Universidade do Vale do Itajaí – UNIVALI
Escola do Mar, Ciência e Tecnologia
Curso de Análise e Desenvolvimento de Sistemas
Disciplina Hands on Work VII
Aluno: Paulo Luciano Silva dos Santos


# Projeto de Hands on Work VII

Relatório
FINAL

API –CONSULTA SQL PARA GERAÇÃO DE GRÁFICO
BACKEND

# Sumário

# Introdução:

Um API foi desenvolvido para fazer a consulta em um banco de dados, retornando informações para a equipe de FrontEnd montar os gráficos, solicitados pela empresa.

Os arquivos estarão no GitHub, onde será postado o caminho dentro do projeto.

# Proposta

O Banco de dados que foi usado para o projeto foi o MySql, que é um banco de dados relacional, foram adicionados 60 registros com informações para poder utilizar no que foi solicitado pelo Fórum Temático.

A Linguagem de programação foi usada a Python uma linguagem de programação orientado a objeto, que será entregue as informações através de arquivos em formato json, para a equipe de FrontEnd montar os gráficos.

# Fórum Temático

**Hands on Work VII**

Nesta disciplina de Hands on Work VII serão desenvolvidos serviços de *backend*, os quais deverão empegar os paradigmas de programação orientado a objetos e também o paradigma de programação funcional, de acordo com o solicitado no escopo da atividade.

**Informações importantes:**

- Este trabalho pode ser realizado em grupos de até 3 participantes.
- Entregas com atraso terão nota atribuída como "0" (ZERO), sem postergação.

**Escopo da atividade:**

Uma grande empresa na área de gestão imobiliária necessita de painéis gráficos que sejam consultáveis por meio de interfaces web. Os gráficos necessários são:

a. Gráfico de barras: Exibir para cada imóvel o valor acumulado em pagamentos considerando toda a série histórica disponível.
b. Gráfico de dispersão ou linhas: exiba por mês/ano, considerando todos os dados disponíveis no banco de dados, o total de vendas ocorridas naquele período.
c. Gráfico de pizza: Exibir o percentual de vendas de cada uma das modalidades de imóvel disponível no sistema (ex. Terrenos, apartamentos, sala comercial, galpão)

Porém sua "empresa" foi contratada não para construir os gráficos (frontend), mas sim os módulos de backend. Deste modo sua responsabilidade é acessar os dados no banco de dados e processá-los na linguagem de backend, assim como disponibilizar serviços para consumo. **A construção das telas (frontend) não faz parte das atribuições deste contrato.**

A origem dos dados é um banco de dados relacional (SGDB). Entretanto os filtros e agrupamentos de dados não devem ser realizados pelo servidor de banco de dados, e sim pela linguagem de programação de backend, utilizando técnicas de programação funcional como map/filter/reduce/forEach.

- *Em linguagem técnica: Os dados **não devem ser filtrados** com a cláusula where do SQL e nem agregados pela cláusula group by do SQL. Os dados devem ser carregados **na íntegra** na memória do backend e lá serão processados pela linguagem de programação.*

Esta decisão e foi motivada pela empresa cliente qual possui alocadas máquinas com grande capacidade de processamento paralelo para suportar o volume dados que serão transacionados quando o sistema estiver operando em ambiente de produção.

O banco de dados deve prover para a aplicação uma estrutura de dados como a representada abaixo, qual inclui informações sobre os pagamentos realizados.:

| id_venda | data_do_pagamento | valor_do_pagamento | codigo_imovel | descricao_imóvel | tipo_imovel |
|---|---|---|---|---|---|
| 1 | 2023-08-10 | 5000 | 4356 | Apartamento 100 m2 em condomínio fechado. | Apartamento |

**Parte 1:**

a. Criar as tabelas necessárias para representar a estrutura de dados apresentada (ex. Imóvel, tipo imóvel, pagamento)
b. Inserir ao menos 30 registro de pagamentos com ocorrência em pelo menos 5 meses distintos.
c. Inserir ao menos 8 imóveis e certifica-se que todos terão apenas 1 ocorrência na tabela de pagamentos.
d. Escrever uma consulta SQL que retorne como resultado uma estrutura análoga aquela apresentada no enunciado (deverá ser realizado um join entre todas as tabelas criadas).
e. Escrever na linguagem de programação de backend que consulte o banco de dados executando a instrução realizada no item d.

