
Universidade de Évora

Disciplina de Sistemas Operativos II

# Reservas de Espaços

*Autores:*
Ricardo Fusco, 29263
Marcus Santos, 29764

*Professor:*
José Saias

April 18, 2014

# Índice

**1 Introdução** **3**

**2 Desenvolvimento** **4**
2.1 Base de dados . . . . . 5
2.2 Funcionalidades . . . . . 5
2.3 Build da aplicação . . . . . 8
2.4 JavaDoc . . . . . 9

**3 Conclusão** **10**

# Lista de Figuras


1 Árvore de directorias da aplicação . . . . . 4
2 Modelo Entidade Relacional . . . . . 5
3 Menu Reservas Espaços . . . . . 5
4 Listagem espaços . . . . . 6
5 Espaço Livre . . . . . 6
6 Listagem Reservas de um dia . . . . . 7
7 Efectuar Reserva . . . . . 7
8 Listar Reservas feitas por uma pessoa . . . . . 8
9 Javadoc - Index . . . . . 9

# 1 Introdução

No âmbito da unidade curricular de Sistemas Operativos II pretende-se, usando as metodologias de comunicação e soluções de Middleware abordadas nas aulas, desenvolver um sistema de reservas de espaços duma instituição de ensino baseado no conceito cliente-servidor. Esse sistema permite ao cliente a execução dum conjunto de operações sobre uma base de dados que contém os dados acerca do conjunto de espaços disponiveis e reservas que já foram efectuadas. O conjunto de operações consiste na listagem de espaços disponíveis, reservar um espaço, saber se um dado espaço está livre num determinado dia e hora, etc. Para o nosso trabalho optámos por utilizar a Invocação Remota de Métodos (javaRMI) por ser extremamente simples de utilizar e por facilitar ao máximo a programação da aplicação permitindo que nos concentremos nos aspectos de mais alto nível, mais virados para a aplicação em sí e não tanto nos aspectos de mais baixo nível(Hardware,etc) e da comunicação.

# 2 Desenvolvimento

O sistema de reservas está organizado em sete classes/objectos que em conjunto servem os mais variados propósitos, desde estabelecer a ligação à base de dados, assinatura e implementação do métodos remotos até ao menu do cliente onde é feita toda a interacção entre o cliente e o servidor, que graças à camada do midleware, é feita de maneira transparente, ou seja, o cliente não necessida saber onde estão localizados os dados.
As classes Espaco e Reserva são as classes que foram criadas para representar objectos do tipo Espaco e Reserva, tendo cada um destes objectos um conjunto de características único. Tendo em conta que estamos a utilizar o javaRMI precisamos do interface remoto ReservasEspacos e a classe do objecto remoto ReservasEspacoImpl que contém a parte funcional e a implementação de todos os métodos de invocação remota declarados no interface remoto. Temos ainda a classe BD, esta é a classe onde estabelecmos a ligação com a base de dados e executamos as querys de acordo com a função que queremos executar. No ReservasEspacosImpl é criado um objecto desta classe BD e os métodos de invocação remota basicamente vão chamar os métodos desse objecto de BD. As restantes classes são o servidor e o cliente.

A figura 1 ilustra a organização dos ficheiros no projecto, faltando apenas referir o Makefile e o ficheiro de criação da base de dados que estão localizados na raiz da directoria.


Figure 1: Árvore de directorias da aplicação

## 2.1 Base de dados

A base de dados é composta pelas tabelas Espaços e Reservas que, como o próprio nome indica, estas guardam informação das salas/espaços a serem requisitadas e das respectivas reservas associadas a esses espaços. Procurou-se desenhar uma base de dados simples mas que ao mesmo tempo respondesse a todas às necessidades da aplicação. A figura 2 ilustra o modelo de entidade relacional da base de dados onde se pode visualizar todos os campos e relações().

