
**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores**

**LEIC & LMATE**

**Introdução aos Sistemas de Informação**

**Semestre de Inverno 2023-2024**

# Reserva de bicicletas — Bicycle reservation

Trabalho prático
( Fase 2 )

Ana Beire, Matilde Pato e Nuno Leite

**Novembro, 2023**

# Planeamento — Planning

As datas importantes a recordar são:

- Lançamento do enunciado: **6 de Outubro 2023**
- Entrega intermédia (Fase 1): **24 de Outubro de 2023**
- Entrega intermédia (Fase 2): **28 de Novembro de 2023**
- Entrega intermédia (Fase 3): **19 de Dezembro de 2023**

[PT]

Cada entrega intermédia deve apresentar o relatório e código (se houver) referentes **exclusivamente** a essa fase. O relatório deve seguir um dos *templates* fornecidos, obrigatoriamente, sob pena de penalização. **Este deve ser conciso e apresentar a justificação de todas as decisões tomadas** (ver Critérios de Avaliação). A capa do relatório deve indicar a composição do grupo, a unidade curricular e a fase do trabalho que relata. Caso tenha adendas e/ou correcções a fazer a modelos já entregues, deve indicá-las de forma explícita no relatório seguinte.

O pdf (e, o zip) gerado deve seguir o nome da seguinte forma: 'TP**N**ISI-2324-**MM**.ext' (N representa um dígito correspondente ao número da fase do trabalho, MM representa os dígitos do número do grupo, e 'ext' a extensão do ficheiro), e.g.: TP1ISI-2324-01.pdf.

[EN]

*Each intermediate delivery must present the report and code (if any) referring* **exclusively** *to that phase. The report must obligatorily follow one of the templates provided, or penalties may ensue.* **This must be concise and present the justification for all decisions made** *(see Assessment Criteria). The report's cover must indicate the group composition, the curricular unit and the phase of the work being reported. If you have additions and corrections to deliver to models previously submitted, tell them explicitly in the following report.*

*The generated pdf (and zip) must follow the name as follows: 'TP***N***ISI-2324-***MM***.ext' (N represents a digit corresponding to the work phase number, MM represents the digits of your group number, and 'ext' the file extension), e.g.: TP1ISI-2324-01.pdf.*

1 de Novembro de 2023, Ana Beire, Matilde Pato e Nuno Leite

## Objectivos de aprendizagem

No final da **segunda fase do trabalho**, os alunos devem ser capazes de:

- Utilizar correctamente a álgebra relacional, com os seus vários operadores, para expressar interrogações sobre um modelo relacional:
  - operadores relacionais unários: `select`, `project` e `rename`;
  - operadores relacionais binários: `division`;
  - operadores sobre conjuntos: `difference`, `intersect` e `union`;
  - operadores de junção: interna e externa;
  - operador de agregação;
- Utilizar correctamente SQL/DDL para criar as tabelas num sistema de gestão de bases de dados (SGDB);
- Garantir as restrições de integridade identificadas enumerando aquelas que têm de ser garantidas pelas aplicações;
- Inserir dados em lote através da cláusula SQL `INSERT`, garantindo que as restrições de integridade são cumpridas;
- Garantir a atomicidade de instruções, utilizando processamento transaccional;
- Utilizar correctamente os operadores da teoria dos conjuntos em PostgreSQL;
- Utilizar correctamente as cláusulas `INNER JOIN` e `OUTER JOIN`;
- Utilizar correctamente sub-interrogações correlacionadas;
- Utilizar correctamente funções de agregação;
- Utilizar correctamente a cláusula `HAVING`;
- Utilizar correctamente a cláusula `ORDER BY`;
- Utilizar correctamente o termo `DISTINCT`;
- Utilizar correctamente os predicados `IN` e `EXISTS`.

Após a realização da 1ª fase do trabalho, segue-se a implementação do modelo físico do sistema, i.e. deverá ser construído em PostgreSQL contemplando todas as restrições que consigam garantir na forma declarativa.

**Nota**: Deverão preencher a base de dados com informação necessária que permita, em seguida, realizar interrogações que apresentem resultados pertinentes. Na etapa de preenchimento da base de dados, os alunos deverão ter particular atenção ao cumprimento das **restrições de integridade, utilizando de forma adequada o controlo transaccional (a atomicidade)**.

