Guia 1

Projeto Final de Banco de Dados - Linhas de Ônibus da Cidade do Rio de Janeiro

**Nomes**:

[Davi Mattos](mailto:davism@dcc.ufrj.br) - 119133049

Guilherme Charret de Vasconcellos - 121053093

**Matéria**: ICP489 - Banco de Dados I

**Profª**: [Vivian dos Santos Silva](mailto:vivian@dcc.ufrj.br)

# Sumário

[**Sumário 2**](#_ep6py0lhq3v8)

[**1. Introdução 3**](#_7pb1uyt2msd1)

[**2. Datasets 3**](#_pdypvir8oms0)

[**3. Projeto do Banco de Dados 6**](#_q1ojc7ywoltc)

[a. Modelo Conceitual 6](#_xdbvdu3m30bq)

[b. Modelo Lógico 6](#_n628sqa4gjnr)

[c. Modelo Físico 7](#_8p96l7wei09e)

[● Consórcio 7](#_d7qgjlwyzotj)

[● Escala 8](#_775jdjv2as59)

[● Tarifa 9](#_yh2w8y614hyp)

[● Linha 10](#_vy5v4zpg1g93)

[● Viagem 11](#_d6cp95tbns1w)

[● Pontos de Ônibus 12](#_ykzwyqnm7blq)

[● Pontos de Parada 13](#_n2l63nvoxtee)

[● Tarifa - Consorcio 14](#_u68oct6iksj)

[Código SQL Completo do Modelo Físico 15](#_27xy9hmoxz9n)

[d. Consultas 18](#_zfn12jqhffjj)

[i. Número de Linhas por consórcio 18](#_2t2ny5xudzq0)

[ii. Quantidade de linhas por valor de tarifa 19](#_ozout02fe0s)

[iii. Status das linhas de ônibus 20](#_1np1hrdq2mei)

[iv. Linhas que não operam aos finais de semana 21](#_ydux9gil9pcm)

[v. Status dos pontos de ônibus 22](#_oyjjwyj8u1l6)

[vi. Trajeto da linha (Consulta Interativa) 23](#_yydi1e6grk2z)

[vii. Destinos da linha (Consulta Interativa) 24](#_izbbwtoyf8yp)

[viii. Linhas que passam por determinado ponto de ônibus 24](#_qaupqupfb4pa)

[ix. Quantidade de viagens associadas a cada consórcio 25](#_m1rs90cx7k8x)

[x. Valor médio da tarifa 26](#_xr9zkw2vo4ne)

[xi. Valor da tarifa por tipo de ônibus 26](#_8fairvnbmsac)

[xii. Linhas Experimentais 27](#_n3c1uqffh2dv)

[xiii. Linhas com serviço variante 28](#_pk42ldjm49k9)

[xiv. Linhas com serviço parcial 28](#_y36v0wixvjr7)

[**4. Aplicação 29**](#_gyncqdu3dnbl)

[**5. Distribuição do Trabalho 32**](#_jhx9vr4u1bbz)

[**6. Considerações Finais 32**](#_vc03p7hjki0v)

[**7. Referências e Links 32**](#_q7v8p8yp1lf4)

# Introdução

O trabalho tem como propósito elucidar informações relacionadas ao transporte público terrestre, ônibus e BRT da cidade do Rio de Janeiro, através de um dashboard integrado com um banco de dados. Para popular o banco utilizaremos dois datasets, GTFS (Especificação Geral sobre Feeds de Transporte Público) da cidade do Rio que é atualizado mensalmente pela Secretaria Municipal dos Transportes, e consiste em uma coleção de *.csv*, e o Itinerário da linhas de ônibus da Cidade do Rio de Janeiro, que consiste em um único arquivo *.csv*.

