Trabalho Prático - Reserva de Voos

Marco Costa, a93283 Rita Teixeira, a89494
Tomás Francisco, a93193

4 de agosto de 2023

Sistemas Distribuídos

Licenciatura em Engenharia Informática

Índice

Capítulo 1

Introdução

Este relatório foi realizado no âmbito do projeto proposto na Unidade Curricular de Sistemas Distribuídos da Licenciatura em Engenharia Informática. A realização deste projeto teve como objetivo a construção duma plataforma de reserva de voos sob a forma dum par cliente-servidor utilizando *sockets* e *threads*.

Com este relatório, pretendemos explicar as abordagens tomadas pelo grupo de trabalho que levaram à estrutura e boa implementação do projeto.

Este relatório encontra-se orientado por quatro secções, onde será explorada a arquitetura do projeto e sua postura implementação, as funcionalidades básicas definidas e ainda funcionalidades extras.

Capítulo 2

Arquitetura

O sistema está dividido em cliente e servidor. Os clientes fazem os pedidos ao servidor, que se encarrega de os executar e de enviar uma resposta ao cliente.

O Cliente, como não tem acesso à base de dados, não tem camada de dados. Isto significa que apenas é composto pela camada de interface e de negócios, cujo objetivo é fazer os pedidos ao servidor e de receber as respostas.

Relativamente ao Servidor, este, como não é usado diretamente por um utilizador, não tem camada de interface. Isto significa que apenas é composto pelo servidor que atende os pedidos, pela camada de negócios e pela camada de dados que acede à base de dados.

