# Objetivos

Criar APIs REST com as seguintes funções:

1. Registro de imóvel;
2. Atualização de um imóvel;
3. Consulta de um imóvel;
4. Exclusão de um imóvel;
5. Filtro de imóveis: O filtro consiste em realizar uma consulta de imóveis por número de quartos e/ou metragem e/ou número de vagas.

# Problema

Desenvolver um servidor com APIs REST que recebem parâmetros, manipulam os dados armazenados no banco e retornam o resultado.

# Desenvolvimento

O algoritmo foi estruturado seguindo regras da arquitetura MVC. Dentro do código, pode ser visto que existem Controllers e Models

Todas as funções das APIs foram estruturadas com “async/await”, dentro de um bloco de “try/catch”

Foram criados ErrorHandlers para retornar erros para o usuário, junto com o código de status da requisição HTTP

## Controllers:

ImovelController.js – contém todas as funções das APIs do model “Imovel”. Extende o modelo “Imovel” e a interface Op do Sequelize

## Models:

Imovel - Extende a classe Model e DataTypes do Sequelize

### Bibliotecas utilizadas:

Mysql2 – para conexão entre a biblioteca Sequelize e o banco MySQL

Express – para inicialização do servidor e criação de APIs REST

Swagger-jsdoc – utilizado para documentar as APIs e mostra-las com Swagger

Swagger-ui-express – utilizado para montar uma interface em que o usuário possa testar as APIs

Para desenvolvimento foi utilizado:

Nodemon – para reinicialização automática quando um dos arquivos são alterados

Sequelize-cli – para criação de banco, tabelas e migrações

# Instalação

Importar estrutura da tabela “imoveis” para o banco de dados a ser usado (arquivo dump\_banco\_imoveis.sql)

Configurar acesso do banco de dados no arquivo /src/config/database.js

Configurar Porta do servidor no arquivo /src/server.js (Porta padrão está para 3333)

Inicializar arquivo server.js pelo código node server.js

Acessar localhost:porta/keycash-apis

Deve mostrar a tela de testes da ferramenta Swagger UI, disponibilizando todas as APIs

# Input e Output

/listar, ImovelController.listAllImoveis

-Irá listar todos os imóveis

/cadastrar, ImovelController.create

- Cadastra um novo imóvel

/deletar/:imovelId, ImovelController.deleteById

- Deleta um imóvel

/atualizar/:imovelId, ImovelController.update

- Atualiza um imóvel

/buscarImovelPorTipoCompra/:tipoCompra, ImovelController.getImovelByTipoCompra

- Busca imóvel por tipo da compra (aluguel, venda)

/buscarImovelPorId/:imovelId, ImovelController.getImovelById

- Busca imóvel pelo Id do imóvel

/buscarImovelPorBanheiro/:totalBanheiro, ImovelController.getImovelByBanheiro

- Busca imóveis pela quantidade de banheiros

/buscarImovelPorQuarto/:totalQuarto, ImovelController.getImovelByQuarto

- Busca imóveis pela quantidade de quartos

/buscarImovelPorVaga/:totalVaga, ImovelController.getImovelByVaga

- Busca imóveis pela quantidade de vagas

/buscarImovelPorPet/:aceitaPet, ImovelController.getImovelByPet

- Busca imóveis que aceitam ou não aceitam animais de estimação

/buscarImovelPorMobilia/:temMobilia, ImovelController.getImovelByMobilia

- Busca imóveis que possuem ou não possuem mobília

/buscarImovelPorTipo/:tipo, ImovelController.getImovelByTipo

- Busca imóveis pelo tipo da compra (Venda, Aluguel)

# Restrições e Regras

É preciso passar a estrutura inteira do imóvel, mesmo se for atualizar apenas um dado

Quando um imóvel é atualizado, o imóvel retornado contém os atributos já atualizados

# Comentários e Conclusões

Utilizei diversas ferramentas neste teste e tive a oportunidade de me desenvolver ainda mais ainda com elas. Foquei bastante no desenvolvimento e tratamento de erros das APIs. Acredito que o banco de dados não seja o foco, logo, deixei ele o mais simples possível.

Resolvi experimentar a ferramenta Swagger-js neste projeto, pois nunca tinha usado. Foi uma ótima experiência.

Ainda dá para escalar muito mais o projeto, tanto as APIs quanto o banco de dados.