Relatório do Projeto Final (RPF)

1. Correção de Erros na Criação da Base de Dados

Durante os testes ao código de criação da base de dados no MySQL, encontrámos apenas um erro, que surgiu repetidamente em dois pontos distintos. Este erro estava relacionado com a definição do atributo data, onde inicialmente utilizámos:

```
data DATE NOT NULL DEFAULT CURRENT_DATE
```

Contudo, ao executar o script no MySQL, verificámos que esta sintaxe gerava erro, uma vez que o uso de DEFAULT CURRENT_DATE para colunas do tipo DATE já não é suportado em algumas versões mais recentes do MySQL. A correção adequada consistiu em alterar o tipo de dados para DATETIME e utilizar CURRENT_TIMESTAMP como valor por defeito:

```
data DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
```

Este ajuste permitiu à base de dados registar automaticamente a data e hora atuais no momento da inserção de novos registos, sem causar conflitos de sintaxe.

2. Adaptações no Modelo Relacional

Durante o desenvolvimento do projeto, foi necessário realizar algumas adaptações ao modelo relacional inicialmente planeado, de forma a garantir a compatibilidade com o LoopBack 4, que foi utilizado para a criação da API. Embora o modelo estivesse corretamente normalizado desde o início, com todas as tabelas estruturadas até à 3ª Forma Normal (3FN), algumas das decisões teóricas não foram práticas do ponto de vista da implementação técnica.

Uma das alterações foi a simplificação da representação dos números de telemóvel. Inicialmente, o modelo previa uma tabela separada chamada "TELEMOVEL", associada ao cliente através de uma relação de 1 para N, uma vez que cada cliente poderia ter mais do que um número. No entanto, o LoopBack não lida bem com atributos multivalorados representados dessa forma, o que causava complicações na geração automática dos endpoints. Por esse motivo, decidimos incorporar o número de telemóvel diretamente na tabela "CLIENTE", restringindo temporariamente o sistema a um número por cliente, o qual foi posteriormente atualizado no próprio diagrama e no respetivo relatório. Esta decisão foi feita para garantir a simplicidade e o bom funcionamento da API, sem comprometer a integridade ou coerência dos dados.

Outra alteração relevante foi a reestruturação da relação "incluido_em", que representava a associação entre produtos e encomendas. Inicialmente modelada como uma relação com atributos, essa estrutura causava limitações técnicas na manipulação de dados via LoopBack. Para resolver esse problema, decidimos transformar essa relação numa entidade autónoma chamada "ITEMENCOMENDA". Esta entidade armazena o preço unitário e a quantidade de cada produto incluído numa encomenda, e está associada tanto à tabela "PRODUTO" como à tabela "ENCOMENDA" através de chaves estrangeiras.

Contudo, como o LoopBack também não lida bem com entidades fracas (aquelas que dependem totalmente de outras chaves para a sua identificação), fomos obrigados a adicionar

um atributo artificial de identificação (um id autogerado) na tabela "ITEMENCOMENDA". Este campo não tem utilidade lógica no contexto do modelo relacional puro, mas foi incluído apenas para satisfazer as exigências técnicas do LoopBack e simplificar a criação dos modelos e endpoints correspondentes. Apesar disso, todas as ligações entre encomendas e produtos continuam a ser devidamente representadas e a lógica do sistema mantém-se correta.

3. Melhorias na Normalização e nas Restrições

Para além da correção dos erros, aprofundámos a explicação do processo de normalização, garantindo que a estrutura da base de dados segue os princípios até à terceira forma normal. Também detalhámos a implementação das restrições no MySQL, como as chaves primárias, chaves estrangeiras, restrições de unicidade, e validações com NOT NULL e CHECK.

Durante a conceção e implementação da base de dados, foram tomadas várias decisões técnicas com base na coerência, integridade e desempenho. Por exemplo, inicialmente optámos por utilizar o tipo de dados BIGINT para chaves primárias em várias tabelas com a intenção de permitir escalabilidade futura. No entanto, ao detetar incompatibilidades com colunas que usavam INT, foi necessário padronizar os tipos de dados para garantir a compatibilidade nas restrições de chave estrangeira. Devido a estas melhorias e às correções feitas, efetuámos ainda alguns ajustes no próprio código DDL em SQL, assegurando que toda a estrutura da base de dados fosse coerente e funcional.