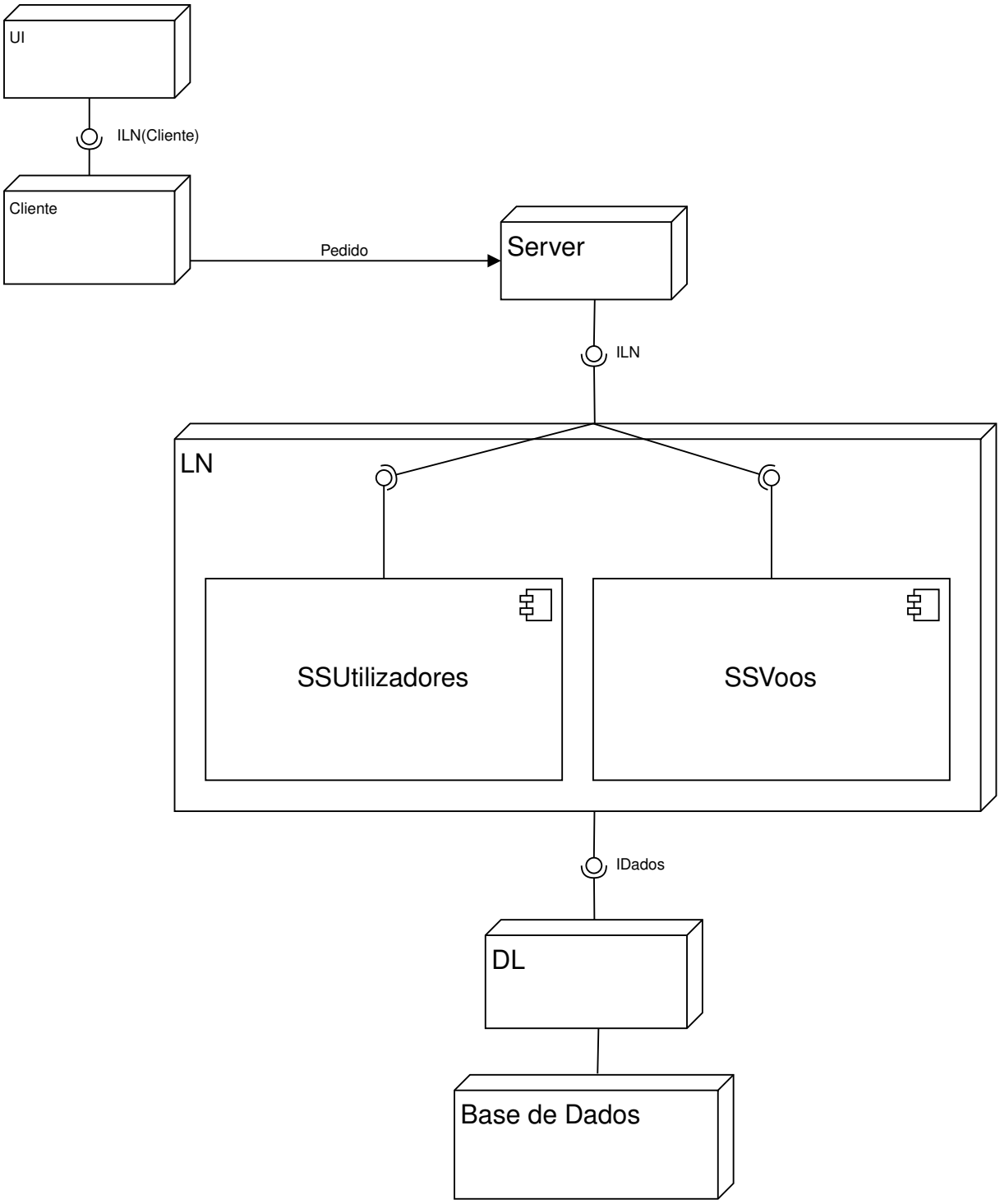

Figura 2.1: Diagrama de Componentes

Capítulo 3

Implementação

Neste capítulo do relatório, iremos entrar em detalhe sobre as várias estruturas e classes que definem o cliente e o servidor como entidades no sistema, bem como comunicam umas com as outras e qual o seu propósito no sistema.

3.1 Cliente

O *Cliente* é utilizada pelos utilizadores, (utilizadores normais e administradores), para efetuar pedidos ao sistema. Relativo ao cliente estão disponíveis as seguintes classes:

- **Client** - Esta classe executável é uma classe muito simples, que corre o menu, permitindo ao utilizador uma *interface* com a qual pode aceder às funcionalidades do sistema.
- **LN(Cliente)** - A camada lógica do cliente. Esta classe é composta por métodos que permitem ao utilizador fazer pedidos ao servidor e de receber as suas respostas, de forma a permitir que o utilizador aproveite todas as funcionalidades do programa.

3.2 Servidor

O *Servidor* permite que os clientes utilizem o sistema através de pedidos. Relativamente a esta entidade, estão presentes os seguintes aspetos:

- **Server** - Esta classe executável é uma classe simples, que aceita conexões de qualquer potencial utilizador e que lhe associa um Worker, servindo como porta de acesso ao sistema.
- **Worker** - Uma classe criada pelo servidor cujo propósito é receber *Frames* em forma de *String* que indicam ao sistema qual a funcionalidade a ser executada. O sistema interpreta os frames e executa o respetivo método.

3.2.1 Lógica de negócios

Esta camada contém todos os métodos que podem ser executados por um utilizador. Esta é subdividida em duas componentes, sendo estas:

SSUtilizadores

Aqui estão os utilizadores do sistema, (UtilizadorNormal e Administrador). A cada um destes tem associado a si um email único, um username e uma password. Para além disto, é lhes associado um nível de autoridade dependendo do tipo de utilizador que são, que indica o tipo de métodos que pode executar. A título de exemplo, um utilizador normal não pode criar novos voos, nem abrir/fechar o dia.

SSVoos

Esta componente abrange, não só os voos, mas também as reservas.
As classes pertencentes a este subsistema são as seguintes:

- **Voo** - Guarda toda a informação relevante a um voo, nomeadamente a partida, o destino, a capacidade de passageiros e a duração do voo.
- **Reserva** - Representa uma reserva feita por um utilizador. Aqui são guardados o email do utilizador, a partida e o destino do voo, assim como as datas da reserva e do voo.