# Datasets

Os dados foram pegos do site da prefeitura do rio o DATA.RIO e apresenta dados referentes a rotas, paradas e horários de serviços de ônibus (urbanos e executivos) e BRTs. Segue os links para os datasets:

* GTFS do Rio de Janeiro: <https://www.data.rio/datasets/8ffe62ad3b2f42e49814bf941654ea6c/about>

Esse dataset em arquivo comprimido (*.zip*) que contém diversos dados em formato *.csv* sobre rotas, paradas e horários de serviços de ônibus (urbanos e executivos) e BRTs da cidade do Rio de Janeiro

Deste dataset nós optamos por utilizar os seguintes dados e seus campos:

1. Consórcio (*Agency*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id\_consorcio | int | 22003 |
| nome\_consorcio | string | “Intersul” |
| site | varchar | “http://www.rioonibus.com/” |

1. Tarifas (*Fare Attributes*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id\_tarifa | char | “BRT” |
| valor | Decimal | 4.7 |

1. Linha (*Routes*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id\_linha | varchar | “O0636AAA0A” |
| numero | varchar | “636” |
| nome | varchar | “Merck - Saens Peña” |
| descricao | varchar | “LECD94” |
| tipo | int | 700 |

1. Escala (*Calendar*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id\_escala | varchar | “D\_REG” |
| seg\_sex | bool | 0 ou 1 |
| sab\_dom | bool | 0 ou 1 |

1. Viagem (*Trip*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id\_viagem | varchar | “eaa08f9e-76a4-4736-acc4-d3cacaed4ec3” |
| destino | string | “Madureira” |
| sentido | bool | 0 ou 1 |

1. Pontos de Ônibus (*Stops*)

| Campo | Tipo | Valor Típico |
| --- | --- | --- |
| id | varchar | “1003VZ0052E9” |
| nome | string | “AquaRio” |
| ponto\_vizinho | varchar | “1003VZ0052E9” |
| cod\_plataforma | string | “Central |

* Itinerários de Serviços de Ônibus Regulares: <https://www.data.rio/datasets/1bf2032a693746a68b70c2a3b2544859_1/explore?location=-22.925706%2C-43.441100%2C9.93>

Esse dataset consiste em um arquivo *.csv* acerca dos itinerários dos ônibus regulares, contendo os seguintes dados

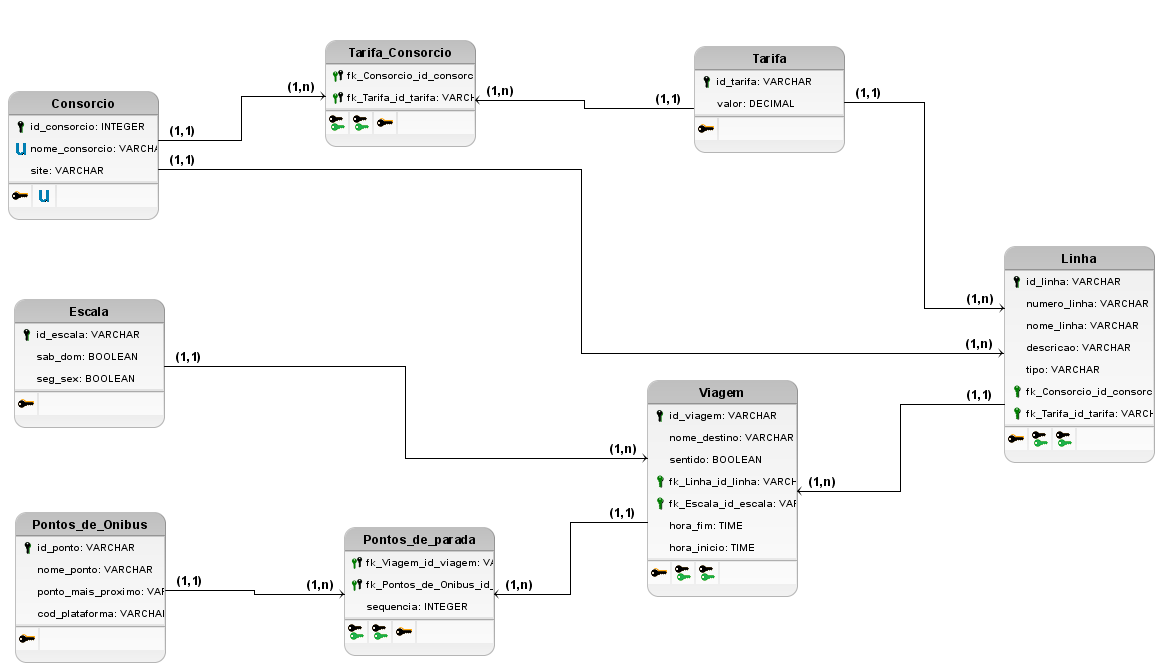
| Campo | Tipo | Valor típico |
| --- | --- | --- |
| id | int | Número decimal |
| extensao | int | 10980 |
| consorcio | string | Texto |
| descricao\_desvio | string | Texto |
| tipo\_rota | string | (“BRT”, “Regular”, “Frescao”) |
| direcao | bool | 0 ou 1 |
| destino | string | Nome |
| servico | varchar | Números ou Letras seguidos de números |

