



Faculdade de Design,
Tecnologia e Comunicação
Universidade Europeia

Curso de Licenciatura

LISBON SPOTS

ESTATÍSTICA

Dashboard

20220523-Abhay Kumar

20221605-Edja da Silva

Lisboa

2023-2024

20220523-Abhay Kumar

20221605-Edja da Silva

LISBON SPOTS

ESTATÍSTICA

Dashboard

Trabalho de estatística do segundo ano, no curso de engenharia informática da Faculdade de Design, Tecnologia e Comunicação.

Orientadores: Rodolfo Bendoyro.

Lisboa

2023-2024

“O melhor de uma imagem é que ela nunca muda, mesmo quando as pessoas nela mudam.”

-Andy Warhol

ÍNDICE

INTRODUÇÃO	8
Gráfico de Barras	9
Código Gráfico de Barras	11
Histograma	14
Código Histograma	17
Gráfico Circular	21
Código Gráfico Circular	23
Estatísticas	25
Código Estatísticas	25
Código Geral	28
Visão geral do Dashboard	31

INTRODUÇÃO

O presente relatório descreve em detalhe o processo de desenvolvimento do dashboard para o projeto "Lisbon Spots", um website concebido para satisfazer as necessidades dos entusiastas de fotografia, em especial os turistas em busca de experiências visuais memoráveis durante as suas estadias em Lisboa. O dashboard inclui três tipos de gráficos - **Gráfico de Barras, Gráfico Circular e Histograma** - e fornece três estatísticas relacionadas aos museus: o **melhor**, o **pior** e a **média das avaliações**.

Gráfico de Barras

Para o gráfico de barras, foi requisitado pelo professor o rating para cada categoria não comercial. É importante recordar que o nosso projeto é composto por spots, os quais incluem spots comerciais e não comerciais.

Os spots não comerciais herdam das características dos spots comerciais, conforme a seguinte estrutura de tabelas:

```
CREATE TABLE tb_nonCommercialSpot(  
    category nonComSpotCategory,  
    add_id int REFERENCES tb_address(id) on delete set null  
) INHERITS (tb_spot);
```

E os spots possuem a seguinte estrutura:

```
CREATE TABLE tb_spot (  
    id SERIAL primary key not null,  
    description TEXT,  
    name TEXT,  
    rating rating,  
    loc_id int REFERENCES tb_location(id) on delete set null  
);
```

Dentro das categorias dos spots não comerciais, temos as seguintes opções:

```
CREATE TYPE nonComSpotCategory AS ENUM('beach', 'park', 'river',  
'waterfall', 'mountain', 'church');
```

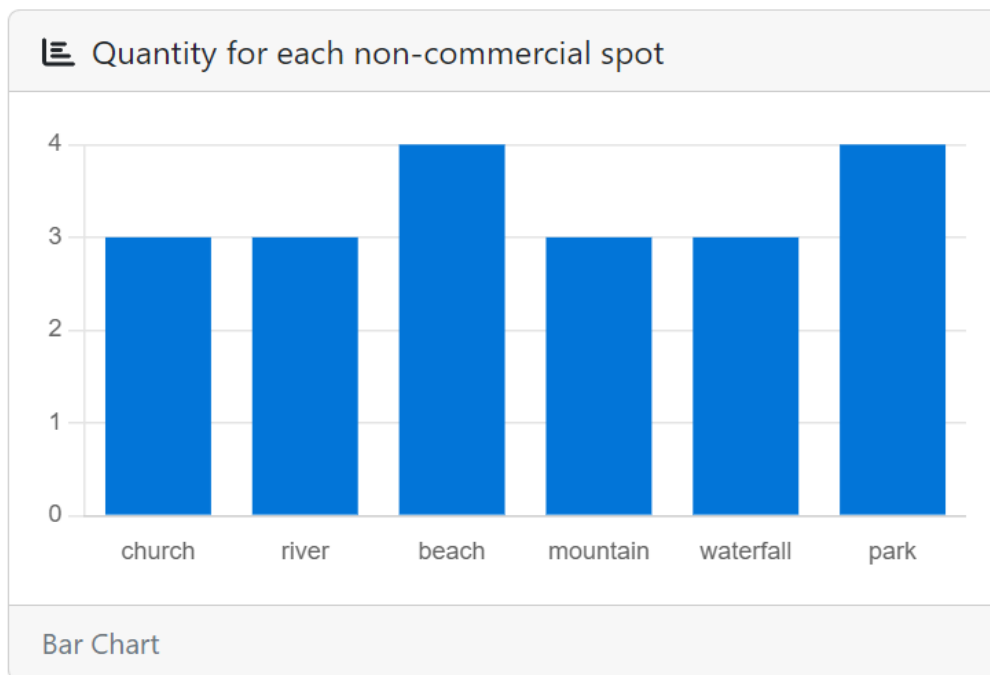


Gráfico de barras

	<div>category</div>	<div>count</div>
1	church	3
2	river	3
3	beach	4
4	mountain	3
5	waterfall	3
6	park	4

