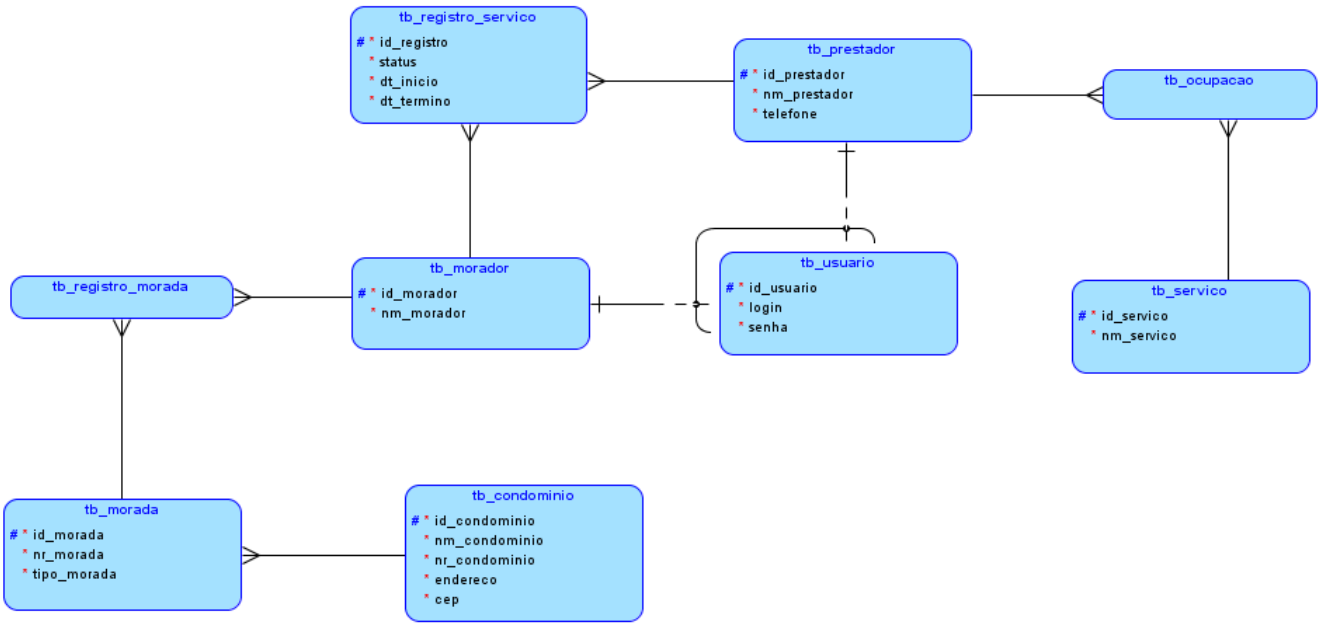
FIAP Fase 2 : Atividade 3

Início

Prosseguindo com o nosso `track record`, é chegado o momento de adicionarmos serviços `RESTful` ao nosso projeto. Seguindo a mesma linha que foi elaborada no projeto do capítulo 5 da fase 1, vamos elaborar um sistema que facilite o encontro de moradores de condomínios com prestadores de serviços. Nosso novo modelo de negócios ficou da seguinte forma:



O que muda desse modelo em relação ao anterior:

- Agora estamos adicionando acessos para a nossa aplicação. Tanto `Morador` quanto `Prestador` vão precisar ter um login e uma senha de acesso caso desejem entrar na aplicação.
- Criamos a **tb_usuario** para armazenar os acessos, **tb_usuario** e **tb_prestador** vão herdar os dados armazenados dentro da tabela.
- A relação entre as tabelas será de 1x1. `Morador` e `Prestador`podem ter apenas um `Usuário de acesso`.
- Também adicionamos mais dados para **tb_condominio** com o intuito de incluir restrições. Não será aceito `Condomínios` com cep e número repetidos.

