

Base de dados Avançadas

Mobilidade Urbana

Ano letivo 2021/2022

Mestrado em Engenharia Informática
Escola Superior de Tecnologia
Instituto Politécnico do Cávado e do Ave

Grupo de trabalho

Alexandre da Cunha Cepa 10207

André Filipe Silva Carvalho 14062

Paulo José Capa Azevedo Meneses 17611

Resumo

O presente documento enquadra-se no âmbito da cadeira de Base de Dados Avançada, com este projeto pretende-se documentar o estudo, desenho e análise de três modelos diferentes de bases de dados relativos aos requisitos para o desenvolvimento da plataforma de mobilidade.

Como resultado, esperamos que este projeto cumpra todos os requisitos propostos pelo docente.

Índice

1.	Introdução	6
1.1.	Objetivos	6
1.2.	Estrutura do documento.....	8
2.	Desenvolvimento.....	9
2.1.	Esquema relacional.....	9
2.2.	Esquema de coleções.....	10
2.3.	Esquema de grafos.....	14
3.	Testes dos modelos de dados.....	16
3.1.	Modelo relacional	16
3.2.	Modelo de coleções.....	17
3.3.	Modelo de grafos.....	20
3.4.	Conclusões dos testes das bases de dados.....	22
4.	Conclusão.....	23
5.	Bibliografia.....	24

Índice de Figuras

Figura 1 - Esquema Relacional Inicial.....	9
Figura 2 - Esquema Relacional Final	10
Figura 3 - Collection-Relationship Diagram	11
Figura 4 - Esquema Grafos Inicial	14
Figura 5 - Esquema Grafos Final	14
Figura 6 – Grafo a representar uma viagem.....	15
Figura 7 – Resultado da querie de custo e duração média por utilizador.....	16
Figura 8 – Resultado da querie de seleccionar uma viagem.....	16
Figura 9 – Resultado da querie de contar o número de viagens por utilizador	17
Figura 10 – Volume de dados existentes no MongoDB.....	17
Figura 11 – Resultado da querie de custo e duração media por utilizador.....	18
Figura 12 – Resultado da querie de selcionar uma viagem.....	19
Figura 13 – Resultado da querie de contar o número de viagens por utilizador	19
Figura 14 – Grafo representando a base de dados existente.....	20
Figura 15 – Tempo de carregamento dos dados no Neo4j	20
Figura 16 – Resultado da querie de media de custo e duração por utilizador.....	21
Figura 17 – Resultado da querie de seleccionar uma viagem.....	21
Figura 18 – Resultado da querie de seleccionar uma viagem.....	22
Figura 19 – Resultado da querie de contar o número viagens por utilizador	22

1. Introdução

Com a crescente urbanização da nossa sociedade, existe a necessidade de tornar os grandes centros urbanos menos poluídos, ambientalmente sustentáveis e mais humanos, para isso estamos a começar a ver muitos sistemas de gestão e auxílio integrados nas infraestruturas das cidades.

Estes sistemas recorrem muito a recolha de informação atrás de sistemas IoT (*Internet of Things*).

“Using Internet of Things (IoT) all relevant data can be collected providing an integrated overview of all city processes. The intensive use of models and data analytics, processed most likely in computing clouds, completes the understanding of the city as a machine and allows for acting in the real word as to adapt it to new circumstances.” [1]

Uma das maneiras que este processo ocorre também é no segmento de mobilidade das pessoas dentro da cidade, tornando a mobilidade mais sustentável. Este projeto pretende o desenvolvimento de uma aplicação de micro mobilidade, proposta no mestrado de engenharia informática do ano letivo 2021/2022, na qual irá disponibilizar serviços de mobilidade, nomeadamente trotinetes elétricas, através de uma aplicação, tendo como principal utilizador o publico geral e a funcionalidade central o aluguer de trotinetes.

1.1. Objetivos

Os objetivos principais deste projeto é desenvolver uma base de dados de apoio a uma solução integrada de aluguer de veículos urbanos para uso pessoal e que permite o registo das seguintes entidades:

- **Utilizadores:** As pessoas que fazem um registo para utilizar os veículos elétricos;
- **Veículos:** Os veículos elétricos para uso pessoal: bicicletas e triciclos;
- **Localização:** Lista das localizações do centro urbano onde o sistema opera;

A base de dados deve suportar um conjunto de operações centrais, como por exemplo:

- **Nova Viagem:** Local e Hora de início e fim, custos;
- **Pagamento:** Recargas do saldo associado ao utilizador;
- **Procurar Veículo:** Mostrar os veículos disponíveis nas proximidades do local do utilizador;
- **Feedback:** os utilizadores podem dar feedback no final de uma viagem;
- **Localização de partilha:** os utilizadores podem partilhar a localização do seu veículo com outros utilizadores;
- **Rotas:** Mostrar a lista de lugares visitados durante uma viagem;
- **Incidentes:** Relato de problemas ocorridos durante uma viagem;
- **Deslocações:** Transporte de veículos pela entidade de um local para o outro;
- **Notificações:** Os utilizadores podem aceitar o pedido de outro utilizador para ser notificado sempre que chegarem a um local específico;

O registo das entidades foi necessário para identificar e mapear as entidades e suas relações.

Depois devemos criar um modelo de base de dados para cada um dos três diferentes modelos, considerando as melhores práticas e características de cada modelo:

- *Entity-Relationship Diagram (ERD)*
- *Collection-Relationship Diagram (CRD)*
- *Graph-based data model*

Implementar cada modelo de dados no seu sistema específico de gestão de base de dados e povoar as bases de dados com dados significativos.

A Base de dados deve ter tamanho suficiente para tirar conclusões sobre o desempenho e a adequação do modelo de base de dados.

1.2. Estrutura do documento

O presente documento obedece a uma estrutura planeada de forma adequada a criar uma solidez entre os conteúdos com o objetivo de documentar todo o trabalho desenvolvido de forma completa, sucinta e estruturada.