Gráfico de barras: pesquisa

Código Gráfico de Barras

HTML

```
<div class="row">

  <div class="col-lg-6">

    <!-- Cartão para o gráfico de barras -->

    <div class="card mb-4">

      <!-- Cabeçalho do cartão -->

      <div class="card-header">

        <i class="fas fa-chart-bar me-1"></i> <!-- Ícone do gráfico de barras -->

        Quantity for each non-commercial spot <!-- Título do gráfico -->

      </div>

      <!-- Corpo do cartão -->

      <div class="card-body">

        <canvas id="myBarChart" width="100%" height="50"></canvas> <!-- Elemento do gráfico de barras -->

      </div>

      <!-- Rodapé do cartão -->

      <div class="card-footer small text-muted">Bar Chart</div> <!-- Descrição do tipo de gráfico -->

    </div>

  </div>

</div>
```

JAVASCRIPT

```
// URL base para as requisições HTTP

const base_url = 'http://localhost:5555';

// Espera o carregamento completo do DOM antes de executar o código

document.addEventListener('DOMContentLoaded', async function() {

  // Faz uma requisição GET para a rota /dashboard/data/spots-by-category

  const response = await fetch(`${base_url}/dashboard/data/spots-by-category`, {

    headers: {"Content-Type": "application/json"}, // Define o cabeçalho Content-Type como application/json

    method: 'GET' // Usa o método GET

  });

  // Verifica se a resposta da requisição foi bem-sucedida
```



```

if(response) {

    // Converte a resposta para JSON e a armazena na variável data
    var data = await response.json();

} else {

    // Se a resposta não foi bem-sucedida, lança um erro com o status da resposta
    throw new Error(response.status);

}

// Imprime os dados recebidos da requisição no console do navegador
console.log(data);

// Gráfico de Barras

// Define o contexto do gráfico de barras com base no elemento HTML <canvas> com o ID myBarChart
const barCtx = document.getElementById('myBarChart').getContext('2d');

// Cria um novo gráfico de barras usando a biblioteca Chart.js
const myBarChart = new Chart(barCtx, {

    type: 'bar', // Tipo do gráfico: barra

    data: {

        labels: data.map(row => row.category), // Rótulos do gráfico com base nos dados recebidos da requisição

        datasets: [{

            label: 'Quantidade', // Rótulo do conjunto de dados

            backgroundColor: "rgba(2,117,216,1)", // Cor de fundo das barras

            borderColor: "rgba(2,117,216,1)", // Cor da borda das barras

            data: data.map(row => row.count), // Dados do gráfico com base nos dados recebidos da requisição

        ]},

    },

    options: {

        // Configurações das escalas x e y

        scales: {

            x: {

                grid: {

                    display: false // Não exibir grade no eixo x

                },

                ticks: {

                    maxTicksLimit: 6 // Limitar o número máximo de ticks no eixo x

                }

            },

        },

    },

```

```

y: {
  ticks: {
    minTicksLimit: 6, // Limitar o número mínimo de ticks no eixo y
    beginAtZero: true, // Iniciar o eixo y no valor zero
    maxTicksLimit: 6 // Limitar o número máximo de ticks no eixo y
  },
  grid: {
    display: true // Exibir grade no eixo y
  }
},
plugins: {
  legend: {
    display: false // Não exibir legenda
  }
}
});
});

