Licenciatura Engenharia Informática Multimédia

Instituto Superior de Engenharia de Lisboa - ISEL

2023/2024 SV

Infraestruturas Computacionais Distribuídas

Relatório: Trabalho Prático 2

Turma: 41D

Nome: Ricardo Paiva Número: 50770

Nome: Rodrigo Caldeira Número: 50754

Nome: Miguel Cordeiro Número: 49765

9 de junho de 2024

Índice

Introdução.

Este projeto tem como objetivo aplicar os conhecimentos adquiridos durante as aulas práticas de IeCD através de dois trabalhos práticos. A intenção final é ensinar-nos a desenvolver um simples jogo, neste caso, o jogo do Othello. O projeto engloba o desenvolvimento de uma comunicação entre servidor e clientes em simultâneo através de uma rede, bem como, posteriormente, uma extensão para a web que utilize um JavaServer Pages (JSP) de modo a diminuir a complexidade e manter a consistência em termos de linguagens de programação. Na primeira parte do projeto, desenvolvemos o servidor. Este vai ter o comando do jogo e vai controlar a comunicação entre os dois computadores ligados à mesma rede com a finalidade de jogar o jogo de tabuleiro, Othello. Desenvolvemos também o cliente, que vai receber dados do servidor e enviar a sua resposta. Cada cliente vai ter um sistema de autenticação à entrada, e no segundo trabalho, temos mais um menu depois da autenticação que vai pedir ao cliente para escolher o seu adversário. (com 3 opções, o cliente pode optar por: jogar contra alguém aleatório; escolher alguém da lista de utilizadores online ou ficar simplesmente à espera que alguém o escolha). À semelhança do primeiro trabalho, neste segundo teremos também a utilização de um *XML* como *database* para armazenarmos as contas dos clientes e fazer a gestão de vitórias, empates e derrotas, tudo com a correção e validação de um *XSD*, e o auxílio da classe disponibilizada pelo Docente da cadeira.

Nesta segunda e última parte deste trabalho prático temos então como principal objetivo consolidar, através da aplicação prática em grupo, os conhecimentos adquiridos sobre tecnologias de desenvolvimento para a World Wide Web, tanto do lado do “client side” quanto do “server side”, com destaque em JavaServerPages (JSP).

O foco foi estender a aplicação computacional desenvolvida na primeira parte para funcionar sobre a Internet, implementando uma aplicação web que complementa a versão existente para consola/GUI. Essa nova aplicação web deve operar simultaneamente com a solução anterior, utilizando máquinas distintas para o servidor de dados e para o acesso via browser.

Diagrama Geral da Arquitetura.