O mesmo segue a seguinte estrutura:

1. Introdução
2. Desenvolvimento
3. Testes dos modelos de dados
4. Conclusão

2. Desenvolvimento

2.1. Esquema relacional

No modelo de dados relacional é utilizado um conjunto de tabelas para representar dados e relações entre dados. As entidades são coisas/objetos que são um conjunto de atributos que são distinguíveis de outros objetos, e as relações são as ligações entre esses objetos, isto pode ser representado visualmente através de um diagrama, ao qual se chama digrama Entidade-Relação (ER).

Neste as entidades representam tabelas, e as relações são criadas através de chaves primárias e estrangeiras. Este modelo permite um nível de abstracção o a entidades e relações que facilita o desenvolvimento destes modelos.

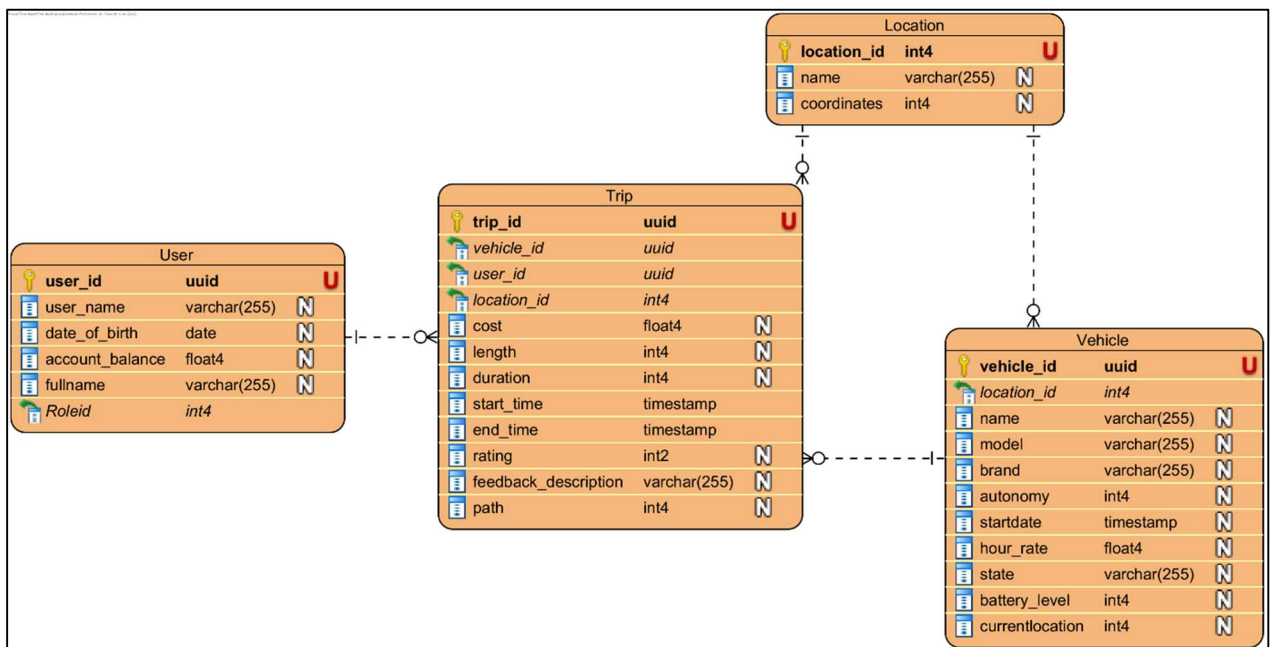


Figura 1 - Esquema Relacional Inicial

O desenvolvimento do esquema ER para a base de dados iniciou-se pelo um modelo mais simples como mostra a figura abaixo, com 4 tabelas.

A tabela “**User**” para os dados dos utilizadores, a tabela “**Location**” para armazenar as localizações e as suas áreas no mapa nas quais as trotinetes podem andar, a “**Vehicle**” para armazenar os veículos do sistema e as suas propriedades e a tabela “**Trip**” que é uma tabela de factos que vais agregar dados sobre a viagem e vai ter chaves das outras 3 tabelas.

