

# Assignment #2

## Three-tier Programming with Object- Relational Mapping

---

Coimbra, 15 de novembro de 2021

2018296643 Inês Margarida Silva Teixeira

[ines.margarida18@hotmail.com](mailto:ines.margarida18@hotmail.com)

2018298731 João Pedro Pacheco Silva

[joaopedro@student.dei.uc.pt](mailto:joaopedro@student.dei.uc.pt)

## 1. INTRODUÇÃO

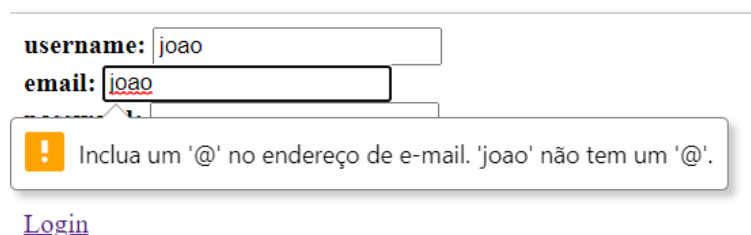
Propõe-se neste trabalho implementar uma aplicação web para gerir uma companhia de autocarros, de forma a ganhar familiaridade com o desenvolvimento de aplicativos empresariais de três camadas usando o modelo Java Enterprise Edition (Java / Jakarta EE). De modo a realizar a divisão por camadas acima referida usamos uma pasta para cada uma delas, tendo assim a camada Web que comunica com o utilizador contendo as interfaces do projeto, a camada Enterprise JavaBeans(EJB) que lida com os negócios, fornecendo um desenvolvimento rápido e simplificado de aplicações java e por fim a Java Persistence API (JPA) onde criamos a estrutura dos dados guardados. Neste projeto usufruímos do Docker onde tivemos acesso a todos os recursos necessários para executar o programa.

## 2. CAMA DE APRESENTAÇÃO

Todas as páginas da aplicação foram implementadas em JPS e seguem as seguintes estruturas:

### Formulários:

Páginas onde é necessário que o utilizador introduza informação são apresentadas como formulários em que cada campo a ser preenchido é uma linha com a respetiva identificação à esquerda e zona de input à direita. Cada input tem restrições sobre o tipo de informação que os utilizadores podem submeter, como, por exemplo, não poder submeter números negativos ou não deixar campos por preencher.



The screenshot shows a login form with two input fields. The first field is labeled 'username:' and contains the text 'joao'. The second field is labeled 'email:' and contains the text 'joao'. Below the email field, there is a red error message box with a yellow warning icon. The message reads: 'Inclua um '@' no endereço de e-mail. 'joao' não tem um '@'.'. Below the error message, there is a blue link labeled 'Login'.

Figura 1 - Página de Sign in

### Listas:

Páginas que fazem listagem de informação apresentação os dados como uma sequência de itens na vertical onde cada item é composto pelo conjunto atributos que o caracterizam. Em baixo dos atributos há ainda um conjunto de botões que permitem realizar operações com respetivo o item, os quais só aparecem caso a operação seja possível. Caso haja a possibilidade de filtrar os dados, os filtros iram estar no topo da lista.

---

**Trips:**

|                  |  |                  |  |        |
|------------------|--|------------------|--|--------|
| 17/11/2021 22:36 |  | 29/11/2021 22:36 |  | update |
|------------------|--|------------------|--|--------|

**From:** lisboa **To:** viseu  
**Date:** 2021-11-20 12:36:00.0  
**Price:** 10.0€

**From:** coimbra **To:** lisboa  
**Date:** 2021-11-21 22:34:00.0  
**Price:** 7.5€

Figura 2 - Página com lista de viagens

### 3. CAMADA DE NEGÓCIO

Os serviços implementados na camada de negócio são disponibilizados através de dois EJBs, um com os serviços disponíveis para os utilizadores da aplicação chamado “BusinessLogic” e outro com os serviços disponíveis para os gerentes da companhia chamado “AdminServices”. Ambos são *stateless*, não retendo qualquer informação sobre o cliente ao longo das invocações de métodos.

#### 3.1. Gerenciamento de transações

O gerenciamento de transações foi feito usando a *Java Transaction API (JTA)*, que através da anotação `@Transactional` possibilita demarcar um método como sendo uma transação onde há *commit* se este terminar sem erros ou um *rollback* caso ocorra uma exceção. Essa notação foi usada em todos os métodos que fizessem alterações na base de dados e todos os métodos onde fosse necessário garantir a atomicidade das operações.

```
@Transactional
public boolean returnTicket(String auth, long id){...}
```

Figura 3 - Uso da notação `@Transactional`

### 3.2. Autenticação

A autenticação dos usuários é feita inicialmente nas páginas de login através dos seus emails e passwords. Após o login é gerado um *authentication token*, que será usado autenticar os utilizadores nas invocações de métodos seguintes, evitando ter que guardar o email e a password na sessão.