**Entrega:** Um arquivo PDF devidamente estruturado apresentando todos os resultados produzidos (*ex. scripts SQL e o código do backend). Na capa deve constar o nome de todos os integrantes do grupo, apenas estes terão nota atribuída.

**Parte 2:**

Considerando o retorno dos dados do banco de dados (produzidos na Parte 1 "e") faça:

a. Implementar uma função que retorne uma lista com o id de cada imóvel e sua respectiva soma de todos os pagamentos. (ex. [18] => 7000, [3] => 12000, [9] => 10000).
b. Implementar uma função que retorne uma lista com cada mês/ano e o total de vendas ocorridas no período. (ex. [11/2022 => 23000, 12/2022 => 15000, 01/2023 =>1800])
c. Implementar uma função que retorne uma lista com cada tipo de imóvel registrado e o seu respectivo valor percentual no total de vendas (quantitativas) considerando toda série histórica. (ex. [Apartamento => 50 %, Casas => 30%, Sala comercial => 20%])
d. Cada uma das três funções deve retornar os dados em formato JSON como resposta de uma chamada "REST"/GET.

**Entrega:**

- O arquivo PDF entregue na parte 1 deve ser atualizado incluindo novas seções para demonstrar o código desenvolvido na parte 2. Também deve constar resultado dos testes de cada um dos 3 serviços a serem evidenciados por "print screens" (imagens da tela) que exibam as respostas recebidas para as chamadas dos serviços REST desenvolvidos.
- Grave e poste um vídeo (máximo 3 minutos de duração) apresentando cada uma das três requisições e seu respectivo resultado JSON. Também apresente o código fonte do backend e a estrutura do banco de dados.
- Entregas sem o vídeo de apresentação serão atribuídas nota 0.

Para implementação recomenda-se o conjunto de tecnologias:

- Banco de dados: MySQL
- Servidor web (backend): Node.js com módulo express.

# Códigos e scripts

SQL

Script, para a criação do banco de dados a ser utilizado.

**###### - INÍCIO DO CÓDIGO**

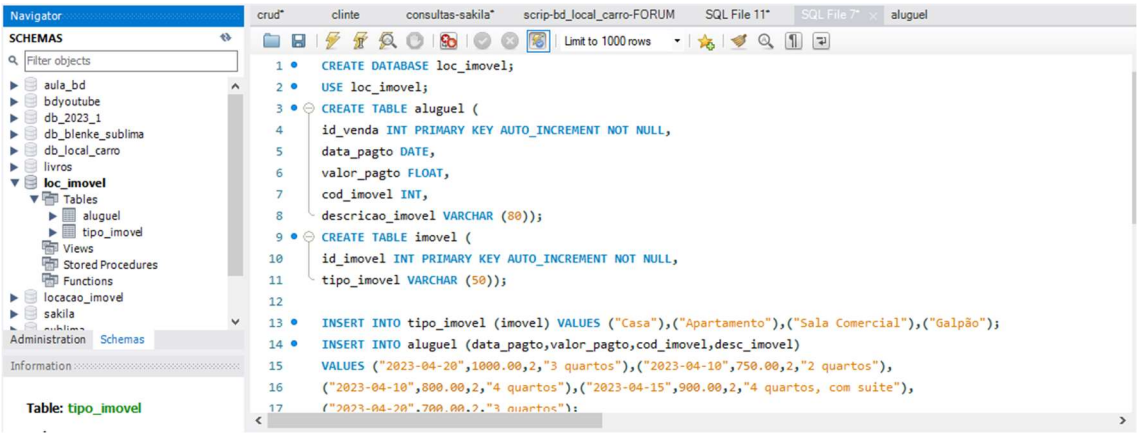
```
CREATE DATABASE loc_imovel;
USE loc_imovel;
CREATE TABLE aluguel (
id_venda INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
data_pagto DATE,
valor_pagto FLOAT,
cod_imovel INT,
descricao_imovel VARCHAR (80));
CREATE TABLE imovel (
id_imovel INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
tipo_imovel VARCHAR (50));
INSERT INTO tipo_imovel (imovel) VALUES ("Casa"),("Apartamento"),("Sala Comercial"),("Galpão");
INSERT INTO aluguel (data_pagto,valor_pagto,cod_imovel,desc_imovel)
VALUES ("2023-04-20",1000.00,2,"3 quartos"),("2023-04-10",750.00,2,"2 quartos"),
("2023-04-10",800.00,2,"4 quartos"),("2023-04-15",900.00,2,"4 quartos, com suite"),
("2023-04-20",700.00,2,"3 quartos");
INSERT INTO aluguel (data_pagto,valor_pagto,cod_imovel,desc_imovel)
VALUES ("2023-04-20",1000.00,1,"3 quartos e suite"),("2023-04-10",750.00,1,"2 quartos"),
("2023-04-10",800.00,1,"4 quartos, 1 suite"),("2023-04-15",900.00,1,"4 quartos, com suite"),
("2023-04-20",700.00,1,"3 quartos");
INSERT INTO aluguel (data_pagto,valor_pagto,cod_imovel,desc_imovel)
VALUES ("2023-03-20",1000.00,3,"100 metros quadrado"), ("2023-04-10",950.00,3,"50 metros quadrado"),
("2023-04-10",2000.00,3,"150 metros quadrado"),("2023-04-15",2600.00,3,"200 metros quadrado"),
("2023-04-20",5000.00,3,"3500 metros quadrado");
INSERT INTO aluguel (data_pagto,valor_pagto,cod_imovel,desc_imovel)
```

