UNIVERSIDADE DE ÉVORA

Sistemas Operativos 2

Relatório do primeiro trabalho prático

Autores:
João Marques, 39996
Tiago Martinho, 35735

Docente:
José Saias

Abril de 2020

Índice

1 Introdução

No âmbito da unidade curricular de Sistemas Operativos 2, pretende-se pôr em prática conhecimentos de sistemas distribuídos lecionados ao longo do semestre.
Com este propósito o grupo foi encarregue de desenvolver um sistema capaz de informar a população da existência de **produtos** e em que **lojas** os mesmos se encontram. Para além disso pretende-se que o utilizador possa **requisitar** um produto e que quando este for encontrado, o utilizador seja informado em que **loja** o produto foi encontrado.
Uma vez que se trata apenas da primeira fase do trabalho final alguns factores como é o caso da autenticação, segurança e não repúdio foram então deixados para a segunda fase do trabalho.
A linguagem de programação usada será o *Java* indo de encontro com o programa da disciplina.

2 Tecnologías Utilizadas

2.1 Base de Dados

Uma vez que se pretende a permanência de todos os dados e o fácil acesso aos mesmos utilizou-se então uma **base de dados** para armazenar os dados do sistema. Esta BD foi desenvolvida através do *postreSQL*.
Usando os conhecimentos lecionados nas aulas práticas, e através duma API (*postgresql.jar*), foi possível então manipular a BD em *Java* executando assim *querys* e introduzindo novos dados na mesma.

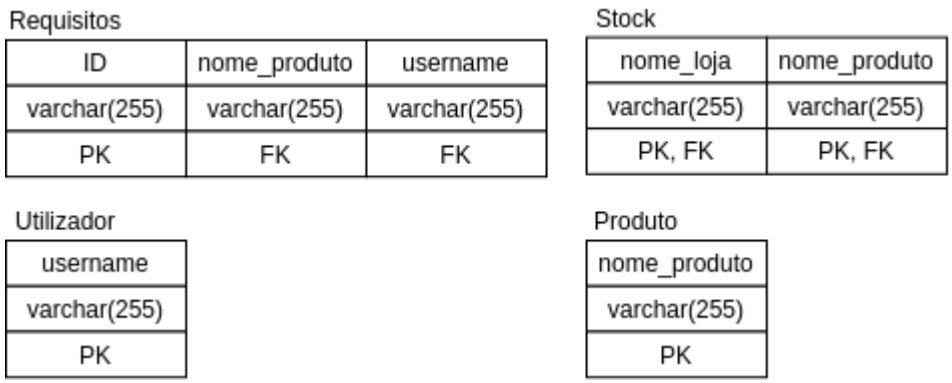

Figura 1: Diagrama da base de dados

2.2 Java RMI

Seria impossível fazer um trabalho de Sistemas Operativos 2 sem falar de ***middleware***. *Java RMI* foi a solução de *middleware* escolhida para este trabalho.
Usando esta **abstração das camadas inferiores** foi então possível criar um objeto remoto no lado do servidor, com o qual os clientes poderiam então interagir **invocando métodos** sobre o mesmo.
Através do *registry* os clientes podem então criar um ***proxy*** do objeto remoto assegurando assim aspectos de **segurança** importantíssimos, entre outros aspectos críticos para um SD.