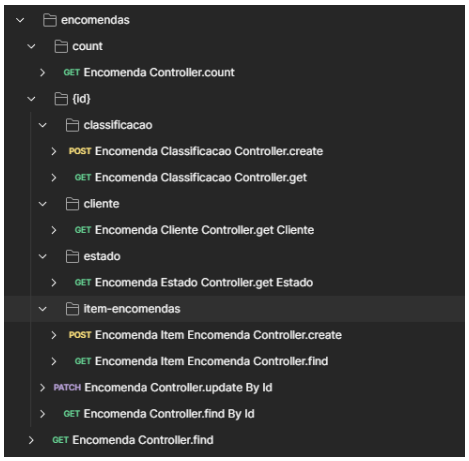
4. Dificuldades Encontradas

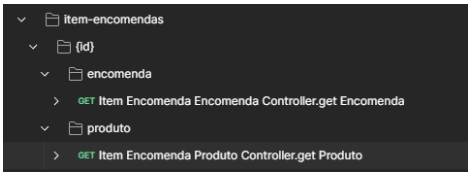
Durante o desenvolvimento deste projeto, enfrentámos algumas dificuldades, especialmente relacionadas com a definição correta das restrições e tipos de dados nas tabelas da base de dados em MySQL. Um dos principais desafios foi garantir a compatibilidade entre os campos relacionados por chaves estrangeiras, o que exigiu vários ajustes e testes no código DDL. Também tivemos dificuldades iniciais na importação do ficheiro SQL para o servidor através da ferramenta de Data Import do MySQL, assim como na gestão de erros menos explícitos que exigiram alguma pesquisa adicional.

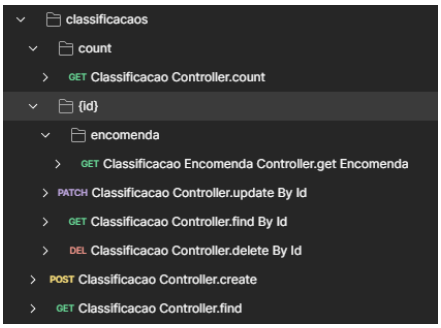
5. Estrutura e Justificação da API

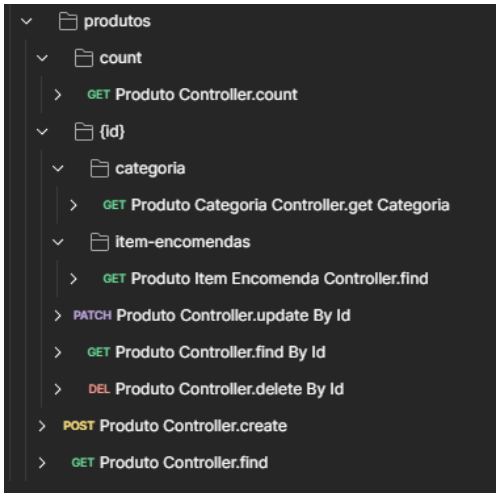
A simplificação da API no Postman foi realizada de forma a garantir uma estrutura mais clara, funcional e adequada aos requisitos obrigatórios. Focámo-nos nos métodos essenciais, como GET, POST, PATCH e DELETE, para assegurar o cumprimento das operações básicas de gestão de dados (CRUD). A escolha dos métodos a manter foi feita com base nas instruções fornecidas e na necessidade de garantir que cada funcionalidade essencial estivesse devidamente implementada, testada e documentada, sem incluir endpoints redundantes ou fora do escopo definido.