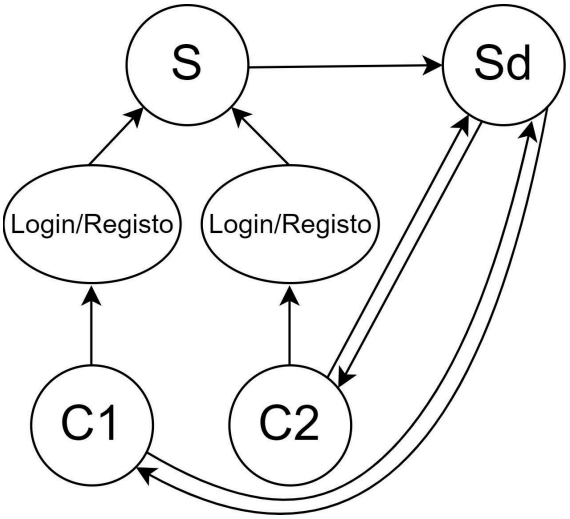

Figura 1: Diagrama geral da arquitetura antigo

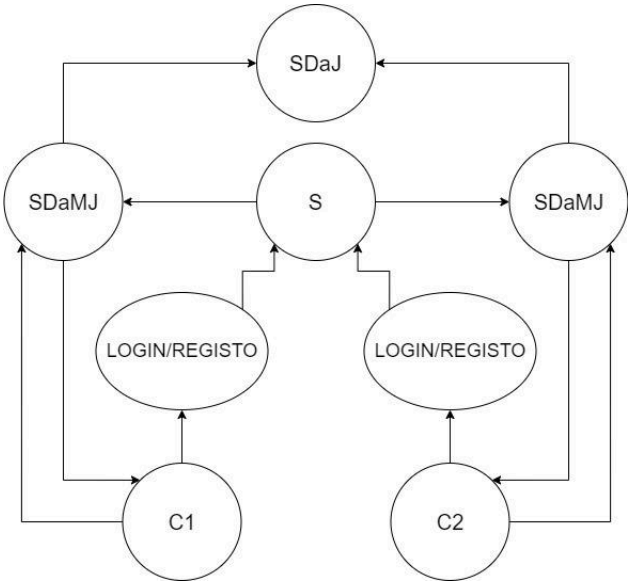

Figura 2: Diagrama geral da arquitetura atualizado

O diagrama geral da arquitetura (figura 1), representa as principais componentes do programa, assim como a transferência de dados realizada entre elas. O processo é iniciado quando dois clientes (C1 e C2), enviam os dados de login ou registo e o realizam com sucesso para aceder ao servidor.

Assim que o servidor recebe os dados de ambos os clientes, ele inicia um servidor dedicado (*Thread*, em Java) para executar o jogo entre os dois. Em seguida, os clientes vão receber do servidor dedicado o jogo a partir da consola e vão enviar a posição da peça para a sua jogada (linha e coluna), e assim sucessivamente até que haja um vencedor e o jogo termine.

Este é o modelo da parte 1 do trabalho. Já na parte 2, vamos ter uma *Thread* especial para cada cliente que se ligar ao servidor, significando assim que enquanto um cliente está a escolher o seu adversário, outro pode já estar a ligar-se ao servidor sem ter de esperar que o primeiro complete a ação que está a realizar. Quando dois clientes escolhem a opção de jogar um com o outro, partimos para a mesma *Thread* que já existia previamente na parte 1, onde se conectam os dois *Sockets* (clientes) e se vai mandar mensagens alternadamente conforme as regras do jogo que estamos a implementar.

Protocolo de Transporte TCP.

O que é um protocolo TCP:

O TCP (Protocolo de Controle de Transmissão) é um protocolo de alta confiança, que garante uma baixa perda de informação, usado para a comunicação entre computadores e redes, integrando-se à camada de transporte do conjunto de protocolos TCP/IP. Este protocolo utiliza o protocolo IP para encaminhar os datagramas pela rede onde cada ponto de acesso à aplicação é identificado por uma porta, permitindo múltiplas conexões em cada host. Cada porta geralmente está associada a um serviço específico na camada de aplicação, como por exemplo, HTTP (porta 80) e o HTTPS (porta 443), que requer a implementação de código adicional para lidar com a camada de segurança SSL (Secure Socket Layer).

Uma conexão TCP está repartida em 3 fases:

- **Estabelecimento da Conexão:** Nesta primeira fase, o cliente envia uma solicitação de conexão para o servidor, este por sua vez responde com uma confirmação do pedido, e para finalizar a conexão, o cliente confirma a recepção da mensagem, significando o estabelecimento bem sucedido da conexão;

- **Transferência de dados:** Quando a conexão é estabelecida, os dados são transmitidos (a conexão TCP garante-nos uma entrega ordenada e confiável, por mais que se comprometa mais a rapidez, quando comparado com uma conexão do tipo UDP);
- **Conclusão da Conexão:** Quando a comunicação é concluída ou já não é mais necessária, ambas as partes podem encerrar a conexão.

Descrição de cada componente.

Servidor:

Numa primeira fase, tínhamos um servidor TCP iterativo, visto que lidava com uma só conexão de cada vez, criando uma *Thread* que vai receber os dois clientes, e tratar do ciclo do jogo, até voltar a aguardar por novos clientes.

Agora, alterámos a estrutura para um servidor TCP concorrente. Este lida com as *Threads* de maneira diferente, criando conexões distintas para cada cliente, dando a possibilidade de ter vários pares de jogadores a ter vários jogos em simultâneo.

Novo protocolo de funcionamento do servidor:

Inicialmente o servidor fica bloqueado, até receber um *Socket* (cliente) na porta específica - 5025. Assim que um cliente estabelece conexão, é criada uma *Thread* para o servidor conseguir continuar a admitir clientes enquanto aquele que já foi admitido escolhe o seu oponente. Assim que é escolhido o oponente, começa-se então, à semelhança do trabalho final, uma *Thread* responsável por realizar o ciclo do jogo.

Cliente:

A arquitetura do cliente está feita de modo que o cliente exiba um menu inicial com a opção entre realizar um registo ou login para autenticação. Essa autenticação vai-nos ser relevante, quando no fim do jogo, se realiza o registo de vitórias, empates ou derrotas.