```

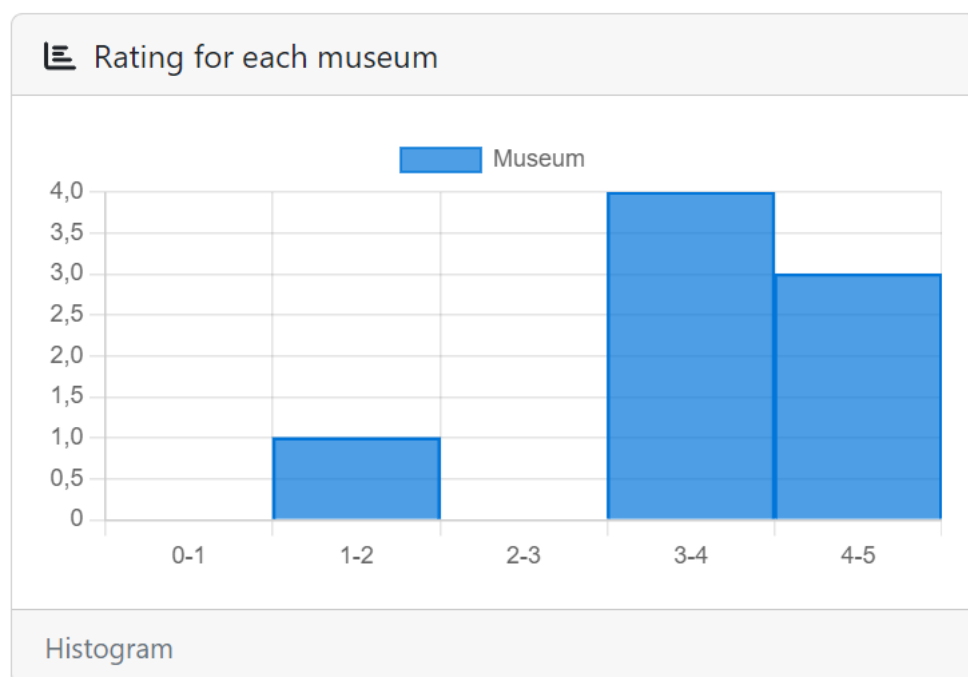
Histograma

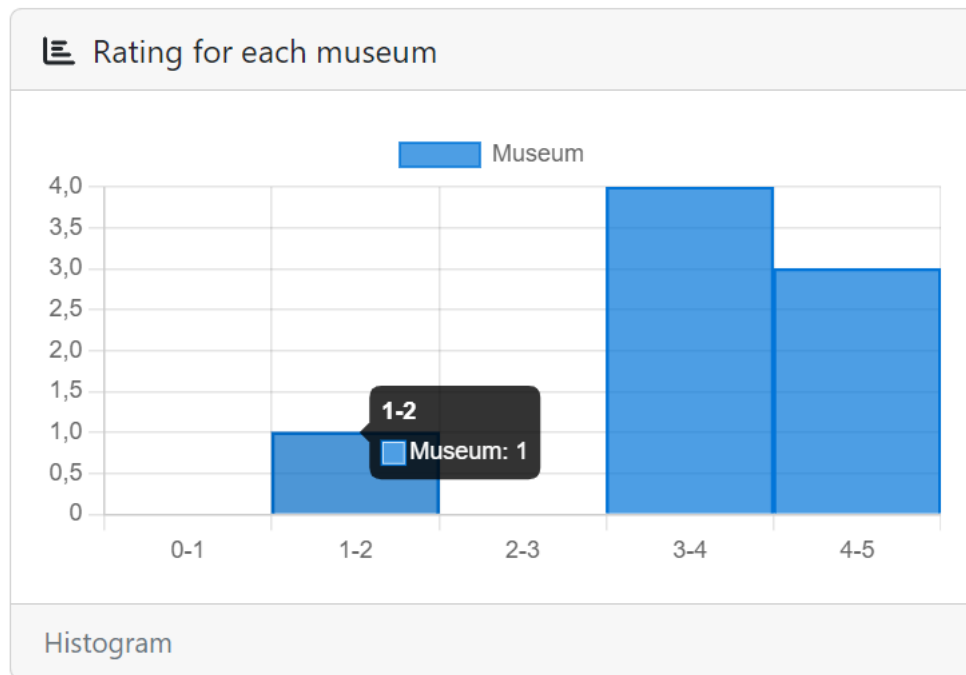
Para o Histograma, o professor requisitou o rating para cada museu. Como já foi explicado anteriormente, vou apenas apresentar os scripts para a tabela comercial e suas categorias.

Aqui está a definição da tabela comercial e os tipos de enumeração das suas categorias:

