Ghibli

Este projeto foi desenvolvido entre 01 e 03 de julho de 2019 como um teste técnico para a GBLIX. O projeto, desenvolvido com o framework Laravel, consistem em uma API que fornece dados a respeito dos personagens do mundialmente conhecido Estúdio Ghibli. Os dados são providos pela API https://ghibliapi.herokuapp.com/.

Instalação

Como um projeto desenvolvido em Laravel 5.8 é necessário que o ambiente de testes atenda aos requisitos do framework. Feito isso, configure o arquivo `.env` na raiz do projeto com as informações a respeito da conexão com o seu banco de dados.
O banco de dados desse sistema também possui requisitos para que o sistema possa operar. Certifique-se de criar o banco `ghibli`, para que o sistema possa armazenas as tabelas. Se você está usando MySQL, use o comando `php artisan mysql:createDB` para criar o banco automaticamente.
Execute o comando `php artisan migrate` para criar as tabelas necessárias para o funcionamento do sistema. Você pode preencher as tabelas recém criadas com dados randômicos usando o comando: `php artisan db:seed`, ou, pode preencher com os dados reais, para isso invoque o Crawler através do comando: `php artisan api:crawl`.

API

Para usar a API desse sistema acesse `http://localhost/pessoas?fmt=html`. Serão exibidos os dados devidamente extraídos e formatados em uma tabela HTML. Alterando o parâmetro fmt para *json* ou *csv* os dados serão exibidos em JSon e CSV respectivamente ao invés do HTML.
Use o parâmetro **sort** para ordenar a tabela mostrada de acordo com uma das colunas. Os valores válidos são:

- *name*. Nome do personagem.
- *age*. Idade do personagem.
- *movieName*. Nome do filme.
- *movieYear*. Ano de lançamento do filme.
- *rtScore*. Pontuação no Rotten Tomatoes.

Combinando o parâmetro **sort** com **order=asc** ou **order=desc** você poderá alternar entre crescente e decrescente respectivamente.
Exemplo: `http://localhost/pessoas?fmt=html&sort=movieYear&order=asc`.

Também é possível filtrar uma coluna por um valor especifico, para isso use o parâmetro **filter** da seguinte forma: **filter=chave:valor**.
Por exemplo: `http://localhost/pessoas?fmt=html&filter=movieYear:2002`.

O Crawler

Para obter os dados a respeito dos filmes e personagens um web crawler foi desenvolvido, seu código está contido na classe Crawler no

arquivo `app/Crawler.php`. O Crawler executa uma busca na API por todos os filmes e personages, em seguida, insere estes dados no banco. Por fim, a função `relationate($movies, $characters)` preenche a tabela de relacionamento: `movies_characters` dessa forma criando um relacionamento muitos para muitos, onde cada personagem pode estar em mais de um filme e, cada filme pode ter múltiplos personagens.
O crawler pode ser executado pelo comando `php artisan api:crawl` ou via agendamento schedule a cada duas horas, apenas certifique-se de configurar o cron do sistema operacional para executar os scredules do Laravel.

O Banco de Dados

Um banco de dados SQL é responsável por guardar e relacionar as informações do sistema. As classes responsáveis por gerenciar a criação do esquema do banco estão contidas na pasta `database`. Existem dois modelos de dados implementados em `App/Model`, um para os filmes e outro para os personagens, além disso, factores para estes modelos foram implementadas em `database/factories/` estas factores permitem ao seeder do banco de dados criar dados fictícios e preencher o banco para testes.

Testes

O Laravel possue uma configuração nativa para o uso do phpunit.
Alguns testes para este sistema foram desenvolvidos e estão presentes das subpastas do diretório `tests`.
`tests/Feature/ResponseTest.php` guarda os testes responsáveis por

checar se a API do sistema está respondendo conforme esperado.

Nota: Se o banco estiver vazio testes falharão.

`tests/Unit/DBTest.php` guarda os testes unitários para checar se o banco contém dados. Este teste falha se o banco não estiver preenchido.

Controller

Este sistema possui apenas uma rota, sendo esta a GET `/pessoas`, esta rota é totalmente gerenciada pelo controller `app/Http/Controllers/PessoasController.php` este controller é responsável por responder a requisição formatando os dados conforme solicitado pelo parâmetro `fmt`, além de aplicar a ordenação e o filtro solicitados na requisição. Caso o formato *html* seja solicitado o controller responderá com uma view implementada com auxilio do blade, esta view pode ser encontrada em: `resources/views/pessoas.blade.php`.

Autor

Este sistema foi desenvolvido por Matheus Alves.

contatos: https://matheusalves.com.br