3.2.2 Camada de dados

Aqui estão implementados os *Data Access Objects* que fazem o acesso à base de dados. Existe uma destas classes para cada uma das classes Utilizador, Voo e Reserva, que permitem adicionar, remover ou atualizar cada um destes. Para além disto, é através destas que se fazem pesquisas à base de dados.

3.3 Base de dados

É na base de dados que é guardada toda a informação do sistema. Esta foi criada com MySQL com ajuda do *MySQL Workbench* e contém as seguintes tabelas:

3.3.1 Utilizador

Guarda a informação comum a todos os tipos de utilizador.
Para além da tabela Utilizador, também existem as tabelas UtilizadorNormal e Administrador que indicam a classe de um dado utilizador. Apesar destas tabelas não terem um propósito no sistema no seu estado atual, decidimos ainda assim inclui-las para facilitar a introdução de nova informação específica a cada um dos tipos de utilizador.

Utilizador

- **idUtilizador INT** - ID único do utilizador que é atribuído ao inserir na base de dados.
- **Email VARCHAR(45)** - Email único do utilizador.
- **Nome VARCHAR(20)** - Username do utilizador.
- **Password VARCHAR(30)** - Palavra-passe utilizada no login.
- **Tipo INT** - Identifica o tipo de utilizador. (UtilizadorNormal ou Administrador)

UtilizadorNormal

- **idUtilizadorNormal INT** - ID do utilizador normal na tabela Utilizador

Administrador

- **idAdministrador INT** - ID do administrador na tabela Utilizador

3.3.2 Voo

Guarda a informação relativa a um voo.

- **idVoo INT** - ID único do voo que é atribuído na inserção na base de dados.
- **Parida VARCHAR(100)** - Cidade de partida do voo.
- **Destino VARCHAR(100)** - Cidade de destino do voo.
- **Capacidade INT** - Capacidade de passageiros do voo.
- **Duracao INT** - Duração, em minutos, do voo.

3.3.3 Reserva

Guarda a informação relativa a uma reserva.

- **idReserva INT** - ID único da reserva atribuído ao inserir na base de dados.
- **idUtilizador INT** - ID do utilizador que fez a reserva.
- **idVoo INT** - ID do voo em que a reserva foi feita.
- **Data_Reserva DATETIME** - Data em que a reserva foi feita.
- **Data_Voo DATETIME** - Data em que o voo será feito.

