UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

PRÓ-REITORIA DE GRADUAÇÃO

CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUCAS GOMES DE ARAÚJO

MATHEUS GOES RIBEIRO

MOSSORÓ

2021

1. Introdução

REST (*Representational state transfer*) é um estilo de arquitetura destinada ao desenvolvimento de aplicações em rede. Tal tecnologia promove uma facilidade de comunicação entre os serviços devido a separação entre o servidor, aquele que armazena os recursos, e o cliente, que irá consultar, alterar e consumir essas informações através de uma requisição às rotas HTTP disponibilizadas pela aplicação REST.

O protocolo HTTP possui métodos que determinam o tipo de interação que o cliente poderá fazer com o servidor. Os mais comumente usados, dentro destes, são os GET, POST, PUT e DELETE, que se assemelham àqueles que regem um banco de dados.

Agora, em questão de funcionalidades dos métodos HTTP, o método GET é aquele que tem como função retornar algum recurso específico, armazenado no servidor, ao cliente que fez a requisição. O método POST é destinado para aquelas interações que podem alterar o estado do servidor, seja inserindo algum novo recurso ou realizando, por exemplo, uma autenticação de um usuário. Para a atualização de alguma propriedade de um recurso armazenado no servidor, utiliza-se o método PUT. E, finalmente, o método DELETE tem como função remover uma informação, especificada pelo cliente, do servidor.

Parte dos recursos armazenados no servidor, geralmente, estão dentro de um banco de dados, no qual existem tabelas e colunas que ditam, respectivamente, o tipo do recurso e as suas propriedades. A gestão desses bancos é feita a partir de um conjunto de softwares, denominado SGBD (Sistema de gerenciamento de banco de dados).

Para uma maior abstração e, consequentemente, uma maior facilidade, acessibilidade e produtividade no desenvolvimento de um software, foram elaboradas, por engenheiros de software, as Frameworks, que são conjuntos de funcionalidades capazes de gerenciar as camadas de baixo nível de uma aplicação, de forma a permitir com que o desenvolvedor trabalhe, geralmente, apenas com a camada alto nível.

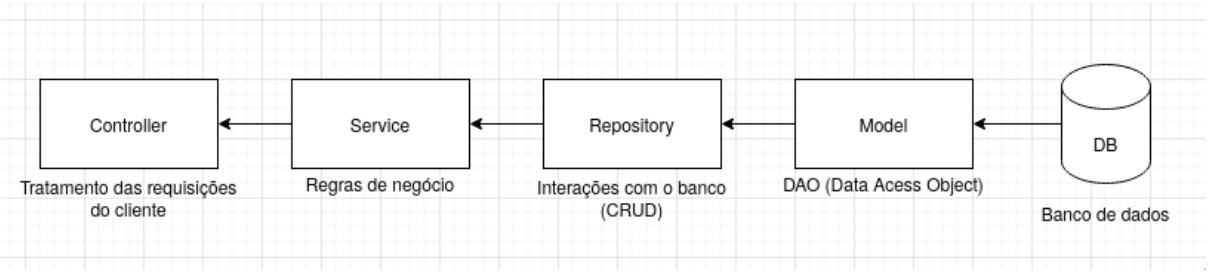
2. Objetivo

Desenvolver uma API REST que irá se conectar com o banco de dados para prover rotas de deleção, persistência e retorno das informações acerca do mundo dos filmes e que, posteriormente, poderá ser consumida por uma aplicação cliente para criar um catálogo visual.

3. Arquitetura

O SGBD utilizado será o Postgresql e os frameworks para o funcionamento do projeto serão o JPA/Hibernate, que ajudará no gerenciamento do banco através das suas anotações, classes, interfaces e métodos CRUD (*create, read, update* e *delete*), e Spring boot, que fornecerá meios de inicializar a aplicação com as configurações necessárias para a criação e requisição de rotas que utilizarão os métodos HTTP: GET, POST, PUT e DELETE.