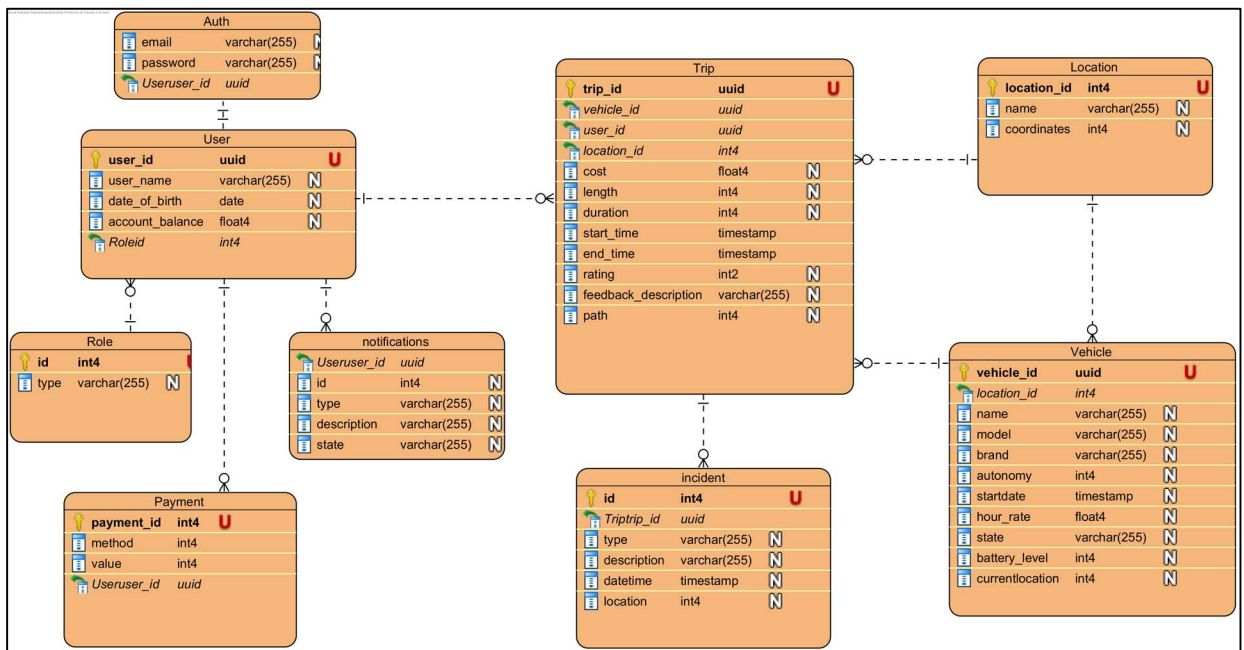


Figura 2 - Esquema Relacional Final

Para suportar mais funcionalidades é preciso mais tabelas, como a baixo está representado.

As tabelas adicionadas são, a tabela **“Auth”** gere o armazenamento das credenciais de acesso como o email e a palavra-passe, a tabela **“Notifications”** para as notificações de cada utilizador, guardando o seu tipo, descrição e estado, a tabela **“Role”** servirá para distinguir os utilizadores por tipo, por exemplo administradores, gestores e clientes, a tabela **“Payment”** guarda todos os pagamentos feitos pelo utilizador e a tabela **“Incident”** irá conter incidentes que ocorram nas viagens.

2.2. Esquema de coleções

O modelo de dados que o MongoDB utiliza:

- Esquema para Users no mongodb

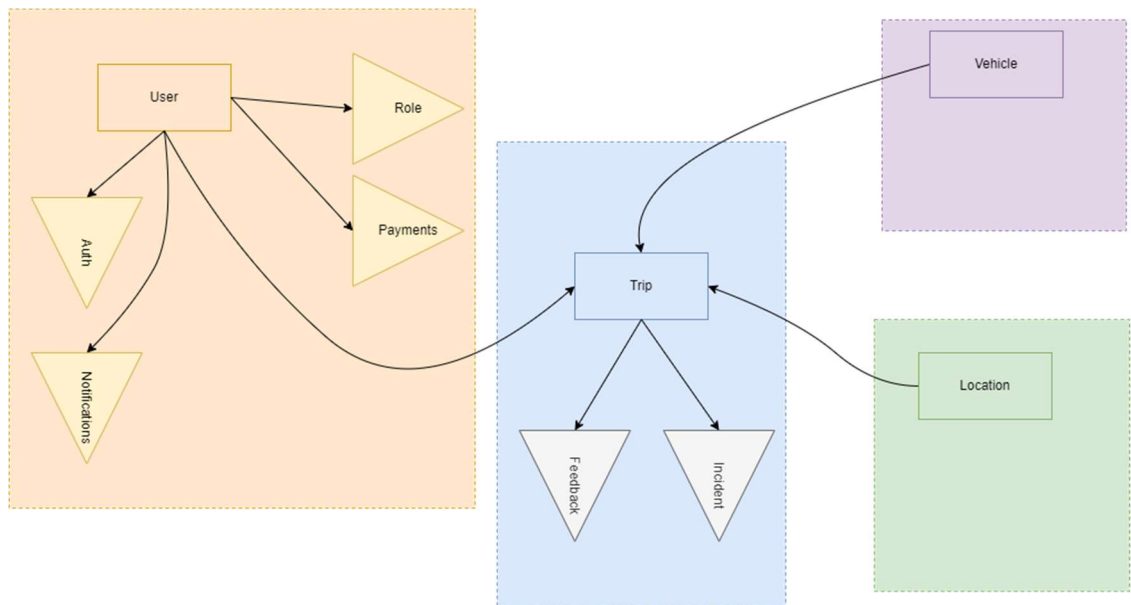


Figura 3 - Collection-Relationship Diagram

Os dados sobre as coordenadas para a localização das viagens, localizações e trotinetas estão no formato “**GeoJSON**”, que é uma norma definida

<https://datatracker.ietf.org/doc/html/rfc7946>

Esquema User

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'user_name'
    ],
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      user_name: {
        bsonType: 'string'
      },
      date_of_birth: {
        bsonType: 'string'
      },
      fullname: {
        bsonType: 'string'
      },
      account_balance: {
        bsonType: 'double'
      },
      role: {
        bsonType: 'object',
        properties: {
```

Esquema Locations

```
{
  $jsonSchema: {
    bsonType: 'object',
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      name: {
        bsonType: 'string'
      },
      features: {
        type: 'array',
        items: {
          type: 'object',
          properties: {
            type: {
              type: 'string'
```

```
id: {
    bsonType: 'int'
},
type: {
    bsonType: 'string'
}
}
}
}
```

```
},  
coordinates: {  
  type: 'array',  
  items: {  
    type: 'array',  
    items: {  
      type: 'array',  
      items: {  
        type: 'number'  
      }  
    }  
  }  
}  
  
}  
  
}  
  
}  
  
}  
  
}
```

Esquema Vehicles

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      '_id',
      'hour_rate',
      'battery_level',
      'currentLocation',
      'brand',
      'model'
    ],
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      name: {
        bsonType: 'string'
      },
      model: {
        bsonType: 'string'
      },
      brand: {
        bsonType: 'string'
      },
      autonomy: {
        bsonType: 'int'
      },
      startdate: {
```

Esquema Trips

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'vehicle_id',
      'location_id',
      'user_id',
      'cost',
      'length',
      'duration'
    ],
    properties: {
      _id: {
        bsonType: 'objectId'
      },
      vehicle_id: {
        bsonType: 'string'
      },
      location_id: {
        bsonType: 'string'
      },
      user_id: {
        bsonType: 'string'
      },
      cost: {
        bsonType: 'double'
      },
      length: {
```

