# Documentação do Projeto - API RESTful Laravel

## 1. Arquitetura do Software

O projeto segue a arquitetura MVC (Model-View-Controller) utilizando o framework Laravel.

* https://github.com/REnamoto/api\_rest

Para representar visualmente a arquitetura e os fluxos do sistema, foi criado um diagrama no Draw.io. O diagrama descreve:

* A interação entre os componentes da aplicação;
* Os fluxos de comunicação entre o usuário, a API e o banco de dados;
* Os principais módulos e entidades (Clientes, Produtos, Pedidos, Estoques).

Figura 1 – C4 Model nível 1

A diagram of software system

AI-generated content may be incorrect.

Figura 2 – C4 Model nível 2

A diagram of a api

AI-generated content may be incorrect.

Figura 3 – C4 Model nível 3

A diagram of a company

AI-generated content may be incorrect.

Figura 4 – C4 Model nível 4

A diagram of a company

AI-generated content may be incorrect.

## 2. Estrutura de Pastas do Projeto (MVC)

A aplicação foi estruturada seguindo o padrão de diretórios do Laravel, organizando os arquivos de acordo com a arquitetura MVC. Abaixo está a descrição dos principais diretórios:

├── app  
│ ├── Http  
│ │ ├── Controllers # Controladores da aplicação (lógica de controle)  
│ │ └── Middleware # Middlewares HTTP  
│ ├── Models # Representação das entidades do banco de dados (regras de negócio)  
│ └── Providers # Configurações dos providers da aplicação  
├── database  
│ ├── migrations # Arquivos de migração para criar as tabelas  
│ └── seeders # Dados iniciais para o banco (opcional)  
├── routes  
│ └── api.php # Arquivo de rotas para a API  
├── resources  
│ └── views # Arquivos de visualização (utilizado para o Swagger UI)  
├── public # Pasta pública para acesso web  
└── storage # Logs e arquivos gerados (inclui documentação Swagger)

## 3. Explicação da Estrutura e Elementos do Código

### a) Models (Regras de Negócio)

Os arquivos da pasta app/Models representam as entidades do sistema e contêm as regras de negócio relacionadas.

* Clients.php - representa os clientes da empresa
* Products.php - representa os produtos da empresa
* ProductsStocks.php - representa os estoques dos produtos da empresa
* Orders.php - além de representar pedidos, contém lógica para controle de estoque (métodos como createWithStockControl(), updateWithStockControl() e deleteAndRestoreStock())

### b) Controllers (Orquestração das Requisições)

Estão em app/Http/Controllers. Recebem as requisições HTTP, validam dados, e chamam os métodos do Model.

* ClientsController - CRUD de clients;
* ProductsController - CRUD de produtos;
* ProductsStocksController - CRUD de estoques de produtos;
* OrdersController - CRUD de pedidos, com controle de estoque.

### c) Rotas

Definidas em routes/api.php. Utiliza Route::apiResource para definir as rotas REST. Exemplo:

* Route::apiResource('clients', ClientsController::class);
* Route::apiResource('products', ProductsController::class);
* Route::apiResource('products\_stocks', ProductsStocksController::class);
* Route::apiResource('orders', OrdersController::class);

### d) Swagger (Documentação da API)

A documentação foi gerada com o pacote l5-swagger, utilizando anotações **@OA** diretamente nos métodos dos controllers. A interface pode ser acessada via:

* http://localhost/api/documentation

Nela é possível testar os endpoints e visualizar todos os detalhes da API.