INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIC

Sistemas de Informação

Semestre de Verão 2024-2025

Partilha de Trotinetas Eléctricas — Electric Scooter Sharing

Projecto

(Fase normal v.1.01)

Ana Beire, Gustavo Jacinto, João Vitorino, Matilde Pato e Nuno Datia

Abril, 2025

Planeamento — Planning

As datas importantes a recordar são:

- Lançamento do enunciado: **24 de abril de 2025**
- Entrega final: **03 de junho de 2025**

[PT]

A entrega deve apresentar o relatório e código. O relatório deve seguir obrigatoriamente um dos modelos fornecidos, 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 o nome do projeto.

O pdf (e, o zip) gerado deve seguir o nome da seguinte forma: 'ProjSisInf-2425-**MM**.ext' (MM representa os dígitos do número do grupo, e 'ext' a extensão do ficheiro), e.g.: ProjSisInf-2425-01.zip.

[EN]

You must present the report and code. 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 name of the project being reported.*

*The generated pdf (and zip) must follow the name as follows: 'ProjSisInf-2425-***MM***.ext' (MM represents the digits of your group number, and 'ext' the file extension), e.g.: ProjSisInf-2425-01.zip.*

24 de Abril de 2025, Ana Beire, Gustavo Jacinto, João Vitorino, Matilde Pato e Nuno Datia

Objectivos de aprendizagem

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

- Garantir a correta implementação das restrições de integridade e/ou lógica de negócio;
- Usar funções, procedimentos armazenados e gatilhos para implementar restrições de integridade na base de dados;
- Desenvolver uma camada de acesso a dados, que use uma implementação de JPA (Jakarta Persistence);
- Conhecer um conjunto de dos padrões de desenho, incluindo `ValueObject` `DataMapper`, `LazyLoad`, `IndentityMap` `Repository`, `QueryObject` e `UnitOfWork`, e relacioná-los com o seu uso em JPA;
- Desenvolver uma aplicação em Java, que use adequadamente a camada de acesso a dados;
- Utilizar corretamente processamento transacional, através de mecanismos disponíveis no JPA ;
- Garantir a correta libertação de ligações e recursos, quando estes não estejam a ser utilizados.

Preambulo

Os sistemas de partilha de trotinetas eléctricas tornaram-se cada vez mais predominantes nas zonas urbanas, oferecendo aos cidadãos uma alternativa moderna e amiga do ambiente para as deslocações de curta distância. Estas soluções de transporte inovadoras têm várias vantagens, tais como zero emissões, transporte activo, facilidade de utilização, entre outras. Estes sistemas de partilha de trotinetas elétricas transformaram os cenários da mobilidade urbana, proporcionando uma forma sustentável, eficiente e agradável de percorrer distâncias mais curtas dentro das cidades. A sua popularidade continua a crescer à medida que as cidades procuram soluções inovadoras para reduzir o congestionamento do tráfego e promover opções de transporte mais amigas do ambiente.

Enunciado do trabalho (Documento de requisitos do sistema)

Pretende-se desenvolver um sistema de informação para o projecto "CITES". Trata-se de um serviço de partilha de trotinetas eléctricas numa cidade. Cada trotineta é caracterizada por um identificador (único), uma velocidade máxima (em km/h), uma autonomia (em km), um peso (em kg), um modelo, uma marca, e o custo da viagem. O serviço tem um custo de €1.00, para o desbloqueio da trotineta, acrescido de €0.15 por minuto de utilização. Cada modelo tem uma autonomia de fábrica (km). Para ser utilizador do serviço, a pessoa deve efectuar o registo. Esse registo deve conter o nome, o NIF (único), o *email* (único) e a data de registo. Quando o utilizador efectua o registo adquire um passe. Cada utilizador tem apenas, um passe. O passe deve guardar as seguintes informações: um número (único), a data de aquisição e o saldo (em €). O saldo é o balanço entre os carregamentos e as viagens realizadas.

Existem vários tipos de passe. O tipo de passe é descrito por uma referência (única), o número de dias e o preço (em €). A título de exemplo, uma referência pode ser: "passe residente anual". Podem ser efectuados vários carregamentos para o mesmo passe. Ao serem realizados os carregamentos, o sistema de informação regista uma data (único) e o valor (em €).

As trotinetas estão numa doca de uma determinada estação. Uma estação é caracterizada por um identificador (único) e a localização. Uma localização é constituída por latitude e longitude em base decimal. As docas são caracterizadas por um número (único) em cada estação e por um estado. O estado pode tomar os valores *livre*, *ocupado* e *indisponível*.