```

      bsonType: 'string'
    },
    hour_rate: {
      bsonType: 'double'
    },
    battery_level: {
      bsonType: 'int'
    },
    features: {
      type: 'array',
      items: {
        type: 'object',
        properties: {
          type: {
            type: 'string'
          },
          coordinates: {
            type: 'array',
            items: {
              type: 'number'
            }
          }
        }
      }
    }
  }
}

```

```

      bsonType: 'int'
    },
    duration: {
      bsonType: 'int'
    },
    start_time: {
      bsonType: 'string'
    },
    end_time: {
      bsonType: 'string'
    },
    rating: {
      bsonType: [
        'int',
        'null'
      ]
    },
    feedback_description: {
      bsonType: [
        'string',
        'null'
      ]
    },
    features: {
      type: 'array',
      items: {
        type: 'object',
        properties: {
          type: {
            type: 'string'
          },
          coordinates: {
            type: 'array',
            items: {
              type: 'array',
              items: {
                type: 'number'
              }
            }
          }
        }
      }
    }
  }
}

```

2.3. Esquema de grafos

O esquema de grafos é representado por nodos e ligações entre eles formando uma relação entre os dados

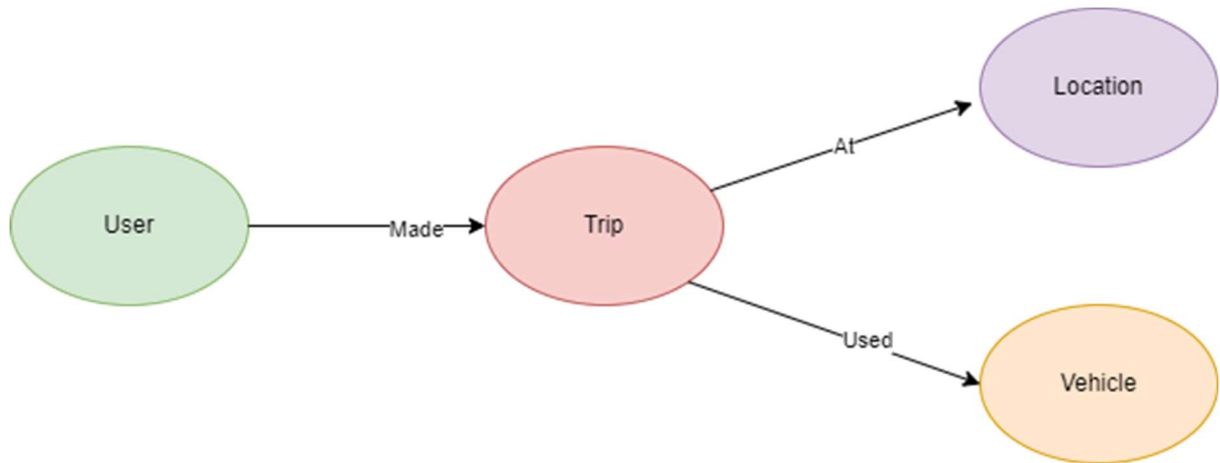


Figura 4 - Esquema Grafos Inicial

O esquema final contempla mais nodos e mais ligações para representar os dados do sistema.

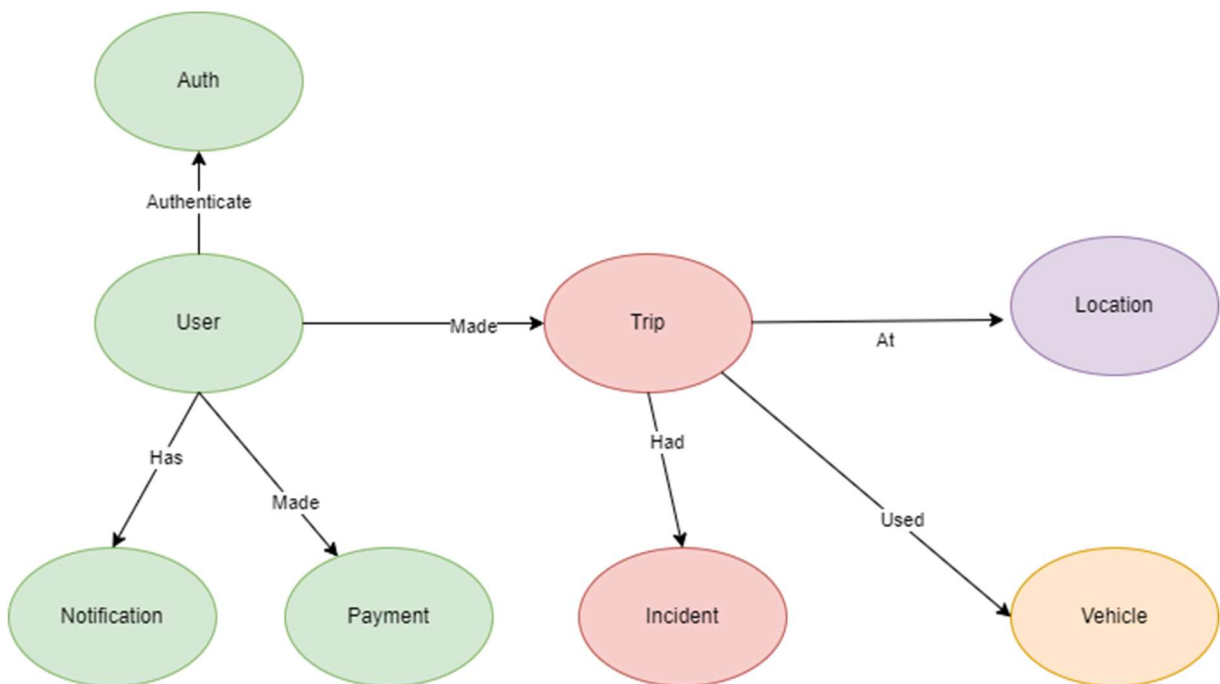


Figura 5 - Esquema Grafos Final

Utilizando a linguagem **Cypher**, cria-se utilizadores, veículos, localizações e viagens com as seguintes expressões.