OBJETIVO DO PROJETO:

Este projeto tem como objetivo mostrar as operações CRUD dessas entidade utilizando REST. O documento terá os seguintes passos: **1) Demonstrando os metodos GET, PUT, POST e DELETE; 2) Considerações finais / Instalação**

Demonstrando os métodos GET, PUT, POST e DELETE

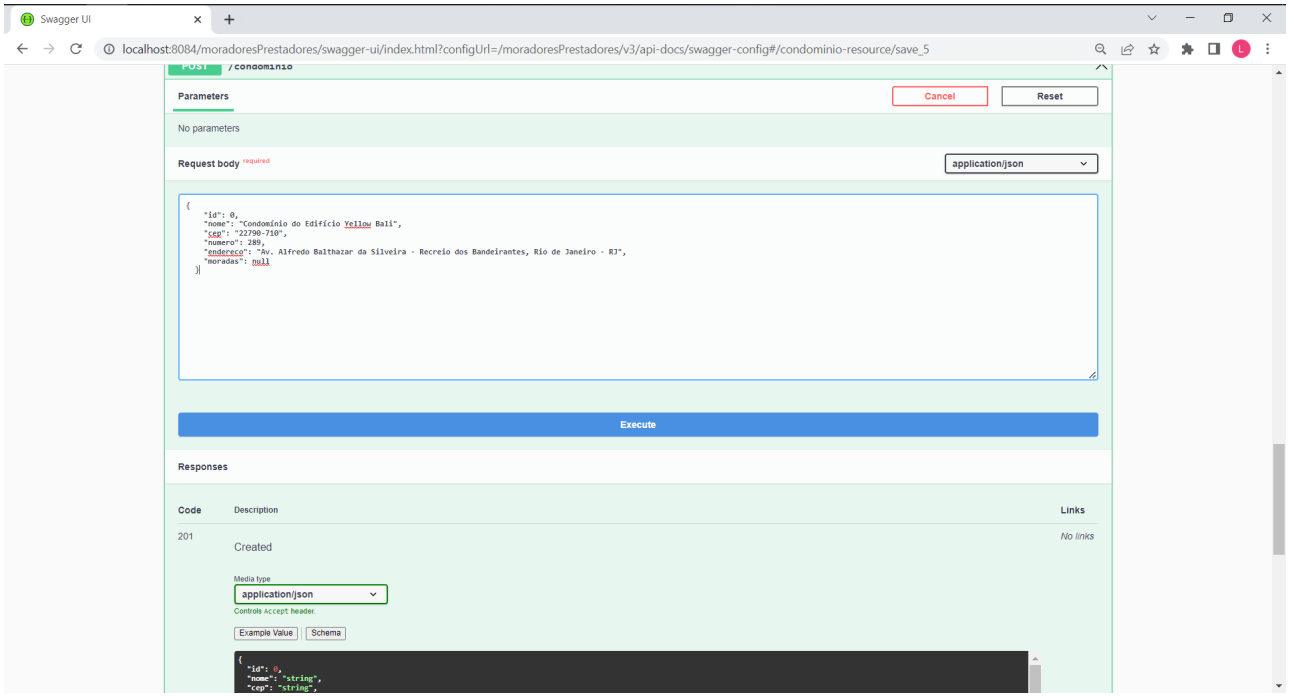
Optei por mostrar os métodos utilizando a documentação do `swagger` pela sua praticidade.

Estou utilizando o banco de dados `postgres` para realizar a armazenagem dos dados. As consultas estão sendo realizadas via `visual studios code` com uma extensão que possibilita a conexão com o banco.