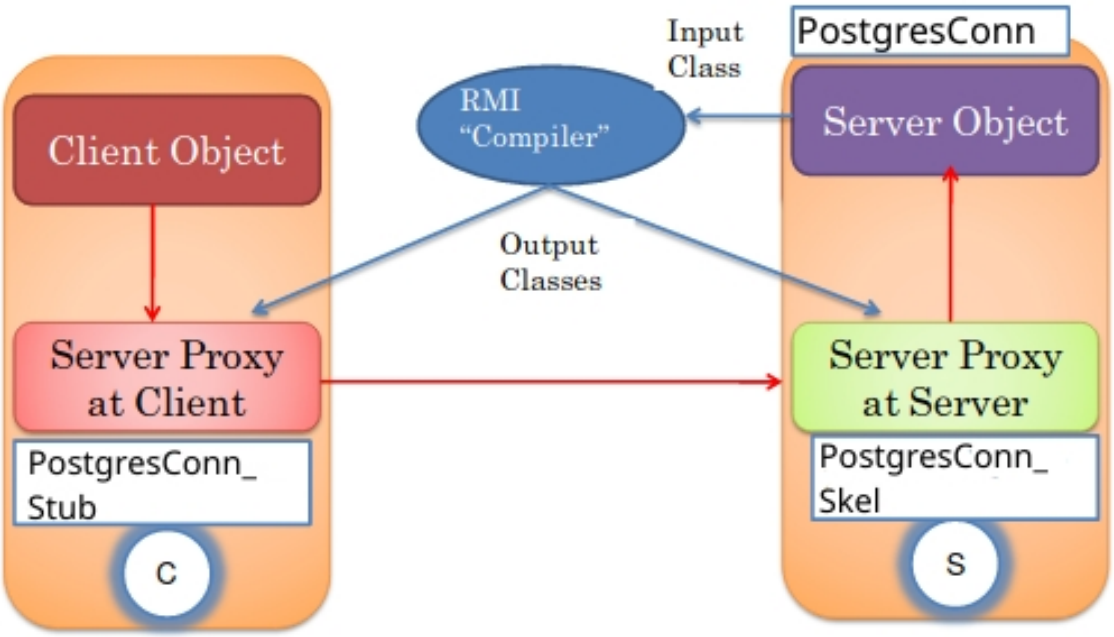

Figura 2: Java RMI

2.3 Servidor Concorrente

De modo a poder **notificar os utilizadores** que os produtos requisitados pelos mesmos já se encontra disponível numa loja foi preciso encontrar uma forma de disponibilizar a informação a múltiplos utilizadores **concorrentes**.
Para tal baseamo-nos na atividade 4 das aula prática que foi adaptada às necessidades encontradas.
Assim, o servidor tem um ciclo de atendimento que recebe novas ligações. Uma vez **validadas** estas ligações cria um novo ***thread*** através do qual comunica o nome da loja onde foi encontrado o produto e de seguida fecha a *socket*.

2.4 *Crash proof*

Caso ocorra um *crash* ou um *reboot* por parte do cliente pretende-se que o sistema **guarde** a última resposta recebida bem como um **registo** do último pedido feito ao servidor.
Guardando a última interação feita com o servidor sempre com o **mesmo nome**, assegura-se que o registo irá ser o pedido feito pelo utilizador antes do *crash*, ou se já recebeu a resposta, a mesma.
Para tal usou-se ***ObjFile*** método lecionado e disponibilizado na atividade 3.

3 Desenvolvimento

3.1 Servidor

O servidor trata-se apenas da criação de um ***binder*** ao qual se adiciona o objeto remoto *PostgresConn*. Os clientes liga-se então a este *registry*.
O objeto *PostgresConn* faz a comunicação toda com a BD. O servidor tem também num segundo *thread* um servidor concorrente a correr, onde é feita a comunicação de produtos requisitados que sejam encontrados.