## Resultados pretendidos

Tendo em conta os objectivos de aprendizagem, deverão ser produzidos os seguintes resultados:

1. Construção do modelo físico do sistema, contemplando todas as restrições de integridade passíveis de ser garantidas declarativamente, assim como a **atomicidade** nas operações. O código PostgreSQL que permite:

   (a) Criar o modelo físico (1 *script* autónomo): "`createTable.sql`";

   (b) Remover o modelo físico (1 *script* autónomo): "`removeTable.sql`";

   (c) Preenchimento inicial da base de dados (1 *script* autónomo): "`insertTable.sql`". **Os dados introduzidos devem permitir validar todas as interrogações pedidas nesta fase do trabalho. Todos os gestores que são, também, clientes deverão constar na BD com o valor "C" no `atrdisc` na tabela `PESSOA`**;

   (d) Apagar todos os dados existentes e remover as tabelas (1 *script* autónomo): "`deleteTable.sql`".

2. Considerando o esquema relacional obtido anteriormente e fornecido no final deste documento ("Adenda"), apresente as expressões em álgebra relacional (AR) que respondam às seguintes alíneas. Por forma a recorrer a todos os operadores, apresente **2 soluções alternativas** para a mesma questão.

   (a) Pretende-se obter informação (`nome`, `morada` e `telefone`) dos clientes e dos gestores (**Nota**: lembre-se que na BD os clientes poderão ser clientes, mas têm um valor único no `atrdisc`).

   (b) Liste, agora, informação (`nome`, `morada` e `telefone`) sobre os clientes que também são gestores.

   (c) Pretende-se saber que pessoas (`nome`, `morada` e `telefone`) não estão associadas a nenhuma reserva.

   (d) Apresente a lista de bicicletas (`marca`, `modelo` e `estado`) que não estão associadas a nenhuma reserva e não são eléctricas.

(e) O conjunto de dispositivos (`noserie`, `latitude` e `longitude`) de bicicletas cujo estado encontra-se em "em manutenção".

(f) O nome dos clientes que realizaram reservas com bicicletas eléctricas. Apresente informação sobre os clientes e o número de reservas.

(g) Pretende-se obter a lista de clientes que efectuaram reservas com um valor total superior a €100 (e.g.).

(h) Liste informações (`email`, `endereço` e `localidade`) sobre lojas e respectivos números de telefone associados, incluindo lojas que podem não ter um número de telefone associado.

(i) Para o cliente de nome "José Manuel", pretende-se a lista de reservas (`noreserva` e `loja`) que efectuou, nomeadamente a sua data e as horas de início e de fim, e o preço final desta.

(j) Apresente a lista do(s) cliente(s) (`nome`, `morada`, `telefone` e `nacionalidade`), com mais reservas no ano de 2023[1].

(k) Apresente o número de clientes de nacionalidade portuguesa e outros. O resultado deve mostrar os atributos `nacionalidade` e o número de clientes.

3. Conceba, na linguagem PostgreSQL, as interrogações que produzam os resultados a seguir indicados, utilizando apenas uma instrução PostgreSQL. Guarde todas num *script* autónomo de nome "`queries.sql`". **Para cada instrução deve ser também apresentada, em comentário, a descrição do raciocínio seguido**.

(a) Implemente em PostgreSQL as interrogações pedidas na alínea 2 (Pode escolher uma das soluções que apresentou acima, desde que nesta fase consiga fazer uso de todos os operadores).

(b) Obtenha os nomes de todos os clientes que fizeram pelo menos uma reserva numa loja localizada em Lisboa.

(c) Encontre os números de série dos dispositivos com uma percentagem de bateria superior a 50%, e liste-os por ordem crescente da sua percentagem de bateria.

(d) Apresente a `marca` e o `modelo` da bicicleta com maior autonomia dentro das bicicletas disponíveis.

(e) Obter o número total de reservas para cada loja, bem como o seu código e o número total de reservas.

(f) Liste todas as lojas (`codigo` e `email`) que tenham efectuado mais de 5 reservas, até à presente data, e enumere-as por ordem decrescente do número de reservas (**Nota**: Faça uso dos operadores/funções de data e tempo).