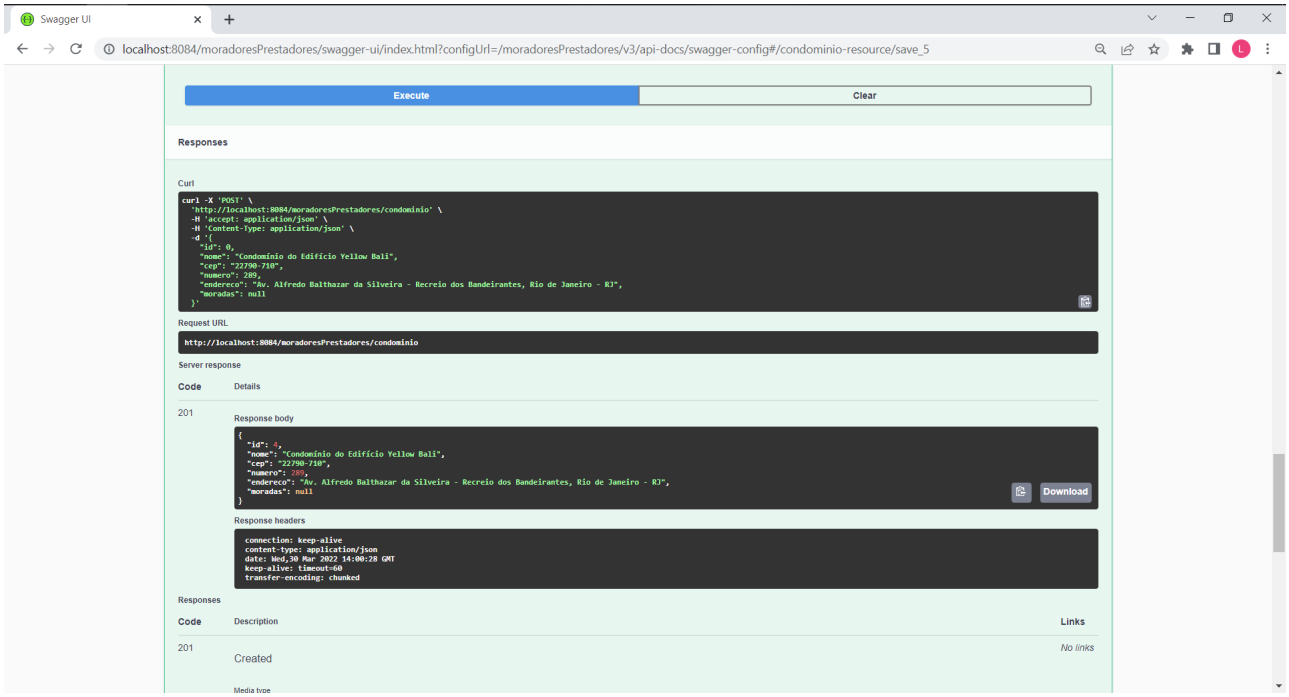
Clique aqui para ter acesso a documentação.

- **POST**

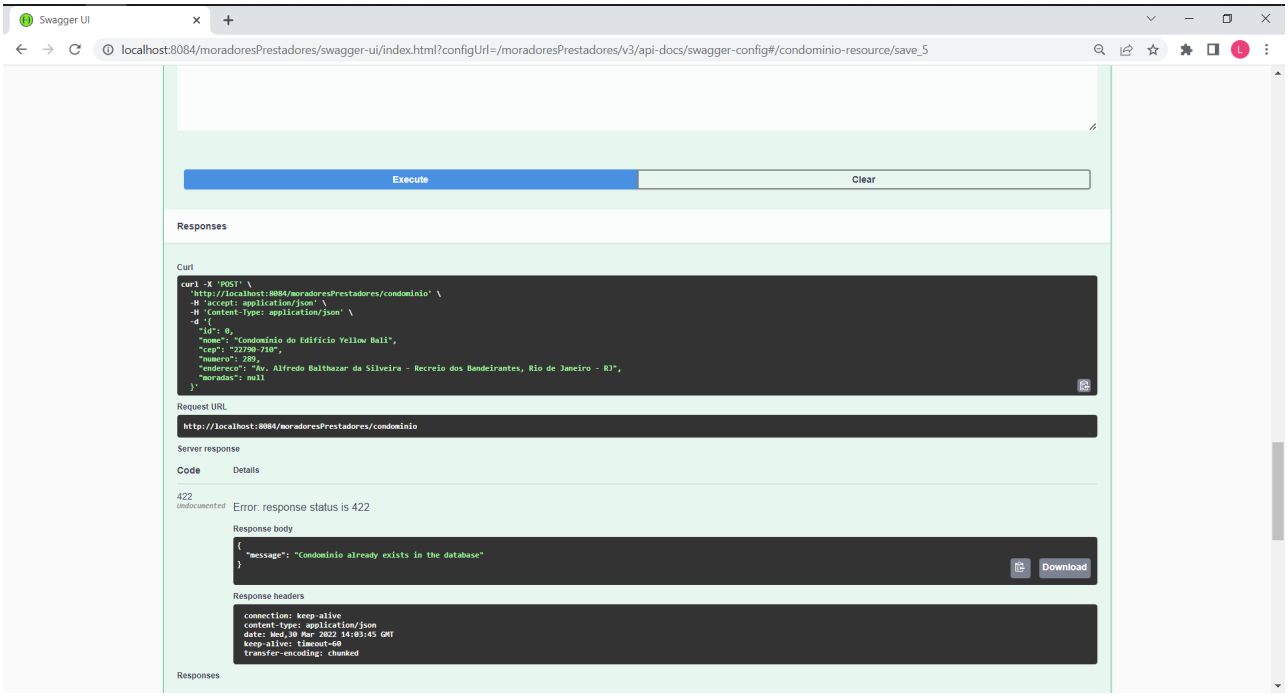
  O comando POST possibilita incluir novos dados ao nosso banco, no exemplo abaixo estou incluindo um novo condomínio.