VALUES ("2023-04-20",1000.00,4,"100 metros quadrado"), ("2023-04-10",950.00,4,"50 metros quadrado"),

("2023-04-10",2000.00,4,"150 metros quadrado"),("2023-04-15",2600.00,4,"200 metros quadrado"),

("2023-04-20",5000.00,4,"3500 metros quadrado");

**###### - FIM DO CÓDIGO**




Código em Python

```python
import mysql.connector
import array
from datetime import datetime, timedelta

# cria uma conexão com o banco de dados
conexao = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    password = '123456',
    database = 'loc_imovel',
)
# variável, que comtem a conexão
cursor = conexao.cursor()

# variavel que será usado para fazer a consulta no banco de dados
comando = f'SELECT * FROM aluguel INNER JOIN tipo_imovel ON
cod_imovel=id_imovel'

# executa a consulta no banco de dados.
cursor.execute(comando)

# variavel que possui todos os dados peito pela consulta esta variavel é
um tipo de array
resultado = cursor.fetchall()
```

```
# imprime o resultado da consulta em formato tipo json
#print (resultado)
#print ("Numero Total de registro retornados:",cursor.rowcount,"\n")


### Calcula o valor do aluguies por imovei
valor = 0
total_registro = cursor.rowcount / 100 + 1.0
perc_movel = []
for linha in resultado:
      if linha[5] == 1:
            valor = valor + 1
            movel = linha[5]
perc_movel.append(movel)
perc_movel.append(total_registro * valor)
valor = 0

for linha in resultado:
      if linha[5] == 2:
            valor = valor + 1
            movel = linha[5]
perc_movel.append(movel)
perc_movel.append(total_registro * valor)
valor = 0

for linha in resultado:
      if linha[5] == 3:
            valor = valor + 1
            movel = linha[5]
perc_movel.append(movel)
perc_movel.append(total_registro * valor)
valor = 0

for linha in resultado:
      if linha[5] == 4:
            valor = valor + 1
            movel = linha[5]
perc_movel.append(movel)
perc_movel.append(total_registro * valor)
valor = 0

# cria o arquivo html como os dados do banco de dados, todos os imoveis
cadastrado.
with open("imoveis_todos_porcetagem.html","w") as arquivo:
      arquivo.write(str(perc_movel))

#print (perc_movel)

valortotal = 0
```

```
timoveis = []

for linha in resultado:
        if linha[5] == 1:
           valortotal = valortotal + linha[2]
           casas = valortotal
           imovel = linha[5]

timoveis.append(imovel)
timoveis.append(valortotal)

for linha in resultado:
        if linha[5] == 2:
           valortotal = valortotal + linha[2]
           apto = valortotal
           imovel = linha[5]

timoveis.append(imovel)
timoveis.append(valortotal)

#print (imovel," => ", valortotal)

for linha in resultado:
        if linha[5] == 3:
           valortotal = valortotal + linha[2]
           slcom = valortotal
           imovel = linha[5]

timoveis.append(imovel)
timoveis.append(valortotal)
#print (imovel," => ", valortotal)

for linha in resultado:
        if linha[5] == 4:
           valortotal = valortotal + linha[2]
           galpao = valortotal
           imovel = linha[5]

timoveis.append(imovel)
timoveis.append(valortotal)

#print (imovel," => ", valortotal)
#print (imovel," => ", valortotal)
#print ("variavel timoves",timoveis)

# cria um arquivo com todos os imoveis cadastrado, com valor total de
aluguel:
with open("imoveis_aluguel_total.html","w") as arquivo:
      arquivo.write(str(timoveis))
```

```
    #for linha in resultado:
     #    arquivo.write(str(linha))
         #print(x)

# cria o arquivo html como os dados do banco de dados, todos os imoveis
cadastrado.
with open("imoveis_todos_cadastro.html","w") as arquivo:
    for linha in resultado:
        arquivo.write(str(linha))

### FIM do calculo

### verifca a data por imovel

# variavel que será usado para fazer a consulta no banco de dados
comando = f'SELECT data_pagto, valor_pagto FROM aluguel'

# executa a consulta no banco de dados.
cursor.execute(comando)

# variavel que possui todos os dados peito pela consulta esta variavel é
um tipo de array
resultado = cursor.fetchall()

# imprime o resultado da consulta em formato tipo json
#print ("Valores de data e pagto \n", resultado)

vlr=0
total_imovel = []
for linha in resultado:
      if linha[0].month== 2:
