Faculdade de Ciências e Tecnologia da
Universidade de Coimbra
Departamento de Engenharia Informática

Integração de Sistemas 2021/2022

A2 - Three-tier Programming with Object-Relational Mapping

Henrique Teixeira Fonseca	uc2021162362@student.uc.pt	2021162362
Nuno Marques da Silva	uc2018285621@student.uc.pt	2018285621

Professor:
Filipe João Boavida Mendonça Machado de Araújo

Versão 1.0

Coimbra, 6 de novembro de 2021

Índice

1 Introdução **ii**

2 Nível de Apresetação **iii**

3 Nível de Trabalho **v**

4 Nível de Dados **vi**

5 Estrutura do Projeto **vii**

1. Introdução

Este relatório foi redigido no âmbito da cadeira de Integração de Sistemas (IS), do Mestrado em Engenharia Informática da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (FCTUC), regida pelo professor Filipe João Boavida Mendonça Machado de Araújo.

O âmbito deste trabalho é implementar uma aplicação de três camadas (Apresentação, Funcional, Dados) para uma empresa de viagens de autocarro, que ofereça funcionalidades específicas para utilizadores e administradores. Para iniciar a implementação do projeto foram seguidas as indicações do capítulo 8 do livro disponibilizado na plataforma UC-Student, Jakarta EE in Practice (inclusive a utilização da versão 16 do Java).

2. Nível de Apresetação

Para implementação do Nível de Apresentação do projeto, contido no ficheiro *web*, foram usados ficheiros *.jsp* e *.html*, correspondentes à parte gráfica, e ficheiros *.java*, correspondentes a *servlets* que efetuam a operações de redirecionamento de páginas e comunicação entre a UI e o Nível Funcional (textitBusiness), e *filters* que impedem o acesso de utilizadores a certas páginas (ou por não terem permissões ou não terem sessão iniciada). Estes ficheiros correspondentes aos *filters* (inclusivé ficheiros *.html* correspondentes às páginas de erro) não estão a ser utilizados apeser de implementados, uma vez que a funcionar, passavam a existir *bugs* de redirecionamento sobre o *flow* normal da aplicação (apesar de o seu objetivo ser alcançado). A organização destes ficheiros pode ser vista na Figura 2.1.

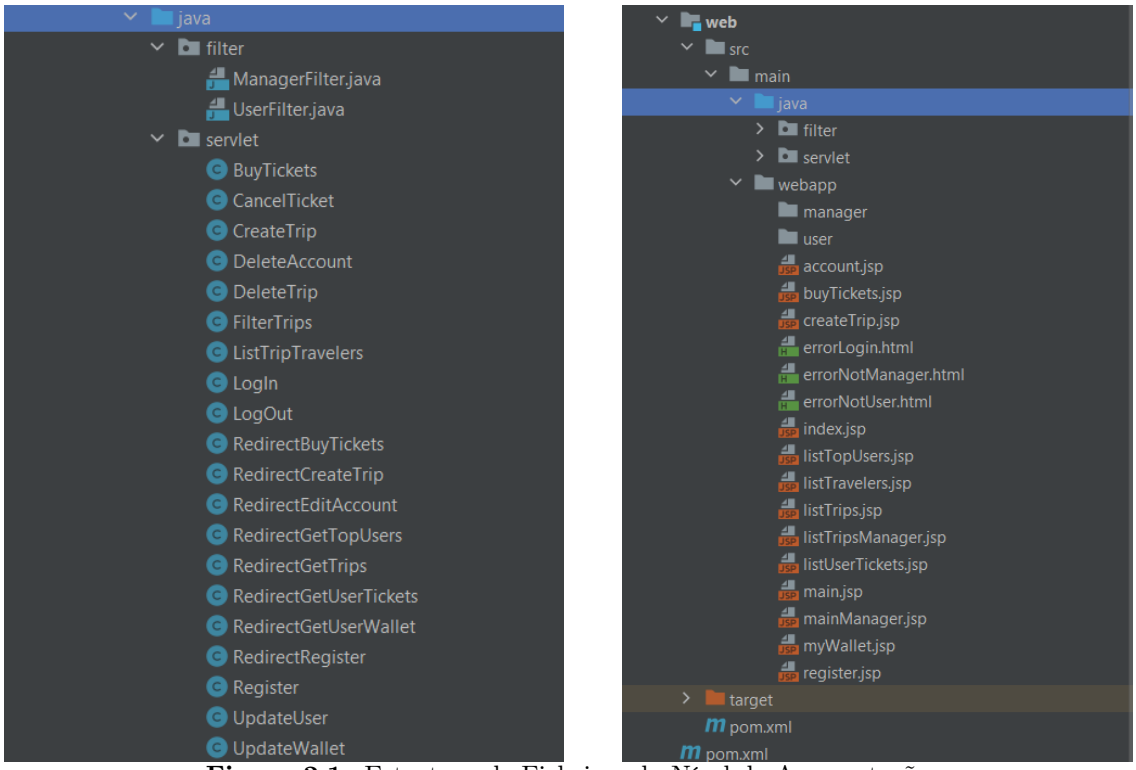

Figura 2.1: Estrutura de Ficheiros do Nível de Apresentação

Quanto à estrutura visual das páginas da interface, foi criado um esquema inicial que a representa, para às páginas do utilizador comum. A diferença para um utilizador administrador são as operações a efetuar, sendo que há páginas diferentes para todas as operações (listar users, listar viagens - utilizador, listar viagens - administrador, comprar bilhete, etc.). O acesso a estas páginas está à volta de um menu principal, apresentado imediatamente após o *login* ou o registo (início de sessão). Assim, o esquema referido neste parágrafo pode ser visualizado na Figura 2.2.

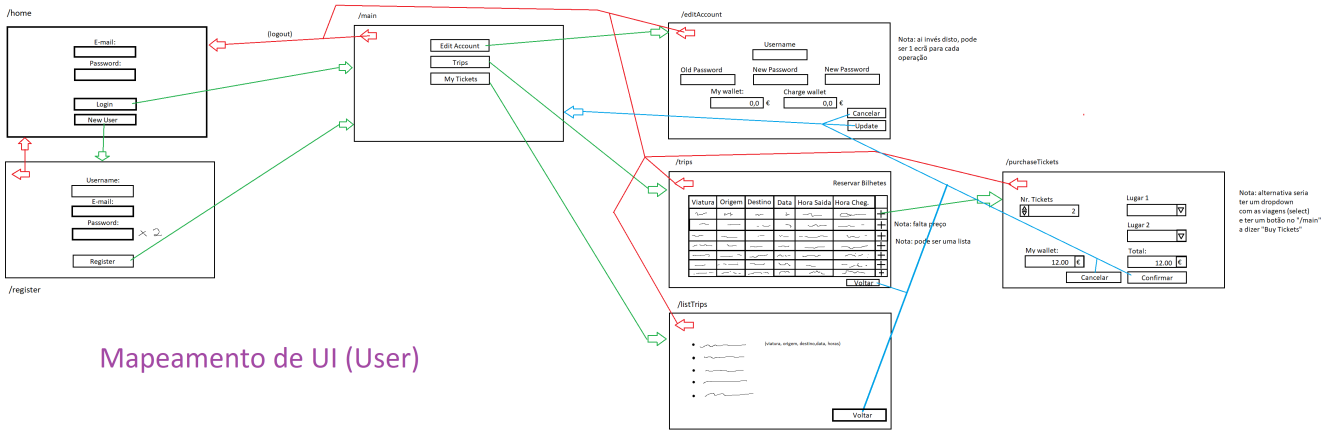

Figura 2.2: Flow da Interface

Quanto ào aspeto visual, a interface consiste num conjunto de inputs de texto, numéricos, selects de datas, botões para executar operações e links para o retorno ao menú principal da aplicação. Um exemplo pode ser visto formulário apresentado na Figura 2.3.

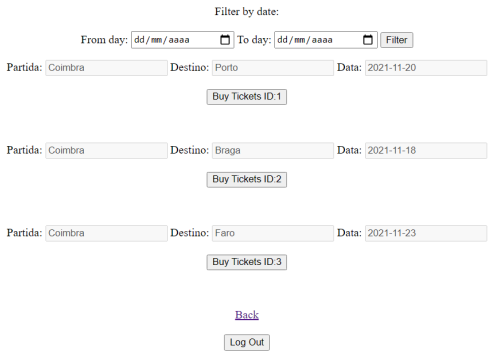

Figura 2.3: Exemplo UI

Tal como requisitado, as palavras passe são encriptadas para posteriormente serem guardadas na base de dados em conjunto como uma chave de encriptação.

Ao fazer *login*, a palavra passe fornecida pelo utilizador é encriptada com uma chave de encriptação associada ao seu endereço de *e-mail*, e só depois é comparada com a palavra passe encriptada que está guardada na base de dados. Se forem iguais significa que o utilizador se autenticou com sucesso e pode avançar no programa, caso contrário este é levado a inserir novamente os dados de *login*.

3. Nível de Trabalho

O Nível Funcional corresponde à comunicação entre o Nível de Apresentação (através dos *servlets*) e o Nível de Dados, através do *Entity Manager* (usando as funções *find* e *merge*) e *Typed Queries*.

Este nível está no módulo *ejbs* e contém um *stateless bean*, o que significa que não guarda informação no servidor sobre a sessão do cliente, necessitando este de se identificar a cada pedido, e um *bean* de inicialização de uma instância(@Startup e @Singleton), que permite a criação automática de um utilizador administrador. O primeiro *bean* referido é o que, efetivamente, executa as operações de comunicação com a Base de Dados (BD) (implementada usando PostgresSQL), passando todo o tipo de informações sobre utilizadores e operações da sua conta (editar conta, eliminar conta, adquirir bilhetes, cancelar bilhetes, etc.), bilhetes, a sua criação e remoção, e viagens, as mesmas operações que os bilhetes. Para extrapolação das funções deste *bean*, foi necessária a implementação de uma interface do mesmo.

A informação só é enviada da BD para o Nível de Apresentação quando necessária, ou seja, quando há um redirecionamento para uma página que necessite de apresentar os valores de certos campos de uma entidade, o Nível de Apresentação efetua um pedido ào Nível Funcional para que este aceda ao Nível de Dados e retorne os valores a apresentar nesta nova página a apresentar.

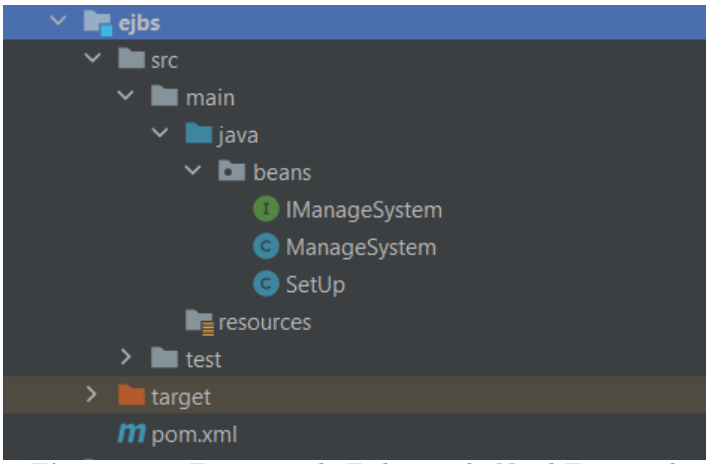

Figura 3.1: Estrutura de Ficheiros do Nível Funcional

4. Nível de Dados

Analisando o contexto do problema foi decidido que iriam ser necessárias três entidades, sendo que duas mantêm uma relação de *One-To-Many* com a terceira.

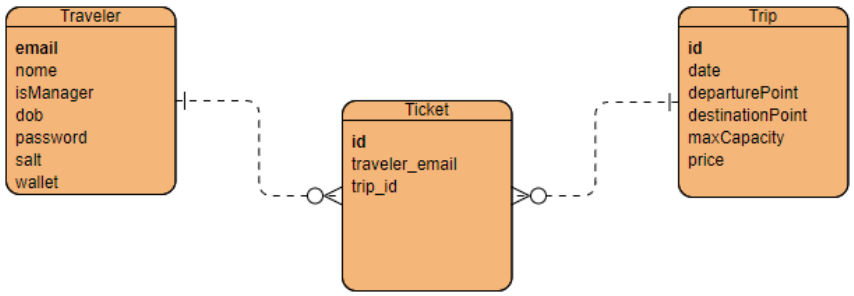

Figura 4.1: Diagrama ER

- A entidade *Traveler* é referente ao utilizador. Para chave primária foi definido o e-mail, dado que este é único. A única diferença entre um utilizador comum e um *Manager* é simplesmente a variável binária *isManager* que tem valor positivo quando se trata de um.
- A entidade *Trip* é referente a uma viagem. Para chave primária foi criado um *ID* auto-incremental, aquando da criação de uma nova viagem, de maneira a este seja único para cada viagem.
- A entidade *Ticket* refere-se a um bilhete de uma viagem e por isso tem associado a si apenas 1 utilizador e 1 viagem. Por consequência, para que um utilizador pudesse comprar mais do que um bilhete a chave primária é também um *ID* auto-incremental.

```java
@Entity
public class Traveler implements Serializable {
    private static final Long serialVersionUID = 1L;

    private String name;
    @Id
    private String email;
    private String password;
    private String salt;
    private LocalDate dob;
    private Boolean isManager;
    private Double wallet;
    @OneToMany(mappedBy = "traveler", cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Ticket> tickets;
```

Figura 4.2: JPA - Entidade *Traveler*

5. Estrutura do Projeto

Como referido anteriormente e visível no ficheiro *pom.xml*, o programa é constituido pelos módulos *jpa*, correspondente aos ficheiros das entidades, ou seja, Nível de Dados, *ejbs*, correspondente ao Nível Funcional, *web*, correspondente ao Nível de Apresentação, e *ear*, que contém um conjunto de ficheiros *.jar* que efetua a compilação dos restantes módulos para serem executados pela aplicação *Wildfly*. Isto pode ser visualizado na Figura 5.1.

Figura 5.1: Estrutura do Projeto e parte do ficheiro *pom.xml* do mesmo