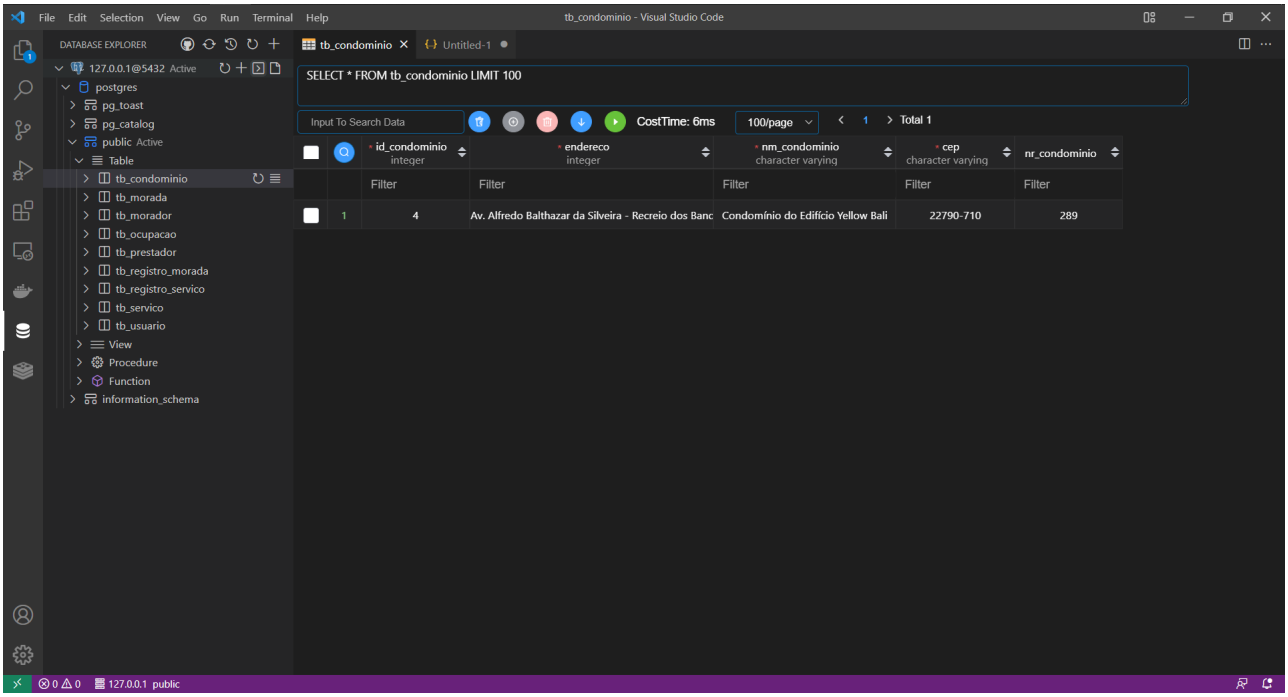
Após executamos a solicitação, vamos receber o corpo do objeto criado e o codigo de sucesso 201.



Configuramos o programa para não criar novos condomínios que tenham o mesmo número e cep existentes no banco. Caso haja alguma inclusão desse tipo, vamos receber um erro de código 422 informando que já existe um condomínio com estes dados.