A reposição consiste na colocação ou remoção de uma ou mais trotinetas das respectivas docas da estação destino, para manter um relação entre docas livres e ocupadas. Quem gere o sistema pode registar um pedido de reposição numa determinada data (único) para uma dada estação, para atingir um valor máximo de ocupação. Quando o pedido é satisfeito deve ficar registado a data de reposição. Para satisfazer o pedido podem existir várias reposições, caracterizadas por um número (único) dentro do pedido de reposição e por uma data. A cada reposição está associado um único funcionário, podendo este efectuar várias reposições (colocar/remover). O funcionário é caracterizado por um nome, o NIF (único), o *email* (único), e um número (único) na empresa.

O utilizador quando inicia uma viagem dirige-se a uma estação e dependendo do estado da doca retira uma trotineta. A viagem é caracterizada por uma data de início, uma data final, uma avaliação, e uma mensagem, uma estação de início, e uma estação de destino (eventualmente nula). Não podem existir duas viagens para o mesmo utilizador na mesma data de início. Se a viagem estiver a decorrer, i.e., se data final é `NULL` a trotineta não pode ser utilizada noutras viagens, nem o utilizador iniciar outra viagem. A avaliação pode tomar valores entre 1 e 5. A mensagem só pode ser diferente de `NULL` se a avaliação também o for, i.e., se for registada uma

avaliação.

PS.1 Todos os atributos são obrigatórios, excepto se indicado em contrário.
PS.2 Os valores monetários devem ser apresentados com 2 casas decimais.
PS.3 Considere que todos os atributos com informação temporal têm data (dd/mm/aa) e tempo (hh:mm:ss).
PS.4 Crie apenas os tipos JPA necessários para atingir os objetivos de aprendizagem e cumprir com os requisitos pretendidos.

Nota: Devem ter em conta o código SQL entregue com o enunciado, que cria o modelo físico a usar, bem como o projeto Maven. Podem preencher as tabelas como os dados que considerem pertinentes para o desenvolvimento, bem como adicionar novas classes java e dependências. Não altere o nome das tabelas nem dos atributos; Não alterem a estrutura do projeto.

Resultados pretendidos

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

1. Adicione as restrições de integridade programaticamente em pl/pgsql, com recurso a gatilhos, nomeadamente:
   (a) só uma trotineta que está numa doca pode ser usada no início de uma viagem;
   (b) uma trotineta e um utilizador só podem participar numa única viagem a decorrer.
2. Complete a função `fx-dock-occupancy`, que calcula e devolve o nível de ocupação de uma estação (em percentagem, valor entre 0 e 1);
3. Garanta que a vista `RIDER` suporta inserções e atualizações;
4. Complete o procedimento armazenado `startTrip`, que inicia uma viagem.
5. Complete a aplicação Java para permitir:
   (a) Criar um cliente (usando a *VIEW* usada na alínea 3);
   (b) Listar um cliente, mostrando o tipo de passe associado;
   (c) Listar as docas e a sua ocupação (deve usar direta ou indiretamente a função criada na usada na alínea 2);
   (d) Iniciar uma viagem (Usar obrigatoriamente procedimentos armazenados);

(e) Colocar uma bicicleta na doca (Não pode usar procedimentos armazenados e deve usar *optimistic locking*).

6. Teste o *optimistic locking*, apresentando uma mensagem de erro adequada em caso de alteração concorrente conflituante que inviabilize a operação. No relatório deve estar descrita a forma como as situações de erro foram criadas para teste desta alínea.

Data limite para entrega: 3 de junho de 2024 até às 23:59.

A entrega deve incluir um relatório, enviados de forma eletrónica através do Moodle. Os documentos **são entregues** em formato `zip` e o relatório em `zip` (obrigatório).

Nota: Sugere-se que o relatório seja organizado de acordo com os pontos anteriores. Deve ser possível aferir cada um dos objetivos de aprendizagem no material que entregar.

Learning Objectives

At the end of the **first phase of the project**, students should be able to:

- Ensure the correct implementation of integrity constraints and/or business logic;
- Use functions, stored procedures, and triggers to implement integrity constraints in the database;
- Develop a data access layer that uses a JPA (Jakarta Persistence) implementation;
- Know a set of design patterns, including `ValueObject`, `DataMapper`, `LazyLoad`, `IdentityMap`, `Repository`, `QueryObject`, and `UnitOfWork`, and relate them to their usage in JPA;
- Develop a Java application that appropriately uses the data access layer;
- Correctly use transactional processing through mechanisms available in JPA;
- Ensure the correct release of connections and resources when they are not being used.

