Sistemas Distribuídos

Mestrado Integrado em Engenharia Informática e Computação

3º Ano - 2º Semestre

Jogo 24

Grupo T01G01

Inês Santos Carneiro - up201303501@fe.up.pt

António Pedro Fraga - up201303095@fe.up.pt

Francisco Carvalho Rodrigues - up201305627@fe.up.pt

Pedro Vieira Lamares Martins - ei10096@fe.up.pt

1 Introdução

Este projecto foi desenvolvido no âmbito da unidade curricular de Sistemas Distribuídos e consiste no desenvolvimento de uma aplicação *RESTful*, tendo por base uma arquitetura do tipo Cliente-Servidor na qual um computador tanto funciona como Servidor como Cliente, isto é, um inicia-se como Servidor podendo também ser Cliente e os restantes ligam-se a este por *TCP*.

O nosso programa consiste numa plataforma que permite fazer torneios do jogo 24. Ao iniciar o programa é dada a opção a cada utilizador de criar uma sala de jogo ou se juntar a alguma que já exista. Apenas terá que indicar o nome que deseja dar a sala e o seu *nickname* caso pretenda criar uma nova ou apenas o *nickname* caso pretenda se juntar a uma.

Ao longo deste relatório será abordada a especificação do projecto, o desenvolvimento deste e as conclusões retiradas após o seu desenvolvimento.

2 Especificação

2.1 Análise do Tema

Como mencionado anteriormente o projecto consiste numa aplicação do jogo 24, sendo importante numa primeira fase definir este tipo de jogo e todas as regras a ele associadas.

O jogo 24 consiste num conjunto de 4 dígitos, que serão lidos de um ficheiro, e 4 operadores (soma, subtração, multiplicação e divisão) que combinados terão de resultar em 24. Cada digito apenas poderá ser usado uma única vez.

O utilizador que chegar mais rápido á solução ganha essa ronda e é gerado um novo tabuleiro para todos os jogadores na sala. A sala de jogo terá também um *chat* associado onde os jogadores poderão trocar mensagens.

2.2 Protocolos implementados

2.2.1 RESTful

Relativamente á comunicação entre Servidor e Cliente optamos por comunicação do tipo *Restful.* Assim quando um novo utilizador se conecta é feito um pedido ao servidor por forma a que seja enviada uma mensagem a todos os utilizadores presentes na mesma sala. Quando um utilizador pretende enviar uma mensagem, essa mensagem também é redirecionada a partir do servidor para todos os outros clientes.

2.2.2 TCP

Relativamente à comunicação entre salas, o grupo escolheu *TCP*. Numa primeira fase de implementação optamos por uma comunicação *peer-to-peer* entre as mesmas, no entanto devido á escassez de tempo não nos foi possível de implementar. Por forma a aproveitar toda a implementação *TCP* desenvolvida, implementação essa que inicialmente tinha o propósito de fazer parte do protocolo que viria a ser criado, achamos por bem deixar a comunicação em *TCP*.

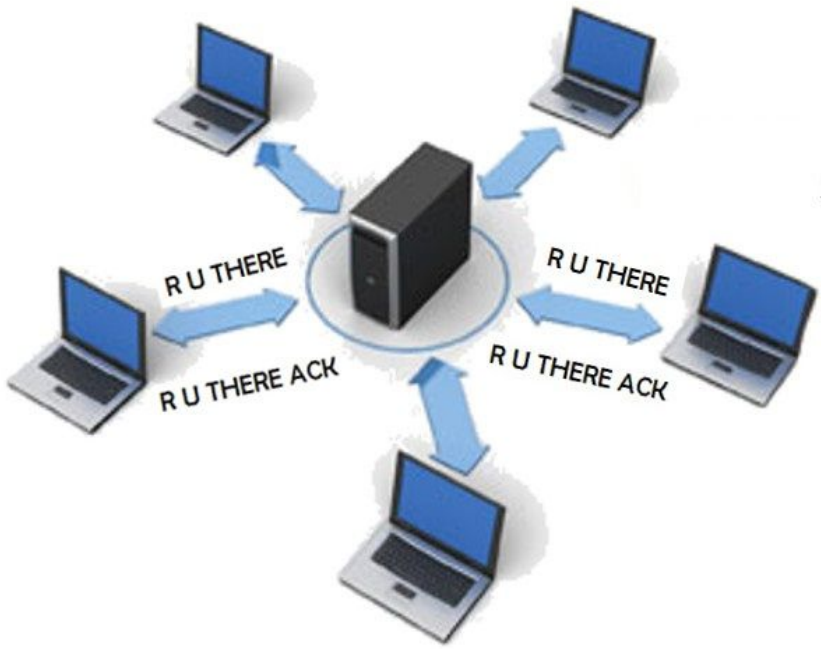
3 Implementação

3.1 Ferramentas utilizadas

Todo o projecto foi desenvolvido na linguagem de programação Java com recurso ao IDE *Intellij*. Optamos por esta linguagem por ser aquela com que mais familiarizados e por ser a que foi usada na implementação do projecto anterior.