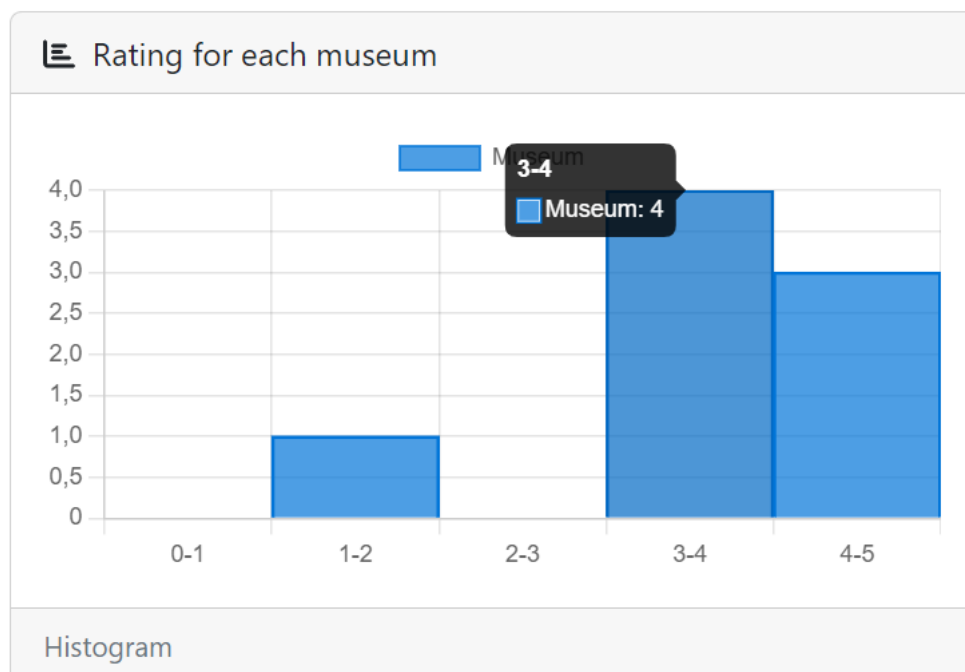
```
CREATE TABLE tb_commercialSpot (  
    category comSpotCategory,  
    add_id int REFERENCES tb_address(id) on delete set null,  
    tt_id int REFERENCES tb_timeTable on delete set null  
)INHERITS (tb_spot);  
  
CREATE TYPE comSpotCategory AS ENUM ('restaurant', 'hotel', 'stadium',  
'shopping', 'museum');
```

Essas são as estruturas das tabelas e enumerações utilizadas para os spots comerciais em nosso projeto. Essas informações serão empregadas na geração do histograma conforme solicitado pelo professor.

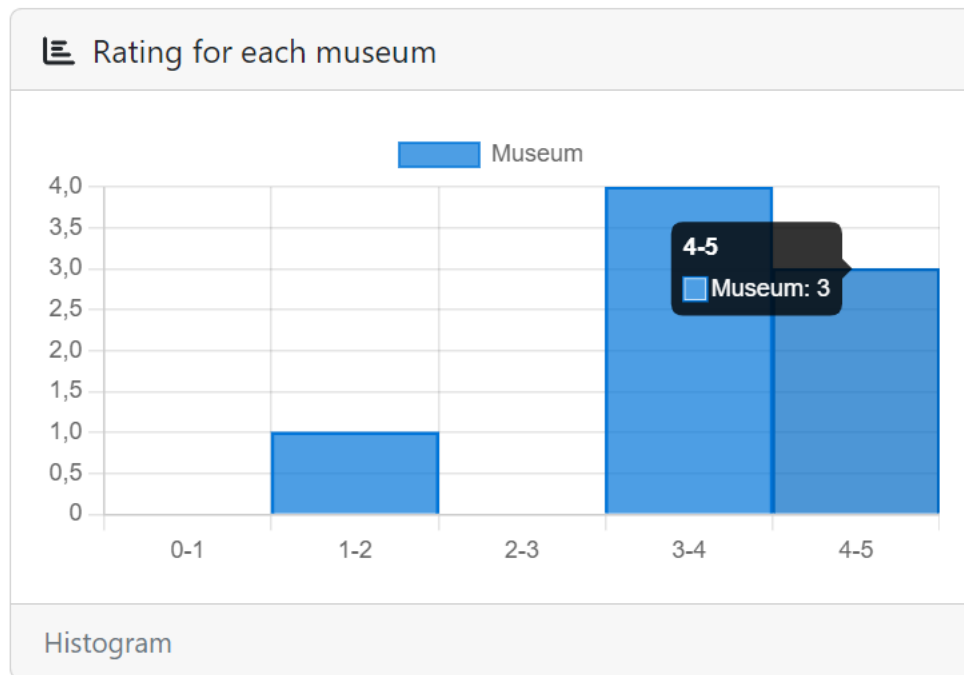




Histograma: um museu



Histograma: 4 museus



Histograma: 3 museus

	category ▼	rating ▼	name ▼
1	museum	2	Museu do Fado
2	museum	4	Museu Arqueológico Do Carmo
3	museum	4	Museu Do Azulejo De Lisboa
4	museum	4	MAAT - Museu De Arte, Arquitetura E Tecnologia
5	museum	4	Casa-Museu Medeiros e Almeida
6	museum	5	MUDE - Museu do Design e da Moda
7	museum	5	Museu Nacional de Arte Antiga
8	museum	5	Museu Calouste Gulbenkian

Histograma: Pesquisa

Código Histograma

HTML

```
<div class="col-lg-6 mx-auto">

  <!-- Div de coluna centralizada -->

  <div class="card mb-4">

    <!-- Cartão para o gráfico de histograma -->

    <div class="card-header">

      <i class="fas fa-chart-bar me-1"></i> <!-- Ícone do gráfico de barras -->

      Rating for each museum <!-- Título do gráfico -->

    </div>

    <!-- Corpo do cartão -->

    <div class="card-body">

      <canvas id="myHistogram" width="100%" height="50"></canvas> <!-- Elemento do gráfico de histograma -->

    </div>

    <!-- Rodapé do cartão -->

    <div class="card-footer small text-muted">Histogram</div> <!-- Descrição do tipo de gráfico -->

  </div>

</div>
```

