Integração de sistemas

Three-tier Programming with Object-Relational Mapping

Mariana Loreto 2018280762
Mariana Lança 2018288500

15 de Novembro de 2021

Introdução

Este trabalho foi realizado no âmbito da disciplina de Integração de Sistemas e tem como objetivo construir uma aplicação web que faça uso do modelo de **três camadas**: A de **apresentação**, correspondente à interface *web* que permite ao utilizador obter e fornecer informações à aplicação, fazendo uso de *Servlets* e páginas do tipo JSP para tal; A camada de **aplicação** que, através do *Enterprise JavaBean* (EJB) processa os dados e faz a conexão entre as outras duas camadas; por fim, a camada de **dados**, responsável por manusear os dados contidos na base de dados com o auxílio da *Java Persistence API* (JPA).

Deste modo, fez-se uso da linguagem *Java* para a codificação do projeto, do *PostgreSQL* para a base de dados e, para o deploy da implementação, o *WildFly Application Server*, contidos num contentor do *Docker*.

Nível de apresentação

O nível de apresentação é referente à estrutura das páginas web e é através desta que o utilizador interage com a aplicação. Para tal, utilizaram-se *JSPs* de modo a criar as páginas e *Servlets* para fazer a conexão com a camada de negócio e fornecer e obter dados da aplicação.

Na implementação deste nível, a maior dificuldade que se encontrou foi perceber quando utilizar o *doPost* e o *doGet* nos *Servlets*. Com o avançar do projeto, reconheceu-se que para enviar dados para a camada de apresentação se devia optar pelo GET, enquanto para extrair dados da mesma, o POST é a opção mais indicada.

Assim, chegou-se ao seguinte fluxo de aplicação:

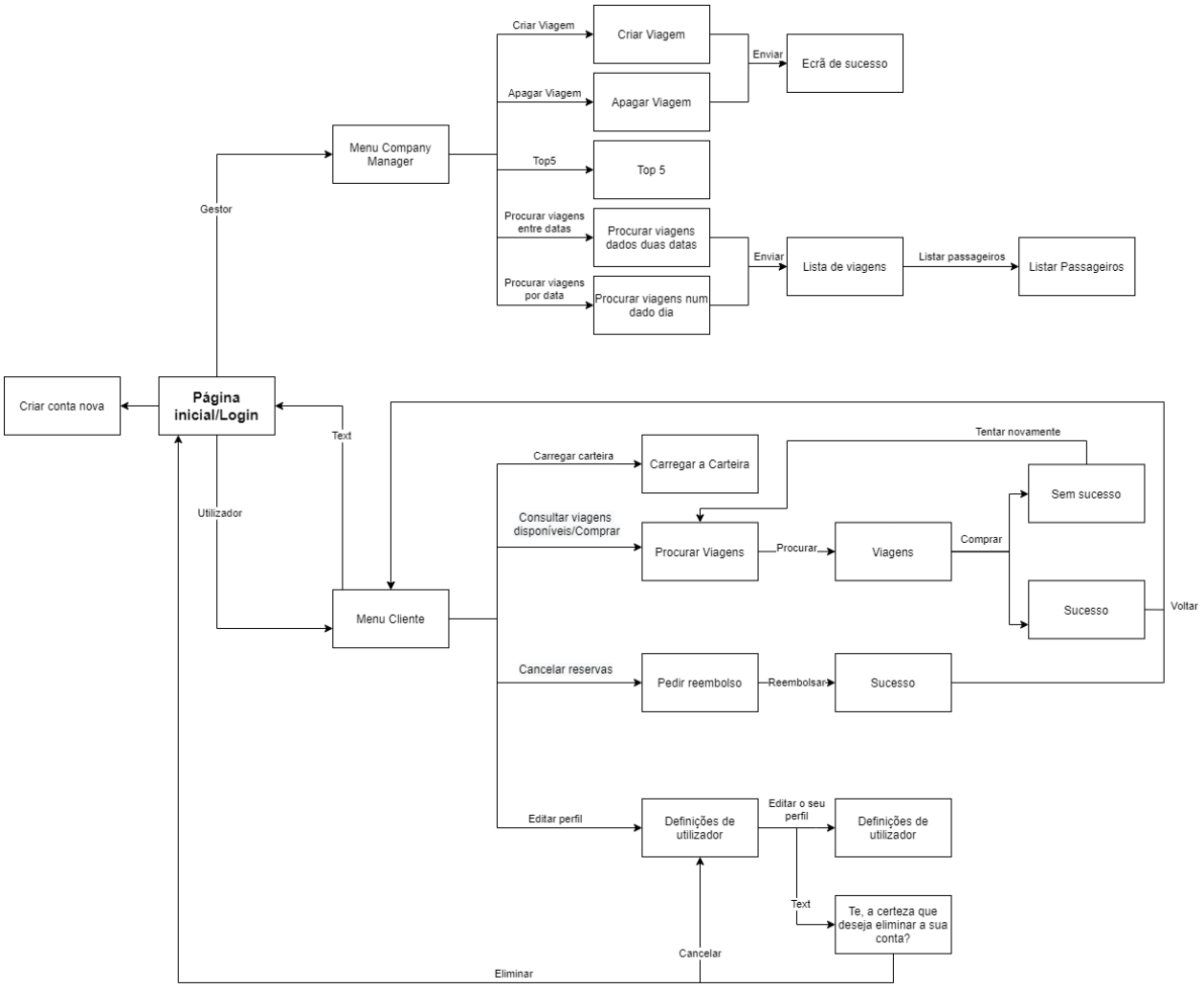

Fig.1 Diagrama de fluxo

A aplicação começa no ecrã de *Login*, possibilitando ao utilizador se registrar, caso ainda não possua conta. Feito o login, o utilizador vai ser redirecionado para o menu principal. Este menu vai diferir consoante o perfil do utilizador, isto é, se é gerente da empresa, ou um utilizador comum. A partir deste menu é possível realizar todas as funcionalidades exigidas no enunciado.

As páginas seguem um modelo de *interface* bastante semelhante ao de seguida apresentado:

Fig.2 Interface da aplicação

A *interface* é bastante simples, contém sempre no topo, pelo menos dois botões: o primeiro para o utilizador fazer *logout* e o segundo para voltar ao menu principal. No caso de haver um terceiro, este terá o propósito de voltar à página anterior. De seguida, é apresentado o título da página e por fim o conteúdo que se pretende apresentar.

Quando é pedido ao utilizador que insira uma *password*, esta é ocultada, à medida que o conteúdo vai sendo inserido, como se pode constatar na figura 3. Tal apenas é pedido no *login* e no registo e, apenas no segundo caso, esta é armazenada. Deste modo, quando o utilizador pressiona o botão “Submeter”, a informação inserida é passada, através do *Servlet*, para a camada de negócio, para o *bean ManageClients*. Este acede a camada de dados e adiciona à base de dados o objeto do tipo *ClientUser*, codificando a password aquando da inserção, a partir da extensão *pgcrypto* do *PostgreSQL*, de modo a que esta não seja armazenada.

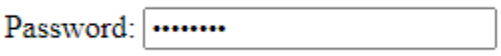

Fig.3 Inserção da *password*

Sobre esta camada é criado um filtro de segurança que permite que, no caso de não haver nenhum *token* de autenticação, o utilizador não logado seja redirecionado para uma página de erro.

Nível de negócio

Já o nível de negócio é responsável por interagir com a base de dados e processar a informação que circula pela aplicação. Esta gestão é feita através de *Beans*.

Para este projeto, criaram-se três *beans*:

- **ManageClientUser** - Responsável por tratar de todas as funcionalidades necessárias para o utilizador interagir com a aplicação. Será neste *bean* que se tratará, entre outros, da compra de uma viagem.
- **CompanyManagers** - Será a partir deste *bean* que a aplicação poderá responder às necessidades dos gerentes para realizar tarefas como criação e eliminação de viagens.
- **Mail -** Criado para possibilitar o envio de e-mails por parte da aplicação. Será este o *bean* responsável pelo envio das *daily revenues* aos gerentes da empresa

Atendendo a que um bean *stateless* não armazena o estado do utilizador na sua sessão, uma instância do bean pode servir qualquer utilizador que esteja ligado à aplicação. Já para um bean *stateful*, é criada uma instância para cada utilizador, que responde aos seus pedidos apenas e armazena o seu estado. Por este motivo, uma sessão deste tipo seria computacionalmente mais pesada. Atendendo a que não é necessário manter um estado conversacional, i.e, não há necessidade de manter informação de um utilizador específico nas variáveis do bean, optou-se por tornar ambos os *beans* deste trabalho *stateless*.

