Departamento de Engenharia Informática

Integração de Sistemas

João Cruz - 2018288464
André Silva - 2018277921

15 de novembro de 2021

1 Introdução

Neste projeto foi pedido que se criasse uma aplicação web para uma empresa de autocarros. A aplicação terá como objetivo facilitar a compra de bilhetes por parte dos clientes da empresa. Além disso, também permitirá aos managers criar e apagar viagens e verificar informações das mesmas.

Esta aplicação, criada usando Jakarta EE com a utilização de um Wildfly Application Server, foi subdividida em três camadas: *Data*, *Presentation* e *Business Tier*, cada uma responsável por parte das operações.

A *Data Tier* é responsável pela definição das entidades e relações entre estas. Para além disso será responsável pela estruturação do armazenamento dos dados na base de dados PostgreSQL. A *Presentation Tier* será composta pelos jsps, que são responsáveis pela estrutura das páginas web e pelos servlets, responsáveis pelo flow entre páginas web e realização das interações com os *Enterprise JavaBeans*.

A *Business Tier* é composta pelos *beans*, servindo de ponte entre a *Presentation* e a *Data Tier* permitindo que a informação passe da base de dados para os jsps.

2 Data Tier

Para a base de dados definiu-se o seguinte ER:

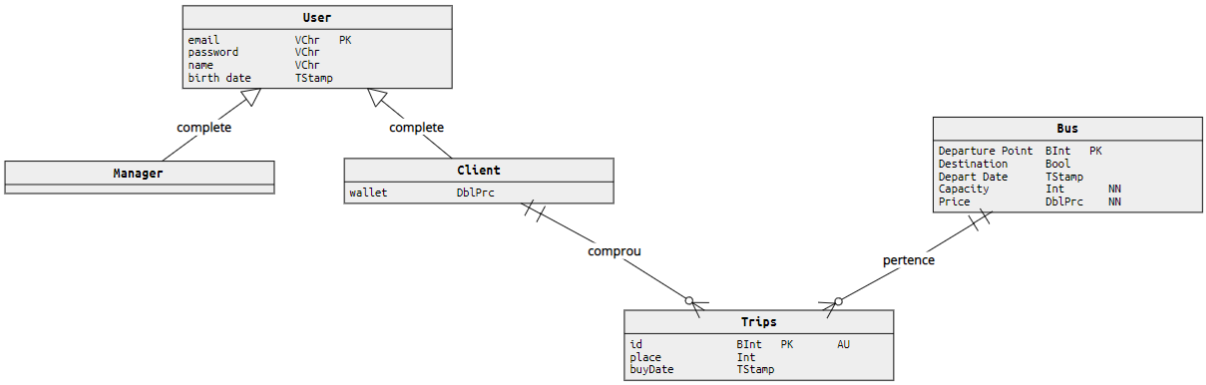

Figura 1: Diagrama ER

A classe User vai ser estendida pelo *Manager* e pelo *Client*. Desta forma vai ser possível diferenciar entre um cliente comum e um manager da empresa. Foram definidos apenas os 4 campos de dados necessários para registo (email, password, name e data de nascimento) devido a ser um projeto pequeno, mas seriam facilmente adicionados mais campos caso necessário. Foi decidido que só um cliente poderá comprar bilhetes, daí ter-se feito a ligação apenas do *Client* até às *Trips*.

A classe *Bus* tem todas as informações necessárias para se saber onde e quando o autocarro irá partir, onde vai chegar e o preço.

A classe *Trips* tem o lugar e a data de compra, de maneira a ser possível fazer o *daily revenue* e uma ligação para o *Bus* e o *Client* de maneira a ser possível que cliente comprou o bilhete e a qual autocarro pertence.

Com base neste ER, criaram-se as seguintes entidades, usando JPA:

- Bus
- Ticket
- Manager
- Client

De maneira a fazer a extensão, usou-se o *@MappedSuperclass*, numa classe chamada Users, da qual o Client e o Manager estendem.

Para serem criados managers, foi criado um projeto a parte seguindo as configurações do livro do capitulo de JPA[1]. Encontra-se disponivel para correr com o seguinte comando: java -jar target/JPA-standalone.jar <email> <nome> <password> <data de nascimento>.