Após o login bem-sucedido, o cliente estabelece uma conexão com o servidor, criando uma *Thread* para lidar com a leitura das mensagens recebidas, através de um menu de opções de escolha de adversário. Depois dessa escolha feita, o servidor junta o cliente com o seu oponente e vão ambos entrar num loop que só terá fim quando o jogo terminar. Isto vai assim dar a capacidade a um utilizador de receber atualizações do jogo enquanto tem a capacidade de enviar novas mensagens de forma iterativa.

Estruturas usadas para manter dados.

Para manter os dados do jogo de modo consistente no servidor, utilizamos diversas informações, entre elas, o estado do tabuleiro, os registos dos jogadores, e os dados de autenticação.

Para o bom funcionamento do jogo, desenvolvemos duas classes, a classe Jogo e a classe Jogador que agora nesta nova parte vai servir também de cliente. O Jogo vai apenas ter os métodos básicos para funcionar:

- Iniciar o tabuleiro;
- Jogar;
- Terminar o jogo;
- Validar o movimento;
- Converter o tabuleiro para uma String;
- Determinar o vencedor;

A classe Jogador vai ter os métodos complementares, que permitem que o ciclo do Servidor funcione. Vai então ter as seguintes funções:

- O menus de autenticação;
- Print do tabuleiro para os clientes visualizarem as peças do Jogo;
- Dois construtores, um para o JSP outro para o Java App;
- Jogar na consola do cliente a jogada (linha e coluna);
- Método *main* que vai garantir a conexão consistente ao server;
- Fechar os *Sockets*;

Quanto aos dados de login e registo, foi criada uma classe à parte, para gerir os Utilizadores (User Class), que vai utilizar duas classes criadas pelo Docente e nos foi disponibilizada (MyImage e XMLDoc) e usamos também 3 métodos do docente para modificar e validar o documento *XML* com suporte do *XSD*. Resumidamente, quando inicializada, vai carregar o documento inteiro e verificar a sua validade para uma variável dentro da classe. De seguida, vai ter setters com mecanismos de validação para garantir que a informação está bem formatada antes de a armazenar.. Esta classe vai também utilizar uma classe dedicada a realizar o registo das estatísticas dos jogos, em *XML*.

Os registos dos utilizadores vão ser armazenados conforme o id único de cada *User* registado. Quando uma instância é criada, é logo passado de início esse id. De seguida, vai se obter, se o utilizador já tiver registo, os dados de vitórias, empates e derrotas, para mais tarde, ser atualizado, com um valor superior, dependendo do resultado do jogo. Se não houver nenhum registo anteriormente, ou seja, se for a primeira vez do utilizador a jogar, o cliente é notificado, e cria-se automaticamente registo.

Descrição de protocolos a nível da Aplicação.

Para mantermos os dados no servidor de maneira persistente optámos por usar uma estrutura com arquivos *XML* que seguem um procedimento *XSD* pré-definido. O servidor usa Java e TCP/IP com um protocolo *Thin Client* para permitir que os jogadores que ambicionam jogar se comuniquem sem falhas. O código do servidor, conhecido como Servidor.java, descreve como são geridas as conexões dos clientes e como a lógica do jogo é implementada. Os sockets TCP/IP são a base do protocolo de comunicação, que permite uma conexão confiável entre o cliente e o servidor.

A troca de mensagens é realizada através de fluxos de entrada e saída que são usados para trocar informação valiosa entre o servidor e os clientes. Isso permite que os jogadores e o servidor se comuniquem de forma bidirecional (cada parte consegue ler e escrever). Os clientes e o servidor estão sempre em contato um com o outro durante a

operação do servidor. Os jogadores recebem atualizações sobre o estado do jogo enquanto jogam até que haja um vencedor ou um empate (os clientes nunca falam diretamente. Têm sempre um servidor como intermediário).