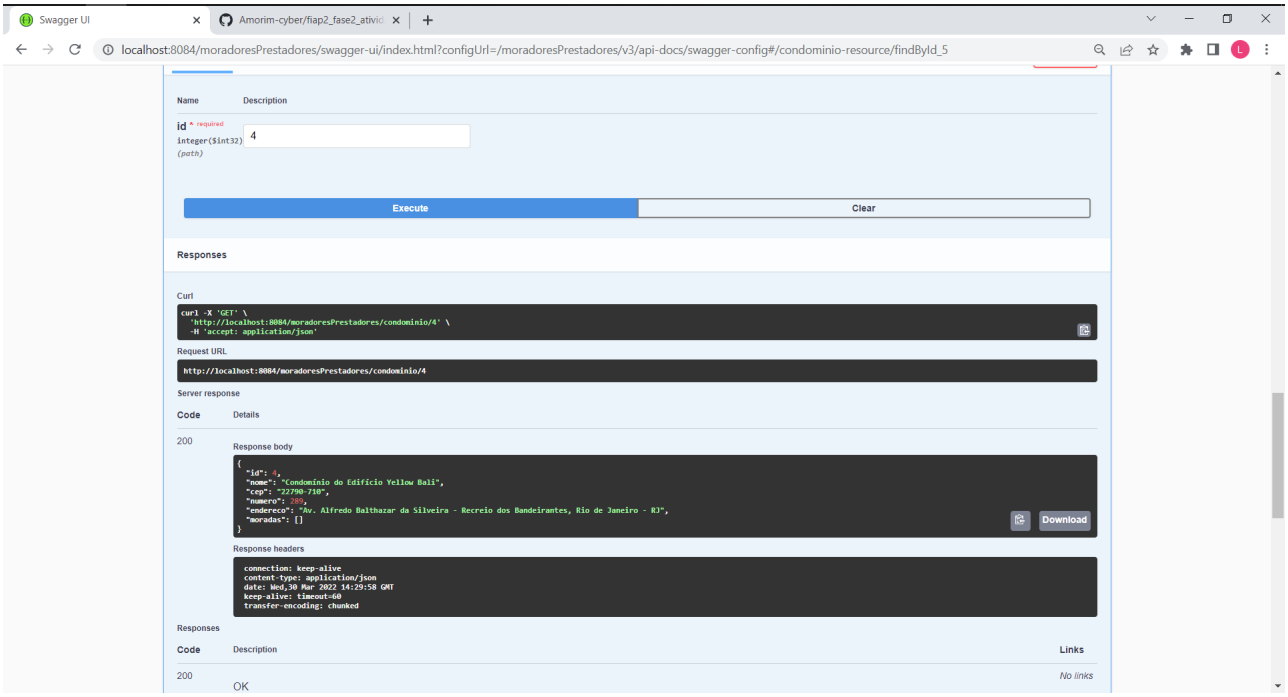
Consultando nosso banco, podemos observar que o condomínio foi inserido com sucesso.