Na base de dados foi criada a extensão pgcrypto de maneira a ser possível fazer a encriptação, que vai ser falada no próximo ponto, para criar a extensão é preciso correr o seguinte comando:
create extension pgcrypto;

3 Presentation Tier

A presentation tier que é composta por vários jsps de forma a cumprir os requisitos da aplicação tem o seguinte flow:

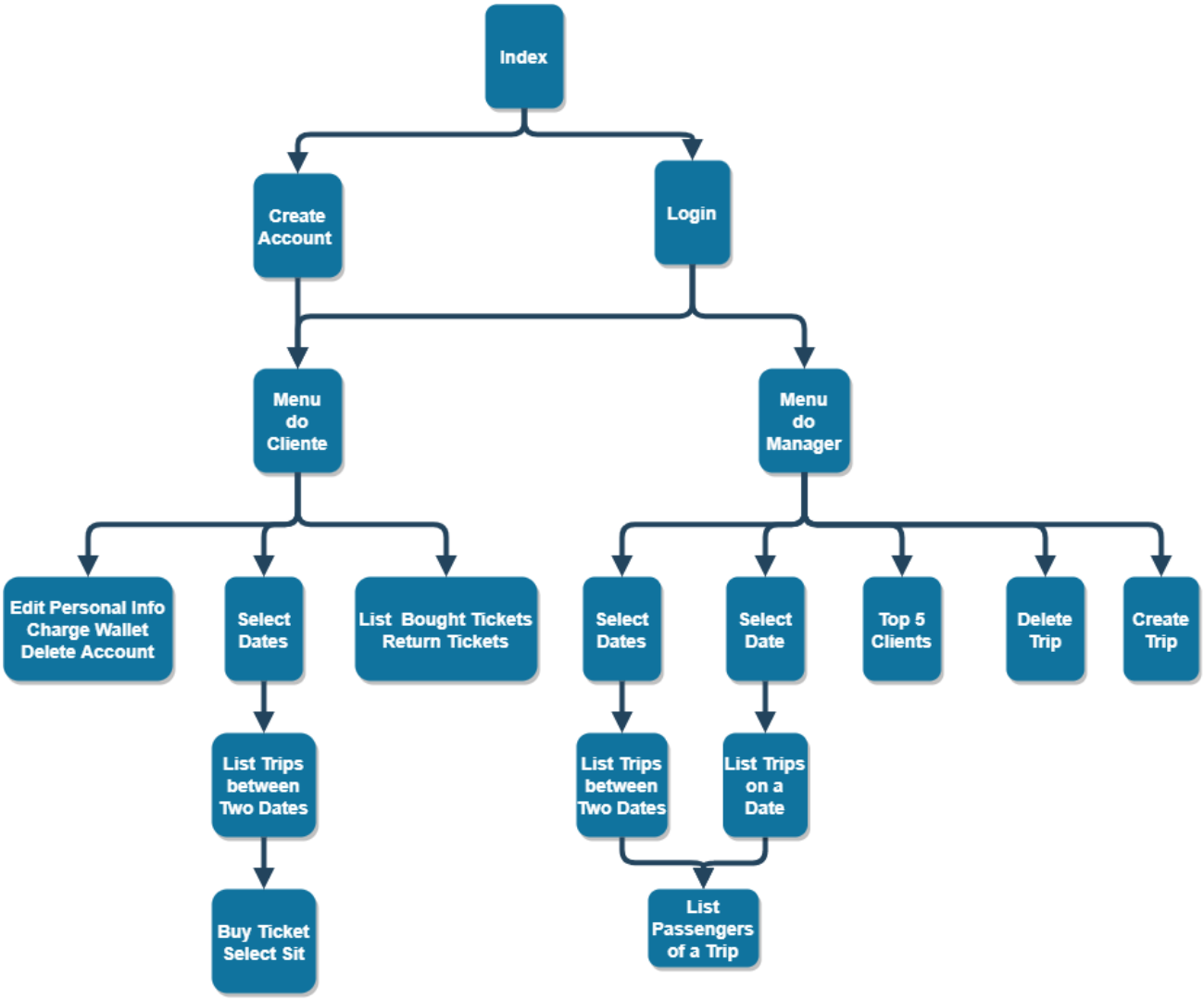

Figura 2: Diagrama de fluxo da aplicação

Não foi criado nenhum template para as páginas já que todas apresentam um layout bastante simples.

Para cada jsp foi criado um servlet com uma função de get e post, de maneira a melhorar a organização do projeto. É através deste servlet que é possivel o jsp comunicar com as outras tiers e que

receber os dados necessários.

