Leia-me

Este projeto tem como o intuito de estudar estruturas de arquitetura de software aplicando padrões de projeto e arquitetando acesso a banco de dados e injeções de dependências em rotas.

O mini framework como eu o chamarei a partir de agora foi criado por mim nos últimos dias, inclusive o acesso a banco de dados, gerenciamento de rotas e injeção de dependências em métodos.

Como instalar:

- Execute o arquivo docker-compose com o comando *docker-compose up*
- Entre no container com o comando *docker exec -i -t <container_name> /bin/bash*, onde container_name é o nome que o ambiente criou para o container
- Estamos neste ponto na pasta raiz do projeto xampp, entre na pasta school (ou outra pasta do nome de projeto desejado), dentro da pasta /school do projeto raiz execute *composer install*
- Entre no link http://127.0.0.1:8888/phpmyadmin e crie 2 bancos, school e school_testing, (dois nomes escolhidos pelo usuário)
- Execute o dump school.sql dentro dos bancos para criar as tabelas, tanto no ambiente quando no banco de testes
- Caso seja necessário, edite o arquvo *.env.yml* e *.env.test.yml* apontando para os bancos criados e com os dumps.
- no arquivo na pasta *raiz school-requests.json* temos as requisições, este arquivo está preparado para ser acessado via insomnia (https://insomnia.rest/download) e mostra as requisições possíveis para o projeto.

Como rodar os tests:

- Após o projeto está rodando com sucesso, entre no container docker com o comando *docker exec -i -t <container_name> /bin/bash*
- Entre na pasta /school
- Execute o comando *composer install*
- Execute o script *./vendor/bin/pest*

Considerações importantes

- Arquitetura e responsabilidades:
  A estrutura foi pensada para separar regras de negócio da camada de infraestrutura. A pasta Core contém elementos genéricos e reaproveitáveis, enquanto as entidades, controladores e rotas podem ser criados de forma simples e rápida para atender às necessidades da aplicação.

- Injeção de dependências:
  O projeto suporta injeção automática de dependências em métodos de controladores. Esse mecanismo foi inspirado em frameworks consagrados, como o Laravel, mas implementado de forma manual, servindo como um exercício prático de reflexão sobre o tema.

- Acesso a dados:
  A BaseRepository atua como um mini query builder. Durante o desenvolvimento, percebi a oportunidade de evoluir para uma camada mais robusta, separando melhor o repositório do acesso direto ao banco de dados por meio de um QueryBuilder, o que também abriria espaço para o uso de padrões como Proxy.

- Serviços auxiliares:
  O PictureUploadService exemplifica a injeção de serviços. Embora simples, ele poderia ser facilmente substituído por integrações reais, como upload para S3 ou outro provedor.

- Rotas:
  No momento, as rotas aceitam apenas os métodos GET e POST, mas a estrutura foi planejada para facilitar a expansão e incluir outros verbos HTTP futuramente.

- Testes e validação:
  A suíte de testes foi implementada com Pest. Embora não esteja completa, já cobre cenários suficientes para demonstrar a organização e a viabilidade da abordagem. O validador também foi escrito de forma enxuta, inspirado em implementações anteriores que desenvolvi em Node.js.

Em resumo, este projeto é um mini framework experimental, desenvolvido com foco em aprendizado e em demonstrar minha capacidade de estruturar soluções backend, aplicando conceitos de arquitetura limpa, desacoplamento e extensibilidade.