Mestrado em Engenharia Informática

Integração de Sistemas - Projecto 2

Leandro Pais - 2017251509
João Branco - 2021179862

15 de novembro de 2021

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

# Índice

# 1 Introdução

O 2º trabalho prático da disciplina de Integração de Sistemas, tem por base a criação de uma aplicação de 3 camadas, relativamente a uma plataforma de gestão de bilhetes de viagens.

Esta aplicação divide-se nas seguintes camadas:

- Camada de Apresentação
  - Interface gráfica da aplicação.
- Camada de Negócio/Lógica
  - Secção onde está implementada a lógica da aplicação e a invocação aos métodos CRUD (*Create, Read, Update, Delete*).
- Camada de Dados/Base de Dados
  - Camada onde se encontram definidas as entidades utilizadas para a construção da base de dados e comunicação entre camadas necessária para este projecto.

Para o desenvolvimento deste projecto utilizaram-se certas tecnologias como por exemplo, *Enterprise Java Beans* (camada de Negócio), *Java Persistence API* (camada de Dados) e JSP/JSF (camada de apresentação).

# 2 Camada de Apresentação

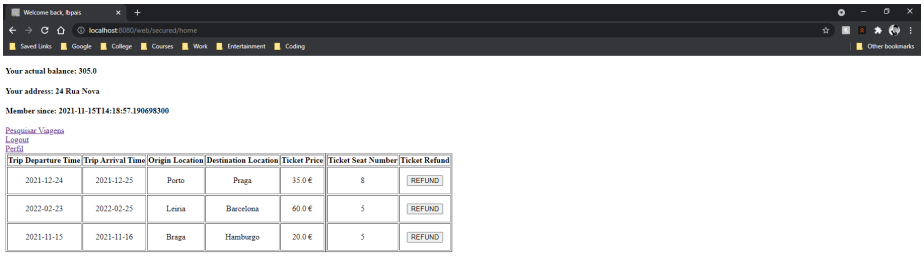


Figura 1: Estrutura Web

O módulo do projecto *Presentation Tier* tem como objectivo fornecer uma interface gráfica ao utilizador para a utilização do nosso serviço. Como é descrito na Figura 1, decidimos criar uma estrutura prática para organizar a parte *web* do nosso trabalho.

A parte *web* pode ser dividida nos seguintes módulos:

- Dados, *code-behind*
  - *Servlets*, que gerem toda a informação apresentada na interface gráfica, estando responsáveis por invocar os *java beans*.
  - *WebFilter*, responsável por gerir as autorizações de acesso à nossa aplicação.
- Visual
  - *.jsp*, páginas baseadas em HTML (*HyperText Markup Language*), que são responsáveis por apresentar a informação necessária ao utilizador.

Em relação ao armazenamento da *password* do utilizador, é guardada de forma encriptada na base de dados. Esta encriptação é realizada com o auxílio do algoritmo MD5, onde é gerada uma *hash* equivalente à *password* introduzida pelo utilizador, sendo essa mesma *hash* armazenada na base de dados.

Uma das tarefas que mais dificuldade ofereceu foi a implementação do *WebFilter*, funcionalidade essa que tem como objectivo filtrar e, restringir caso ne-

cessários, todos os pedidos feitos à aplicação web, oferecendo algum tipo de autorização no acesso à aplicação.

# 3 Camada de Negócio

O módulo *Business Tier* do projecto tem como responsabilidade fornecer toda a lógica necessária para garantir a comunicação entre a interface gráfica e a base de dados.

Esta camada, tendo em conta o nosso projecto, pode ser dividida em 2 componentes: os Enterprise Java Beans (com a sua interface e implementação) e uma pasta de Utilidades (definida por *Utils*) onde se encontra, por exemplo, a classe para se proceder à encriptação da palavra-chave do utilizador). Para se encriptar a *password* utiliza-se o algoritmo *MD5*, para posteriormente se armazenar a *hash*, equivalente à palavra-chave do utilizador, gerada por este algoritmo na base de dados.