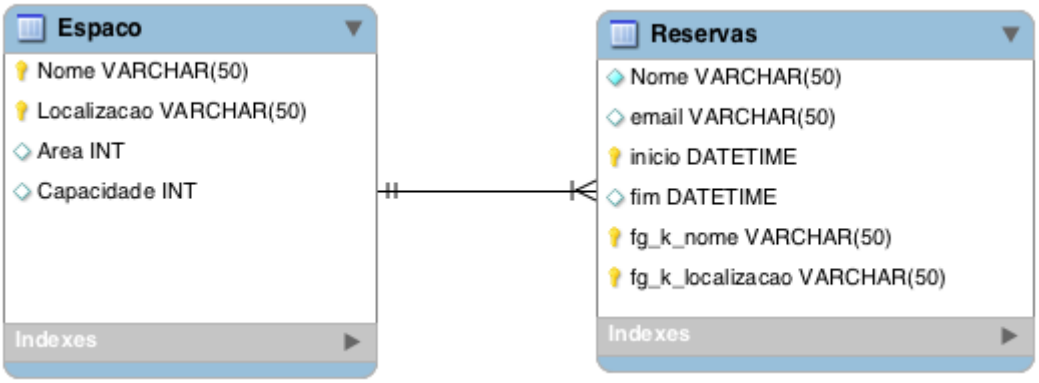


Figure 2: Modelo Entidade Relacional

## 2.2 Funcionalidades

Tal como requerido as funcionalidades do sistema de reservas são as que se poderão observar na figura seguinte:

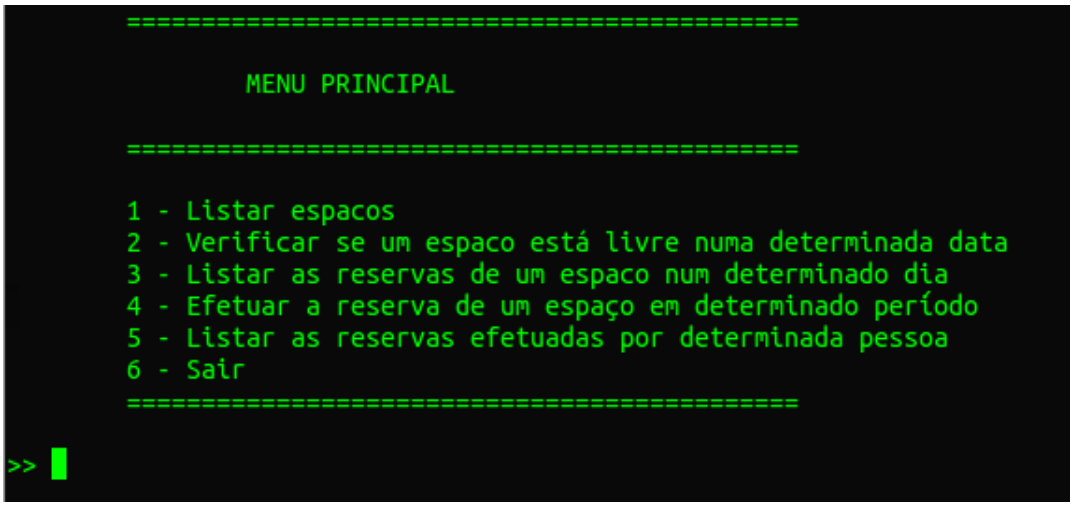


Figure 3: Menu Reservas Espaços

- Listagem de todos os espaços

  - Quando é seleccionada esta funcionalidade basicamente irá ser chamado o método remoto que por sua vez irá ligar à base de dados, verificar todos os espaços disponíveis e colocá-los num vector, de seguida é então percorrido o

vector de espaços para "imprimir" no terminal as informações acerca de cada espaço que se encontra disponível (fig. 4).