Preamble

Electric scooter sharing systems have become increasingly prevalent in urban areas, offering citizens a modern and environmentally friendly alternative for short-distance travel. These innovative transport solutions offer several advantages, such as zero emissions, active transportation, ease of use, among others. These electric scooter sharing systems have transformed urban mobility landscapes, providing a sustainable, efficient, and pleasant way to cover shorter distances within cities. Their popularity continues to grow as cities seek innovative solutions to reduce traffic congestion and promote more environmentally friendly transport options.

Project Statement (System Requirements Document)

The goal is to develop an information system for the "CITES" project. It is a service for sharing electric scooters within a city. Each scooter is characterised by: a unique identifier, a maximum

speed (in km/h), a range (in km), a weight (in kg), a model, a brand, and the cost per trip. The service has a cost of €1.00 for unlocking the scooter, plus €0.15 per minute of use. Each model has a factory-specified range (km). To use the service, a person must register. This registration must contain: the name, the NIF (unique), the *email* (unique), and the registration date. When the user completes registration, they acquire a pass. Each user may only have one pass. The pass must store the following information: a unique number, the acquisition date, and the balance (in €). The balance represents the difference between top-ups and trips made.

There are several types of pass. The type of pass is described by: a unique reference, the number of days, and the price (in €). As an example, a reference might be: "annual resident pass". Multiple top-ups can be made for the same pass. When top-ups are made, the information system records: a date (unique), and the amount (in €).

The scooters are located at a dock within a given station. A station is characterised by: a unique identifier, and its location. A location consists of latitude and longitude in decimal format. Docks are characterised by a unique number within each station and by a state. The state can take the following values: *free*, *occupied*, and *unavailable*.

Restocking consists of placing or removing one or more scooters from the respective docks at the destination station, in order to maintain a balance between free and occupied docks. System managers can record a restocking request on a given date (unique) for a specific station, aiming to achieve a maximum occupancy level. When the request is fulfilled, the restocking date must be recorded. To satisfy the request, multiple restocking operations may occur, each characterised by a unique number within the restocking request and by a date. Each restocking operation is associated with a single employee, although an employee may carry out multiple restocking operations (placing/removing scooters). An employee is characterised by: a name, the NIF (unique), the *email* (unique), and a unique number within the company.

When a user starts a trip, they go to a station and, depending on the state of the dock, retrieve a scooter. The trip is characterised by: a start date, an end date, a rating, and a message, a start station, and a destination station (which may be null). There cannot be two trips for the same user with the same start date. If the trip is ongoing, i.e., if the end date is `NULL`, the scooter cannot be used in other trips, nor can the user start another trip. The rating can take values between 1 and 5. The message can only be different from `NULL` if a rating is also recorded.

PS.1 All attributes are mandatory, except where indicated otherwise.
PS.2 Monetary values must be presented with 2 decimal places.
PS.3 Assume that all attributes with temporal information include both date (dd/mm/yy) and time

(hh:mm:ss).

PS.4 Create only the JPA types necessary to achieve the learning objectives and fulfil the specified requirements.

Note: You must take into account the SQL code provided with the project statement, which creates the physical model to be used, as well as the Maven project. You may populate the tables with any data you deem pertinent for development, and you may add new Java classes and dependencies. Do not change the name of tables or attributes; Do not modify the project structure.

Expected Results

Considering the learning objectives, the following outcomes must be produced:

1. Add the integrity constraints programmatically in PL/pgSQL, using triggers, specifically:
   (a) only a scooter that is in a dock can be used to start a trip;
   (b) a scooter and a user may only participate in one ongoing trip at a time.
2. Complete the function `fx-dock-occupancy`, which calculates and returns the occupancy level of a dock-station (as a percentage, a value between 0 and 1);
3. Ensure that the view `RIDER` supports insertions and updates;
4. Complete the stored procedure `startTrip`, which initiates a trip.
5. Complete the Java application to enable:
   (a) Creating a customer (using the *VIEW* referred to in item 3);
   (b) Listing a customer, showing the associated type of pass;
   (c) Listing the docks and their occupancy (you must directly or indirectly use the function created in item 2);
   (d) Starting a trip (you must use stored procedures);
   (e) Placing a bicycle in a dock (You must not use stored procedures and must use *optimistic locking*).
6. Test the *optimistic locking*, presenting an appropriate error message in the case of a conflicting concurrent update that prevents the operation. The report must describe how the error situations were created to test this item.

Submission deadline: **3 de junho de 2024 by 23:59**.

The submission must include a report, sent electronically via Moodle. The documents **must be submitted** in `zip` format, and the report must also be included in the `zip` file (mandatory).

Note: It is suggested that the report be organised according to the points above. It must be possible to assess each learning objective in the material you submit.