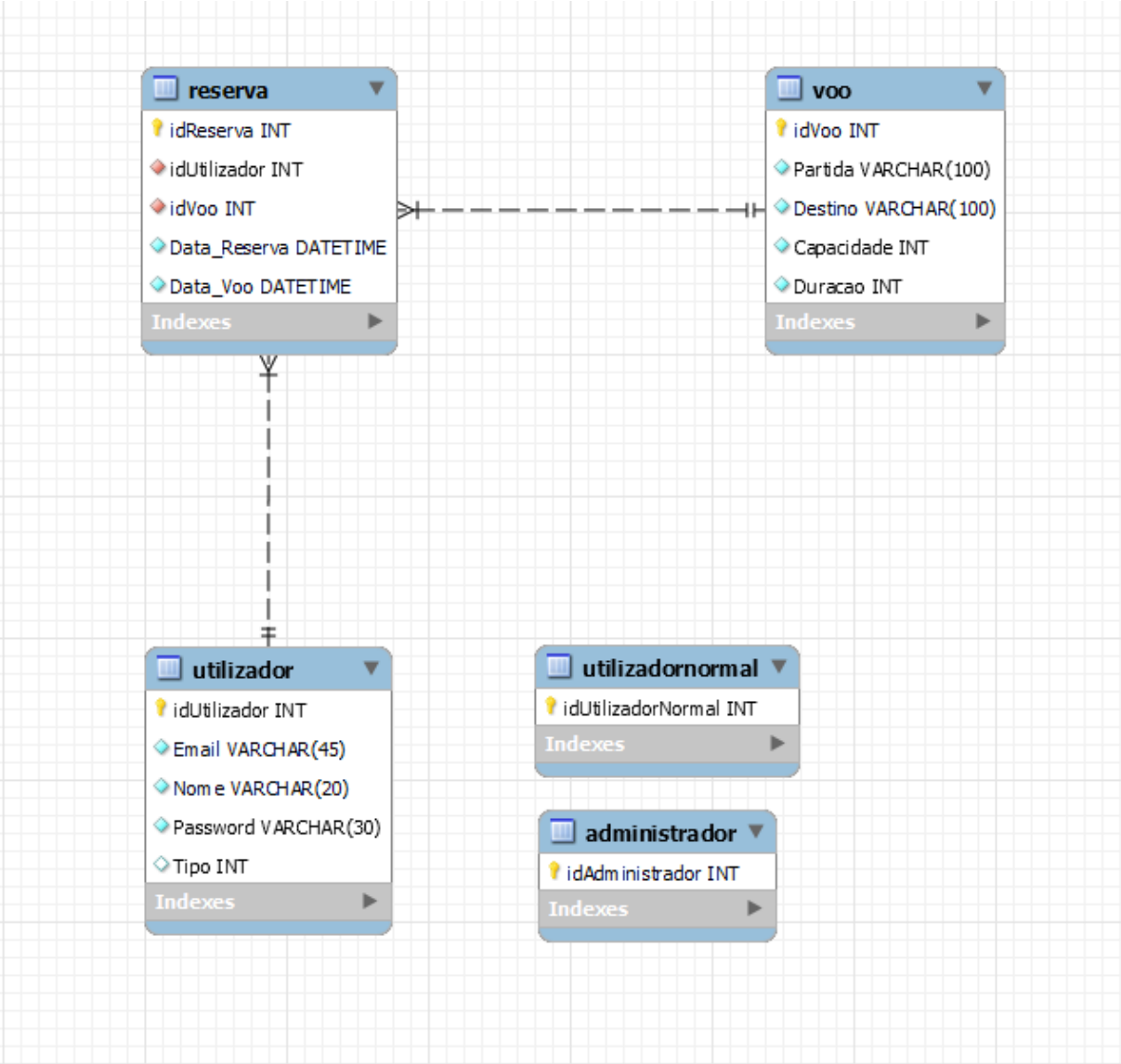

Figura 3.1: Diagrama da base de dados

3.4 Interface

A *interface* representa o método com o qual o utilizador comunica com o sistema e pode ser representada no nosso código exclusivamente pelas classe *MenuPrincipal* e pela classe *Menu*.

Em *MenuPrincipal*, é possível observar cada um dos aspetos da comunicação entre o utilizador e o sistema, divididos através de vários métodos que correspondem a diferentes maneiras do sistema lidar com vários pedidos. Esta classe é capaz de fornecer um meio do utilizador comunicar com o sistema, e isto é possível através de *Menu*, que contém métodos que criam menus e definem opções para mostrar ao utilizador como consegue utilizar o sistema. Este faz isso ao definir as opções que serão utilizadas perante o contexto do programa e associando números a cada uma destas opções, pedindo ao utilizador que prima o número correspondente à opção desejada.

Por exemplo, quando um utilizador efetua o *login* no programa, o sistema, após obter as credenciais fornecidas pelo o utilizador, irá percorrer a base de dados à procura de um

utilizador com as essas mesmas credenciais obtidas. Se não o encontrar, então o programa informa o utilizador que houve um erro, quer seja pelas credenciais serem inválidas ou por o dia estar encerrado e não ser possível efetuar mais operações. Mas caso encontre o utilizador, o programa irá obter o nome do mesmo e um valor *int* designado de *autoridade*. A razão pela qual ele faz isto é para saber qual o menu que o utilizador irá aceder, o menu de um utilizador normal ou o menu de um administrador do sistema. Ele faz isto através da função **redirecionarMenu** que, através do valor de autoridade, determina qual a função que é chamada e, subsequentemente, transmite ao utilizador as suas opções dependendo da autoridade, após lhe dar as boas vindas.