[1]Nota: Poderão fazer uso da função `YEAR()`, por questões de simplificação

(g) Apresente os nomes dos clientes que efectuaram reservas de bicicletas cujo estado tem como valor "em manutencao" no ano passado (**Nota**: Faça uso dos operadores/funções de data e tempo).

(h) Listar as informações (`nome`, `morada`, `telefone`) das pessoas que geriram uma loja e efectuaram reservas.

(i) Obter o(s) nome(s) do(s) cliente(s) que realizaram reservas numa loja gerida por uma pessoa chamada "João".

(j) Crie uma vista `LISTAJOAOFILIPE` que inclui informação sobre os clientes que efectuaram reservas num loja gerida pelo "João Filipe". (**Nota**: O João Filipe é, ele também, cliente).

(k) Crie uma vista `BICICLETASEMNUMEROS` que inclui informação sobre as bicicletas e o seu número. A informação disponível deverá ser, o tipo de bicicleta (eléctrica e clássica), o seu estado ("em manutenção" e "outras"[2]) e o número total.

4. Apresente o(s) comando(s) que permite(m): a) alterar o atributo `mudanca` em `BICICLETA` adicionando um novo valor, neste caso 40; b) afectar este novo valor à(s) bicicleta(s) já registada(s) na BD com o `modelo` "Modelo-B" da `marca` "Marca-A". **Devem garantir atomicidade nas alterações**.

Todas as simplificações e optimizações realizadas ao modelo devem ser indicadas e justificadas.

**PS.1** Sugere-se que consulte o manual do SGDB para obter informação sobre as funções de manipulação de datas.

**PS.2** Na resolução não poderá recorrer aos operadores `with`, `limit` e `cross` **`join`**

**Data limite para entrega: 28 de Novembro de 2023 até às 23:59**.

A entrega deve incluir um documento com as respostas de AR (em formato `PDF`, devidamente identificado, com o número do grupo, os números de aluno e respectivos nomes) e o código PostgreSQL (com extensão .sql). Os ficheiros produzidos **deverão ser** comprimidos e enviados de forma electrónica através do Moodle.

**Nota**: Deve ser possível aferir cada um dos objectivos de aprendizagem no material que entregar.

[2]Estas incluem todas as bicicletas que não estão em manutenção e poderão ser reservadas

## Learning Objectives

At the end of **the second step** of the work, students should be able to:

- Correctly use relational algebra, with its various operators, to express questions about a relational model:
  - unary relational operators: SELECT, PROJECT and RENAME;
  - binary relational operators: DIVISION;
  - binary set theoretic operations: UNION, INTERSECTION, and SET DIFFERENCE;
  - different `JOIN` operations called: THETA JOIN, EQUIJOIN, and NATURAL JOIN, INNER and OUTER;
  - aggregate function operation;
- Use SQL / DDL correctly to create the tables in a database management system of data (DBMS);
- Guarantee the identified integrity constraints, not forgetting those that result from the transition from ER to relational, enumerating those that have to be guaranteed by the applications;
- Insert batch data through the SQL INSERT clause, ensuring that integrity constraints are met;
- Ensure the atomicity of instructions, using transactional processing;
- Use set theory operators correctly in PostgreSQL;
- Use correctly the INNER JOIN and OUTER JOIN clauses;
- Correctly use correlated subqueries;
- Use aggregate functions correctly;
- Use the `HAVING` clause correctly;
- Use the `ORDER BY` clause correctly;

- Use the term `DISTINCT` correctly;
- Use `IN` and `EXISTS` predicates correctly.

After the first phase of the work has been completed, and based on the obtained logical model, the students should built in SQL, the physical model of the system, covering all the restrictions they can guarantee in the declarative form.

**Note**: The students should fill in the database with information that will allow queries to be executed that present relevant results. At the stage of completing the database, students should keep particular attention to performance with integrity restrictions, using transactional control appropriately.