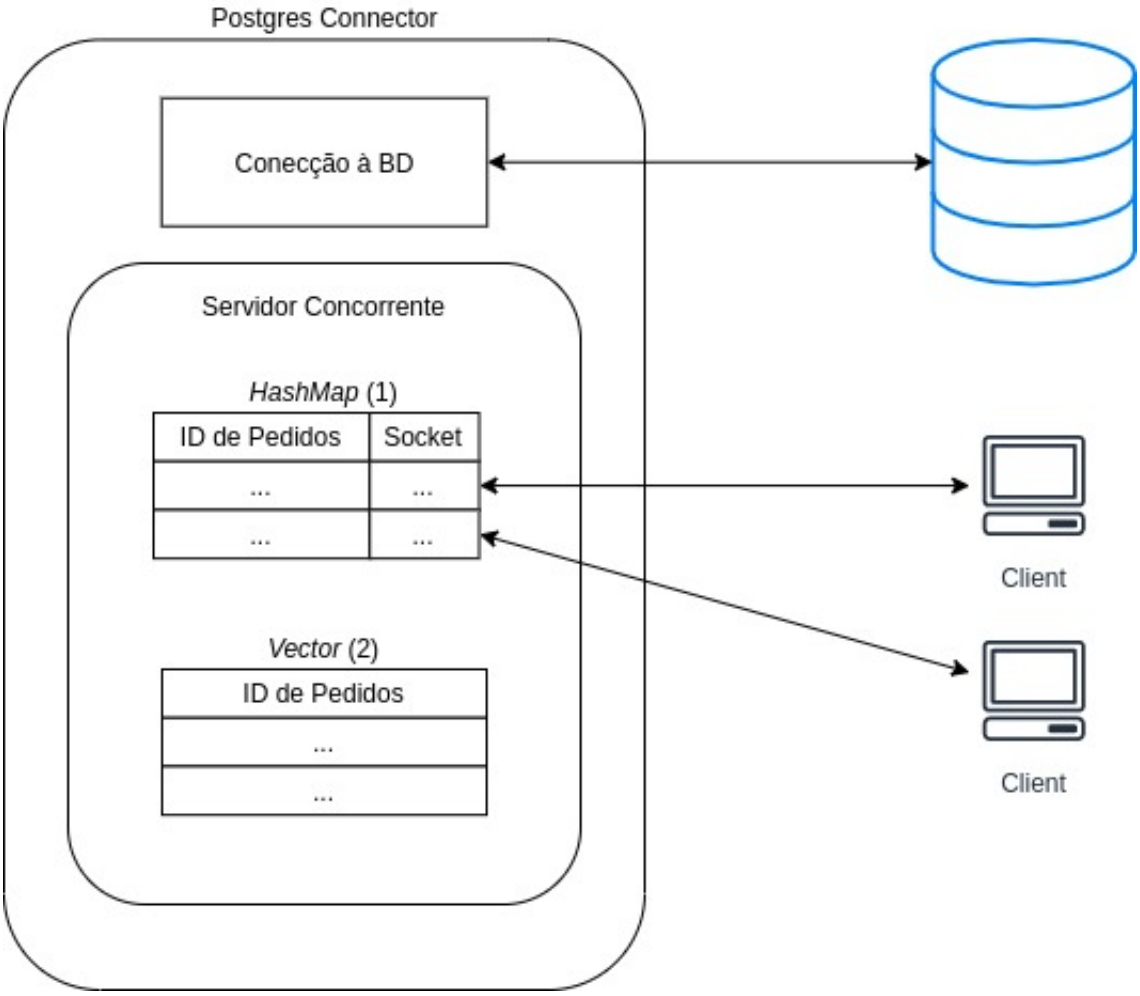

Figura 3: Diagrama do servidor

3.2 Requisitar produto

Ao requisitar um produto cria-se uma nova entrada na BD na tabela dos *Requisitos*.
De seguida cria-se um **ID único** para o pedido. Este pedido é adicionado no *Vector* 2 e envia-se o ID para o cliente.
O cliente guarda este ID **localmente** de modo a ter uma lista de todos os requisitos efectuados pelo mesmo.
O cliente cria então um novo ***thread*** por onde irá receber a informação caso o produto seja encontrado. Esse thread cria uma *socket* e liga-se ao servidor concorrente **enviando o ID** de modo a autenticar-se para o servidor **associar** o ID ao *socket* de comunicação (*Hashmap* 1).

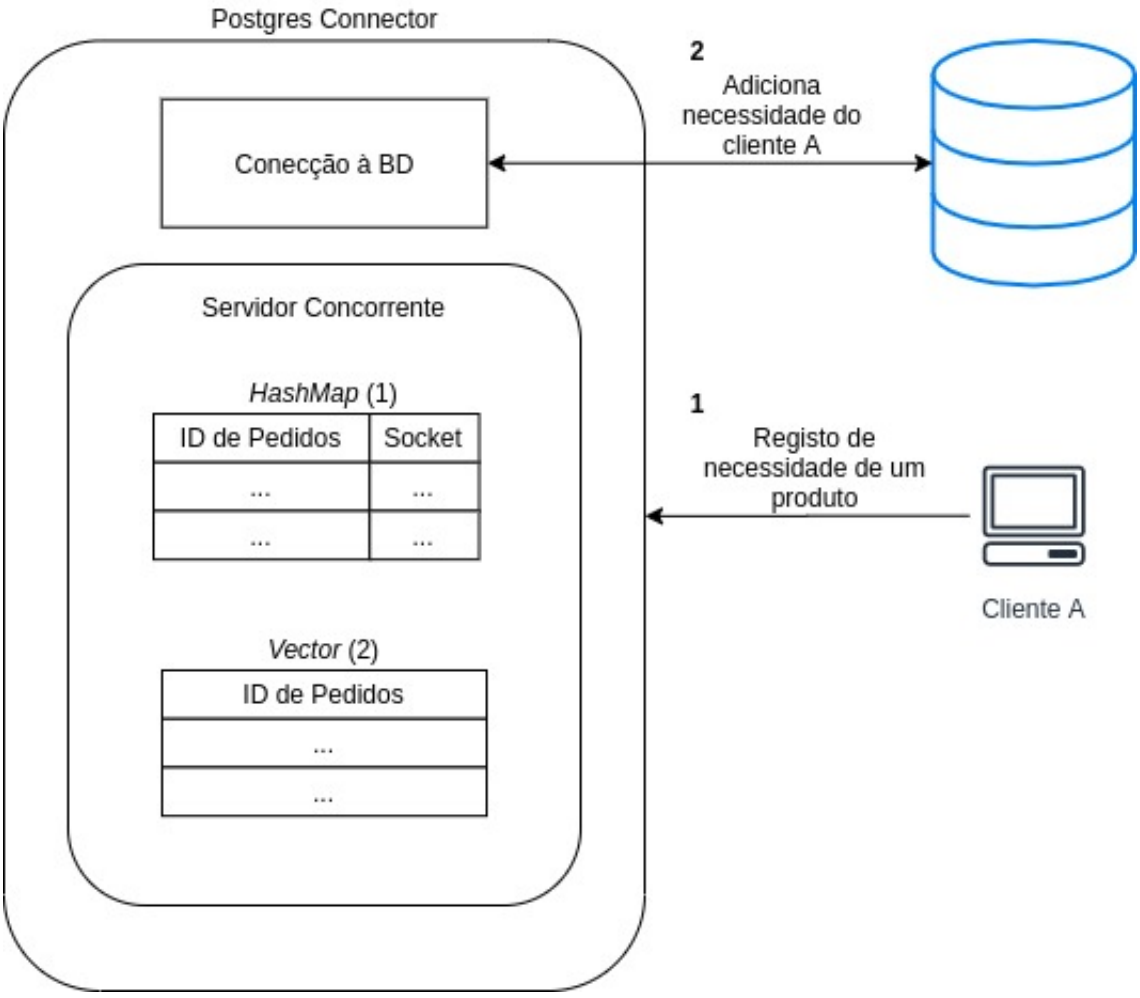

Figura 4: Requisitar produto 1

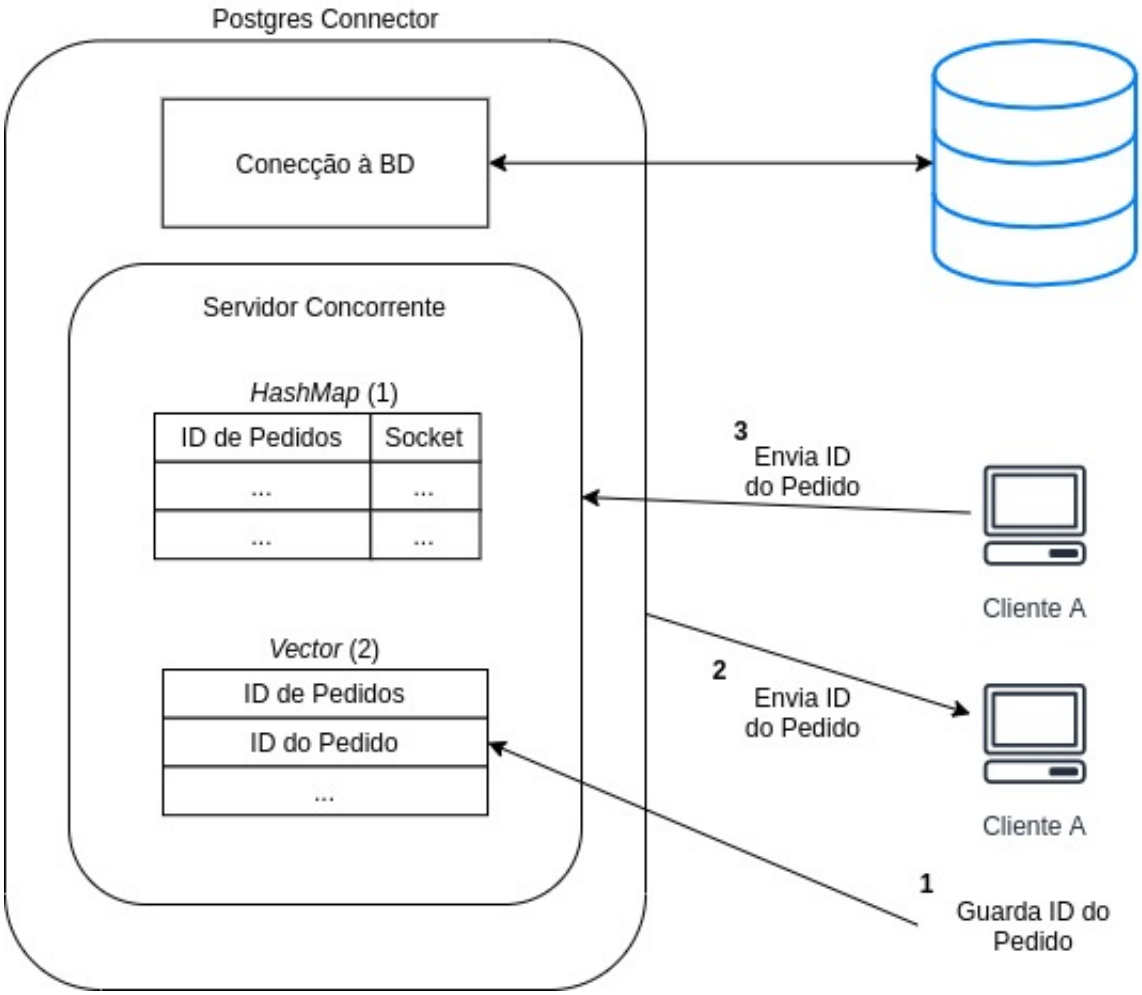

Figura 5: Requisitar produto 2

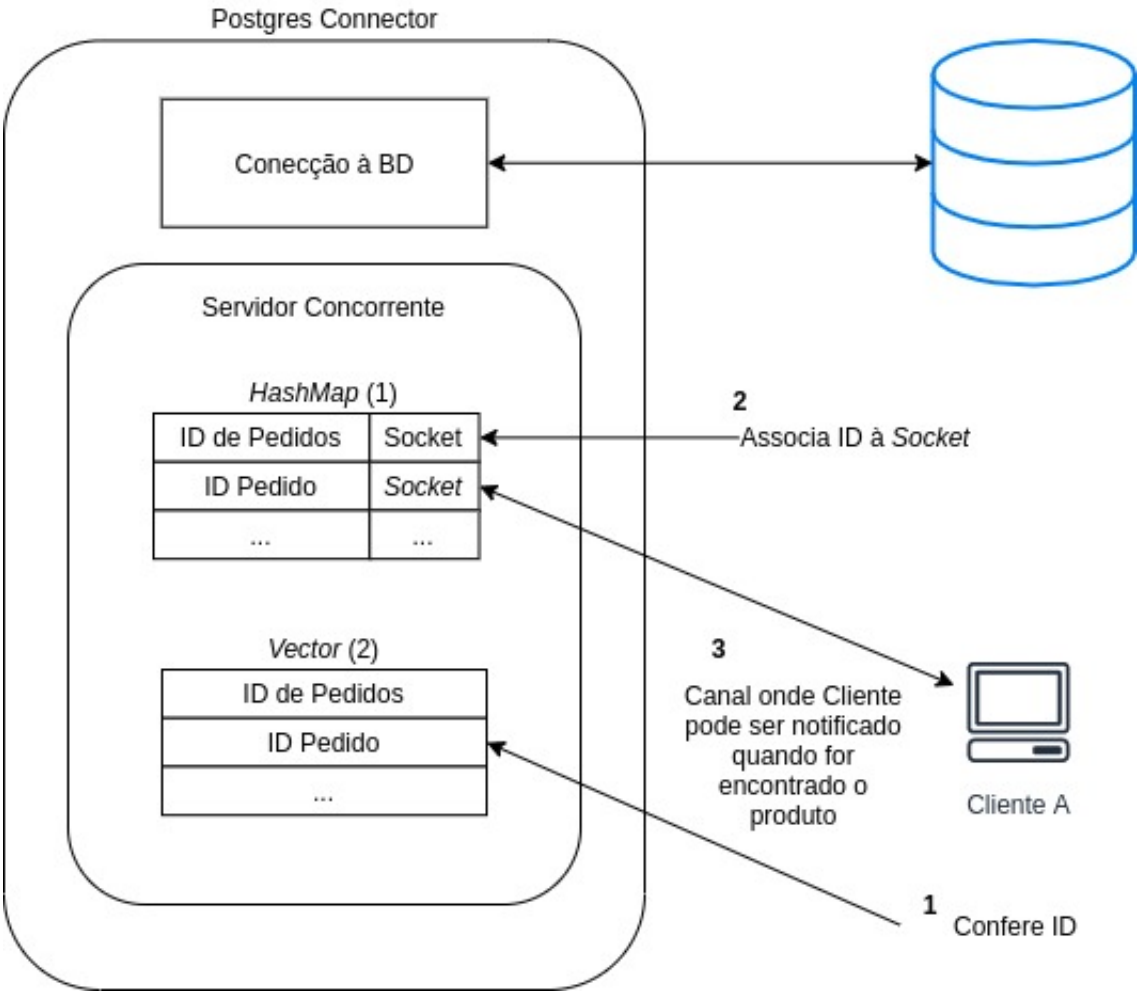

Figura 6: Requisitar produto 3

3.3 Reportar produto

Quando um produto é encontrado numa loja procura-se na tabela dos *Requisitados* se algum utilizador procura esse produto.
De seguida **envia-se** para os respectivos *sockets* desses utilizadores o nome da loja onde foi encontrado o produto. Uma vez respondidos aos clientes remove-se os requisitos da tabela *Requisitados* e adiciona-se os novos dados na tabela *Stock*.

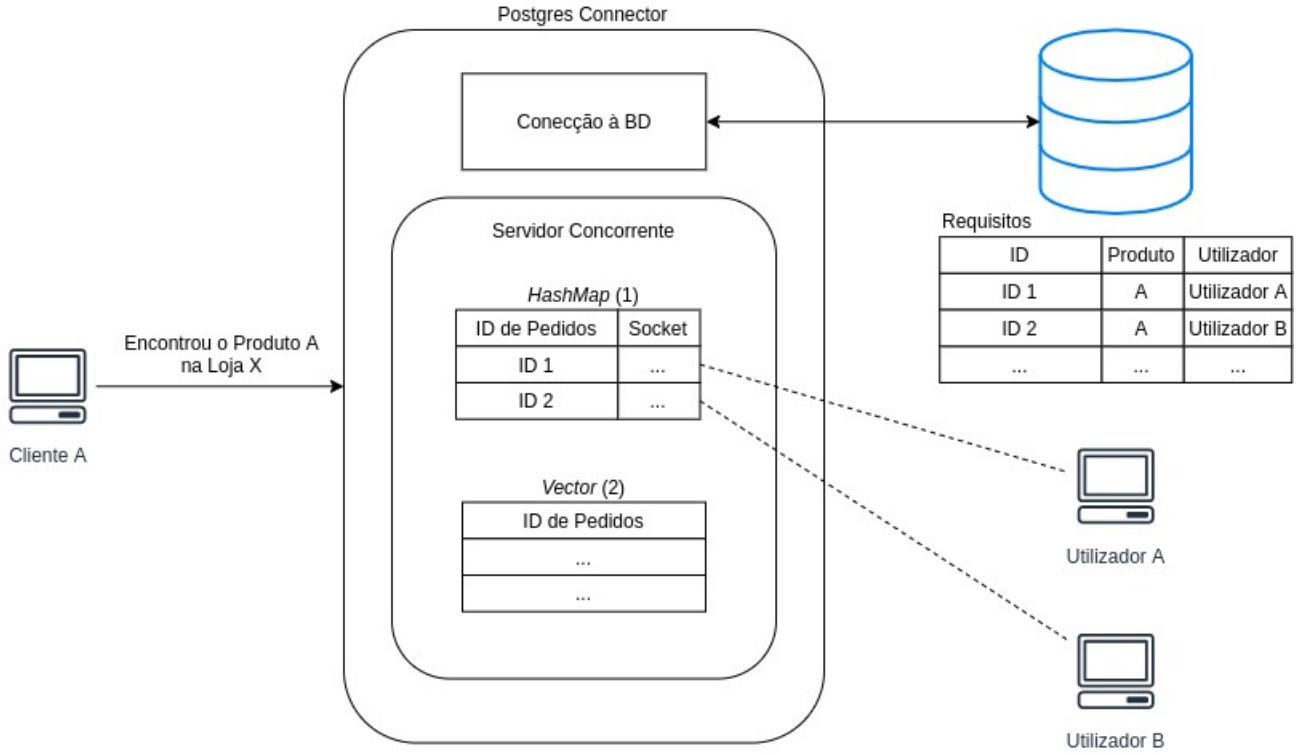

Figura 7: Reportar produto 1

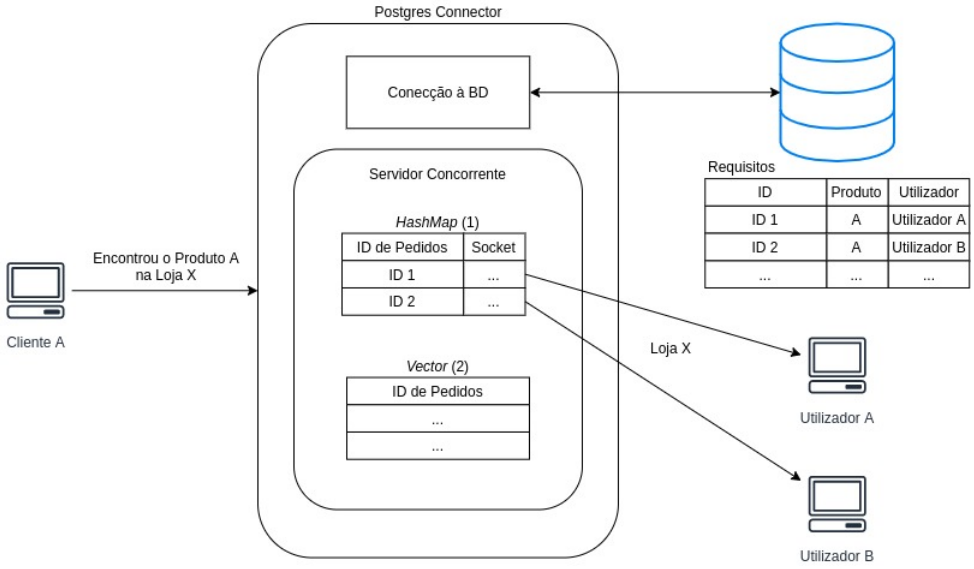

Figura 8: Reportar produto 2

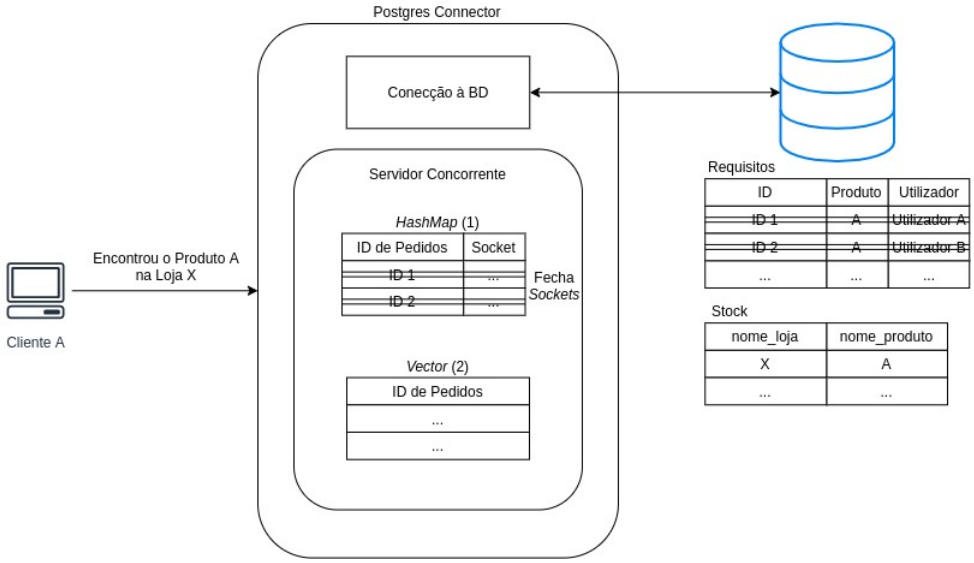

Figura 9: Reportar produto 3

3.4 Cliente

Aqui é onde o utilizador pode **interagir** com o sistema. O utilizador faz login ou regista um nome de utilizador e uma vez verificado pode fazer pedidos ao sistema. Estes pedidos incluem ver stocks de lojas, procurar produtos, procurar lojas, ver os produtos que outros utilizadores precisam, requisitar um produto ou reportar um produto.
Como foi referido no enunciado o cliente está sujeito a *crash* ou interrupções não esperadas. Por esta razão o cliente **guarda localmente** todos os pedidos e respostas enviados e recebidas. A cada utilizador está associado **um ficheiro** de registo e requisitos.
Ao requisitar um produto cria-se um novo ***thread*** que irá ser usado para fazer a comunicação com o servidor.

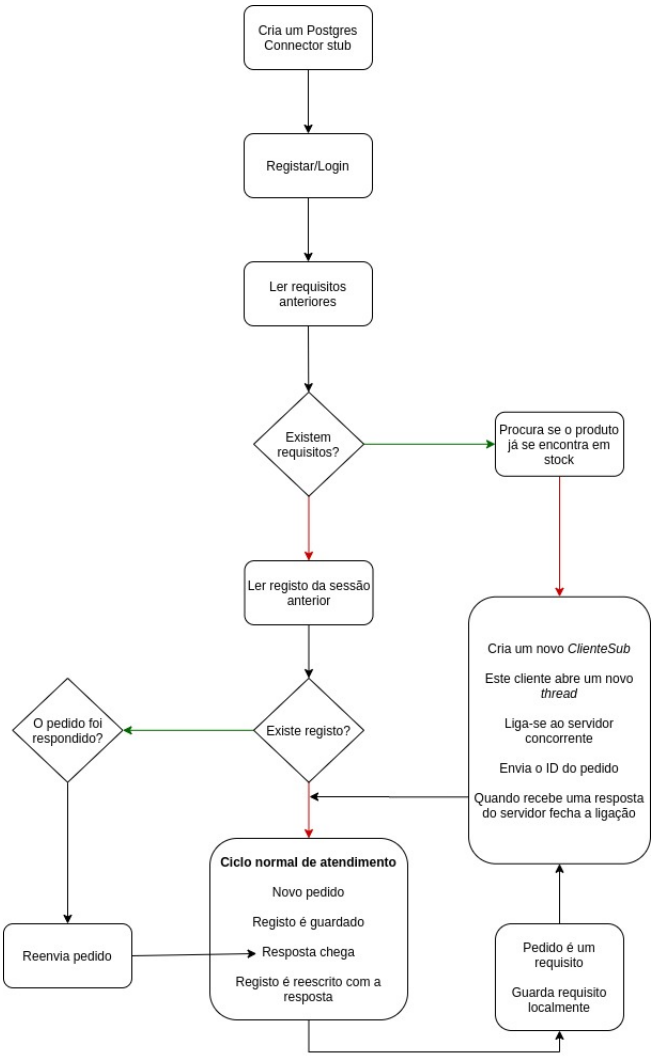

Figura 10: Diagrama do cliente

4 Conclusão

Ao longo do desenvolvimento deste trabalho o grupo foi ajustando perante as dificuldades encontradas.
O trabalho foi compilado no NetBeans sendo que não foi criado um ficheiro makefile, pelo que a entrega não se encontra como o professor deseja e é então um pasta projeto do NetBeans.
Tendo sido atingidos todos os aspectos do enunciado o grupo sente-se satisfeito com o trabalho desenvolvido.

5 Referências

Para o desenvolvimento deste trabalho prático foram consultadas as atividades das aulas práticas.
A imagem 2 foi adaptada do *pdf* leitura complementar sobre RMI em Java.