# Projeto do Banco de Dados

## Modelo Conceitual

## 

## Modelo Lógico



## Modelo Físico

### Consorcio

| Código SQL | |
| --- | --- |
| CREATE TABLE Consorcio (  id\_consorcio INTEGER PRIMARY KEY,  nome\_consorcio VARCHAR(255) UNIQUE,  site VARCHAR(255)  ); | |
|  |  |

### Escala

| Código SQL | |
| --- | --- |
| CREATE TABLE Escala (  id\_escala VARCHAR(255) PRIMARY KEY,  seg\_sex TINYINT(1),  sab\_dom TINYINT(1)  ); | |
|  |  |

### Tarifa

| Código SQL | |
| --- | --- |
| CREATE TABLE Tarifa (  id\_tarifa VARCHAR(255) PRIMARY KEY,  valor DECIMAL(10,2)  ); | |
|  |  |

### Linha

| Código SQL | |
| --- | --- |
| CREATE TABLE Linha (  id\_linha VARCHAR(255) PRIMARY KEY,  numero\_linha VARCHAR(255),  nome\_linha VARCHAR(255),  descricao VARCHAR(255),  tipo VARCHAR(255),  fk\_id\_consorcio INTEGER,  fk\_id\_tarifa VARCHAR(255)  ); | |
|  | |
|  | |

### Viagem

| Código SQL | |
| --- | --- |
| CREATE TABLE Viagem (  id\_viagem VARCHAR(255) PRIMARY KEY,  nome\_destino VARCHAR(255),  sentido TINYINT(1),  fk\_id\_linha VARCHAR(255),  fk\_id\_escala VARCHAR(255),  hora\_inicio TIME,  hora\_fim TIME  ); | |
|  | |
|  | |

### Pontos de Ônibus

| Código SQL | |
| --- | --- |
| CREATE TABLE Pontos\_de\_Onibus (  id\_ponto VARCHAR(255) PRIMARY KEY,  nome\_ponto VARCHAR(255),  ponto\_mais\_proximo VARCHAR(255),  cod\_plataforma VARCHAR(255)  ); | |
|  | |
|  | |

### Pontos de Parada

| Código SQL | |
| --- | --- |
| CREATE TABLE Pontos\_de\_parada (  fk\_id\_viagem VARCHAR(255),  fk\_id\_ponto VARCHAR(255),  sequencia INTEGER,  PRIMARY KEY (fk\_id\_viagem, fk\_id\_ponto)  ); | |
|  | |
|  | |

### Tarifa - Consorcio

| Código SQL | |
| --- | --- |
| CREATE TABLE Tarifa\_Consorcio (  fk\_id\_consorcio INTEGER,  fk\_id\_tarifa VARCHAR(255),  PRIMARY KEY (fk\_id\_consorcio, fk\_id\_tarifa)  ); | |
|  | |
|  | |

### Código SQL Completo do Modelo Físico

CREATE TABLE Consorcio (

id\_consorcio INTEGER PRIMARY KEY,

nome\_consorcio VARCHAR(255) UNIQUE,

site VARCHAR(255)

);

CREATE TABLE Escala (

id\_escala VARCHAR(255) PRIMARY KEY,

seg\_sex TINYINT(1),

sab\_dom TINYINT(1)

);

CREATE TABLE Tarifa (

id\_tarifa VARCHAR(255) PRIMARY KEY,

valor DECIMAL(10,2)

);

CREATE TABLE Linha (

id\_linha VARCHAR(255) PRIMARY KEY,

numero\_linha VARCHAR(255),

nome\_linha VARCHAR(255),

descricao VARCHAR(255),

tipo VARCHAR(255),

fk\_id\_consorcio INTEGER,

fk\_id\_tarifa VARCHAR(255)