```
CREATE (u:User{username: "admin", fullname: "Paulo Meneses", date_of_birth: "2020-09-09"}) return u
CREATE (v:Vehicle{name: "abc", model: "top", brand: "xpto", startdate: datetime("2020-05-15T14:30:00"), location: [42.12312,-8.324]}) return v
CREATE (l:Location {name:"Braga", coordinates:[-45.2131,-8.42341,-45.321,-8.1231,54.6324,-6.42141] }) return l
Create (t:Trip {cost: 4.05, duration: 23, coordinates: [45.1331,-4.2131,45.4123,-4.54314,45.313,-8.635],end_time: datetime("2020-05-09T15:45:38.132"), start_time: datetime("2020-05-09T15:38:38.132")})
```

Para fazer as ligações entre os nodos, primeiro os nodos têm de ser localizados e depois criada as ligações entre eles.

O comando abaixo demonstra a criação de 3 ligações.

```
Match (u),(l),(v),(t)
Where ID(u)=0 AND ID(l)=3 AND ID(v)=1 AND ID(t)=6
Create (u)-[m:Made]->(t),(t)-[a:At]->(l),(t)-[w:With]->(v)
```

No fim resulta o seguinte grafo.

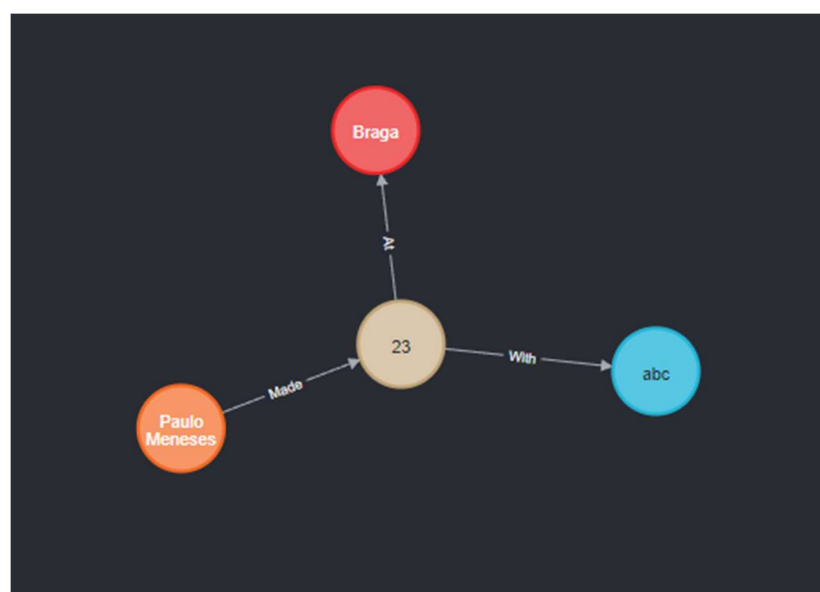


Figura 6 – Grafo a representar uma viagem

3. Testes dos modelos de dados

3.1. Modelo relacional

Custo e duração média por utilizador

A query para fazer esta consulta demora 0.019 segundos a ser executada numa base de dados com 30 mil viagens, 10 mil utilizadores e 325 veículos. O carregamento destes dados na base de dados demorou cerca de 10 minutos.

```
SELECT user_acc.user_id, user_acc.fullname AS Nome, AVG(trip.cost) AS
CustoMedio, AVG(trip.duration) AS DuracaoMedia
FROM user_acc INNER JOIN trip ON user_acc.user_id = trip.user_id
GROUP BY user_acc.user_id, trip.cost
> OK
> Time: 0,019s
```

user_id	nome	customedio	duracaomedia
86870251-1046-45ce-9686-3d81d59c7498	yealsdnoigkormkj	1,5499999523162842	13,0000000000000000
9aea7e7b-fc0a-4174-9cf4-cff6fc920b77	fpkpcqdatbotlchw	1,5499999523162842	13,0000000000000000
9e377a20-3ddd-41ea-9af7-1d8d4830006a	ovyvbtcdwsdwnsvx	1,5499999523162842	13,0000000000000000
a633f267-acb3-40cf-91c8-4e4e59386a08	kvgfveknlsntetog	1,5499999523162842	13,0000000000000000
a8d2d460-b173-480c-b405-53c953b5b471	orcoxvfqagtswrgp	1,5499999523162842	13,0000000000000000
b16663d3-e022-4732-8900-582049836d87	xkgxkjjmqjndgatd	1,5499999523162842	13,0000000000000000
b6b6da9d-0760-4921-aafe-3856e33e06aa	lqlmklhpiujurwni	1,5499999523162842	13,0000000000000000
b927eaf0-9c9d-4441-b914-7a0e0a1e3d32	lkhdetkxjmccrsvg	1,5499999523162842	13,0000000000000000
bb03080f-2583-41fb-99b4-745fe0cb7526	qgcdjmsljkwhwfjm	1,5499999523162842	13,0000000000000000
bcff8cba-b057-44c6-a694-8d4b5bf695b6	hvdluclknlrceal	1,5499999523162842	13,0000000000000000

Figura 7 – Resultado da querie de custo e duração média por utilizador

Selecionar uma viagem

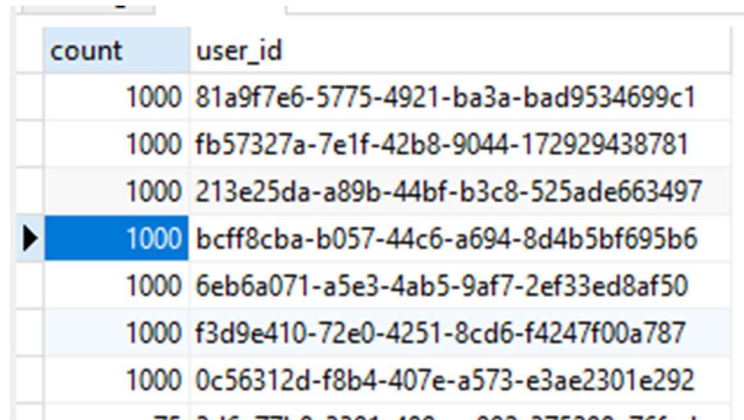
```
SELECT * FROM trip
WHERE trip_id = 'cd3cd86c-0872-42a9-9d59-c1d7b2a3010a'
> OK
> Time: 0,002s
```

trip_id	vehicle_id	user_id	location_id	cost	length	dura
cd3cd86c-0872-42a9-9d59-c1d7b2a3010a	e68e1dcd-af8b-0c56312d-f8b4-407e-a573-e3ae2301e292		5	1,55	650	

Figura 8 – Resultado da querie de selecionar uma viagem

Contar o número de viagens por utilizador