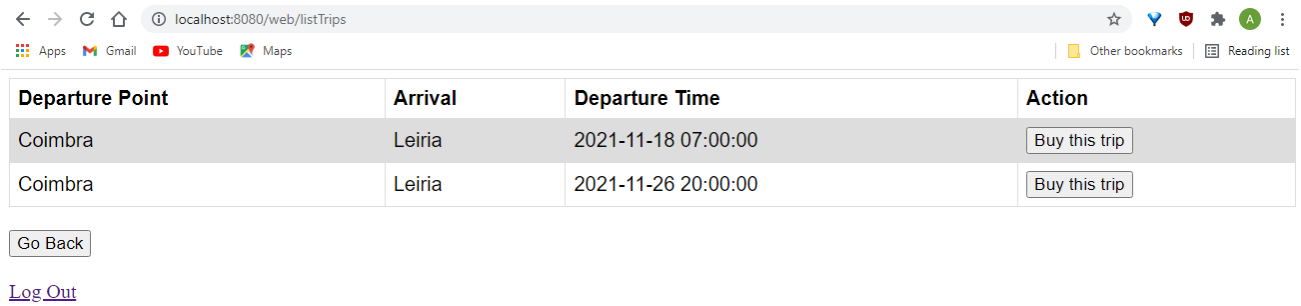

Figura 3: Screenshot da listagem do ecrã da aplicação

Em relação ás dificuldades, não se encontrou nenhuma em especial. A única parte que gerou confusão foi como fazer urls dinâmicos, mas uma vez que usando JSTL não é possivel acabamos por resolver usando *hidden input fields* nos jsps.

4 Business Tier

Durante o projeto foram criados 3 beans, um para as *Trips*, *ManageTrips*, um para os *Clients*, *ManageClients*, e outro para ser possivel enviar email com os ganhos diários, *ManageRevenue*. Para ser possível injetar na *presentation tier* foi necessário criar uma interface para cadas, *IManageTrips*, *IManageClients* e *IManageRevenue*

Os 2 primeiros beans são *stateless*, uma vez que, os métodos necessários são independentes de si e não necessitam do estado do cliente, deste modo é possível melhorar a performance da aplicação. Já o ManageRevenue é *singleton* uma vez que é necessário iniciar apenas uma vez por sessão para ser possível comunicar por email os ganhos diários.

Os beans recebem os dados das servlets através de variáveis, como String ou Integer, e retornam em forma de *Data Transfer Objects*, DTO, de maneira a não sobrecarregar o sistema e não serem enviados dados que não seriam usados, assim através do DTO só são passados para a *presentation tier* os dados extremamente necessários para cada caso.

Serviços de cada EJB:

- ManageTrips
  - addTrip → Permite ao manager criar uma viagem
  - getTrips → Obtem todas as viagens num dia ou entre 2 datas
  - getTripsUser → Obtem todas as viagens compradas por um cliente
  - getSeats → Obtém os lugares disponíveis de um autocarro
  - buyTicket → Compra o bilhete para um cliente
  - deleteTrip → Elimina uma viagem do sistema reembolsando todos os clientes e enviando-lhes emails
  - returnTicket → Elimina um bilhete do cliente, reembolsando-o
- ManageClients
  - addClient → Registo de um cliente
  - login → Indica se o cliente existe ou não no sistema
  - updateInfo → Atualiza as informações do cliente
  - delete → Elimina um cliente, assim como todos os seus bilhetes
  - getClientInfo → Retorna todas as informações do cliente

  - chargeWallet → Permite ao cliente carregar a carteira
  - getTopClients → Obtem os clientes que mais viagens compraram
  - checkWallet → Verifica se o cliente tem dinheiro para comprar a viagem selecionada

- ManageRevenue
  - addClient → Envia aos managers, por email, os ganhos diários

5 Project Management and Packaging

O projeto em si esta dividio em 4 módulos, uma para os *ebjs*, *jpa*, *web* e *ear*.

O módulo dos *ebjs* tem todos os beans do projeto e suas interfaces de maneira a serem injetadas nos servlets.

O módulo do *jpa* contem todas as entidades da base de dados, assim como o *persistance* necessário para a criar. Também foram criados *Data Transfer Objects*(DTOs) para que nas comunicações da*Business Layer* com a *presentation layer* fossem apenas passados os dados estritamentes necessários ao ecrã a ser apresentado.

O módulo do *web* contem tantos os servlets que comunicam com os beans como os jsps que servem como *presentation layer*. Para além disto ainda inclui um filter que permite a realização de uma autenticação sempre que se acede aos jsps impedindo acesso não autorizado.

O módulo do *ear* identifica todos os restantes módulos e está responsável pelo deploy no Wildfly.

Quando o Wildfly está a correr já se pode iniciar a utilização da aplicação. Cada link ou botão num jsp redireciona para um servlet que irá fazer as operações necessárias para concretizar a passagem para o ecrã seguinte. Esse servlet poderá então interagir com a business layer para obter informação necessária para o ecrã seguinte e por fim redirecionar para o ecrã seguinte.

Referências

[1] Nuno Laranjeiro Filipe Araújo. *Jakarta EE*. 2021.