## Expected results

Taking into account the learning objectives, the following results should be produced:

1. Development of the system's physical model, taking into account all integrity restrictions that can be guaranteed in a declarative way, as well as **atomicity** in operations. The PostgreSQL code that allows:
   (a) Create the physical model (1 autonomous *script*): "`createTable.sql`";
   (b) Remove all physical model (1 autonomous *script*): "`removeTable.sql`";
   (c) Create a third SQL script to populate the physical template: "`insertTable.sql`". **The data must allow validating queries requested at this stage of the work. All managers who are also clients must appear in the DB with the value "C" in the `atrdisc` in the table `PESSOA`**;
   (d) Also create a SQL script to erase all data in the tables (1 autonomous *script*): "`deleteTable.sql`".
2. Considering the relational schema obtained earlier and provided at the end of this document ("Addendum"), present the expressions in relational algebra (RA) that answer the following questions. To use all the operators, give **two alternative solutions**, if necessary, to the same question.
   (a) To retrieve a list of unique people (clients and managers) and their contact information (`nome`, `morada` and `telefone`).
   (b) Now, identify people who are also managers of a store (`nome`, `morada` and`telefone`).
   (c) Find out a list of people (`nome`, `morada` and `telefone`) who are not associated with any reservation.

(d) Retrieve bikes (`marca`, `modelo` and `estado`) that are not associated with any booking and are not electric.

(e) The collection of devices (`noserie`, `latitude` and `longitude`) of bicycles whose status is 'under maintenance' ("em manutenção").

(f) The names of users who have made bookings with electric bikes. Display information about customers and the number of bookings.

(g) Find the customers' name who have made reservations with a total value exceeding €100 (e.g.).

(h) Retrieves information (`email`, `endereço` and `localidade`) about stores and their associated phone numbers, including stores that may not have a phone number.

(i) For the client named "José Manuel", we would like a list of the bookings (`noreserva` and `loja`) he has made, including the date and start and end times, and the final price.

(j) List the customer(s) (`nome`, `morada`, `telefone` and `nacionalidade`) having the highest bookings in 2023[3].

(k) Retrieves the number of customers of Portuguese nationality and others. The result should show the attributes `nacionalidade` and the number of customers.

3. Design, in the PostgreSQL language, queries that produce the following results, using only one PostgreSQL statement. Save them all in a separate *script* named "`queries.sql`". **For each statement, the description of the reasoning followed must also be shown in a comment**.

(a) Implement in PostgreSQL the questions requested in 2. (You can choose one of the above solutions if you can use all the operators at this stage).

(b) Retrieve the names of all customers who made a reservation in a store located in Lisbon.

(c) Find the serial numbers of devices with a battery percentage greater than 50%, and list them in ascending order of their battery percentage.

(d) Retrieve the brand and the model (`marca` and `modelo`, respectively) of the bicycle with the highest autonomy (`autonomia`) among the available bikes.

(e) Retrieve the total number of reservations for each store along with their code (`codigo`) and the total number of reservations.

(f) Retrieve all the stores (`codigo` and `email`) that have made more than 5 bookings to date and list them in descending order of the number of bookings (**Note**: Make use of the date and time operators/functions).

[3]Note: You can use the function `YEAR()`, for simplicity

(g) Find the names of the customers who made reservations with bicycles having an `estado` (status) of 'under maintenance' ("em manutencao") last year (**Note**: Make use of the date and time operators/functions).

(h) Retrieve the information (`nome`, `morada` and `telefone`) of people who managed a store and have made reservations.

(i) Retrieve the names of customers who made reservations at a store managed by a person named "John" (assuming "John" is the manager of a store).

(j) Create a view `LISTJOAOFILIPE` that includes information about the customers who have made bookings at a shop run by "João Filipe". (**Note**: João Filipe is also a customer).

(k) Create a view `BICYCLESINNUMBERS` that includes information about the bicycles and their number. The information available should be the type of bike (electric and classic), its status ("under maintenance" and "other"[4]) and the total number.

4. Write a sql code that allows: a) change the attribute `mudanca` in `BICICLETA` by adding a new value, in this case 40; b) assign this new value to the bicycle(s) already registered in the DB with the `model` "Modelo-B" of the `brand` "Marca-A". **The sql code must guarantee atomicity in changes**.

Just so you know, all simplifications and optimisations made to the model must be indicated and justified.

**PS.1** You should consult the SGDB manual for information on date manipulation functions.

**PS.2** In the resolution you can't use the operators `with, limit and cross join` .

**Delivery deadline: 28 de Novembro de 2023 by 23:59**.

The delivery must include a document with the RA answers (in `PDF` format, duly labelled with the group number, student numbers and their names) and the PostgreSQL code (with .sql extension). The files produced **should be** compressed and sent electronically via Moodle.

**Note**: It should be possible to assess each learning objective in the material you hand in.

[4] These include all bikes that are not under maintenance and can be reserved.