Os dados do *XML* da mensagem têm o seguinte formato:

<lista>

<user>

<id> … <lastupdated> … <username> … <password> … <nacionalidade> …

<idade> … <foto> … <cor>

</user>

</lista>

Os dados do *XML* do registo têm o seguinte formato:

<registos> <registo>

<id> … <vitórias> … <empates> … <derrotas>

</registo> </registos>

Os dados do *XSD* vão seguir a regra dos seguintes diagramas:

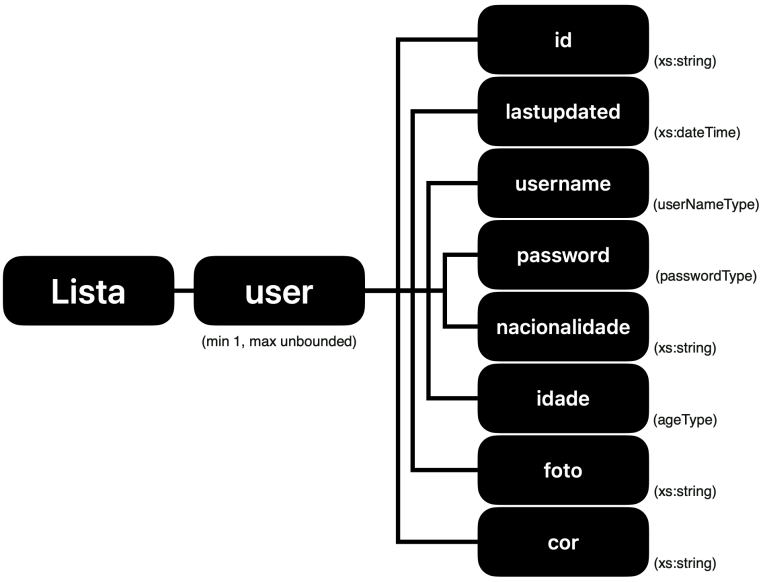

Figura 3: Diagrama do *XSD* da lista de utilizadores

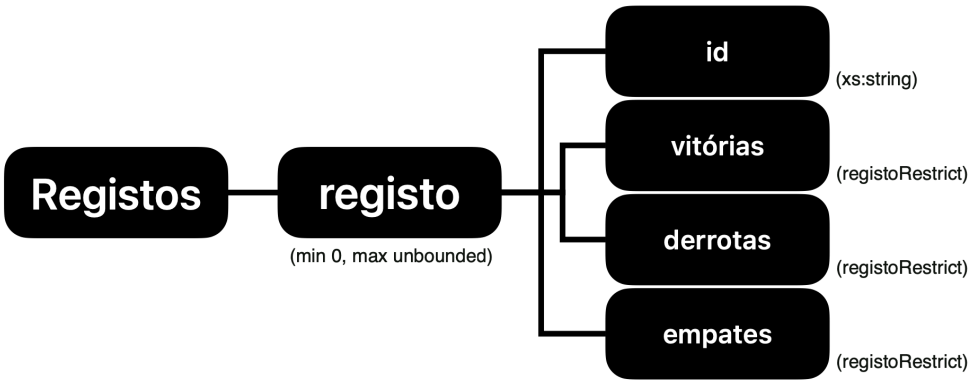

Figura 4: Diagrama do *XSD* da lista de registos

O que os diagramas *XSD* significam, no caso da lista de utilizadores, cada utilizador vai ter de ter obrigatoriamente 8 características (nome de utilizador, palavra-passe, id único, última vez que foi atualizado, nacionalidade, idade, foto e cor preferida) e o *XML* tem de ter pelo menos 1 utilizador para ser válido. No caso da lista de registos, é possível não haver registos e cada registo tem de ter o valor único do utilizador, o nº de vitórias, derrotas e empates.

Exemplos de Utilização[1].

Depois de correr a classe Cliente vai aparecer na consola o Menu Inicial com as opções de registo, login e sair.

```
--------------------
Menu Inicial
--------------------
1. Registo
2. Login
3. Sair
--------------------
Digite a opção desejada:
```

Figura 5: Menu Inicial

[1] Este parágrafo foi inalterado, comparativamente ao relatório passado.

Se optar pela opção 1 vai-lhe ser pedido para inserir os dados para criar um novo utilizador.

```
Digite a opção desejada: 1
Insira os dados do registo: (username, password, nacionalidade, idade, foto)
Nome do utilizador: teste1
Senha para autenticação: 12345
Nacionalidade (pt ou portugues, p.e.): pt
Idade: 19
----- Dados do Utilizador -----
Identificador (UUID): null
Última atualização em: 2024-04-14T19:43:24.980816400
Nome de utilizador: teste1
Senha: 12345
Senha encriptada: 5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
Idade: 19
Nacionalidade: pt
Foto em base64: /9j/4AAQSkZJRgABAQEBLAEsAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBw
```