JAVASCRIPT

```
// Define a fonte padrão e a cor para todos os gráficos

Chart.defaults.global.defaultFontFamily = 'apple-system,system-ui,BlinkMacSystemFont,"Segoe UI",Roboto,"Helvetica Neue",Arial,sans-serif';

Chart.defaults.global.defaultFontColor = '#292b2c';

// Aguarda o carregamento completo do DOM antes de executar o código

document.addEventListener('DOMContentLoaded', async function() {

  // Faz uma requisição GET para a rota /dashboard/data/spots-by-rating

  const response = await fetch(`${base_url}/dashboard/data/spots-by-rating`, {

    headers: {"Content-Type": "application/json"}, // Define o cabeçalho Content-Type como application/json

    method: 'GET' // Usa o método GET

  });

});
```

```

// Verifica se a resposta da requisição foi bem-sucedida
if(response) {

    // Converte a resposta para JSON e a armazena na variável data
    var data = await response.json();

} else {

    // Se a resposta não foi bem-sucedida, lança um erro com o status da resposta
    throw new Error(response.status);

}

// Imprime os dados recebidos da requisição no console do navegador
console.log(data);

// Intervalos dos bins
const bins = [0, 1, 2, 3, 4, 5];

// Função para calcular as frequências dos bins
function calculateBinFrequencies(data, bins) {

    const frequencies = new Array(bins.length - 1).fill(0);

    data.forEach(row => {

        for (let i = 0; i < bins.length - 1; i++) {

            if (row.rating >= bins[i] && row.rating <= bins[i + 1]) {

                frequencies[i]++;

                break;

            }

        }

    });

    return frequencies;

}

// Calcula as frequências dos bins com base nos dados e nos intervalos
const frequencies = calculateBinFrequencies(data, bins);

// Define o contexto do gráfico de histograma com base no elemento HTML <canvas> com o ID myHistogram
const histogramCtx = document.getElementById('myHistogram').getContext('2d');

// Cria um novo gráfico de barras (histograma) usando a biblioteca Chart.js
const hist = new Chart(histogramCtx, {

```

```

type: 'bar', // Tipo do gráfico: barra

data: {

  labels: bins.slice(0, -1).map((bin, index) => `${bin}-${bins[index + 1]}`), // Rótulos do gráfico com base nos intervalos dos bins

  datasets: [{

    label: 'Museu', // Rótulo do conjunto de dados

    data: frequencias, // Dados do gráfico com base nas frequências dos bins

    backgroundColor: 'rgba(2, 117, 216, 0.7)', // Cor de fundo das barras

    borderColor: 'rgba(2, 117, 216, 1)', // Cor da borda das barras

    borderWidth: 1, // Largura da borda das barras

    barPercentage: 1, // Percentual da largura das barras em relação ao espaço disponível

    categoryPercentage: 1 // Percentual da largura da categoria em relação ao espaço disponível

  ]

},

options: {

  scales: {

    x: {

      grid: {

        scaleLabel: {

          display: true,

          labelString: 'Museu' // Rótulo do eixo x

        },

        ticks: {

          maxTicksLimit: bins.length // Limita o número máximo de ticks no eixo x

        }

      },

      ticks: {

        beginAtZero: true // Começa o eixo y no valor zero

      },

      gridLines: {

        display: true // Exibe as linhas de grade no eixo y

      }

    }

  }

}

```



```
    },  
    legend: {  
      display: true // Exibe a legenda  
    }  
  }  
});  
  
});
```

Gráfico Circular

Para o gráfico circular, foi requisitado pelo professor o rating para cada categoria comercial.