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIC — LMATE

# Modelo Relacional a usar no trabalho prático (fase 2)
# Relational Model to be used in the second assignment

**Introdução a Sistemas de Informação**

**Semestre de Inverno 2023/2024**

**Versão 1.0**

# Modelo Relacional

Todos os atributos são obrigatórios nas relações, excepto quando indicado o contrário. As relações são apresentadas por ordem alfabética. **Nota:** Poderão não estar representadas todas as restrições de integridade, uma vez que esta componente também é avaliada. Mais, as restrições que não conseguiremos implementar no modelo físico serão asseguradas, posteriormente, na API que irão desenvolver na 3ª parte.

## BICICLETA

BICICLETA(id, peso, raio, modelo, marca, mudanca, estado, atrdisc, dispositivo).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| id | serial | |
| peso | numeric(4,2) | Em gramas. |
| raio | integer | Em polegadas, entre 13 e 23. |
| modelo | varchar(20) | |
| marca | varchar(30) | |
| mudanca | integer | Sistema de mudanças, que pode tomar como valores {1, 6, 18, 24}. Pode ser `NULL` se `CLASSICA` não tiver mudanças. A descrição pode ser alterada ou adicionada. |
| estado | varchar(30) | Pode tomar como valores {livre, ocupado, em manutenção}. |
| atrdisc | char(2) | Atributo discriminante que contém "C" para representar clássica ou "E" para eléctrica. |
| dispositivo | integer | FK referência de `DISPOSITIVO.{noserie}` |

## CLASSICA

CLASSICA(bicicleta, nomudanca).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| bicicleta | integer | FK referência de `BICICLETA.{id}`. |
| nomudanca | integer | Pode ser 0 a 5. O valor 0 representa sem mudanças. |

## CLIENTERESERVA

CLIENTERESERVA(cliente, reserva, loja).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| cliente | integer | FK referência de `PESSOA.{id}`. |
| {reserva, loja} | integer | FK referência de `RESERVA.{noreserva, loja}`. |

## DISPOSITIVO

DISPOSITIVO(noserie, latitude, longitude, bateria).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| noserie | integer | |
| latitude | numeric(6,4) | Corresponde à latitude numa coordenada GPS em graus decimais. |
| longitude | numeric(6,4) | Corresponde à longitude numa coordenada GPS em graus decimais. |
| bateria | integer | Em percentagem, i.e. pode tomar valores entre 0 e 100. |

## ELECTRICA

ELECTRICA(bicicleta, autonomia, velocidade).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| bicicleta | integer | FK referência de `BICICLETA.{id}`. |
| autonomia | integer | Em km. |
| velocidade | integer | Em km/h. |

## LOJA

LOJA(codigo, email, endereco, localidade, gestor).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| codigo | integer | |

continua na próxima página

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| email | varchar(40) | Inclui "@". Chave candidata. |
| endereco | varchar(100) | |
| localidade | varchar(30) | |
| gestor | integer | FK referência de PESSOA.{id} |

## PESSOA

PESSOA(id, nome, morada, email, telefone, noident, nacionalidade, atrdisc).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| id | serial | Valor sequencial. |
| nome | varchar(40) | |
| morada | varchar(150) | |
| email | varchar(40) | Inclui "@". Chave candidata. |
| telefone | varchar(30) | Móvel ou fixo. Chave candidata. |
| noident | char(12) | Número de identificação (cartão de cidadão ou passaporte).Chave candidata. |
| nacionalidade | varchar(20) | |
| atrdisc | char(2) | Atributo discriminante que contém "G" para representar o gestor ou "C" a clientes. |

## RESERVA

RESERVA(noreserva, loja, dtinicio, dtfim, valor, bicicleta).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| noreserva | serial | |
| loja | integer | FK referência de LOJA.{codigo} |
| dtinicio | timestamp | Tem o formato "aaaa-mm-dd hh:mm:ss". |
| dtfim | timestamp | Tem o formato "aaaa-mm-dd hh:mm:ss". Valor superior a `dtinicio`. O atributo pode tomar valor `NULL`. |
| valor | numeric(4,2) | Valor em euros. |
| bicicleta | integer | FK referência de BICICLETA.{id} |

### TELEFONELOJA

TELEFONELOJA(loja, numero).

| Atributo | Tipo | Restrições Integridade |
|---|---|---|
| loja | integer | FK referência de LOJA.{codigo} |
| numero | varchar(10) | Número de telefone fixo ou móvel. |

[EN]

## Relational model

All attributes are mandatory in relationships unless otherwise indicated. The relationships are presented in alphabetical order. **Note:** Not all integrity restrictions may be represented, as this component is also evaluated. Furthermore, the restrictions you cannot implement in the physical model will be ensured in the API developed in the 3rd part. The name of the attributes must remain in Portuguese, i.e. they must not be translated.