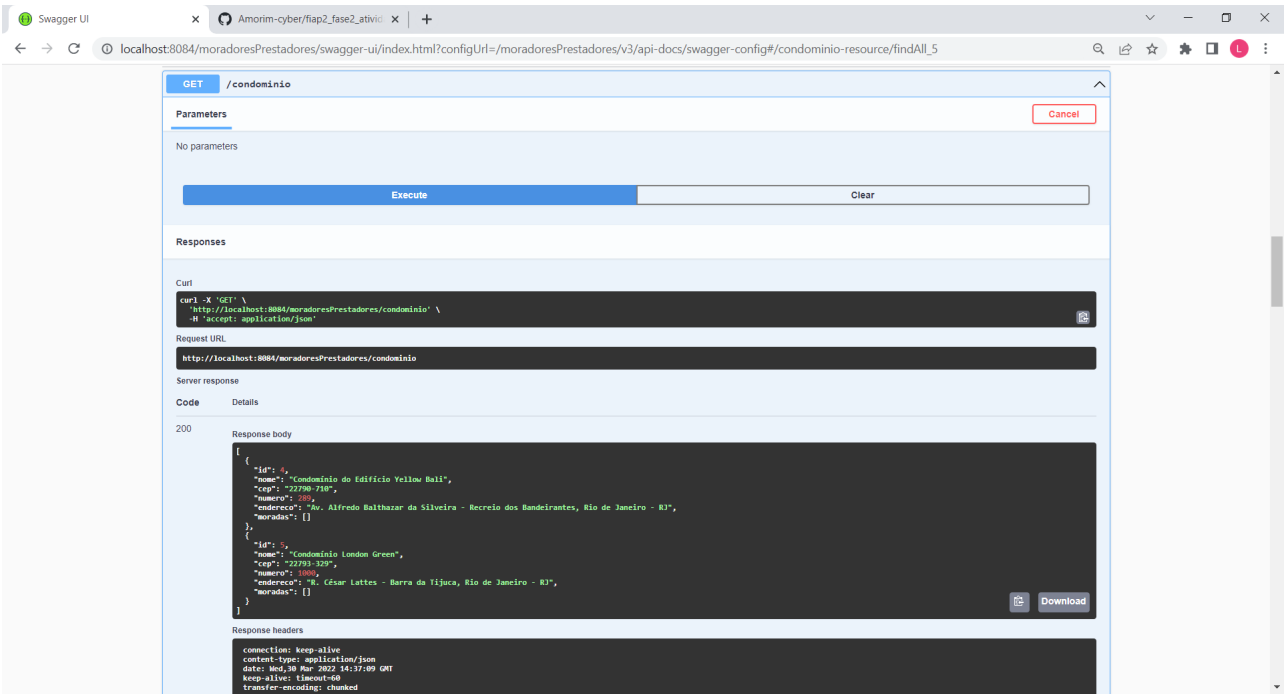
- **GET**

O comando GET tem o papel de retornar nossas entidades armazenadas no banco em formato de objeto, podemos retornar um ou mais elementos.

No exemplo abaixo, retornamos apenas um único condomínio por meio da sua chave de identificação.



Podemos chamar mais de um condomínio se quisermos.

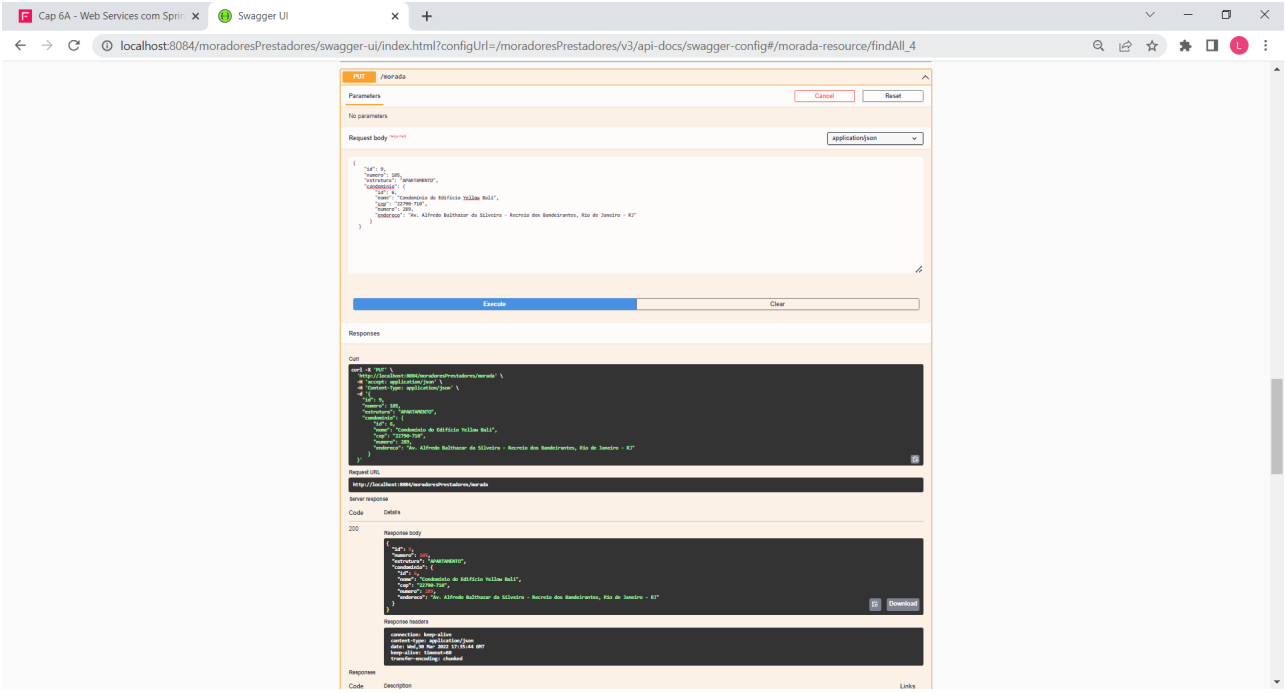


Deixo aqui esse vídeo demonstrando os métodos POST e GET.