3.2 Arquitetura do projecto

O projecto encontra-se dividido em duas grandes partes: servidor e cliente. O servidor é responsável por receber mensagens dos clientes, e reenviar para todos os clientes dependendo do tipo de mensagem. O servidor, tal como mencionado anteriormente é responsável pela criação do tabuleiro de cada uma das salas, enviando-o para todos os clientes associados a essa sala. Quando um cliente cria ou se junta a uma sala, é criada uma conexão *TCP* entre as duas partes, conexão essa que serve para verificar se um cliente se desconectou de determinada sala, se chegou a uma equação válida ou mesmo para troca de mensagens.



3.3 Detalhes relevantes da implementação

3.3.1 Segurança

Sendo que a nossa aplicação não requer qualquer tipo de registo ou autenticação do utilizador e também não foi necessário garantir uma segurança em bases de dados, optou-se por implementar um *socket* de SSL, fazendo recurso de uma *Keystore* e uma *password* associada, que geram um certificado válido, que garante segurança entre a troca de mensagens.

3.3.2 Concorrência

De forma a minimizar todos os problema de concorrência foram implementados os programas da seguinte forma:

3.3.2.1 Cliente-Servidor

Para a comunicação entre Servidor e Cliente escolhemos utilizar uma *thread* que é independente do servidor, esta é responsável pela comunicação constante entre as duas partes, assim que o cliente realiza um pedido, o servidor responde consoante a informação que deve ser redirecionada para esse cliente. É de realçar que o servidor nunca toma iniciativa de enviar uma mensagem ao cliente, este apenas responde a clientes. Todas as mensagens trocadas entre eles são enviadas em mensagens *JSON*, mensagens essas que incluem um tipo de pedido e campos associados a esse mesmo pedido. Sempre que é feito um pedido o *MessageHandler* é responsável por processar o mesmo e enviar a resposta ao Cliente.

3.3.3 Tolerância a falhas

No que toca a tolerância a falhas relativas aos utilizadores, caso algum se desconecte por algum motivo externo ao sistema o servidor tomará conhecimento do sucedido e a sala continuará a estar disponível para o utilizador por um período de tempo. Ao fim deste a sala será eliminada. Caso o Servidor se desligue por qualquer motivo as salas ficam ativas apenas por um período de tempo. Caso tivéssemos tido mais tempo, iriamos implementar um servidor de *backup*, um servidor de *backup* que iria ser iniciado caso o servidor inicial falhasse. O servidor de *backup* teria que enviar uma mensagem para o servidor principal num intervalo de tempo previamente definido, e caso não tivesse obtido resposta então tomaria ele o papel de servidor principal.

4 Interfaces da aplicação

4.1 Menu inicial

A partir deste menu o utilizador poderá criar uma nova sala ou juntar-se a uma já existente

4.2 Menu secundário

A partir deste menu o utilizador poderá juntar-se a uma sala já existente indicando apenas o seu *nickname*.

4.3 Sala de jogo

Aqui o utilizador poderá jogar o jogo do 24 e ainda poderá comunicar com todos os utilizadores presentes na sala, através do *chat* á direita.

5 Conclusões

5.1 Dificuldades

Uma das maiores dificuldades foi sem dúvida o tempo que tivemos para desenvolver este projecto. Seria bem mais simples se tivéssemos tido mais tempo para o implementar. Encontramos também algumas dificuldades com o desenvolvimento da parte gráfica.

5.2 Esforço dedicado

António Pedro Fraga - 25%

Inês Santos Carneiro - 25%

Francisco Carvalho Rodrigues - 25%

Pedro Vieira Lamares Martins - 25%

Horas totais dedicadas: 150 horas.

5.3 Melhorias

Algumas das melhorias discutidas pelo grupo consistem em:

- Comunicação *peer-to-peer* em vez de *TCP*.
- Possibilidade de se jogar online em vez de ser apenas em rede local.
- Cada utilizador passar a ter um *login* associado em vez de um simples *nickname*.
- Cada sala passar a ter uma *password* associada.
- Possibilidade de tabela de *scores* das jogadas de cada utilizador.
- Possibilidade de ter um *backup server* que caso o Servidor principal falhe entra em funcionamento.

6 Referências

- http://mavensearch.io/repo/org.json/json/20160212 - JSON 20160212 lib;
- http://stackoverflow.com/ - StackOverflow;
- http://www.tutorialspoint.com/ - Tutorial Point;
- https://docs.oracle.com/javase/tutorial/networking/urls/index.html - Para as conexões em REST
- https://en.wikipedia.org/wiki/Peer-to-Peer_Protocol_(P2PP) - Para uma primeira fase em que pensamos em fazer uma implementação peer-to-peer
- https://web.fe.up.pt/~pfs/aulas/sd2016/labs/lab3.html - para ligação TCP
- https://web.fe.up.pt/~pfs/aulas/sd2016/labs/7tcp.pdf - Sockets