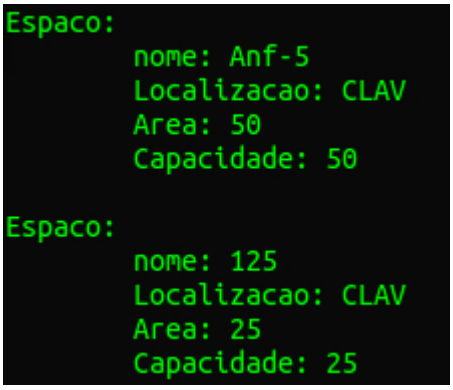


Figure 4: Listagem espaços

- Saber se um dado espaço está livre numa determinada data e hora

  - Quando é chamado o método remoto que lida com esta funcionalidade, irá primeiro ser executada uma query que verifica a contagem de espaços ocupados durante a data e hora especificadas e caso o resultado da contagem seja 0 significa que o espaço se encontra livre nessa data e hora e devolve true, caso contrário devolve false (fig. 5).

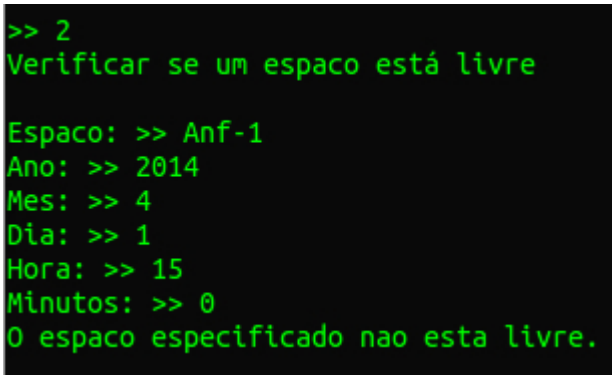


Figure 5: Espaço Livre

- Listar as reservas de um espaço num determinado dia

  - Dada uma data(ano, mês e dia) é executada uma query que selecciona todas as reservas para a data fornecida da base de dados e coloca-as num vector de reservas para que depois se percorra o array e imprima as informações de cada reserva que foi feita na data especificada (fig. 6).

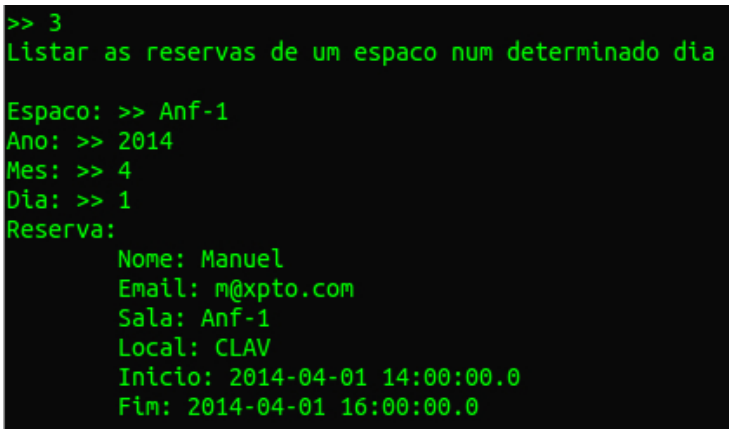


Figure 6: Listagem Reservas de um dia

- Efectuar uma reserva

  - Para se efectuar esta operação há que primeiro assegurar que a data e hora de inicio da reserva é anterior à data e hora do fim da reserva. Quando isto se verifica é necessário primeiro executar uma query para contar quantas reservas há entre o periodo de reserva que se pretende, caso a contagem seja 0 é possível inserir a reserva com sucesso caso haja 1 ou mais reservas nesse horarios não é permitido fazer a reserva no horário pretendido. Este método devolve uma string que será a mensagem que será mostrada no terminal consoante o caso que se verifique (fig. 7).