Figura 6: Novo utilizador

No caso da opção 2 vai-lhe ser pedido para inserir o nome do utilizador e a password de um utilizador que já foi previamente criado. Caso a password não corresponder ao user indicado vai ser pedido que insira os dados novamente (figura 4), se a password coincidir com a associada ao user pode então jogar o jogo (figura 5), que vai ser apresentado na consola respetivamente, tal como se pode ver na (figura 6).

```
--------------------
Menu Inicial
--------------------
1. Registo
2. Login
3. Sair
--------------------
Digite a opção desejada: 2
Nome do utilizador: werwer
Senha para autenticação: 12345
Utilizador especifico nao encontrado.
Nome do utilizador:
```

Figura 7: Utilizador não encontrado

```
----- Dados do Utilizador -----
Identificador (UUID): null
Última atualização em: 2024-04-14T19:43:24.980816400
Nome de utilizador: teste1
Senha: 12345
Senha encriptada: 5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
Idade: 19
Nacionalidade: pt
Foto em base64: /9j/4AAQSkZJRgABAQEBLAEsAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBw
----- -------------------- -----
```

Figura 8: Utilizador encontrado

```
Java-> Ligação estabelecida: Socket[addr=localhost/127.0.0.1,port=5025,localport=64013]
Simbolo atribuído: X
Othello Game.
$|1|2|3|4|5|6|7|8
1
2
3
4       O X
5       X O
6
7
8
------------------
Player: X
Inserir linha e coluna (p.e. 1 3):
```

Figura 9: Consola cliente

Extensão para Aplicação Web.

Para estender a funcionalidade do sistema para a web, implementamos uma solução alternativa baseada em JavaServer Pages (JSP) e servlets, que pode funcionar em simultâneo com a solução de consola. Utilizamos um servidor de aplicação web (por exemplo, Apache Tomcat v.10.1) para lidar com solicitações HTTP, comunicando-se com o servidor de jogo original.

Inicialmente, criamos uma página introdutória, tal e qual o menu de autenticação onde o utilizador escolhe a opção de registar ou de iniciar sessão.

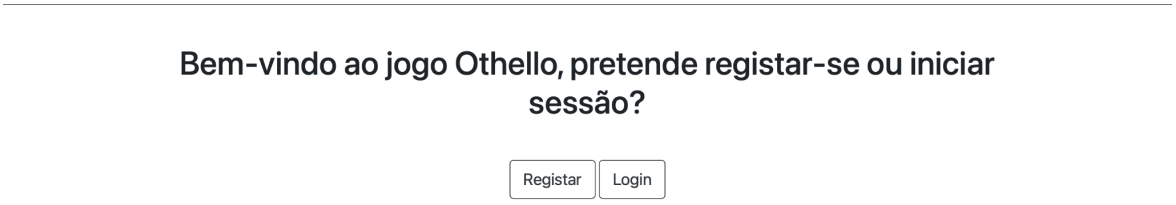

Figura 10: Página inicial

Se o utilizador selecionar a opção registar, temos o ecrã de registo onde introduz os dados todos necessários.

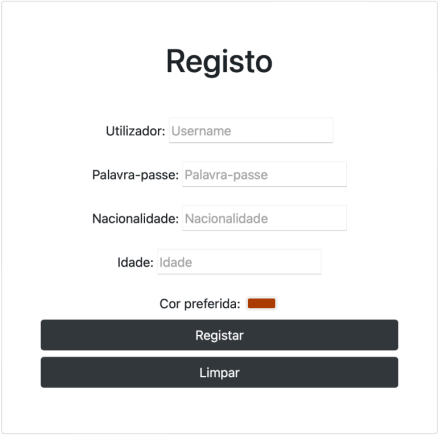

Figura 11: Página de registo

Se o utilizador selecionar a opção para iniciar sessão, o ecrã muda para uma página onde pede apenas o nome de utilizador e a palavra-passe.

Figura 12: Página para iniciar sessão

Depois desta fase passada, vamos então entrar no ecrã do jogo que tem um tabuleiro 8x8 interativo, onde o jogador faz uma jogada ao clicar na quadrícula que ambiciona jogar.