            vlr = vlr + linha[1]

total_imovel.append("02/2023")
total_imovel.append(vlr)
#print ("Total mes 02/2023 => ",vlr)

for linha in resultado:
      if linha[0].month== 3:
            vlr = vlr + linha[1]

total_imovel.append("03/2023")
total_imovel.append(vlr)
#print ("Total mes 03/2023 => ",vlr)

for linha in resultado:
      if linha[0].month== 4:
            vlr = vlr + linha[1]
```

```
total_imovel.append("04/2023")
total_imovel.append(vlr)

with open("imoveis_pagto_mensal.html","w") as arquivo:
      arquivo.write(str(total_imovel))

# cria o arquivo html como os dados do banco de dados.
with open("imoveis.html","w") as arquivo:
    for linha in resultado:
        arquivo.write(str(linha))


# fecha as conexões abertas.
cursor.close()
conexao.close()
```

```
PS C:\Documentos_particular\! Download\how-aula> & C:/Users/Lageano/AppData/Local/Programs/Python/Python311/python.exe "c:/Documentos_particular/! Download/how-aula/api_imovel.py"
[(7, datetime.date(2023, 2, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa'), (8, datetime.date(2023, 2, 10), 750.0, 1, '2 quartos', 1, 'Casa'), (9, datetime.date(2023, 2, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa'), (10, datetime.date(2023, 2, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa'), (11, datetime.date(2023, 2, 20), 700.0, 1, '3 quartos', 1, 'Casa'), (27, datetime.date(2023, 3, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa'), (28, datetime.date(2023, 3, 10), 750.0, 1, '2 quartos', 1, 'Casa'), (29, datetime.date(2023, 3, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa'), (30, datetime.date(2023, 3, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa'), (31, datetime.date(2023, 3, 20), 700.0, 1, '3 quartos', 1, 'Casa'), (47, datetime.date(2023, 4, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa'), (48, datetime.date(2023, 4, 10), 750.0, 1, '2 quartos', 1, 'Casa'), (49, datetime.date(2023, 4, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa'), (50, datetime.date(2023, 4, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa'), (51, datetime.date(2023, 4, 20), 700.0, 1, '3 quartos', 1, 'Casa'), (1, datetime.date(2023, 2, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento'), (2, datetime.date(2023, 2, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento'), (3, datetime.date(2023, 2, 10), 750.0, 2, '2 quartos', 2, 'Apartamento'), (4, datetime.date(2023, 2, 10), 800.0, 2, '4 quartos', 2, 'Apartamento'), (5, datetime.date(2023, 2, 15), 900.0, 2, ' 4 quartos, com suite', 2, 'Apartamento'), (6, datetime.date(2023, 2, 20), 700.0, 2, '3 quartos', 2, 'Apartamento'), (22, datetime.date(2023, 3, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento'), (23, datetime.date(2023, 3, 10), 750.0, 2, '2 quartos', 2, 'Apartamento'), (24, datetime.date(2023, 3, 10), 800.0, 2, '4 quartos', 2, 'Apartamento'), (25, datetime.date(2023, 3, 15), 900.0, 2, '4 quartos, com suite', 2, 'Apartamento'), (26, datetime.date(2023, 3, 20), 700.0, 2, '3 quartos', 2, 'Apartamento'), (42, datetime.date(2023, 4, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento'), (43, datetime.date(2023, 4, 10), 750.0, 2, '2 quartos', 2, 'Apartamento'), (44, datetime.date(2023, 4, 10), 800.0, 2, '4 quartos', 2, 'Apartamento'), (45, datetime.date(2023, 4, 15), 900.0, 2, '4 quartos, com suite', 2, 'Apartamento'), (46, datetime.date(2023, 4, 20), 700.0, 2, '3 quartos', 2, 'Apartamento'), (12, datetime.date(2023, 2, 20), 1000.0, 3, '100 metros quadrado', 3, 'Sala Comercial'), (13, datetime.date(2023, 2, 10), 950.0, 3, '50 metros quadrado', 3, 'Sala Comercial'), (14, datetime.date(2023, 2, 10), 2000.0, 3, '150 metros quadrado', 3, 'Sala Comercial'), (15, datetime.date(2023, 2, 15), 2600.0, 3, '200 metros quadrado', 3, 'Sala Comercial'), (16, datetime.date(2023, 2, 20), 5000.0, 3, '3500 metros quadrado', 3, 'Sala Comercial'), (32, datetime.date(2023,
```

```
Numero Total de registro retornados: 61

id_venda = 7
data_pagto = 2023-02-20
valor_pagto = 1000.0
cod_imovel = 1
descricao_imovel = 3 quartos e suite
id_imovel = 1
Imovel =  Casa

id_venda = 8
data_pagto = 2023-02-10
valor_pagto = 750.0
cod_imovel = 1
descricao_imovel = 2 quartos
id_imovel = 1
Imovel =  Casa
```