Capítulo 4

Funcionalidades básicas do Sistema

A classe *SSVooFacade* trata de concretizar as funcionalidades básicas pedidas no enunciado, como por exemplo, permitir ao cliente realizar uma reserva. Assim sendo, esta classe trata de pedir informações sobre os clientes, reservas e voos e, posteriormente, implementar os métodos pedidos. Seguidamente, a *Interface* demonstra o funcionamento do sistema.

Aproveitamos esta secção para mostrar o funcionamento do programa, porém iremos apenas apresentar algumas das opções possíveis. As imagens seguintes irão conter o menu principal com a opção de gerar um novo registo selecionada e preenchida; o menu do cliente com a opção de apresentação de todos os voos da base de dados; o menu do admin com a opção de adicionar informação sobre um novo voo.

```
 *** Menu ***
1 - Entrar no sistema, usando credenciais
2 - Registrar novo utilizador
0 - Sair
Opção: 2
Email :
marco@email.com
Username :
Marco
Password :
Password123
Autoridade :
0

Utilizador registado com sucesso
```

Figura 4.1: Menu principal - Registar novo utilizador

```
 *** Menu ***
1 - Fazer uma reserva de voo
2 - Cancelar uma das reservas de voo
3 - Reservar um voo através de um percurso específico
4 - Obter uma lista de todos os voos
5 - Obter uma lista de percursos possíveis
0 - Sair
Opção: 4

Partida: Porto; Destino: Paris; Capacidade: 100; Duração: 120.
Partida: Paris; Destino: Munique; Capacidade: 150; Duração: 180.
Partida: Munique; Destino: Tóquio; Capacidade: 100; Duração: 210.
Partida: Lisboa; Destino: Tóquio; Capacidade: 70; Duração: 240.
Partida: Porto; Destino: Lisboa; Capacidade: 50; Duração: 60.
```

Figura 4.2: Menu Cliente - Obter uma lista de todos os voos

```
 *** Menu ***
1 - Adicionar informação sobre um novo voo
2 - Encerrar o dia, não permitindo novas reservas
3 - Reabre o dia, voltando a permitir novas reservas
0 - Sair
Opção: 1
Origem do voo :
Braga

Destino do voo :
Londres

Quantos passageiros podem ir no voo? :
100

Quanto tempo dura o voo? :
90

Informação sobre voo adicionada com sucesso
```

Figura 4.3: Menu Admin - Adicionar informação sobre um novo voo

Capítulo 5

Funcionalidades Adicionais

De maneira a explorarmos o projeto a um nível mais profundo, decidimos implementar uma das funcionalidades extra pedidas no enunciado providenciado. Assim, estudamos o método que permite a obtenção de uma lista com todos os percursos possíveis para viajar entre uma origem e um destino, limitados a duas escalas (três voos).

Para implementar este método, achamos por bem começar por definir uma árvore que descreva todos os percursos (até 3 voos) que se iniciem com a partida definida pelo utilizador. Seguidamente, preenchemos uma lista com as listas dos percursos que descrevem o destino pedido também pelo utilizador.

Para além deste método, também existia no enunciado uma outra funcionalidade adicional que pedia "Possibilidade de um cliente continuar a executar outras operações enquanto se espera pela conclusão duma reserva, incluindo novas operações de reserva". Porém, o grupo não conseguiu implementar este método, uma das razões sendo que não conseguimos interpretar com certezas o que era pedido.

Capítulo 6

Conclusão

Este projeto tratou dum sistema de reserva de voos, utilizando *sockets* e *threads*. Após a conclusão do trabalho, o grupo pode concluir que atingiu o objetivo com sucesso. As funções principais e obrigatórias foram todas implementadas com qualidade e ainda tratamos de parte das funções adicionais.

Para trabalho futuro, ainda poderíamos, por exemplo, implementar uma interface gráfica de maneira a termos uma *Interface* mais apelativa ao utilizador.

Este projeto permitiu-nos melhorar os nosso conhecimentos do funcionamento e implementação de *threads*.