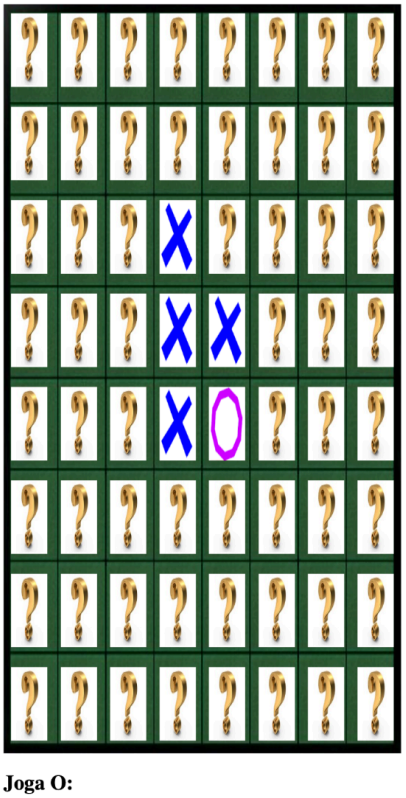

Figura 13: Página para iniciar sessão

Compatibilidade com Browsers.

A compatibilidade de uma página JSP depende de vários fatores, e hoje em dia, a chance de encarar um problema de compatibilidade é mais reduzida, dado haver um grande suporte para todos os browsers modernos (Chrome, Firefox, Safari, Edge) e até versões mais recentes do Internet Explorer (9+). A nossa aplicação web foi testada para garantir a maior compatibilidade entre os browsers mais comuns:

- **Google Chrome**
- **Mozilla Firefox**
- **Safari**

A aplicação mostrou-se compatível com todos os browsers testados, com pequenos ajustes de estilo utilizando CSS específico para cada browser, quando necessário.

Navegadores mais antigos, como é o caso do Internet Explorer, é possível que se enfrente alguns problemas de implementação, dado que estamos a utilizar a versão mais recente do HTML5 e tanto o CSS como o JavaScript (na utilização do Bootstrap) podem ter algumas funcionalidades mais recentes que podem não funcionar, ou funcionar mal.

Conclusões.

Concluindo, a aplicação prática dos conhecimentos adquiridos no âmbito da disciplina de Infraestruturas Computacionais Distribuídas (IeCD) é refletida neste trabalho. O projeto inicialmente desenvolvido teve como objetivo principal criar um servidor TCP iterativo para permitir a comunicação entre dois clientes e a realização de um jogo de tabuleiro (Othello Game). Durante o desenvolvimento, foram abordados conceitos fundamentais, entre eles, o uso e explicação do protocolo TCP para a transmissão dos dados e ligação dos clientes. Além disso, foram utilizadas estruturas de dados, classes para representar o jogo e o funcionamento e ficheiros *XML* e *XSD* para garantir o armazenamento dos dados dentro e fora do jogo.

Os resultados obtidos demonstram a capacidade de aplicar os conhecimentos teóricos adquiridos em situações práticas, bem como a habilidade para desenvolver soluções eficientes. Este projeto proporcionou a todos os membros do grupo uma experiência valiosa no desenvolvimento de um sistema distribuído e no uso de diferentes tecnologias.

Contudo, reparámos que tivemos algumas dificuldades a realizar aquilo que nos foi pedido, entre eles:

- Na primeira parte do trabalho, não conseguimos realizar a procura por jogadores online a aguardar adversários, fazendo com que o cliente jogue com o primeiro que se conectar depois dele, contudo, para a segunda fase, ampliamos o nosso protocolo do servidor e conseguimos efetuar uma procura por utilizadores online na consola e uma espera automática quando conectado pelo browser;

- No jogo, não tivemos sucesso a marcar no tabuleiro as jogadas possíveis, estas estão a ser bem calculadas, porém, é recomendado que o jogador saiba as regras pois não vai ter apoio visual nas jogadas válidas;
- No armazenamento das estatísticas do jogador, no final do jogo, estamos a calcular bem as vitórias / derrotas / empates, porém, não conseguimos calcular o tempo que cada jogo durou;
- Não fomos capazes de gravar os registos a partir do browser, porém na consola tudo funciona;
- Não conseguimos definir a cor de fundo conforme a preferida, e o preenchimento automático por nome completo.

Bibliografia.

- **Official rules for the game Othello** - "Regras oficiais do jogo Othello". *https://www.worldothello.org/about/about-othello/othello-rules/official-rules/english*

- **Exemplos seguidos para ajudar na resolução do problema.**

Exemplo Jogo do Galo Peer-to-Peer - Moodle IeCD 2324SV;

Exemplo JSP-MVC - Moodle IeCD 2324SV;

Exemplo JSP-TCP - Moodle IeCD 2324SV;

Exemplo Jogo do Galo JSP - Moodle IeCD 2324SV.

- **Exemplos seguidos para compreensão e implementação do XML e XSD.**

Exemplo Utilização: XML XSD (include), DTD e XSLT - Moodle IeCD 2324SV.