# Postagem dos arquivos

https://github.com/Luciano-Lageano/Projeto_HandWorkVII.git

Neste endereço esta os arquivos que foram criados.

Com o código pronto, serão criado os arquivos de saída em formato JSON, estes arquivos estão disponível no GITHUB, sendo que foi criado 5 arquivos html.

Arquivos gerado pela API

### imoveis.html

(datetime.date(2023, 2, 20), 1000.0)(datetime.date(2023, 2, 20), 1000.0)(datetime.date(2023, 2, 10), 800.0)(datetime.date(2023, 2, 15), 900.0)(datetime.date(2023, 2, 20), 700.0)(datetime.date(2023, 2, 20), 1000.0)(datetime.date(2023, 2, 10), 750.0)(datetime.date(2023, 2, 10), 800.0)(datetime.date(2023, 2, 15), 900.0)(datetime.date(2023, 2, 20), 700.0)(datetime.date(2023, 2, 20), 1000.0)(datetime.date(2023, 2, 10), 950.0)(datetime.date(2023, 2, 10), 2000.0)(datetime.date(2023, 2, 15), 2600.0)(datetime.date(2023, 2, 20), 5000.0)(datetime.date(2023, 2, 20), 1000.0)(datetime.date(2023, 2, 10), 950.0)
(datetime.date(2023, 2, 10), 2000.0)(datetime.date(2023, 2, 15), 2600.0)(datetime.date(2023, 2, 20), 5000.0)(datetime.date(2023, 3, 20), 1000.0)(datetime.date(2023, 3, 10), 750.0)(datetime.date(2023, 3, 10), 800.0)(datetime.date(2023, 3, 20), 700.0)(datetime.date(2023, 3, 20), 1000.0)(datetime.date(2023, 3, 10), 750.0)(datetime.date(2023, 3, 10), 800.0)(datetime.date(2023, 3, 15), 900.0)(datetime.date(2023, 3, 20), 700.0)(datetime.date(2023, 3, 20), 1000.0)(datetime.date(2023, 3, 10), 950.0)(datetime.date(2023, 3, 10), 2000.0)(datetime.date(2023, 3, 15), 2600.0)(datetime.date(2023, 3, 20), 5000.0)(datetime.date(2023, 3, 20), 1000.0)(datetime.date(2023, 3, 15), 2600.0)(datetime.date(2023, 3, 20), 5000.0)(datetime.date(2023, 4, 20), 1000.0)(datetime.date(2023, 4, 10), 750.0)(datetime.date(2023, 4, 10), 800.0)
(datetime.date(2023, 4, 15), 900.0)(datetime.date(2023, 4, 20), 700.0)(datetime.date(2023, 4, 20), 1000.0)(datetime.date(2023, 4, 10), 750.0)(datetime.date(2023, 4, 10), 800.0)(datetime.date(2023, 4, 15), 900.0)
(datetime.date(2023, 4, 20), 700.0)(datetime.date(2023, 3, 20), 1000.0)(datetime.date(2023, 4, 10), 950.0)(datetime.date(2023, 4, 10), 2000.0)(datetime.date(2023, 4, 15), 2600.0)(datetime.date(2023, 4, 20), 5000.0)(datetime.date(2023, 4, 20), 1000.0)(datetime.date(2023, 4, 10), 950.0)(datetime.date(2023, 4, 10), 2000.0)(datetime.date(2023, 4, 15), 2600.0)(datetime.date(2023, 4, 20), 5000.0)

### imoveis_aluguel_total.html

[1, 12450.0, 2, 24250.0, 3, 58900.0, 4, 90600.0]

### imoveis_pagto_mensal.html

['02/2023', 31650.0, '03/2023', 60200.0, '04/2023', 90600.0]