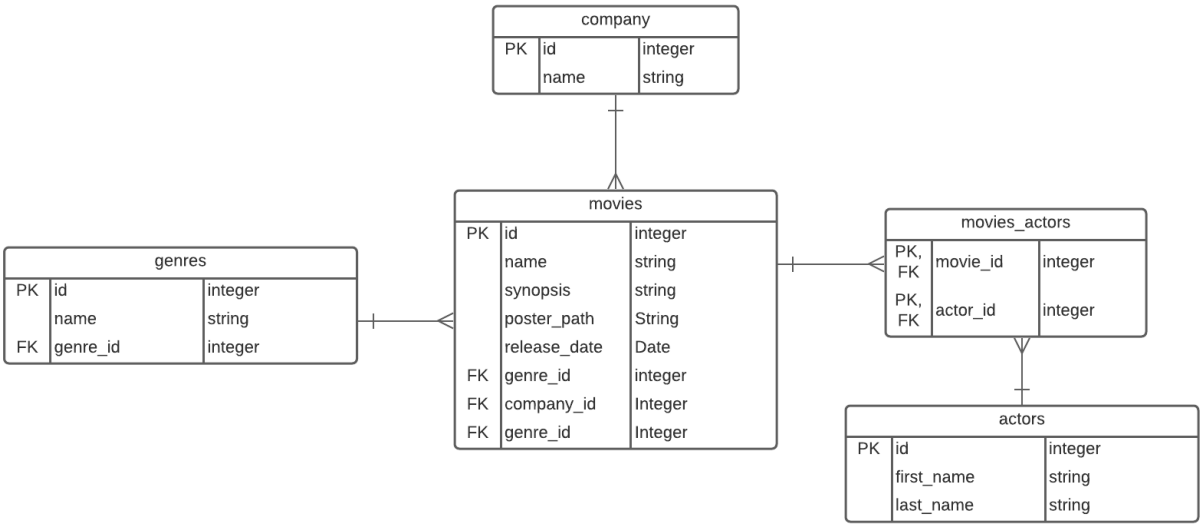
Figura 1: Modelo do projeto



O formato do projeto obedecerá o modelo no qual são contemplados os pacotes *controller*, *service*, *repository* e *model* (figura 1). Estarão contidas, no pacote *model*, as entidades que se comunicam com o banco e possuem atributos. Em *repository* estarão as interfaces que possuem os métodos anotados por uma query específica, seja esta de persistência, atualização, deleção ou seleção. O pacote *service* será consistido por classes que irão implementar métodos que invocarão aqueles do *repository* e, a partir destes, irão realizar regras de negócio a fim de moldar o dado obtido e/ou tratar os possíveis erros provenientes da interação com o banco. O pacote *controller* será composto por classes que possuem métodos

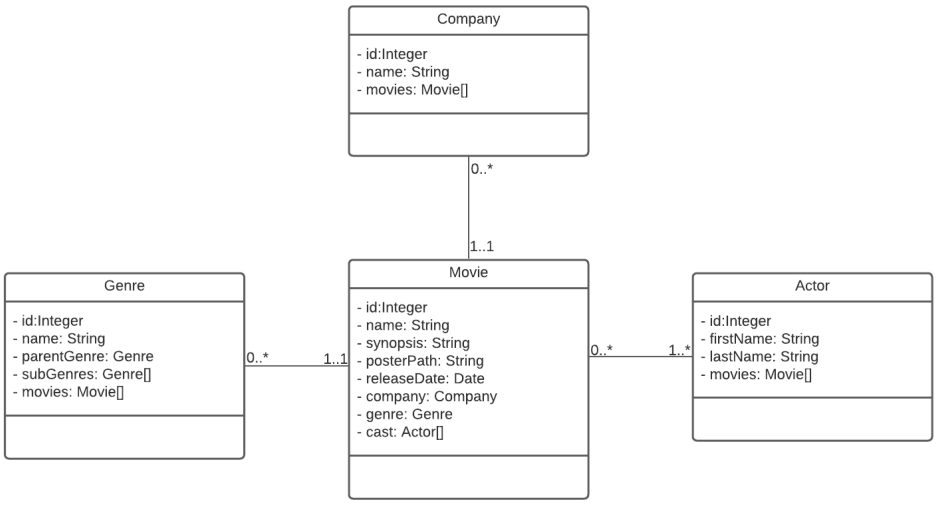
que tratam a requisição vinda de um cliente com a invocação de um método de uma classe específica da camada *service*.

4. Modelagem Entidade-Relacionamento do banco de dados



5. Modelagem UML das entidades

As classes que irão se conectar com o banco de dados e que consequentemente terão atributos, serão:



Os arrays presentes nas entidades serão implementados utilizando a interface Set, já que não existem duplicatas de um mesmo elemento no banco de dados devido a unicidade da chave primária. Como classes concretas, caso a ordenação dos elementos não seja necessária, será usado o HashSet. Agora, caso a ordem dos itens seja importante para o funcionamento do serviço, como uma listagem, numa certa ordem, imutável dos gêneros existentes em um menu árvore, a classe concreta utilizada será a TreeSet.

6. Rotas

As rotas que irão se comunicar com o banco de dados, inicialmente, para cada entidade, serão:

Tabela 1: rotas para as corporações

Company	Método	Descrição
/**companies**	GET	retorna todas as corporações
/**company**/{id}	GET	retorna a corporação que possui o id especificado
/**company**	POST	persiste uma nova corporação no banco
/**company**/{id}	PUT	atualiza uma corporação
/**company**/{id}	DELETE	remove uma corporação do banco
/**company**/{id}/movies	GET	retorna todos os filmes de uma corporação específica
/**company**/{id}/add_movie	POST	adiciona um filme a corporação especificada

Tabela 2: rotas para os filmes

Movie	Método	Descrição
/**movies**	GET	retorna todos os filmes
/**movie**/{id}	GET	retorna o filme que possui o id especificado
/**movie**	POST	persiste um novo filme no banco
/**movie**/{id}	PUT	atualiza um filme
/**movie**/{id}	DELETE	remove um filme do banco
/**movie**/{id}/cast	GET	retorna todos os atores do filme especificado
/**movie**/{id}/add_actor	POST	adiciona um ator ao filme especificado

Tabela 3: rotas para os atores/atrizes

Actor	Método	Descrição
/**actors**	GET	retorna todos os atores
/**actor**/{id}	GET	retorna o ator que possui o id especificado
/**actor**	POST	persiste um novo ator no banco
/**actor**/{id}	PUT	atualiza um ator
/**actor**/{id}	DELETE	remove um ator do banco
/**actor**/{id}/movies	GET	retorna todos os filmes do ator especificado

Tabela 4: rotas para os gêneros

Genre	Método	Descrição
/**genres**	GET	retorna todos os gêneros
/**genre**/{id}	GET	retorna o gênero que possui o id especificado
/**genre**	POST	persiste um novo gênero no banco
/**genre**/{id}	PUT	atualiza um gênero
/**genre**/{id}	DELETE	remove um gênero do banco
/**genre**/{id}/movies	GET	retorna todos os filmes do gênero especificado
/**genre**/{id}/subgenres	GET	retorna todos os subgêneros do gênero especificado
/**genre**/{id}/subgenre	POST	adiciona um novo subgênero ao gênero especificado

A criação de novas rotas virá com a implementação de outras estruturas que, além daquelas usuais das classes entidade ou dos métodos do repositório, irão servir funcionalidades específicas, como as interfaces Queue e Map e a classe Stack.