Relatório do Trabalho Prático 1 – Programação Concorrente (Erlang)

Introdução ao Erlang

Com a evolução do hardware cada vez mais para vertentes paralelas, foi expectável que as linguagens de programação acompanhassem essa tendência. Assim sendo surgiu por volta dos anos 80 a linguagem Erlang, desenvolvida pelo Ericsson's Computer Science Laboratory, com o objetivo de criar uma linguagem que cumprisse as necessidades para as suas novas antenas. Após experimentos com diversas linguagens já existentes nenhuma cumpria na totalidade o que era pretendido. Erlang foi influenciado por linguagens funcionais tais como ML e Miranda, linguagens concorrentes como ADA, Modula e Chill para alem de ProLog. Facto importante foi quando a Ericsson decidiu tornar Erlang open source, em dezembro de 1998, o que levou a que aos dias de hoje seja uma linguaguem com muito sucesso.

Características principais

Uma das principais características de Erlang e que se difere de muitas outras linguagens é o facto de ser uma lingaugem concorrente e não sequencial, ou seja é especializada para programaras concorrentes e como tal tem construtores especiais para expressar concorrência na linguagem em si. A concorrencia em Erlang é garantida pela máquina virtual do Erlang e não pelo sistema operativo o que faz com que ela funcione de forma igual independentemente do sistema operativo.

Arquitetura do trabalho

O trabalho foi divido em 2 ficheiros (server.erl e client.erl) no server.erl está implementada a parte referente ao servidor usando o modulo predefinido no Erlang mnesia, funções de inicialização das tabelas (do_this_only_once(), start()) e uma função loop() para lidar com as mensagens enviadas pelas clientes e funções para tratamento dos pedidos feitos pelos clientes. No ficheiro client.erl foram implementadas as funções que simulam as mensagens envidas para o servidor.

Após serem compilados os respetivos ficheiros de código fonte, quando executado pela primeira vez de ser chamada a função do_this_only_once() e posteriomente a função start(), caso seja desejado restabelecer as tabelas para as originais pode ser feito chamando a função clear_tables().

Exemplos de execução do progrrama

O programa contem as seguintes tabelas inicias:

Tabela Books

```
6> client:list_table(<0.193.0>, books).
[{books,123,"A historia 2","Paulo"},
 {books,1,"A historia","Sergio"},
 {books,1234,"A historia 2","Paulo"},
 {books,12,"A historia","Sergio"},
 {books,12345,"A historia 3","Rui"}]
```

Tabela People

```
7> client:list_table(<0.193.0>, people).
[{people,123456789,"Marta Sousa","Amarante",987654321},
 {people,192837465,"Amadeu Oliveira","Lousada",111222333},
 {people,987654321,"Pedro Carvalho","Amarante",123123123}]
```

Tabela Register

```
8> client:list_table(<0.193.0>, register).
[{register,123,987654321},
 {register,1,987654321},
 {register,1234,123456789},
 {register,12345,123456789}]
```

Requisition (Server, PersonId, PersonName, Adress, MobileNumber, BookId) - Introdução de um registo na base de dados com sucesso e demonstração do estado da tabela após a inserçao;

```
9> client:requisition(<0.193.0>,987654321,"Pedro Carvalho", "Amarante", 123123123,12).
{atomic,ok}
10> client:list_table(<0.193.0>, register).
[{register,123,987654321},
 {register,1,987654321},
 {register,12,987654321},
 {register,1234,123456789},
 {register,12345,123456789}]
```

Requisition (Server, PersonId, PersonName, Adress, MobileNumber, BookId) - Introdução de um registo na base de dados sem sucesso e demonstraçao do estado da tabela após a inserção;

```
11> client:requisition(<0.193.0>,987654321,"Pedro Carvalho", "Amarante", 123123123,1234).
Error
ok
```

Return (Server, BookId, PersonId) - Remoção de um registo da base de dados e demonstração do estado da tabela após a remoção;

```
16> client:return(<0.193.0>,1234,123456789).
{atomic,ok}
17> client:list_table(<0.193.0>, register).
[{register,123,987654321},
 {register,1,987654321},
 {register,12345,123456789}]
```

Books (Server, PersonId) – Lista de livros requesitada por uma pessoa;

```
23> client:books(<0.193.0>,123456789).
[1234,12345]
24> client:books(<0.193.0>,192837465).
[]
```

Loans (Server,BookName) – Lista de pessoas que requesitaram um determinado livro;

```
25> client:loans(<0.193.0>,"A historia").
[987654321]
26> client:loans(<0.193.0>,"A historia 2").
[987654321,123456789]
```

Requested (Server, BookId) – Determina se um livro foi requesitado ou não;

```
28> client:requested(<0.193.0>,1).
true
29> client:requested(<0.193.0>,12).
false
```

Codes (Server, BookName) – Lista de códigos de livros com um determinado título;

```
8> client:codes(<0.194.0>,"A historia 2").
[123,1234]
ok
9> client:codes(<0.194.0>,"A historia 4").
[]
```

Requests_numbers (Server, PersonId) – Número de livros requesitados por determinada pessoa;

```
11> client:requests_numbers(<0.194.0>,987654321).
2
12> client:requests_numbers(<0.194.0>,192837465).
0
```