### imoveis_todos_cadastro.html

(7, datetime.date(2023, 2, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa')(8, datetime.date(2023, 2, 10), 750.0, 1, '2 quartos', 1, 'Casa')(9, datetime.date(2023, 2, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa')(10, datetime.date(2023, 2, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa')(11, datetime.date(2023, 2, 20), 700.0, 1, '3 quartos', 1, 'Casa')(27, datetime.date(2023, 3, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa')(28, datetime.date(2023, 3, 10), 750.0, 1, '2 quartos', 1, 'Casa')(29, datetime.date(2023, 3, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa')(30, datetime.date(2023, 3, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa')(31, datetime.date(2023, 3, 20), 700.0, 1, '3 quartos', 1, 'Casa')(47, datetime.date(2023, 4, 20), 1000.0, 1, '3 quartos e suite', 1, 'Casa')(48, datetime.date(2023, 4, 10), 750.0, 1, '2 quartos', 1, 'Casa')(49, datetime.date(2023, 4, 10), 800.0, 1, '4 quartos, 1 suite', 1, 'Casa')(50, datetime.date(2023, 4, 15), 900.0, 1, '4 quartos, com suite', 1, 'Casa')(51, datetime.date(2023, 4, 20), 700.0, 1, '3 quartos', 1, 'Casa')(1, datetime.date(2023, 2, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento')(2, datetime.date(2023, 2, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento')(4, datetime.date(2023, 2, 10), 800.0, 2, '4 quartos', 2, 'Apartamento')(5, datetime.date(2023, 2, 15), 900.0, 2, ' 4 quartos, com suite', 2, 'Apartamento')(6, datetime.date(2023, 2, 20), 700.0, 2, '3 quartos', 2, 'Apartamento')(22, datetime.date(2023, 3, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento')(23, datetime.date(2023, 3, 10), 750.0, 2, '2 quartos', 2, 'Apartamento')(24, datetime.date(2023, 3, 10), 800.0, 2, '4 quartos', 2, 'Apartamento')(26, datetime.date(2023, 3, 20), 700.0, 2, '3 quartos', 2, 'Apartamento')(42, datetime.date(2023, 4, 20), 1000.0, 2, '3 quartos', 2, 'Apartamento')(43, datetime.date(2023, 4, 10), 750.0, 2, '2 quartos', 2, 'Apartamento')(44, datetime.date(2023, 4, 10), 800.0, 2, '4 quartos', 2, 'Apartamento')(45, datetime.date(2023, 4, 15), 900.0, 2, '4 quartos, com suite', 2, 'Apartamento')(46, datetime.date(2023, 4, 20), 700.0, 2, '3 quartos', 2, 'Apartamento')(12, datetime.date(2023, 2, 20), 1000.0, 3, '100 metros quadrado', 3, 'Sala Comercial')(13, datetime.date(2023, 2, 10), 950.0, 3, '50 metros quadrado', 3, 'Sala Comercial')(14, datetime.date(2023, 2, 10), 2000.0, 3, '150 metros quadrado', 3, 'Sala Comercial')(15, datetime.date(2023, 2, 