaremos dele pois ao clicar em uma célula devemos enviar o id do filme selecionado para a cena seguinte para que assim ela possa utilizá-lo para buscar os detalhes do filme.

#### Detalhes de Filmes

Veja o design:



A implementação dos ViewModels ficará assim:

struct ViewModel {  
 let title: String  
 let originalTitle: String  
 let backDropPath: String  
 let voteAverage: Int  
 let overview: String  
 let revenue: String  
 let budget: String  
 let releaseDate: String  
 let genre: String  
 let runtime: String  
 }

Temos então todos os View models implementados. Na aula que vem iremos criar initializers para estes Models.