### BICICLETA

BICICLETA(id, peso, raio, modelo, marca, mudanca, estado, atrdisc, dispositivo).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| id | serial | |
| peso | numeric(4,2) | In grams. |
| raio | integer | In inches, between 13 and 23. |
| modelo | varchar(20) | |
| marca | varchar(30) | |
| mudanca | integer | Bicycle gear system can take as values {1, 6, 18, 24}. It can be `NULL` if `CLASSICA` has no changes. The description can be changed or added. |
| estado | varchar(30) | You can take the following values {livre, ocupado, em manutenção} ("free", "occupy" and "under maintenance"). |
| atrdisc | char(2) | Discriminant attribute containing "C" to represent classic or "E" for electric. |
| dispositivo | integer | FK references of `DISPOSITIVO.{noserie}` |

## CLASSICA

CLASSICA(bicicleta, nomudanca).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| bicicleta | integer | FK references of `BICICLETA.{id}`. |
| nomudanca | integer | It can be 0 to 5. The value 0 represents no gears. |

## CLIENTERESERVA

CLIENTERESERVA(cliente, reserva, loja).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| cliente | integer | FK references of `PESSOA.{id}`. |
| {reserva, loja} | integer | FK references of `RESERVA.{noreserva, loja}`. |

## DISPOSITIVO

DISPOSITIVO(noserie, latitude, longitude, bateria).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| noserie | integer | |
| latitude | numeric(6,4) | Corresponds to the latitude in a GPS coordinate in decimal degrees. |
| longitude | numeric(6,4) | Corresponds to the longitude in a GPS coordinate in decimal degrees. |
| bateria | integer | As a percentage, i.e. it can take values between 0 and 100. |

## ELECTRICA

ELECTRICA(bicicleta, autonomia, velocidade).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| bicicleta | integer | FK references of `BICICLETA.{id}`. |
| autonomia | integer | In km. |
| velocidade | integer | In km/h. |

## LOJA

LOJA(codigo, email, endereco, localidade, gestor).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| codigo | integer | |
| email | varchar(40) | Include "@". Alternate key. |
| endereco | varchar(100) | |
| localidade | varchar(30) | |
| gestor | integer | FK references of PESSOA.{id} |

## PESSOA

PESSOA(id, nome, morada, email, telefone, noident, nacionalidade, atrdisc).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| id | serial | Sequential value. |
| nome | varchar(40) | |
| morada | varchar(150) | |
| email | varchar(40) | Include "@". Alternate key. |
| telefone | varchar(30) | Mobile or landline. Alternate key. |
| noident | char(12) | ID number (citizen's card or passport). Alternate key. |
| nacionalidade | varchar(20) | |
| atrdisc | char(2) | Discriminant attribute containing "G" to represent the manager or "C" to represent customers. |

## RESERVA

RESERVA(no, loja, dtinicio, dtfim, valor, bicicleta).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| noreserva | serial | |
| loja | integer | FK references of LOJA.{codigo} |
| dtinicio | timestamp | It has the format "yyyy-mm-dd hh:mm:ss". |

continued on next page

| Attribute | Type | Constraints Integrity |
|---|---|---|
| dtfim | timestamp | It has the format "yyyy-mm-dd hh:mm:ss". Value greater than `dtinic`. The attribute can take value `NULL`. |
| valor | numeric(4,2) | Value in euros. |
| bicicleta | integer | FK references of BICICLETA.{id} |

## TELEFONELOJA

TELEFONELOJA(loja, numero).

| Attribute | Type | Constraints Integrity |
|---|---|---|
| loja | integer | FK references of LOJA.{codigo} |
| numero | varchar(10) | Mobile or landline. |

31/10/2023, Ana Beire, Matilde Pato e Nuno Leite