);

CREATE TABLE Viagem (

id\_viagem VARCHAR(255) PRIMARY KEY,

nome\_destino VARCHAR(255),

sentido TINYINT(1),

fk\_id\_linha VARCHAR(255),

fk\_id\_escala VARCHAR(255),

hora\_inicio TIME,

hora\_fim TIME

);

CREATE TABLE Pontos\_de\_Onibus (

id\_ponto VARCHAR(255) PRIMARY KEY,

nome\_ponto VARCHAR(255),

ponto\_mais\_proximo VARCHAR(255),

cod\_plataforma VARCHAR(255)

);

CREATE TABLE Pontos\_de\_parada (

fk\_id\_viagem VARCHAR(255),

fk\_id\_ponto VARCHAR(255),

sequencia INTEGER,

PRIMARY KEY (fk\_id\_viagem, fk\_id\_ponto)

);

CREATE TABLE Tarifa\_Consorcio (

fk\_id\_consorcio INTEGER,

fk\_id\_tarifa VARCHAR(255),

PRIMARY KEY (fk\_id\_consorcio, fk\_id\_tarifa)

);

ALTER TABLE Linha ADD CONSTRAINT FK\_Linha\_2

FOREIGN KEY (fk\_id\_consorcio)

REFERENCES Consorcio (id\_consorcio);

ALTER TABLE Linha ADD CONSTRAINT FK\_Linha\_3

FOREIGN KEY (fk\_id\_tarifa)

REFERENCES Tarifa (id\_tarifa);

ALTER TABLE Viagem ADD CONSTRAINT FK\_Viagem\_2

FOREIGN KEY (fk\_id\_linha)

REFERENCES Linha (id\_linha);

ALTER TABLE Viagem ADD CONSTRAINT FK\_Viagem\_3

FOREIGN KEY (fk\_id\_escala)

REFERENCES Escala (id\_escala);

ALTER TABLE Pontos\_de\_parada ADD CONSTRAINT FK\_Pontos\_de\_parada\_1

FOREIGN KEY (fk\_id\_viagem)

REFERENCES Viagem (id\_viagem);

ALTER TABLE Pontos\_de\_parada ADD CONSTRAINT FK\_Pontos\_de\_parada\_2

FOREIGN KEY (fk\_id\_ponto)

REFERENCES Pontos\_de\_Onibus (id\_ponto);

ALTER TABLE Tarifa\_Consorcio ADD CONSTRAINT FK\_Tarifa\_Consorcio\_1

FOREIGN KEY (fk\_id\_consorcio)

REFERENCES Consorcio (id\_consorcio);

ALTER TABLE Tarifa\_Consorcio ADD CONSTRAINT FK\_Tarifa\_Consorcio\_2

FOREIGN KEY (fk\_id\_tarifa)

REFERENCES Tarifa (id\_tarifa);

## Consultas

### Número de Linhas por consórcio

Retornar a quantidade de linhas agrupadas por cada consórcio

| SELECT nome\_consorcio, COUNT(linhas.numero\_linha) as qtd\_linhas  FROM consorcio JOIN (  SELECT DISTINCT numero\_linha, fk\_id\_consorcio FROM linha  ) AS linhas  ON id\_consorcio = linhas.fk\_id\_consorcio  GROUP BY nome\_consorcio  ORDER BY qtd\_linhas DESC |
| --- |
|  |

### Quantidade de linhas por valor de tarifa

| SELECT valor, COUNT(l.id\_linha) AS total\_linhas  FROM tarifa JOIN (  SELECT DISTINCT id\_linha, fk\_id\_tarifa  FROM linha  ) AS l  ON tarifa.id\_tarifa = l.fk\_id\_tarifa  GROUP BY valor  ORDER BY valor ASC; |
| --- |
|  |

### Status das linhas de ônibus

| SELECT  status\_linha,  COUNT(\*) AS quantidade  FROM (  SELECT  l.id\_linha,  CASE  WHEN COUNT(v.id\_viagem) > 0 THEN 'Ativa'  ELSE 'Inativa'  END AS status\_linha  FROM Linha l  LEFT JOIN Viagem v ON l.id\_linha = v.fk\_id\_linha  GROUP BY l.id\_linha  ) AS sub  GROUP BY status\_linha; |
| --- |
|  |