```
SELECT count(trip_id), user_id
FROM trip
GROUP BY user_id
> OK
> Time: 0,007s
```



count	user_id
1000	81a9f7e6-5775-4921-ba3a-bad9534699c1
1000	fb57327a-7e1f-42b8-9044-172929438781
1000	213e25da-a89b-44bf-b3c8-525ade663497
1000	bcff8cba-b057-44c6-a694-8d4b5bf695b6
1000	6eb6a071-a5e3-4ab5-9af7-2ef33ed8af50
1000	f3d9e410-72e0-4251-8cd6-f4247f00a787
1000	0c56312d-f8b4-407e-a573-e3ae2301e292

Figura 9 – Resultado da querie de contar o número de viagens por utilizador

3.2. Modelo de coleções

A base de dados **NoSQL**, **MongoDB**, foi inserido 31 mil viagens, 10 mil utilizadores e 328 veículos. O carregamento destes dados demorou alguns segundos.






Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Incidents	2	42.5 B	85.0 B	1	36.9 KB	
Locations	1	158.0 B	158.0 B	1	20.5 KB	
Trips	31,167	525.0 B	16.4 MB	1	36.9 KB	
Users	10,006	176.0 B	1.8 MB	1	151.6 KB	
Vehicles	328	267.0 B	87.6 KB	1	36.9 KB	

Figura 10 – Volume de dados existentes no MongoDB

A *querie* de NoSQL para calcular o custo e duração média por utilizador é complexa pois a base de dados não é propositada para fazer relações.

```
db.Users.aggregate(  
  [{  
    $project: {  
      _id: {  
        $toString: "$_id"  
      }  
    }  
  }, {  
    $lookup:  
    {  
      from: "Trips",  
      localField: "_id",  
      foreignField: "user_id",  
      as: "Viagens"  
    }  
  }, {  
    $unwind: '$Viagens'  
  }, {  
    $group:  
    {  
      _id: "$_id",  
      avgAmount: {  
        $avg: "$Viagens.cost"  
      },  
      avgQuantity: {  
        $avg: "$Viagens.duration"  
      }  
    }  
  }]  
)  
> OK  
> Time: 163,557s
```

Message	Result																																										
	<table><tr><th>_id</th><th>avgAmount</th><th>avgQuantity</th></tr><tr><td>61dee8ef523a149bb71c5a85</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5ae5</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5aea</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a19</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5ad2</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5aac</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a6a</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5b30</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a86</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a8e</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5acb</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a09</td><td>2,34</td><td>12</td></tr><tr><td>61dee8ef523a149bb71c5a38</td><td>2,34</td><td>12</td></tr></table>	_id	avgAmount	avgQuantity	61dee8ef523a149bb71c5a85	2,34	12	61dee8ef523a149bb71c5ae5	2,34	12	61dee8ef523a149bb71c5aea	2,34	12	61dee8ef523a149bb71c5a19	2,34	12	61dee8ef523a149bb71c5ad2	2,34	12	61dee8ef523a149bb71c5aac	2,34	12	61dee8ef523a149bb71c5a6a	2,34	12	61dee8ef523a149bb71c5b30	2,34	12	61dee8ef523a149bb71c5a86	2,34	12	61dee8ef523a149bb71c5a8e	2,34	12	61dee8ef523a149bb71c5acb	2,34	12	61dee8ef523a149bb71c5a09	2,34	12	61dee8ef523a149bb71c5a38	2,34	12
_id	avgAmount	avgQuantity																																									
61dee8ef523a149bb71c5a85	2,34	12																																									
61dee8ef523a149bb71c5ae5	2,34	12																																									
61dee8ef523a149bb71c5aea	2,34	12																																									
61dee8ef523a149bb71c5a19	2,34	12																																									
61dee8ef523a149bb71c5ad2	2,34	12																																									
61dee8ef523a149bb71c5aac	2,34	12																																									
61dee8ef523a149bb71c5a6a	2,34	12																																									
61dee8ef523a149bb71c5b30	2,34	12																																									
61dee8ef523a149bb71c5a86	2,34	12																																									
61dee8ef523a149bb71c5a8e	2,34	12																																									
61dee8ef523a149bb71c5acb	2,34	12																																									
61dee8ef523a149bb71c5a09	2,34	12																																									
61dee8ef523a149bb71c5a38	2,34	12																																									

db.Users.aggregate([{ \$project: { _id: { \$toString: "\$_id" } } }, { \$lookup: { from: "Trips", Read Only String Query time: 163.560s

Figura 11 – Resultado da querie de custo e duração media por utilizador

Selecionar uma viagem