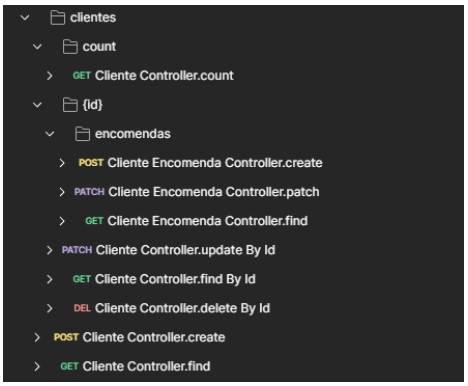
5.1 Coleção de Endpoints da API no Postman

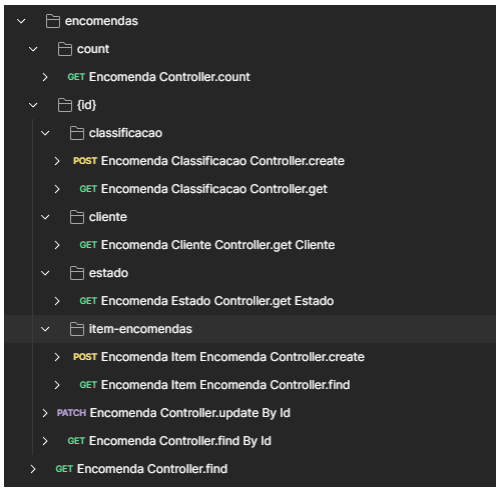

encomendas
count
GET Encomenda Controller.count
{id}
classificacao
POST Encomenda Classificacao Controller.create
GET Encomenda Classificacao Controller.get
cliente
GET Encomenda Cliente Controller.get Cliente
estado
GET Encomenda Estado Controller.get Estado
item-encomendas
POST Encomenda Item Encomenda Controller.create
GET Encomenda Item Encomenda Controller.find
PATCH Encomenda Controller.update By Id
GET Encomenda Controller.find By Id
GET Encomenda Controller.find

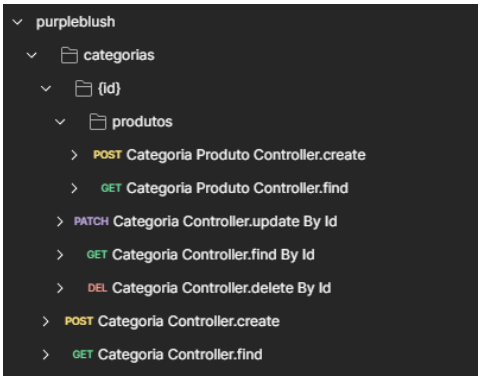

item-encomendas
{id}
encomenda
GET Item Encomenda Encomenda Controller.get Encomenda
produto
GET Item Encomenda Produto Controller.get Produto


classificacaos
count
GET Classificacao Controller.count
{id}
encomenda
GET Classificacao Encomenda Controller.get Encomenda
PATCH Classificacao Controller.update By Id
GET Classificacao Controller.find By Id
DEL Classificacao Controller.delete By Id
POST Classificacao Controller.create
GET Classificacao Controller.find


produtos
count
GET Produto Controller.count
{id}
categoria
GET Produto Categoria Controller.get Categoria
item-encomendas
GET Produto Item Encomenda Controller.find
PATCH Produto Controller.update By Id
GET Produto Controller.find By Id
DEL Produto Controller.delete By Id
POST Produto Controller.create
GET Produto Controller.find


clientes
count
GET Cliente Controller.count
{id}
encomendas
POST Cliente Encomenda Controller.create
PATCH Cliente Encomenda Controller.patch
GET Cliente Encomenda Controller.find
PATCH Cliente Controller.update By Id
GET Cliente Controller.find By Id
DEL Cliente Controller.delete By Id
POST Cliente Controller.create
GET Cliente Controller.find


encomendas
count
GET Encomenda Controller.count
{id}
classificacao
POST Encomenda Classificacao Controller.create
GET Encomenda Classificacao Controller.get
cliente
GET Encomenda Cliente Controller.get Cliente
estado
GET Encomenda Estado Controller.get Estado
item-encomendas
POST Encomenda Item Encomenda Controller.create
GET Encomenda Item Encomenda Controller.find
PATCH Encomenda Controller.update By Id
GET Encomenda Controller.find By Id
GET Encomenda Controller.find


purpleblush
categorias
{id}
produtos
POST Categoria Produto Controller.create
GET Categoria Produto Controller.find
PATCH Categoria Controller.update By Id
GET Categoria Controller.find By Id
DEL Categoria Controller.delete By Id
POST Categoria Controller.create
GET Categoria Controller.find