Em relação aos Java Beans, é nesta camada que são recebidos e processados os pedidos do utilizador, que vêm do *front-end*. Foram criados 4 *Beans* correspondente a cada entidade/tabela criada, de forma a que esta camada se encontre modular e organizada por funcionalidades pertencentes a cada tabela, ou seja por exemplo, tudo o que sejam operações relacionadas com os *tickets* estão no Bean dos *Tickets*.

Todos os *beans* têm implementado a sua interface, de modo a que o *front-end* tenha acesso só às definições dos métodos, onde a implementação da interface contém as *queries* para a realização das operações relativas às funcionalidades requeridas no enunciado do projecto.

Todos os *beans* estão definidos como *Local* e *Stateless*, visto que todas as camadas desta aplicação estão alojadas de forma local e não necessitamos de guardar informações sobre o cliente.

Descrevendo os *beans* criados e as suas funcionalidades mais importantes, podemos organizá-los da seguinte forma:

- *ClientBean*
  - *Login*
  - *Registo*
  - *Reembolso de bilhete*
  - *Listar todos os bilhetes pertencentes a um cliente*
  - *Listar todas as viagens pertencentes a um cliente*
  - *Atualizar os dados do cliente*
- *TicketBean*
  - *Adicionar um ticket novo*
  - *Comprar ticket*
  - *Listar todos os bilhetes disponíveis para uma determinada viagem*
  - *Re-atribuir bilhete*, de forma a que possa ser vendido ao público

- *TripBean*
  - *Listar todas as viagens*
  - *Listar todas as viagens disponíveis entre determinadas datas*
  - *Listar todas as viagens de um determinado cliente*
  - *Listar os passageiros de uma viagem*
- *WalletBean*
  - *Listar todas as carteiras*
  - *Listar uma carteira de um determinado cliente*
- *CmBean*
  - *Criar company manager*
  - *Listar top 5 clientes*
  - *Criar viagem*
  - *Calcular lucros diários*
  - *Remover viagem, reembolsar clientes e notificar via mail*

Em termos de transferência de dados entre as camadas de apresentação e de negócio, segui-se de forma generalizada, a norma sugerida, em que o *front-end* envia somente para os *beans* os identificadores únicos das entidades envolvidas na operação pretendida, sendo depois retornado pelo *bean* uma mensagem de sucesso ou erro conforme a realização, ou não, da operação.

Em operações que envolvam listagem de dados, como por exemplo, listagem de bilhetes/viagens/clientes, entre outros, os *beans* recebem o identificador único relativamente à entidade pretendida, caso necessário, sendo devolvida uma lista de objectos equivalentes à entidade pretendida, por exemplo, para uma listagem de viagens de um cliente, é recebido pelo *bean* o identificador do cliente pretendido, e é devolvido uma lista de *Trips*.