Obs: No vídeo era para ter dito "postman" ao invés de "postgres" na hora de abrir o swagger

- **PUT**

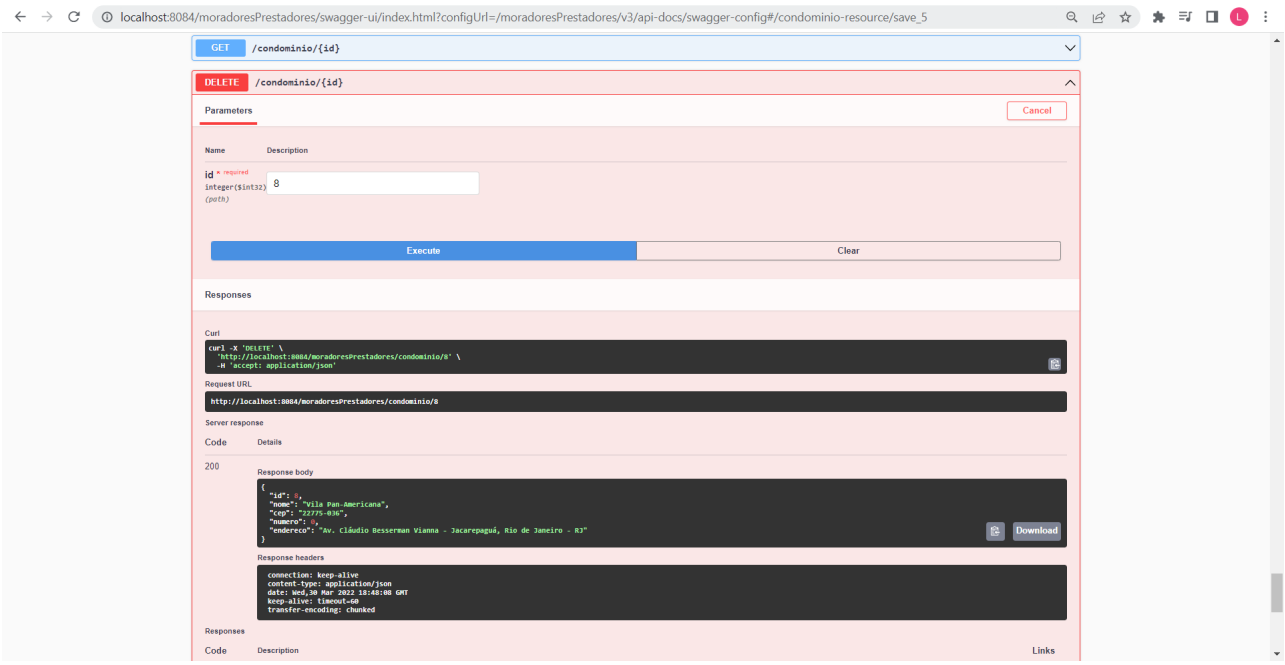
O comando PUT atualiza dados de uma entidade já existente no banco. O programa irá retornar o elemento com os atributos alterados junto com o código 200 de sucesso.



Atenção quando for atualizar, o comando só irá ter êxito se preencher os dados da forma correta além de fornecer um id já existente no banco. Veja aqui o vídeo de demonstração.

- **DELETE**

O comando DELETE exclui uma determinada entidade dando seu id de identificação. Ao deletar o programa retorna o corpo do objeto deletado e código 200 de sucesso.



Caso queira deletar uma entidade que tenham uma chave estrangeira (Condomínio por exemplo) precisa observar se ela não possui relação com outras entidades (Morada por exemplo), quando isso ocorrer é necessário deletar/mudar as entidades da qual ela tem relação antes de deleta-la. Clique aqui para ver demonstração.

Considerações finais / instalação

O projeto deve ter telas e mais funcionalidade em breve.

Foi utilizado as ferramentas do spring boot junto com a IDE `Intellij`.

O servidor está rodando na porta 8084.

Para rodar o programa localmente em sua maquina siga os seguintes passos:

1. Faça o download do arquivo e descompacte-o.
2. Abra o Intellij e abra o arquivo.
3. Remova as credenciais de acesso ao banco postgres heroku e coloque as credenciais de seu banco local.

Segue vídeo de instalação.

Obs: No vídeo eu falei "eclipse" ao invés de "intelliJ" mas acho que deu para pegar a ideia

CLIQUE AQUI PARA OLHAR O GITHUB DESTE PROJETO

Forte abraço! :smile: