Servidor em Erlang

Base de Dados para Requisição de Livros usando Mnesia

FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

15 de março de 2021

Criado por: Rui Cardoso, 201805046

Servidor em Erlang

Base de Dados para Requisição de Livros usando Mnesia

Introdução

Criado nos anos 80, nos laboratórios da Ericsson, o Erlang tem como objetivo ser uma linguagem de programação para suporte de aplicações distribuídas e tolerância a falhas. Efetivamente, algumas aplicações conhecidas são totalmente desenvolvidas em Erlang, tal como o *WhatsApp*.

O objetivo deste projeto é implementar um servidor que corresponde a uma base de dados que guarda requisições de livros numa biblioteca. As funções que têm '*' à frente do nome, tem algum bug que estará relatado no fim do desenvolvimento.

Desenvolvimento

Arquitetura do programa

Para simplificar a base de dados, separei o projeto em 5 ficheiros e foram criadas três tabelas distintas:

Em relação aos ficheiros, temos os ficheiros: ***dataBase.erl*** que contém funções para a inicialização da base de dados, inserir alguns dados nas tabelas e para verificar o conteúdo das tabelas; ***server.erl***, que contém o servidor que vai receber e responder aos pedidos; ***funs.erl***, que contém o código das funções a serem avaliadas; ***client.erl***, que contém pedidos de cada um dos pedidos ao servidor; ***tables.hrl***, que contém apenas os records da base de dados. Em todos os exemplos colocados neste relatório, assume-se que P é o servidor.

Quanto às tabelas:

-Tabela '*person*', que guarda as informações de uma pessoa, tais como Número do Cartão de Cidadão ('cc'), que é a chave primária, o nome ('*name*'), a morada ('*address*') e o número de telemóvel ('*phone*');

-Tabela '*book*', que guarda as informações de um livro, tais como o seu código ('*code*'), que é a chave primária, título ('*title*') e os autores ('*authors*');

-Tabela '*request*', que guarda o código de um livro requisitado ('*book_code*'), que é a chave primária, e o Número do Cartão de Cidadão da pessoa que requisitou esse livro ('*person_cc*').

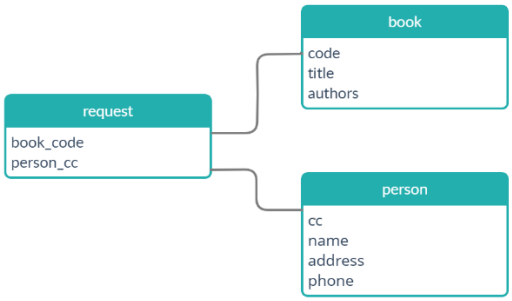

FIGURA 1 - REPRESENTAÇÃO DA BASE DE DADOS

Função 'books'

Esta função tem como objetivo retornar o título dos livros requisitados por uma pessoa, dado o seu número do CC. Para isso, selecionamos os elementos que tem o mesmo Número CC que o dado como

argumento na tabela '*request*'. Depois de termos esses elementos, é só ver os códigos dos livros que essa pessoa requisitou e ir à tabela '*book*' ver os títulos.

Exemplo de comando: *client:books(P, 0123456789).* Retorno: ["*Harry Potter and The Philosophers Stone*"].

Função 'loans'

Esta função tem como objetivo retornar os nomes das pessoas que requisitaram um certo livro, dado o título. Retorna-se uma lista porque como há várias cópias do mesmo livro, pode haver duas pessoas que requisitaram o livro com o mesmo título. Calcula-se então a lista de modo que as requisições digam apenas respeito àquele livro específico. Depois, através do 'person_cc' e do 'cc', verificam-se os nomes das pessoas e retorna-se a lista com esses nomes.

Exemplo de comando: *client:loans(P, "Harry Potter and The Philosophers Stone").* Retorno: ["*Harry Potter*", "*Rui*"].

Função 'request'

Esta função tem como objetivo determinar se um livro está ou não requisitado, dado o seu código. Esta função retorna um booleano. Primeiramente, calcula-se uma lista que contém os dados de requisição do livro com o código dado como argumento. De facto, se o livro não estiver requisitado, a lista vai estar vazia. De outro modo, vai ter um elemento. Depois de ter a lista, através do encaixe de padrões, verifica-se se está ou não vazia. Se estiver vazia, retorna-se falso, caso contrário, retorna-se verdadeiro.

Exemplo de comando: *client:request(P, 7).* Retorno: *true. Client:request(P, 4).* Retorno: *false*.

Função 'codes'

Esta função tem como objetivo determinar os códigos dos livros com o mesmo título, sendo dado o título como argumento. Assim, fazemos uma pesquisa na tabela *book* por cópias do mesmo livro, mas que tenham códigos diferentes.

Exemplo de comando: *client:codes(P, "Harry Potter And The Philosophers Stone").* Retorno: [0, 7].

Função 'req_num'

Esta função tem como objetivo retornar o título dos livros requisitados por uma pessoa, dado o seu número do cartão de cidadão. Primeiro, calcula-se a lista de requisições feita pela pessoa com o número de CC dado como argumento. Depois, vê-se o tamanho da lista.

Exemplo de comando: *client:req_num(P, 4567890123).* Retorno: 0.

client:req_num(P,0123456789). Retorno: 1.

client:req_num(P, 1234567890). Retorno: 2.

*Função 'add'**

Esta função tem como objetivo adicionar uma requisição à base de dados. Aqui, simplesmente escrevemos na tabela de requisições e na tabela pessoa. Uma vez que só é dado o código do livro requisitado, não faz sentido adicionar um livro à base de dados onde só iria contar com o código.

Exemplo de comando: *client:add(P, 5678901234, 7).* (Digitar *dataBase:check_request_table().* para verificar a adição deste dado da BD).

*Função 'remove'**

Esta função tem como objetivo remover uma requisição da base de dados. Efetivamente, não faz sentido retirar uma pessoa ou um livro da base de dados só porque o livro já foi devolvido. Assim, deixamos os dados da pessoa e do livro, mas removemos os da requisição. Então, basta fazer um '*delete*' e eliminamos essa requisição.

Exemplo de comando: *client:remove(P, 0123456789, 0).* (Digitar *dataBase:check_request_table().* para verificar a remoção deste dado da BD).

Bugs

Como na função '*add*' não se adiciona o livro à respetiva tabela (por falta de dados), pode haver erros de pesquisa.

Tanto a função '*add*' como a função '*remove*' não tornam os dados persistentes, isto é, ao desligar o servidor e a base de dados, os dados adicionados e removidos dão *reset*, e passamos a ter a BD original.

Conclusão

Concluindo, penso que se atingiram os objetivos pedidos, apesar de alguns erros menores. Efetivamente, penso que na função '*add*' podiam ser passadas as informações completas de um livro, ao invés de apenas o código, para que, caso o livro ainda não estivesse registado na BD, este registo fosse feito, não havendo assim erros de pesquisa.