# 4 Camada de dados/Base de Dados

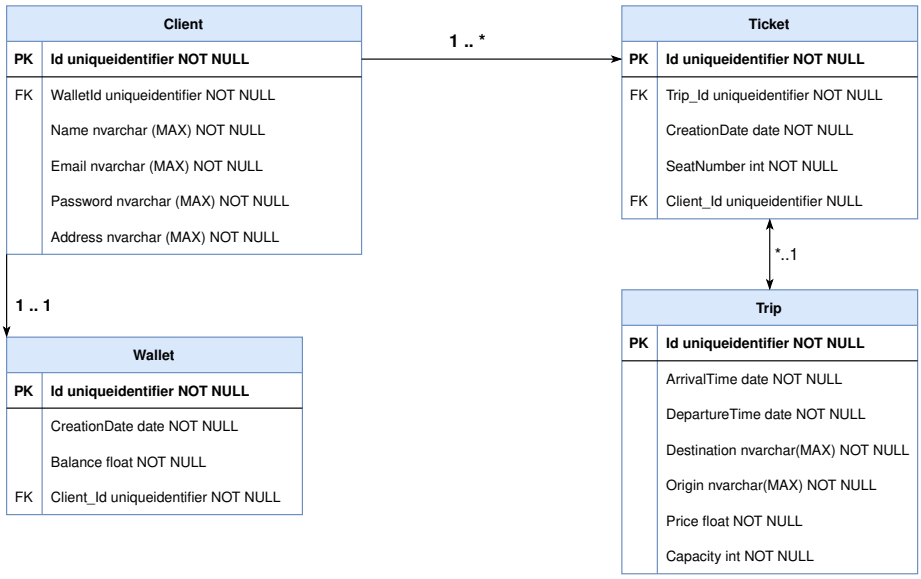


Figura 2: Diagrama Entidade-Relação

Como base para a construção para este diagrama, foi tido em conta que primeiro, o cliente pode comprar vários bilhetes de viagem, e, para reflectir isso, adoptou-se a relação *One-To-Many* entre a entidade *Client* e a entidade *Ticket*. Já entre as entidades *Ticket* e *Trip* foi adoptada a relação *One-To-Many* visto que uma viagem pode ter vários bilhetes disponíveis para venda, e vários bilhetes de uma viagem estão disponíveis para venda.

Adicionalmente criámos uma entidade *Wallet* de modo a reflectir o saldo do cliente, visto que o cliente pode carregar a sua carteira para depois poder utilizar na compra de bilhetes ou para receber possíveis reembolsos. Esta relação é de *One-To-One* entre a *Wallet* e o *Client*.

# 5 Estrutura do Projecto e *Packaging*

Para este projecto foi usado a tecnologia Maven para que tornasse a compilação dos diferentes projectos prática e automatizada. Desta forma, foi necessário acrescentar um novo módulo ao projecto (EAR) para que conseguisse gerar um ficheiro .ear responsável por compactar todas informações e dependências dos três módulos descritos nos 3 capítulos anteriores. Foram ainda usados alguns *plugins* para automatizar algumas tarefas, nomeadamente, a mais importante, o *deploy* do ficheiro .ear no Wildfly. Para este propósito foi utilizado o *plugin wildfly-maven-plugin*.

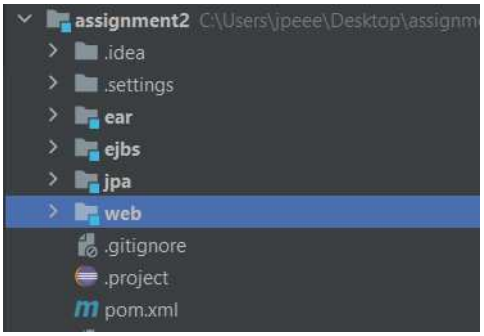


Figura 3: Estrutura do projecto

# 6 Ferramentas utilizadas

Neste capítulo, é feita uma breve descrição de todas as tecnologias essenciais para a realização do projecto.

## 6.1 Maven

O Maven foi utilizado neste projecto para ser a ferramenta responsável pela gestão e compilação do projecto em questão. Esta gestão é feita essencialmente realizada através de um ficheiro, *pom.xml*, onde, por exemplo, estão descritas as dependências e respectivas versões necessárias para a execução do projecto, o nome e versão do projecto, entre outros.

## 6.2 IntelliJ

O Intelij foi o IDE (*Integrated Development Environment*, Ambiente de Desenvolvimento) escolhido por ambos os membros do grupo para se poder desenvolver o projecto.

## 6.3 GitLab

Foi utilizado um gestor de repositório *online* para gerir o código dos dois membros do grupo, facilitando o trabalho dos elementos do grupo para um desenvolvimento mais eficiente.

# Referências

[1] *Wildfly Mail Configuration*
*https://tinyurl.com/4zj8zwvd*

[2] *Jakarta EE in Practice by Filipe Araújo & Nuno Laranjeiro*