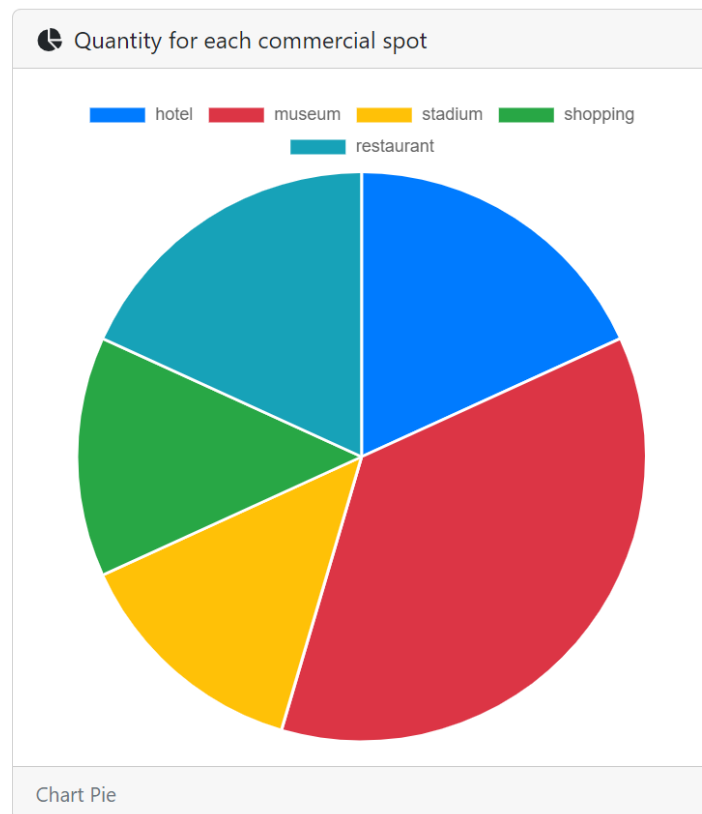


Gráfico circular

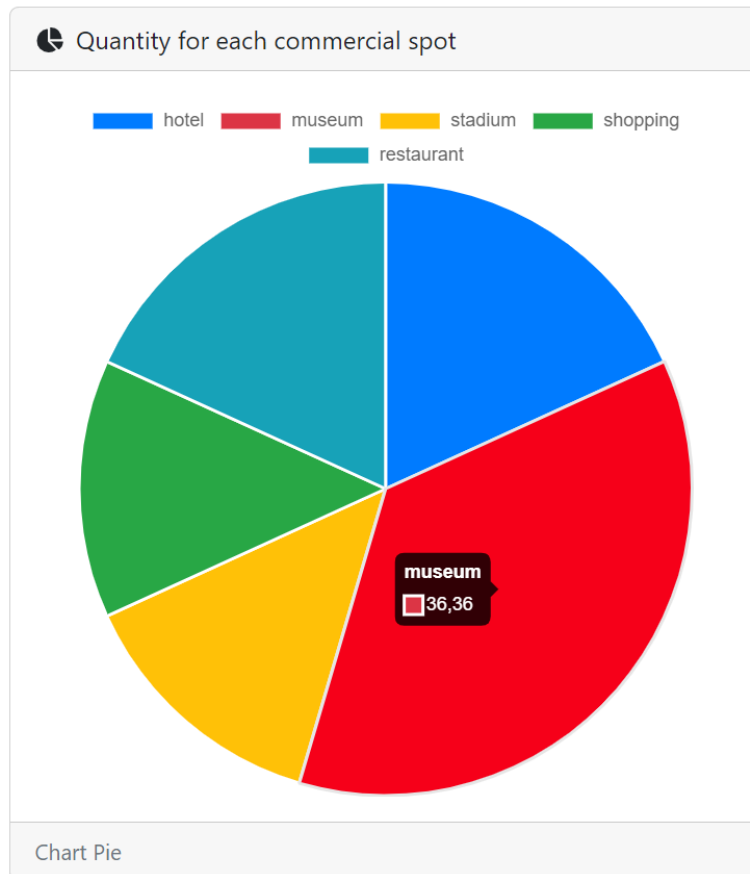


Gráfico circular: museu

	category ▾	count ▾	percentage ▾
1	hotel	4	18.18
2	museum	8	36.36
3	stadium	3	13.64
4	shopping	3	13.64
5	restaurant	4	18.18

Gráfico circular: pesquisa

Código Gráfico Circular

HTML

```
<div class="col-lg-6 mx-auto">

  <!-- Coluna do cartão do histograma -->

  <div class="card mb-4">

    <!-- Cabeçalho do cartão -->

    <div class="card-header">

      <i class="fas fa-chart-bar me-1"></i> <!-- Ícone do gráfico de barras -->

      Rating for each museum <!-- Título do gráfico -->

    </div>

    <!-- Corpo do cartão -->

    <div class="card-body">

      <canvas id="myHistogram" width="100%" height="50"></canvas> <!-- Elemento do gráfico de histograma -->

    </div>

    <!-- Rodapé do cartão -->

    <div class="card-footer small text-muted">Histogram</div> <!-- Descrição do tipo de gráfico -->

  </div>

</div>
```