### 

### Linhas que **não** operam aos finais de semana

| SELECT l.numero\_linha, l.nome\_linha, l.tipo  FROM linha l  LEFT JOIN (  SELECT \* FROM viagem JOIN escala  ON viagem.fk\_id\_escala = escala.id\_escala  WHERE escala.sab\_dom = 1  ) AS v  ON l.id\_linha = v.fk\_id\_linha  WHERE v.id\_escala IS NULL  ORDER BY l.numero\_linha, l.tipo DESC; |
| --- |
|  |

### 

### Status dos pontos de ônibus

| SELECT status\_ponto, COUNT(\*) AS quantidade  FROM (  SELECT  p.id\_ponto,  CASE WHEN COUNT(pp.fk\_id\_viagem) > 0 THEN 'Ativos'  ELSE 'Desativados' END AS status\_ponto  FROM Pontos\_de\_Onibus p  LEFT JOIN Pontos\_de\_parada pp  ON p.id\_ponto = pp.fk\_id\_ponto  GROUP BY p.id\_ponto  ) AS sub  GROUP BY status\_ponto; |
| --- |
|  |

### 

### Trajeto da linha (Consulta Interativa)

| SELECT DISTINCT p.nome\_ponto, pp.sequencia  FROM pontos\_de\_parada pp JOIN pontos\_de\_onibus p  ON p.id\_ponto = pp.fk\_id\_ponto  WHERE pp.fk\_id\_viagem IN (  SELECT id\_viagem FROM linha JOIN viagem  ON id\_linha=fk\_id\_linha  WHERE numero\_linha = "610" AND sentido = 0  )  ORDER BY pp.sequencia ASC |
| --- |
|  |

### 

### Destinos da linha (Consulta Interativa)

| SELECT DISTINCT sentido, nome\_destino  FROM viagem  WHERE fk\_id\_linha = (  SELECT id\_linha FROM linha  WHERE numero\_linha = "610" LIMIT 1  ) ORDER BY sentido; |
| --- |
|  |

### Linhas que passam por determinado ponto de ônibus (Consulta interativa)

| SELECT DISTINCT l.numero\_linha, l.nome\_linha, v.nome\_destino  FROM linha l JOIN (  SELECT DISTINCT fk\_id\_linha, nome\_destino  FROM viagem JOIN (  SELECT fk\_id\_viagem FROM pontos\_de\_parada  JOIN pontos\_de\_onibus  ON fk\_id\_ponto = id\_ponto  WHERE nome\_ponto = "CMS Jorge Saldanha Bandeira de Mello"  ) AS pontos ON id\_viagem = pontos.fk\_id\_viagem  ) AS v ON l.id\_linha = v.fk\_id\_linha  ORDER BY l.nome\_linha; |
| --- |
|  |

### Quantidade de viagens associadas a cada consórcio

| SELECT c.nome\_consorcio, COUNT(v.id\_viagem) AS total\_viagens  FROM Consorcio c  JOIN Linha l ON c.id\_consorcio = l.fk\_id\_consorcio  JOIN Viagem v ON l.id\_linha = v.fk\_id\_linha  GROUP BY c.nome\_consorcio  ORDER BY total\_viagens DESC; |
| --- |
|  |

### 

### Valor médio da tarifa

| SELECT AVG(valor) AS media\_valor FROM Tarifa; |
| --- |
|  |

### Valor da tarifa por tipo de ônibus

| SELECT DISTINCT  CASE l.tipo  WHEN 'regular' THEN 'Regular'  WHEN 'brt' THEN 'BRT'  WHEN '700' THEN 'Especial'  WHEN 'frescao' THEN 'Frescão'  ELSE l.tipo  END AS tipo\_formatado,  t.valor  FROM Linha l  JOIN Tarifa t ON l.fk\_id\_tarifa = t.id\_tarifa  WHERE l.tipo IS NOT NULL AND l.tipo != ''  ORDER BY t.valor; |
| --- |
|  |