```
db.Trips.findOne({_id: ObjectId("61def967901e59744de08700")})  
> OK  
> Time: 0,004s
```

Result							
vehicle_id	location_id	user_id	cost	length	duration	start_time	ent_time
61dee8ff523a149bb71c817a	61bc9ef5538163155768d8b0	61dee8ef523a149bb71c5a04	2,34	344	12	12/01/2022 15:53:11	12/01/2022 15

Figura 12 – Resultado da querie de selcionar uma viagem

Contar o número de viagens por utilizador

```
db.Trips.aggregate([  
  $group:  
  {  
    _id: "$user_id",  
    total: {$count: {}}  
  }  
)  
> OK  
> Time: 0,031s
```

_id	total
61dee8ef523a149bb71c5b2b	100
61dee8ef523a149bb71c5ae7	100
61dee8ef523a149bb71c5a14	100
61dee8ef523a149bb71c5a92	100
61dee8ef523a149bb71c5a5d	100
61dee8ef523a149bb71c5a8b	100
61dee8ef523a149bb71c5a21	100
61dee8ef523a149bb71c5aa5	100
61dee8ef523a149bb71c5ade	100

Figura 13 – Resultado da querie de contar o número de viagens por utilizador

3.3. Modelo de grafos

A base de dados de grafos tem um grafo representando visualmente os dados existentes. O carregamento de dados demorou alguns minutos.

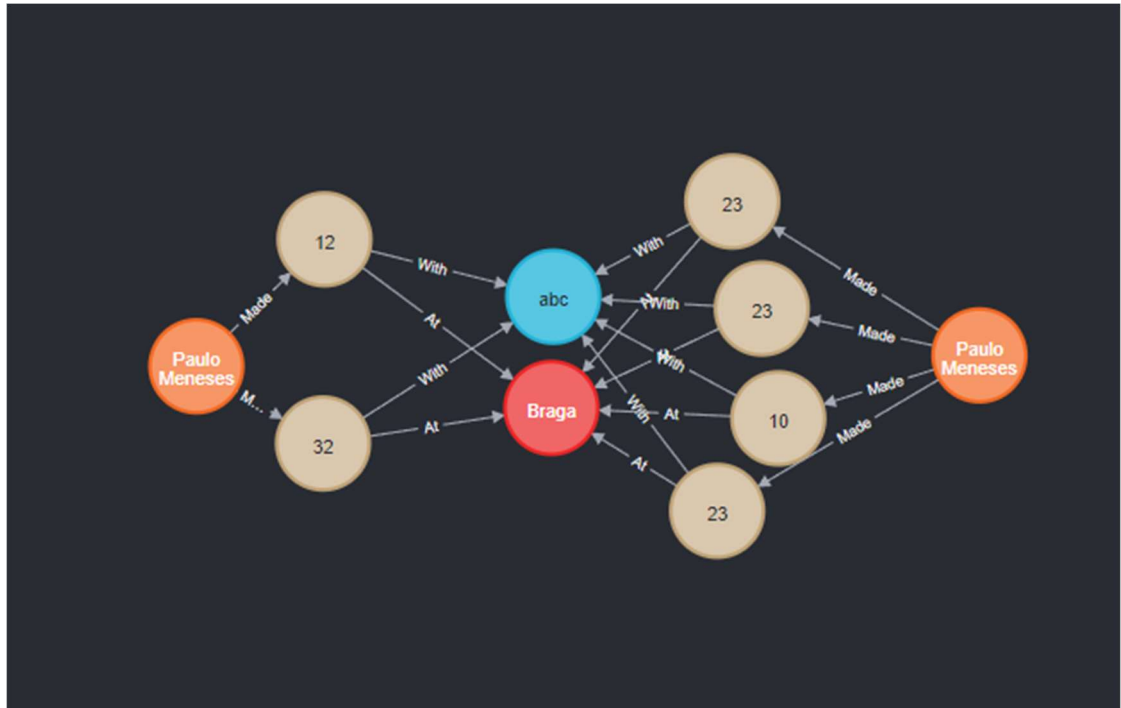


Figura 14 – Grafo representando a base de dados existente

O tempo de carregamento das viagens, cerca de 30 mil.

```
neo4j$ LOAD CSV WITH HEADERS FROM 'file:///Trip.csv' AS row MERGE (c:
Added 33985 labels, created 33985 nodes, set 237895 properties, completed after 928638 ms.
```

The image shows a terminal window with a Neo4j command and its output. The command is `neo4j$ LOAD CSV WITH HEADERS FROM 'file:///Trip.csv' AS row MERGE (c:`. The output is `Added 33985 labels, created 33985 nodes, set 237895 properties, completed after 928638 ms.`

Figura 15 – Tempo de carregamento dos dados no Neo4j

Media de custo e duração por utilizador

```
Match (u:User)-[:Made]-(t:Trip)
Return u, avg(t.cost), avg(t.duration)
```

```
neo4j$ Match (u:User)-[:Made]-(t:Trip) Return u, avg(toFloat(t.cost)), avg(toFloat(t.duration))
```

"u"	"avg(toFloat(t.cost))"	"avg(toFloat(t.duration))"
{"User_id": "00f55329-f696-474c-910d-e8f1442cff60", "username": "null"}	1.55	13.0
{"User_id": "01f7e6b3-016b-471d-822c-724d0145df56", "username": "null"}	1.55	13.0
{"User_id": "2a16e507-0c03-4cc5-8e6f-5ae10353e156", "username": "null"}	1.55	13.0
{"User_id": "344b3af4-8fd5-47ef-8af7-9ce8266a5990", "username": "null"}	1.55	13.0
{"User_id": "3801e4f1-e518-44e7-b572-d507840e747a", "username": "null"}	1.55	13.0
{"User_id": "4116e273-fa19-47c8-b112-a12b375a808e", "username": "null"}	1.55	13.0

Figura 16 – Resultado da querie de media de custo e duração por utilizador

Selecionar uma viagem

