João Pedro Perez 24/03/2025

Situação problema: Sistema de gerenciamento para uma biblioteca.

B)

O software a ser desenvolvido se consiste em ser um sistema para realizar o gerenciamento de uma biblioteca, em que necessidade da mesma é organizar o sistema de empréstimo de seus livros.

Esta aplicação será realizada pelo Beck-End Web, uma vez que o mesmo se caracteriza por realizar o processamento das informações, ou seja, de modo figurado o Back-End é responsável pelo ‘o que está atrás dos panos/interface(visual)’, ele fará todo o controle de entrada e saída de livros, armazenando os dados de empréstimos, por exemplo: O cliente 1 alugou quais livros e em quais dadas. Ou até mesmo saber se o cliente 1 tem empréstimos atrasados.

De modo geral, o Beck-End terá o controle e acesso a todas informações referente a biblioteca, como clientes, livros, prazos, entre outros, afim de ter um gerenciamento mais eficiente. E ela é considerada WEB, pois toda interface gráfica, será exibida para usuário em uma página WEB, facilitando a compreensão e entendimento do sistema.

C)

A arquitetura usada nesse projeto, será baseada no sistema MVC, o qual se consiste em separar em o código em partes, afim de deixa-lo mais organizado, facilitando o a manutenção e futuras alterações.

No sistema MVC, dividimos o código em três partes: MODEL, VIEW e CONTROLLER.

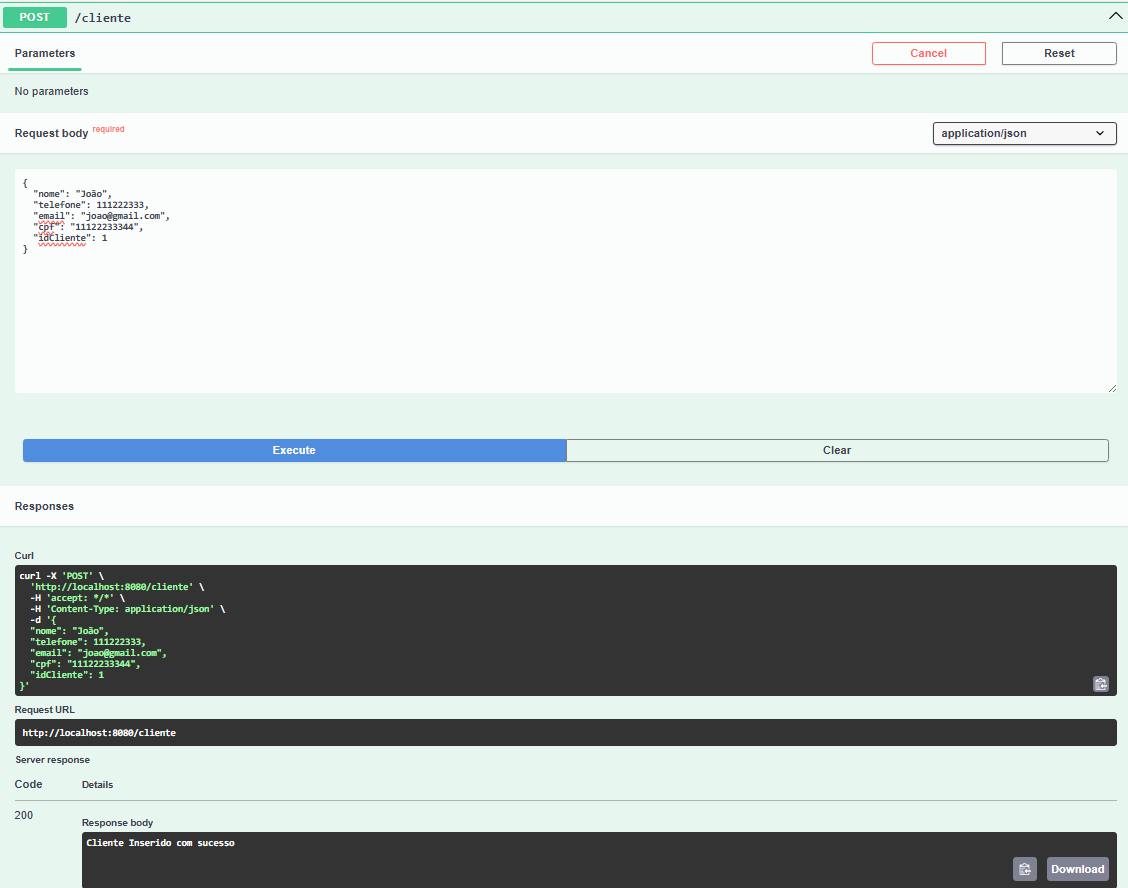
No MODEL, será realizado toda a parte de criação das classes, e no caso do sistema a ser desenvolvido o MODEL será constituído pelas Classes: Empréstimo, Cliente e Livro, contendo em cada seus próprios atributos. Além disso de conter todas a classes, também nessa parte em que se encontra a lógica a ser desenvolvida.

A VIEW, é responsável pela toda parte da interface, ou seja, o usuário irá ver, nessa etapa, acontece o recebimento de dados, em que o usuário inserir, e a partir dessas novas informações a VIEW irá mandar para o MODEL realizar toda a lógica/processamento de tais dados.