### Linhas Experimentais

| SELECT l.numero\_linha, l.nome\_linha  FROM linha l  WHERE l.numero\_linha LIKE "LECD%" |
| --- |
|  |

### 

### Linhas com serviço variante

| SELECT l.numero\_linha, l.nome\_linha  FROM linha l  WHERE l.numero\_linha LIKE "SV%"  ORDER BY l.numero\_linha, l.nome\_linha |
| --- |
|  |

### Linhas com serviço parcial

| SELECT l.numero\_linha, l.nome\_linha  FROM linha l  WHERE l.numero\_linha LIKE "SP%"  ORDER BY l.numero\_linha, l.nome\_linha |
| --- |
|  |

# Aplicação

A criação da API é o coração do projeto. Ela funciona como uma camada de serviço que conecta a interface web a um banco de dados MySQL. Para isso, no back-end, usamos Python como linguagem principal, e o microframework Flask para desenvolver as rotas RESTful. A biblioteca SQLAlchemy foi importante para garantir que as consultas ao banco fossem feitas de forma segura. Além disso, o Pandas foi essencial para transformar os dados obtidos em JSON de maneira eficiente, facilitando o uso por parte do front-end. Essa estrutura permitiu que os dados fossem entregues de forma segura e organizada, simplificando a interação com o banco de dados. Já para a interface do usuário do painel foi criada usando as principais tecnologias da web: HTML5, que organiza o conteúdo de forma clara; CSS3, que dá o visual e organiza o layout em uma grade; e JavaScript, que controla toda a interatividade. Para mostrar os dados de forma visual, usamos a biblioteca Chart.js, que facilita a criação de gráficos de pizza e barras para tornar as informações mais fáceis de entender.

O desenvolvimento da API aconteceu de forma gradual. Inicialmente, ela era estática que fornecia dados agrupados para alimentar gráficos e tabelas. Depois, foi adicionada a interação com o usuário, para isso foram adicionadas rotas dinâmicas que aceitavam parâmetros, como o número de uma linha ou o nome de um ponto específico.

Link para o vídeo da aplicação rodando: [Dashboard da linhas regulares do Rio de Janeiro](https://drive.google.com/file/d/18aAMtJDttSN6IU7Rz-Wiz0XrwDMcJAAv/view?usp=sharing)

|  |
| --- |
| *Imagem 1: API rodando localmente (*[*Valor Médio*](#_xr9zkw2vo4ne)*,* [*Quantidade de Linhas x Tarifa*](#_ozout02fe0s) *e* [*Tarifas x Tipo*](#_8fairvnbmsac)*)* |

|  |
| --- |
| *Imagem 2:* [*Linhas x Consórcio*](#_2t2ny5xudzq0) *e* [*Total de Viagens x Consórcio*](#_m1rs90cx7k8x) |

|  |
| --- |
| *Imagem 3:* [*Status das linhas de ônibus*](#_1np1hrdq2mei) *e* [*Status dos pontos de ônibus*](#_oyjjwyj8u1l6) |

|  |
| --- |
| *Imagem 4: Buscas interativas* |

* Consultar itinerário da linha:

Ao digitar o número da linha e clicar em “Buscar”, é realizada uma [consulta do itinerário](#_yydi1e6grk2z) e os [destinos da linha](#_izbbwtoyf8yp) que aparecerem no menu suspenso, então é exibido o primeiro trajeto registrado no banco de dados da linha.

* Busca linhas por ponto de parada:

Ao digitar o nome de um ponto de ônibus é realizada uma [consulta](#_qaupqupfb4pa), que retorna todos os ônibus em que seu trajeto realiza parada no determinado ponto.

|  |
| --- |
| *Imagem 5:* [*Linhas que não realizam viagem aos fins de semana*](#_ydux9gil9pcm) |

# 

# Distribuição do Trabalho

Como o trabalho foi feito em dupla, ambos contribuíram ativamente em cada parte do trabalho, porém, ficou definido que cada integrante ficasse responsável pelos seguintes tópicos abaixo, enquanto outro integrante ficaria responsável pela revisão (do tópico o qual não era responsável).