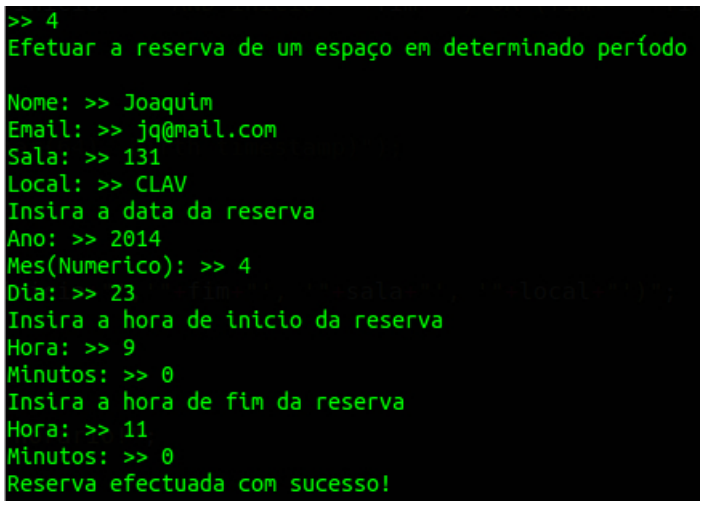


Figure 7: Efectuar Reserva

- Lista as reservas efectuadas por uma pessoa - Dado um nome irá executar uma query para seleccionar todas as reservas feitas por essa pessoa da base de dados e é devolvido um vector de reservas que irá de seguida ser percorrido para que sejam mostradas no terminal todas as reservas e as informações acerca de cada reserva (fig. 8).

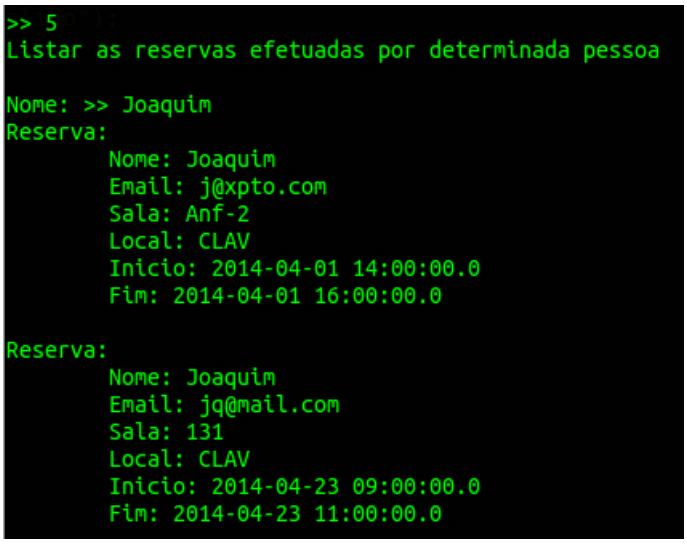


Figure 8: Listar Reservas feitas por uma pessoa

## 2.3 Build da aplicação

Para correr o serviço de reservas é necessário compilar todas as classes da pasta src, activar o serviço de nomes, correr o servidor e por último o cliente. Para mais informações acerca do make rs ver o ficheiro README.txt que se encontra na directoria do trabalho. O trabalho não está feito com um sistema de controlo de concorrência pelo que a execução de diversos clientes em simultâneo pode resultar em erros inesperados ou inconsistências na base de dados. Para correr o programa sem problemas é necessário correr os seguintes comandos pela ordem apresentada:
- make all
- make stubs
- make reg
- make rs A1=HOST A2=DATABASE A3=USER A4=PWD
- make rc

## 2.4 JavaDoc

Toda a documentação da aplicação encontra-se localizada na mesma directoria do projecto na pasta /doc. Lá poderá encontrar uma breve descrição de cada classe e as funcionalidades de cada método, bem como os seus argumentos e retornos. Para abrir a página da documentação basta abrir o ficheiro "index.html".