Para cada um deste beans foi criada uma interface de forma a ser possível usar os métodos criados na camada de apresentação.

Além disso, de forma a enviar os dados da camada de negócio para a camada de apresentação, recorreu-se aos *Data Transfer Objects* (DTOs), que permitem enviar a informação estritamente necessária. Um exemplo claro do uso de um DTO no projeto, é aquando do envio da lista do top 5 para a camada de apresentação, em que a única informação passada é o nome de cada cliente, visto ser a única informação que vai ser exposta.

ManageClientUser

Neste *bean* são realizadas as várias ações referentes a um *ClientUser*, isto é, um utilizador comum da aplicação, nomeadamente:

1. Login de um utilizador
2. Criar/eliminar um utilizador
3. Logout
4. Editar os dados pessoais
5. Carregar a carteira do cliente
6. Listar viagens num dado intervalo
7. Comprar/devolver um bilhete e escolher o lugar

Para algumas destas tarefas é necessário apresentar ao utilizador informação, a partir da base de dados. No entanto teve-se o cuidado de apenas passar para a camada de apresentação a informação estritamente necessária, através dos DTOs. Tal é feito nos pontos 4, em que se permite ao utilizador ver os seus dados pessoais atuais antes e depois de os alterar; nos pontos 5 e 7, o utilizador tem acesso ao valor na sua carteira; no ponto 7, novamente, são apresentadas todas as viagens disponíveis num dado intervalo (no caso de ser uma compra) ou as futuras viagens do cliente, de modo a este poder devolvê-los, no caso de assim o desejar.

Além disto, em vários outros pontos é necessário aceder ao nível de dados, como no 1, em que se faz uma pesquisa através do email para descobrir se o utilizador existe e, no caso de tal se verificar, se a password está correta; nos ponto 2 e 7 faz-se uma inserção/remoção de um utilizador ou de um bilhete de viagem; e nos pontos 4 e 5 update dos dados.

CompanyManagers

Neste segundo *bean* estão contidas as funções que permitem completar as seguintes ações esperadas para um gerente de empresa:

1. Login de um utilizador
2. Criar/eliminar futuras viagens de autocarro
3. Listar os passageiros que contêm mais viagens
4. Listar viagens num dado intervalo ou numa dada data, por ordem
5. Listar todos os passageiros de uma dada viagem

Tal como anteriormente referido, é necessário enviar informação para o nível de apresentação a partir da de dados.

Assim, no ponto 3 obtém-se apenas os nomes dos vários passageiros de uma viagem e, nos 4 e 5, as que estão inseridas no dado intervalo ou numa única data, para que estes sejam enviados para o JSP para o utilizador os poder visualizar.

Ademais, no ponto 1, corresponde ao login, dá-se um processo bastante semelhante ao descrito no *ManageClientUsers* e, no 2, ao de criação/remoção de um utilizador.

(As tarefas de notificação através de email, apesar de estares codificadas, não estão funcionais)

Nível de dados

Por fim, o nível dos dados funciona sobre a base de dados de modo a manter toda a informação.

Neste projeto utilizaram-se 4 entidades. Todas contém um id único.

- **ClientUser** - É o cliente. Esta entidade inclui o email,password e toda a informação pessoal necessária sobre o mesmo, bem como uma lista de bilhetes que este possua.
- **CompanyManager** - Representa um gerente da empresa. Tal como o *ClientUser* inclui o *email* e a *password* imprescindíveis para o *login*.
- **Trip** - A viagem contém a informação básica de uma viagem de autocarro, que inclui o ponto de partida e chegada, a data de partida, e os bilhetes vendidos, entre outros.
- **Ticket** - Esta entidade, bilhete, faz ligação entre o cliente e a viagem e, além desses dois, contém também o lugar reservado.