* Davi Mattos:
  + Escolha do dataset
  + Modelagem conceitual
  + Limpar e Tratar os datasets
  + Popular o banco
  + Aprimorar as consultas para a API
  + Integrar a API ao Banco (Usando SQLAlchemy)
  + Criar o template dos relatórios
  + Preencher os relatórios (1, 2 e 4)
  + Criar o template da apresentação Final
* Guilherme Charret:
  + Apresentação da Parte 1
  + Modelo Lógico
  + Modelo Físico
  + Limpar e Tratar os datasets
  + Popular o banco
  + Desenvolver consultas para parte 3
  + Desenvolver a parte front-end da API (Utilizando Flask)
  + Preencher os relatórios (3, 4)
  + Montar a apresentação Final

Como dito anteriormente, apesar de termos dividido as tarefas, ambos participaram ativamente de cada parte do projeto, pois ao longo do desenvolvimento, devido ao calendário e agenda dos integrantes, se tornou muito difícil um único integrante ficar responsável por uma parte em específico.

# Considerações Finais

A maior dificuldade do projeto foi a modelagem, pois requeriu uma atenção minuciosa, houve inúmeras tentativas de modelagem até chegar na “versão final”, pois apesar de ser um dataset bem completo, não havia uma documentação sequer que auxiliasse e facilitasse no seu entendimento, então o único meio que sobrou, por assim dizer, foi explorar o dataset na mão e tentar entendê-lo, e para dar sequência no trabalho, optamos por filtrar bastante informações do dataset, e bem como limpá-lo, para casar com a modelagem que definimos. Ao longo da implementação da API, as consultas tiveram que ser repensadas e aprimoradas, e algumas até refeitas para que pudesse casar com aquilo que gostaríamos de exibir na aplicação. Apesar de todas as dificuldades, o trabalho cumpre com seu objetivo, claro que não está isento de erros e sempre propenso a melhorias, principalmente na relação escala-viagem. Uma possível melhoria posteriormente, seria mostrar os horários dos ônibus e seus intervalos, mas sempre se atentando aos dados, pois toda vez que o intervalo do ônibus altera, é criada uma nova viagem, motivo o qual optamos por não abordar no projeto no primeiro momento.

# Referências e Links

Todo código fonte se encontra disponível em: <https://github.com/DaviMattos14/Banco_de_Dados_API>.

SILVA, Vivian. Slides de Aula. [Apresentação de slides] – Banco de Dados, Bacharelado em Ciência da Computação. Rio de Janeiro: Universidade Federal do Rio de Janeiro

PREFEITURA DA CIDADE DO RIO DE JANEIRO. Data.rio: Portal de Dados Abertos da Prefeitura da Cidade do Rio de Janeiro. Disponível em: <https://www.data.rio>. Acesso em: 04 jul. 2025.

THE PANDAS DEVELOPMENT TEAM. pandas.: Pandas, 2025. Disponível em: <https://pandas.pydata.org>. Acesso em: 04 jul. 2025.

PALLETS PROJECTS. Flask. : Pallets Projects, 2025. Disponível em: <https://flask.palletsprojects.com>. Acesso em: 04 jul. 2025

GRINBERG, M. SQLAlchemy Documentation. SQLAlchemy, 2024. Disponível em: <https://docs.sqlalchemy.org/en/20/>. Acesso em: 5 jul. 2025.

ORACLE. MySQL Documentation. Oracle, 2024. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 5 jul. 2025.

PyMySQL. In: PYTHON PACKAGE INDEX (PyPI)]. Disponível em: <https://pypi.org/project/PyMySQL/>. Acesso em: 04 jul. 2025.

CHART.JS COMMUNITY. Chart.js – Simple yet flexible JavaScript charting. Disponível em: <https://www.chartjs.org/>. Acesso em: 5 jul. 2025.

ORACLE. MySQL Connector/Python. PyPI, 2024. Disponível em: <https://pypi.org/project/mysql-connector-python/>. Acesso em: 5 jul. 2025.

MDN WEB DOCS. CSS: Folhas de Estilo em Cascata.: Mozilla, 2025. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 04 jul. 2025.

MDN WEB DOCS. HTML: Linguagem de Marcação de Hipertexto.: Mozilla, 2025. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 04 jul. 2025.

MDN WEB DOCS. JavaScript.: Mozilla, 2025. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 04 jul. 2025.