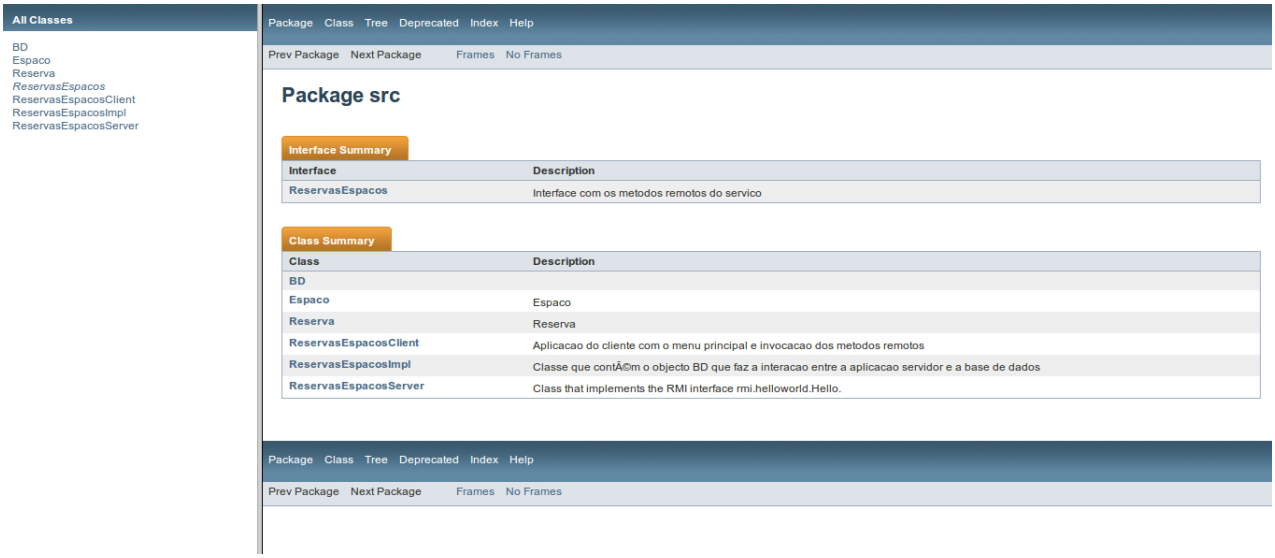


Figure 9: Javadoc - Index

# 3 Conclusão

Em conclusão achamos que conseguimos cumprir com sucesso os ojectivos base do trabalho. Inicialmente tivemos alguns problemas na ligação do servidor com a base de dados, após superado este problema surgiram ainda alguns problemas no que diz respeito à execução de algumas querys levando-nos a fazer alguns testes extensivos sobre a base de dados local e da base de dados da conta de aluno em alunos.di.uevora.pt.

Um dos problemas relativamente aos métodos que executam as mudanças na base de dados foi no inserir pois inicialmente passou-nos um pouco ao lado o facto de ser impossível realizar uma reserva num horário onde já existe uma reserva para a mesma sala pelo que depois tivemos que resolver este problema executando uma query, antes da inserção, que faz a contagem do número de reservas no horário que se pretende e consoante o valor prosseguir ou não com a inserção da reserva. Não conseguimos separar o cliente do servidor no sentido de ter o servidor e o registo de nomes a correr em background na conta de alunos.di.uevora.pt e o cliente a correr num pc, o cliente ligava mas não executava os métodos do menu, mas colocando todos os ficheiros na conta e correndo todos em background (localhost) pela ordem correcta o programa corria sem problemas.

Tentámos ainda, implementar um sistema de controlo de concorrência para permitir a execução de vários clientes sem problemas mas não o conseguimos fazer.

Tendo em conta o que foi feito podemos assumir que o balanço foi positivo, apesar de algumas dificuldades com que nos deparámos, do facto de não termos conseguido implementar tudo aquilo que tínhamos em mente e de alguns eventuais bugs que possam ter passado despercebidos.