JAVASCRIPT

```
// Define a nova fonte padrão e cor da fonte para mimetizar o estilo padrão do Bootstrap

Chart.defaults.global.defaultFontFamily = '-apple-system,system-ui,BlinkMacSystemFont,"Segoe UI",Roboto,"Helvetica Neue",Arial,sans-serif';

Chart.defaults.global.defaultFontColor = '#292b2c';

// Aguarda o carregamento completo do DOM antes de executar o código

document.addEventListener('DOMContentLoaded', async function() {

  // Faz uma requisição GET para a rota /dashboard/data/spots-by-comCategory

  const response = await fetch(`${base_url}/dashboard/data/spots-by-comCategory`, {

    headers: {"Content-Type": "application/json"}, // Define o cabeçalho Content-Type como application/json

    method: 'GET' // Usa o método GET

  });

  // Verifica se a resposta da requisição foi bem-sucedida

  if(response) {
```

```

// Converte a resposta para JSON e a armazena na variável data
var data = await response.json();

} else {

// Se a resposta não foi bem-sucedida, lança um erro com o status da resposta
throw new Error(response.status);

}

// Imprime os dados recebidos da requisição no console do navegador
console.log(data);



// Define o contexto do gráfico de pizza com base no elemento HTML <canvas> com o ID myPieChart
const pieCtx = document.getElementById('myPieChart').getContext('2d');

// Cria um novo gráfico de pizza usando a biblioteca Chart.js
const myPieChart = new Chart(pieCtx, {
  type: 'pie', // Tipo do gráfico: pizza
  data: {
    labels: data.map(row => row.category), // Rótulos do gráfico com base nos dados recebidos da requisição
    datasets: [{
      data: data.map(row => row.percentage ), // Dados do gráfico com base nas percentagens recebidas da requisição
      backgroundColor: ['#007bff', '#dc3545', '#ffc107', '#28a745', '#17a2b8', '#6c757d'], // Cores de fundo das fatias da pizza
    }],
  },
});

```

Estatísticas

Para as estatísticas, o professor requisitou o pior, melhor e o valor médio das avaliações de uma categoria a nossa escolha, nós escolhemos os museus.

Museum	Museum	Museum
Worst Rating	Average Rating Value	Best Rating
 2	 4.125	 5

Estatísticas: pesquisa

Código Estatísticas

HTML

```
<div class="container col-8 my-5">

  <!-- Container com largura 8 colunas e margem superior de 5 -->

  <div class="row">

    <!-- Linha para os elementos do dashboard -->

    {{#if max}}

    <!-- Verifica se há uma avaliação máxima (best rating) -->

    <div class="col order-last border border-3 border-success bg-light mx-1">

      <!-- Coluna para a avaliação máxima -->

      <div class="col-12 mt-2 ">

        <!-- Divisão com margem superior -->

        <p class="text-muted">Museum</p> <!-- Texto "Museum" em cinza -->

        <p class="text-secondary fw-light fs-5"> Best Rating</p> <!-- Texto "Best Rating" em cinza claro e menor tamanho de fonte -->

      </div>

    </div>

  </div>

</div>
```

```

        <p class="fs-4 mx-1 "><i class="fa fa-database my-0 text-secondary"></i> {{max}}</p><!-- Ícone de banco de dados e avaliação
máxima -->

    </div>

</div>

{{/if}}

{{#if avg}}

<!-- Verifica se há uma média de avaliação (average rating value) -->

<div class="col border border-3 border-warning bg-light">

    <!-- Coluna para a média de avaliação -->

    <div class="col-12 mt-2 ">

        <!-- Divisão com margem superior -->

        <p class="text-muted">Museum</p><!-- Texto "Museum" em cinza -->

        <p class="text-secondary fw-light fs-5"> Average Rating Value</p><!-- Texto "Average Rating Value" em cinza claro e menor
tamanho de fonte -->

        <p class="fs-4 mx-1 "><i class="fa fa-database my-0 text-secondary"></i> {{avg}}</p><!-- Ícone de banco de dados e valor médio
de avaliação -->

    </div>

</div>

{{/if}}

{{#if max}}

<!-- Verifica se há uma avaliação mínima (worst rating) -->

<div class="col order-first border border-3 border-danger bg-light mx-1">

    <!-- Coluna para a avaliação mínima -->

    <div class="col-12 mt-2 ">

        <!-- Divisão com margem superior -->

        <p class="text-muted">Museum</p><!-- Texto "Museum" em cinza -->

        <p class="text-secondary fw-light fs-5"> Worst Rating</p><!-- Texto "Worst Rating" em cinza claro e menor tamanho de fonte -->

        <p class="fs-4 mx-1 "><i class="fa fa-database my-0 text-secondary"></i> {{min}}</p><!-- Ícone de banco de dados e avaliação
mínima -->

    </div>

</div>

{{/if}}

</div>

</div>