15), 2600.0, 3, '200 metros quadrado', 3, 'Sala Comercial')(16, datetime.date(2023, 2, 20), 5000.0, 3, '3500 metros quadrado', 3, 'Sala Comercial')(32, datetime.date(2023, 3, 20), 1000.0, 3, '100 metros quadrado', 3, 'Sala Comercial')(33, datetime.date(2023, 3, 10), 950.0, 3, '50 metros quadrado', 3, 'Sala Comercial')(34, datetime.date(2023, 3, 10), 2000.0, 3, '150 metros quadrado', 3, 'Sala Comercial')(35, datetime.date(2023, 3, 15), 2600.0, 3, '200 metros quadrado', 3, 'Sala Comercial')(36, datetime.date(2023, 3, 20), 5000.0, 3, '3500 metros quadrado', 3, 'Sala Comercial')(52, datetime.date(2023, 3, 20), 1000.0, 3, '100 metros quadrado', 3, 'Sala Comercial')(53, datetime.date(2023, 4, 10), 950.0, 3, '50 metros quadrado', 3, 'Sala Comercial')(54, datetime.date(2023, 4, 10), 2000.0, 3, '150 metros quadrado', 3, 'Sala Comercial')(55, datetime.date(2023, 4, 15), 2600.0, 3, '200 metros quadrado', 3, 'Sala Comercial')(56, datetime.date(2023, 4, 20), 5000.0, 3, '3500 metros quadrado', 3, 'Sala Comercial')(17, datetime.date(2023, 2, 20), 1000.0, 4, '100 metros quadrado', 4, 'Galpão')(18, datetime.date(2023, 2, 10), 950.0, 4, '50 metros quadrado', 4, 'Galpão')(19, datetime.date(2023, 2, 10), 2000.0, 4, '150 metros quadrado', 4, 'Galpão')(20, datetime.date(2023, 2, 15), 2600.0, 4, '200 metros quadrado', 4, 'Galpão')(21, datetime.date(2023, 2, 20), 5000.0, 4, '3500 metros quadrado', 4, 'Galpão')(37, datetime.date(2023, 3, 20), 1000.0, 4, '100 metros quadrado', 4, 'Galpão')(40, datetime.date(2023, 3, 15), 2600.0, 4, '200 metros quadrado', 4, 'Galpão')(41, datetime.date(2023, 3, 20), 5000.0, 4, '3500 metros quadrado', 4, 'Galpão')(57, datetime.date(2023, 4, 20), 1000.0, 4, '100 metros quadrado', 4, 'Galpão')(58, datetime.date(2023, 4, 10), 950.0, 4, '50 metros quadrado', 4, 'Galpão')(59, datetime.date(2023, 4, 10), 2000.0, 4, '150 metros quadrado', 4, 'Galpão')(60, datetime.date(2023, 4, 15), 2600.0, 4, '200 metros quadrado', 4, 'Galpão')(61, datetime.date(2023, 4, 20), 5000.0, 4, '3500 metros quadrado', 4, 'Galpão')

### imoveis_todos_porcetagem.html

[1, 23.549999999999997, 2, 21.979999999999997, 3, 23.549999999999997, 4, 20.409999999999997]

# Considerações

Com base na solicitação do Fórum Temático, foi criado o banco de dados, e a na linguagem de programação foi feito a conexão com o banco e um SELECT com INNER JOIN para trazer uma consulta básica.

Estas consultas geraram saída em arquivos html no formato json para a equipe de FRONT END, poder criar gráficos solicitados pela empresa.