De forma a que os utilizadores não autenticados tivessem apenas acessos às páginas de *login* e *sign in*, todas as restantes páginas foram guardadas numa pasta chamada “secured”. Pedidos de recursos dentro dessa pasta são intercetados por um filtro chamado que verifica se a sessão tem um *authentication token* válido e que redireciona para a página de login caso esta não tenha.

```
public void doFilter(ServletRequest request,
                    ServletResponse response,
                    FilterChain chain)
    throws IOException, ServletException {

    System.out.println("Accessing the filter...");
    HttpServletRequest httpReq = (HttpServletRequest) request;
    HttpSession session = httpReq.getSession(false);

    if (session != null && session.getAttribute("auth") != null) {
        if(businessLogic.authentication(
            (String) session.getAttribute("auth"))){
            System.out.println("Valid token...");
            chain.doFilter(request, response);
        } else{
            System.out.println("Invalid token...");
            request.getRequestDispatcher("/Login.jsp").forward(request,
                                                            response);
        }
    } else {
        System.out.println("Token not found...");
        request.getRequestDispatcher("/Login.jsp").forward(request, response);
    }
}
```

Figura 4 - filtro para autenticação

Relativamente ao acesso aos EJBs, todos os métodos à exceção do login e do sign in, requerem um *authentication token* valido de modo a garantir que os métodos não são invocados por utilizadores não autenticados. Caso o *token* não seja valido o método é simplesmente abortado.

### 3.3. Transferência de dados

As entidades com a informação necessária nunca são enviadas para a camada de apresentação. Por norma estas são mapeadas em Data Transfer Objects que posteriormente são enviados no seu lugar. Nos casos mais simples, onde só é necessário um atributo ou uma lista de um dos atributos das entidades (id, preço, ...), optamos por não fazer o mapeamento e em vez disso enviar apenas o atributo ou lista de atributos.

## 4. CAMADA DE DADOS

### 4.1. Diagrama ER

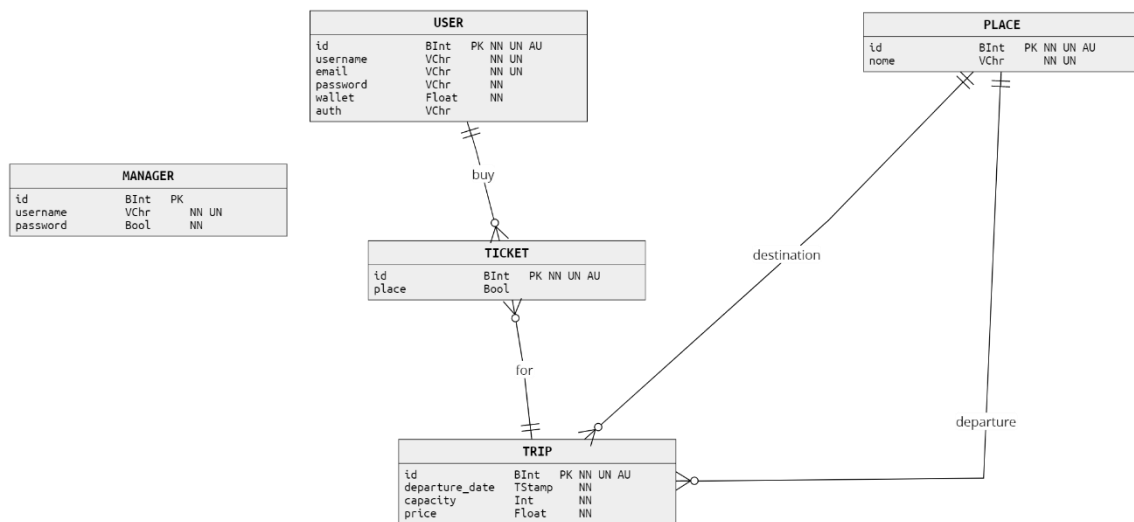


Figura 5 - Diagrama ER

A camada de dados é composta por 5 entidades chamadas:

- **USER** – entidade que contém os dados de cada utilizador registado na aplicação.
- **MANAGER** – entidade que contém os dados de cada gerente registado na aplicação.
- **MANAGER** – entidade que contém os dados necessários para os gerentes se poderem autenticar.
- **PLACE** – entidade que contém os nomes dos locais de partida e de destino disponíveis para as viagens.
- **TRIP** – entidade que contém os dados das viagens.

#### ***4.2. Restrições***

Para além das visíveis no diagrama ER, a base de dados apresenta também as seguintes restrições:

- O par (place, trip\_id) da entidade TICKET é *unique*, para evitar que haja 2 ou mais tickets para o mesmo lugar numa viagem.
- O atributo place da entidade TICKET não pode conter valores menores que 1.
- O atributo wallet da entidade USERS não pode conter valores menores que 0.
- O atributo price da entidade TRIP não pode conter valores menores que 0.01.
- O atributo capacity da entidade TRIP não podem conter valores menores que 1.