Diagrama de Entidades Relacionais

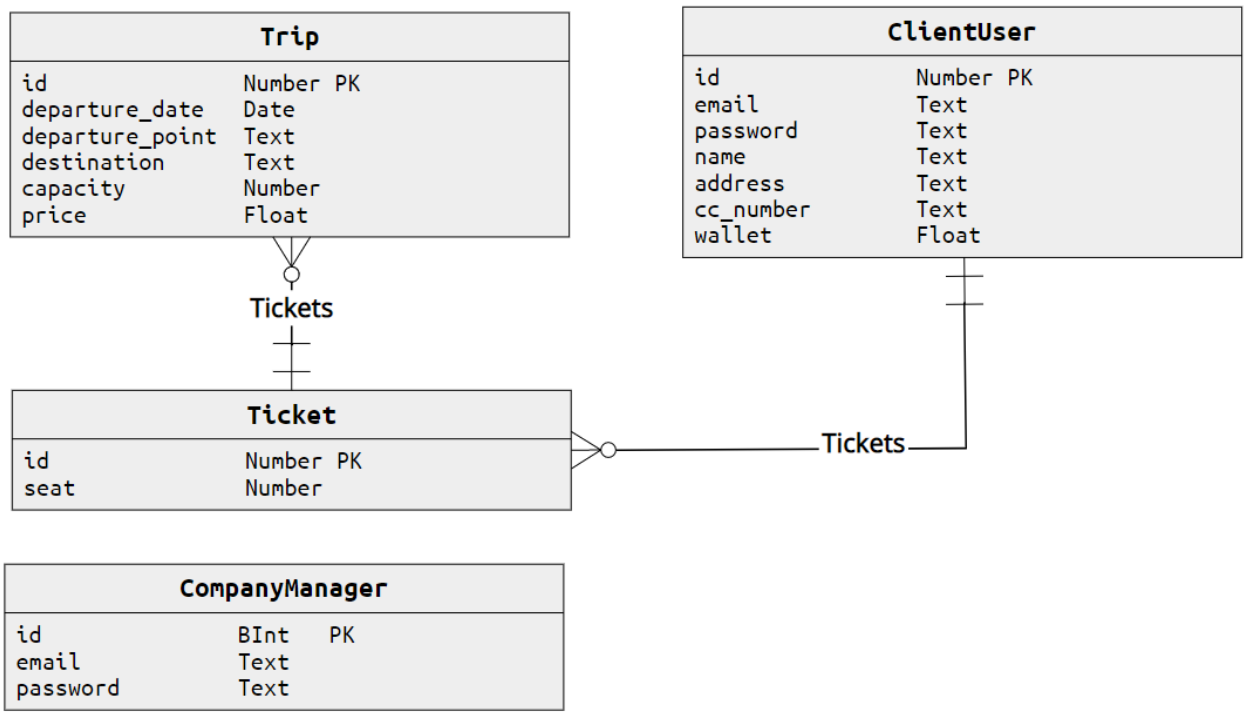

Fig.4 Diagrama ER

Gestão de negócios e packaging

O projeto está dividido em quatro módulos, como se pode ver pelo pacote do programa (pom.xml), o **ejbs**, onde estão os beans e que é correspondente ao nível de negócio; o **jpa**, que contém as entidades da base de dados e os DTOs e está associada à camada de dados; o **ear**, que tem como objetivo fazer empacotamento dos módulos num único arquivo de modo a que o deployment destes seja simultâneo e coerente; o **web** que contém os ficheiros *JSP*, os *Servlets* e o filtro de segurança de login, pelo que se refere ao nível de apresentação.

Conclusão

Em suma, considera-se que os objetivos propostos para este projeto foram cumpridos e que foi possível construir uma aplicação web com base no modelo de três camadas de forma correta e fazendo uso das várias tecnologias baseadas em Java, lecionadas na aula. Foi

possível compreender como é feita a ligação entre as várias camadas e a importância de cada uma delas para o bom funcionamento de uma aplicação.