```
Match (t:Trip {trip_id: "18006b5f-5c5c-4f7c-a8d9-2da0471c73f6"})
return t
```

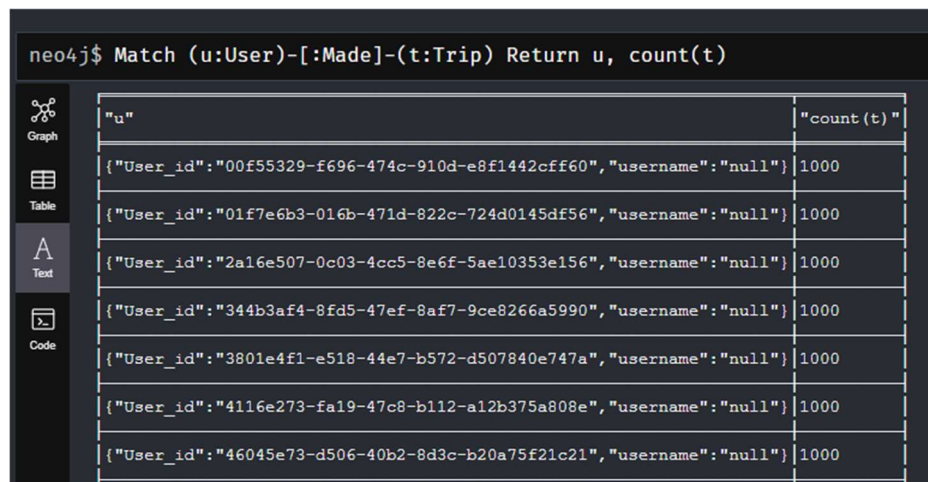
```
neo4j$ Match (t:Trip {trip_id: "18006b5f-5c5c-4f7c-a8d9-2da0471c73f6"}) return
```

"t"
{"duration": "13", "trip_id": "18006b5f-5c5c-4f7c-a8d9-2da0471c73f6", "cost": "1.55", "user_id": "c14f88b2-669d-4ed6-9103-c8a1bf1571f0", "length": "650", "vehicle_id": "768bc3bc-a185-415c-9163-a2d8df05d404", "location_id": "1"}

Figura 17 – Resultado da querie de selecionar uma viagem

Contar o número de viagens por utilizador

```
Match (u:User)-[:Made]-(t:Trip)
Return u, count(t)
```



The screenshot shows the Neo4j interface with a query executed: `neo4j$ Match (u:User)-[:Made]-(t:Trip) Return u, count(t)`. The results are displayed in a table view. The table has two columns: `"u"` and `"count(t)"`. There are eight rows of data, each showing a user ID and a count of 1000.

"u"	"count(t)"
<code>{"User_id": "00f55329-f696-474c-910d-e8f1442cff60", "username": "null"}</code>	1000
<code>{"User_id": "01f7e6b3-016b-471d-822c-724d0145df56", "username": "null"}</code>	1000
<code>{"User_id": "2a16e507-0c03-4cc5-8e6f-5ae10353e156", "username": "null"}</code>	1000
<code>{"User_id": "344b3af4-8fd5-47ef-8af7-9ce8266a5990", "username": "null"}</code>	1000
<code>{"User_id": "3801e4f1-e518-44e7-b572-d507840e747a", "username": "null"}</code>	1000
<code>{"User_id": "4116e273-fa19-47c8-b112-a12b375a808e", "username": "null"}</code>	1000
<code>{"User_id": "46045e73-d506-40b2-8d3c-b20a75f21c21", "username": "null"}</code>	1000

Figura 18 – Resultado da query de selecionar uma viagem

3.4. Conclusões dos testes das bases de dados

Para fazer *queries* mais complexas, com relações, o SQL no modelo relacional é mais simples de as executar que o NoSQL, mas o NoSQL é mais flexível podendo-se alterar o modelo de dados facilmente.

No modelo de grafos é perdida alguma flexibilidade no tipo de dados para armazenar, mas permite uma visualização dos dados diferente. A performance das bases de dados varia dependendo do tipo de *query* a ser executada. Mas são bastante rápidas apesar de existir relações. Todas a *queries* demoraram cerca de 150 milissegundos a executar.

O tempo de execução da *query* de calculo do custo e duração média por utilizador é muito diferente entre o modelo relacional e de coleções. Na base de dados relacional a *query* é executada muito mais rapidamente que no modelo coleções, devido ao facto de que a *query* necessita de cruzar dados com tabelas/coleções diferentes, que é desvantajoso para o NoSQL, mas o tempo de inserção de dados é muito inferior, demorando alguns segundos para a inserção de 30 mil viagens na base de dados comparado a alguns minutos no modelo relacional. As *queries* de seleção de uma viagem são igualmente rápidas nos dois tipos e a contagem de viagens também.

A base de dados de grafos permite visualizar os dados de uma forma diferente permitindo outras abordagens ao armazenamento de dados e as *queries* são bastante eficientes. A inserção de dados também é a mais lenta das três. Existiu dificuldades a inserir grandes quantidades de dados no Neo4j.

4. Conclusão

Uma vez concluída a realização deste trabalho prático o balanço que se pode fazer deste é positivo. Apesar das dificuldades ao decorrer do desenvolvimento, as mesmas foram superadas de várias formas.

As dificuldades neste trabalho foram algumas, uma delas foi o tempo da entrega, que não foi possível entregar o trabalho no dia proposto pelo docente. Como tal tivemos mais tempo para corrigir alguns problemas no projeto e o melhorar.

Esperamos naturalmente ter superado todas as expectativas propostas pelo docente e ter um bom uso em projetos futuros.

5. Bibliografia

- [1] Á. a. C. M. Oliveira, "From Smart Cities to Human Smart Cities," em *2015 48th Hawaii International Conference on System Sciences*, 2015.

[OBJ]