```

JAVASCRIPT

```

// Rota GET para a raiz do servidor
server.get('/', async (req, res) => {
  try {
    // Consulta para obter a média das avaliações dos museus
    const avgResult = await database.query(`
      SELECT round(AVG(CAST(CAST(rating AS text) AS integer)), 3) AS avg_rating
      FROM tb_commercialspot
      WHERE category = 'museum';
    `);
    const avgRating = avgResult.rows[0].avg_rating;

    // Consulta para obter a avaliação máxima dos museus
    const maxResult = await database.query(`
      SELECT MAX(CAST(CAST(rating AS text) AS integer)) AS max_rating
      FROM tb_commercialspot
      WHERE category = 'museum';
    `);
    const maxRating = maxResult.rows[0].max_rating;

    // Consulta para obter a avaliação mínima dos museus
    const minResult = await database.query(`
      SELECT MIN(CAST(CAST(rating AS text) AS integer)) AS min_rating
      FROM tb_commercialspot
      WHERE category = 'museum';
    `);
    const minRating = minResult.rows[0].min_rating;

    // Renderiza a página 'dashboard' com os dados obtidos
    res.render('dashboard', { layout: 'adminLay', avg: avgRating, max: maxRating, min: minRating });
  } catch (error) {
    // Se ocorrer um erro durante a consulta ao banco de dados, registra o erro e retorna uma resposta de erro do servidor
    console.error('Erro ao consultar o banco de dados:', error);
    res.status(500).send('Erro no servidor');
  }
});

```


Código Geral

ROTAS

```
// Importa os controladores do dashboard
import { getAll, getCountByCategory, getRatingByCategory, getCountComCategory } from '../controllers/admin/dashboardControllers.js';

// Importa o Router do Express
import { Router } from 'express';

// Cria uma instância do Router
const router = Router();

// Rota para renderizar a página inicial do dashboard
router.get("/", getAll);

// Rota para obter os dados dos spots por categoria
router.get("/data/spots-by-category", getCountByCategory);

// Rota para obter os dados dos spots por rating
router.get("/data/spots-by-rating", getRatingByCategory);

// Rota para obter os dados dos spots comerciais por categoria
router.get("/data/spots-by-comCategory", getCountComCategory);

// Rota para obter os dados da média de avaliação dos spots
router.get("/data/spots-by-avgValue", getAvgResult);

// Exporta o router para uso em outros arquivos
export default router;
```

CONTROLADORES

```
// Importa o modelo Statistics para acessar os métodos de consulta
import Statistics from "../models/statistics.model.js";

// Controlador para renderizar a página inicial do dashboard
const getAll = async function(req, res){
  res.render('dashboard', {layout: 'adminLay', title: 'LisbonSpots', }
});

// Controlador para obter a contagem de spots por categoria
const getCountByCategory = async (req, res) => {
  const result = await Statistics.getCountByCategory(); // Chama o método do modelo para obter os dados
  res.send(result); // Retorna os dados como resposta
}

// Controlador para obter a avaliação de spots por categoria
const getRatingByCategory= async(req, res) => {
  const result = await Statistics.getRatingByCategory(); // Chama o método do modelo para obter os dados
  res.send(result); // Retorna os dados como resposta
}

// Controlador para obter a contagem de spots comerciais por categoria
const getCountComCategory = async (req, res) =>{
  const result = await Statistics.getCountComCategory(); // Chama o método do modelo para obter os dados
  res.send(result); // Retorna os dados como resposta
}

// Exporta os controladores para uso em outros arquivos
export {getAll, getCountByCategory, getRatingByCategory, getCountComCategory}
```

MODELS

```
// Importa o conector do banco de dados
import db from "../config/db_connector.js";

// Define o objeto Statistics com os métodos para consulta de estatísticas
const Statistics = {

  // Método para obter a contagem de spots não comerciais por categoria
  getCountByCategory: async () => {
    const text = `
      SELECT category, count(*)
      FROM tb_noncommercialspot
      GROUP BY category;
    `;
    // Executa a consulta e retorna as linhas resultantes
    return (await db.query(text)).rows;
  },

  // Método para obter a avaliação de spots comerciais por categoria (específico para museus)
  getRatingByCategory: async () => {
    const text = `
      SELECT category, rating, name
      FROM tb_commercialspot
      WHERE category='museum'
      ORDER BY rating;
    `;
    // Executa a consulta e retorna as linhas resultantes
    return (await db.query(text)).rows;
  },

  // Método para obter a contagem de spots comerciais por categoria e calcular a percentagem
  getCountComCategory: async () => {
    const text = `
      SELECT
        category,
        COUNT(*) AS count,
        ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM tb_commercialspot), 2) AS percentage
      FROM
    `;
  },
}
```

```

        tb_commercialspot

    GROUP BY

        category;

    `;

    // Executa a consulta e retorna as linhas resultantes

    return (await db.query(text)).rows;

    },

}

// Exporta o objeto Statistics para uso em outros arquivos

export default Statistics;

```

Visão geral do Dashboard