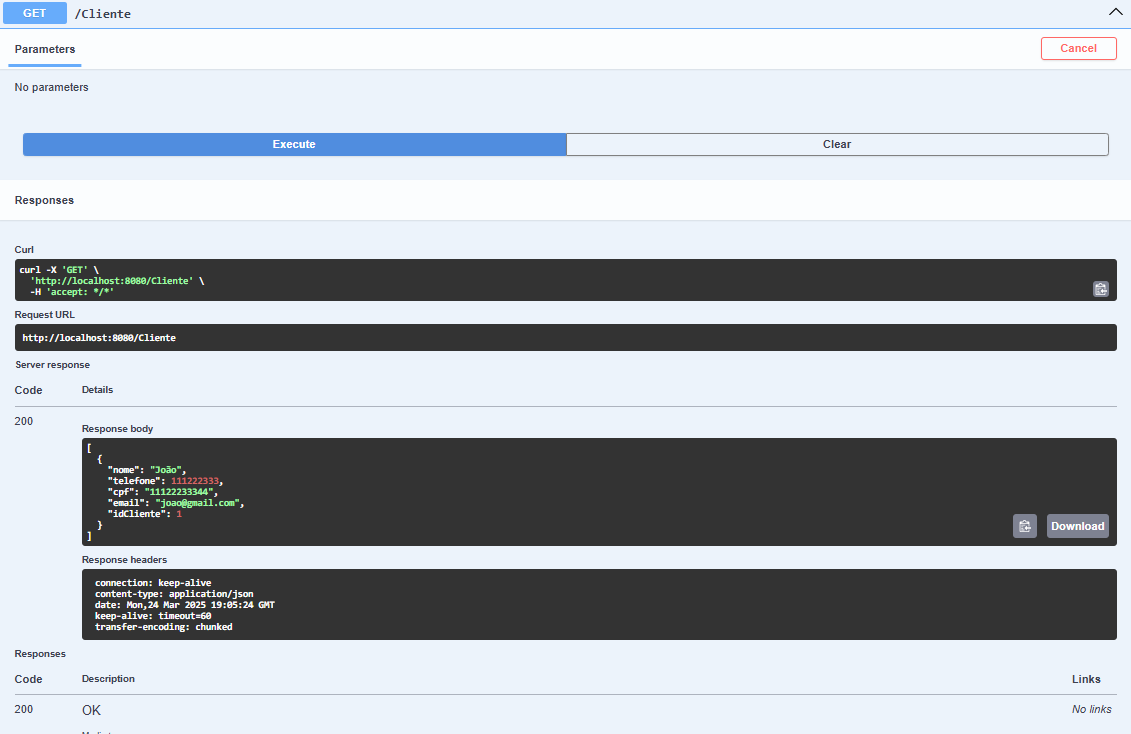
Para que a comunicação entre a VIEW e MODEL, não fique desorganizada, temos o CONTROLLER, a qual sua responsabilidade é justamente essa, ser o intermediário entre a VIEW e MODEL, para que a troca de informação entre as duas seja eficiente, evitando o máximo de conflitos em relação a movimentação de dados.

D)

Na primeira etapa do projeto, criei um End-Point dedicado ao cliente, nele será possível adicionar os dados de todos os clientes. E caso todas as informações estiverem corretas e não houver nenhum imprevisto na hora de adicionar um cliente, o sistema irá retorna uma mensagem de sucesso.



Na Segunda etapa criei outro End-Point dedicado ao cliente, nele será possível visualizar todos os clientes cadastrados.



Essa mesma lógica, será aplicada para a classe livro.

E)

Desings Patterns - Factory Method: Esse desing patterns permite que o criar uma super classe, e as subclasses classes podem alterar o tipo de objeto dessa super Classe.

No contexto da biblioteca, seria possível criar a super classe autor, e dentro dela os objetos seriam os livros que esse autor teria.

Desings Patterns - Adapter: Aqui é possível alterar uma classe de modo que ela consiga trabalhar de forma simultânea com outra classe.

Por exemplo: Caso a biblioteca amplie seu sistema de armazenamento das fichas dos clientes e livros, e também queira deixar essas fichas em uma nuvem, a fim de evitar possíveis perdas. Porém o formato atual não é possível deixa-lo na nuvem, invés de alterar o código fonte, a qual pode oferecer risco ao atual código, o Adapter entre em ação possibilitando a alteração do formato dos dados atual.

Desing Patterns – Builder: O builder permite criar vários objetos e estabelecer um padrão entre eles.

Por exemplo: Para um sistema de padaria, ela pode utilizar a mesma massa para fazer diversos pães com formatos e diferentes. Ou seja, a base está pronta, só foi alterado os moldes em que foram assados.

Desing Patterns – Observer: Esse design permite definir alguma função assim que algum evento ocorrer.

Exemplo: Se o cliente João da biblioteca, está querendo realizar um empréstimo do livro “Como Aprender JAVA, sem surtar”, porém esse livro está com outro cliente. Nesse caso quando o livro fosse devolvido, o sistema notificaria ao João que o livro está disponível. Então o observer iria fazer esse serviço de notar quando o livro solicitado estiver disponível, e notificar o João.

F)

Padrão Factory Method: Ao usar esse desing a sistema da biblioteca poderia organizar os livros de formas diferentes, declarando um super Classe. Exemplo: Declarar uma super Classe com o nome do autor, com isso todos os livros desse autor estariam reunidos. Ou declarar uma super Classe chamada categoria, assim todos os livros da mesma categoria estariam reunidos.

Padrão Observer: Ao usar esse desing auxiliaria a biblioteca facilitando a notificação de clientes. Exemplo: Um cliente deseja um livro em específico, porém esse livro não está disponível, e assim que o livro ficar disponível chegaria uma notificação para o cliente.