



LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA

2022 | 2023

SEMESTRE VERÃO

INFRAESTRUTURAS COMPUTACIONAIS DISTRIBUÍDAS

2ª parte

Docente: Engenheiro Porfírio Filipe

18 DE JUNHO DE 2023

Realizado pelo Grupo:

Pedro Silva A48965 41D

Cláudia Sequeira A49247 41D

Conteúdo

Introdução	4
Alterações	5
Dados.....	5
Código	6
Jogador	7
Login.....	7
Perfil Privado	8
Registo	8
Consulta do Perfil Privado	9
Leaderboard.....	10
Jogo	12
Compatibilidade	12
Jogabilidade.....	12
Código	14
JogoServlet.....	14
PlayerServlet.....	15
RegisterServlet.....	15
Jogador.jsp	16
Perfil.jsp	16
meuJogador.java.....	16
Conclusões	17
Bibliografia	18

Índice de Figuras

Figura 1—Esquema XSD atualizado.....	5
Figura 2 — Página de Login	7
Figura 3 — Página de perfil privado	8
Figura 4 — Perfil Privado Preenchido.....	9
Figura 5 — Leaderboard	10
Figura 6 — Procura Autocomplete	11
Figura 7 — Perfil Publico	11
Figura 8 — Interface Gráfica do Jogo.....	12
Figura 9 — Final do Jogo	13

Introdução

No âmbito da UC de Infraestruturas Computacionais Distribuídas vamos desenvolver um projeto que procura estender o sistema computacional desenvolvido na primeira parte do trabalho, de modo a que possa ser operado sobre o navegador web

A segunda parte não substitui a solução para a consola que foi implementada na primeira parte, que continuará a funcionar em simultâneo com a aplicação web.

No desenvolver deste projeto, vamos preservar as funcionalidades e requisitos do trabalho anterior, e vamos adicionar algumas funcionalidades importantes. Estas incluem permitir que o jogador registe e altere o seu perfil, e vai ser acrescentado a sua cor preferida de modo a definir a cor de fundo da janela.

Vai também ser implementado um mecanismo de pesquisa de outros jogadores utilizando um controle *AutoComplete*. O sistema deverá também permitir que os jogadores joguem vários jogos simultaneamente, mantendo um limite de 30 segundos para cada jogada.

Outras características relevantes a serem implementadas nesta parte do projeto incluem a visualização do quadro de honra ("*Scoreboard*"), ordenado pelo número de vitórias. Adicionalmente, consideramos a compatibilidade com os navegadores mais comuns e apresentamos uma revisão da arquitetura e dos modelos de dados/mensagens.

Além dos requisitos mínimos, fizemos esforços para introduzir melhorias que introduzem facilidade na utilização e maior dinamismo de conteúdo.

Alterações

Dados

Vamos começar por abordar as alterações que foram feitas ao trabalho produzido na primeira parte do projeto. A mudança mais notável é na estrutura do nosso XML, onde introduzimos um novo elemento 'cor'. Este novo elemento permite que o jogador armazene a sua cor preferida, que será posteriormente utilizada como cor de fundo na sua página de perfil no browser.

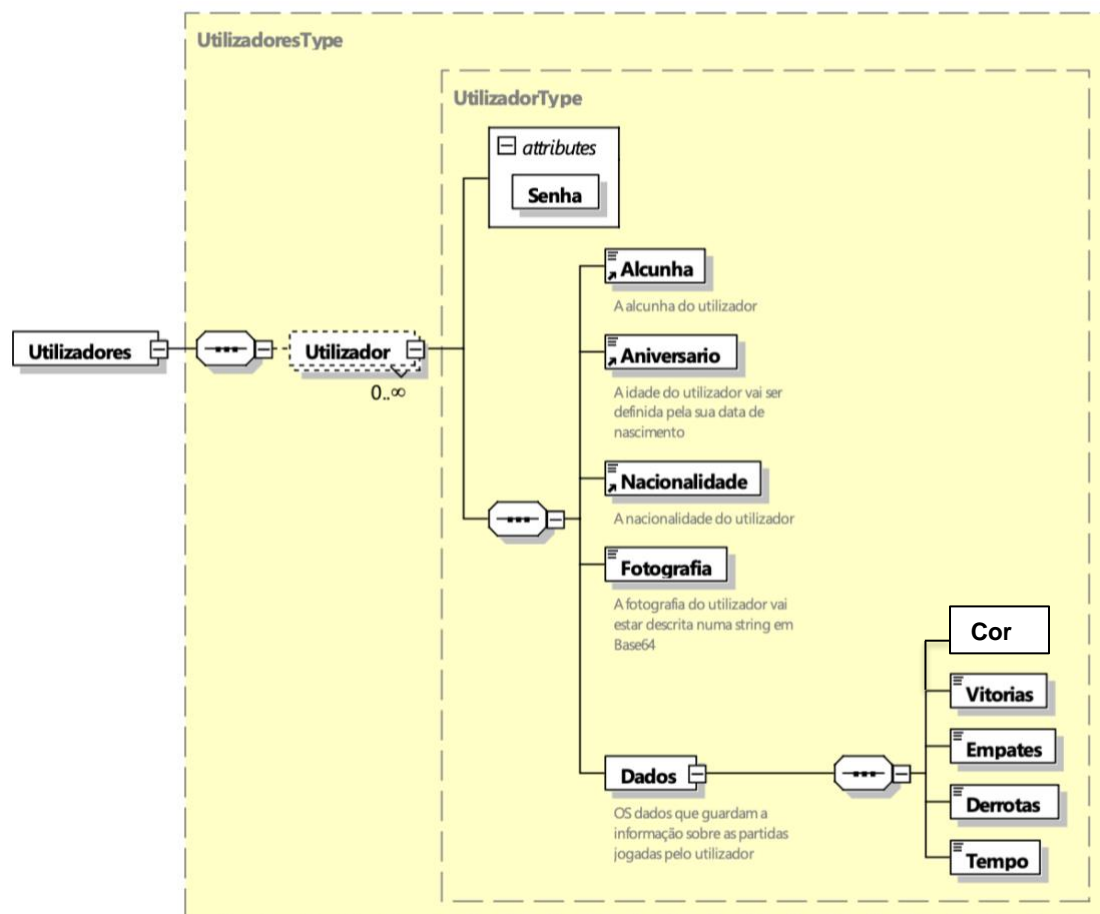


Figura 1—Esquema XSD atualizado

Além disso, adaptamos o formato do elemento da nacionalidade para integrar eficientemente com as bandeiras disponíveis com o uso do Bootstrap. Agora, a nacionalidade é representada por um código de duas letras (como 'PT', 'US', 'ES', etc.), permitindo uma correspondência direta e visualmente apelativa com as bandeiras de cada país na interface do utilizador.

Código

Em relação ao código, as modificações implementadas no trabalho anteriormente desenvolvido foram a inclusão de um método denominado 'loginWeb' na classe 'Conta'. Este método será responsável pela comunicação com o servidor, tratando dos dados recebidos a partir do navegador.

Na classe "Servidor", foi realizado um ajuste para incorporar um limite de tempo de 30 segundos para realizar uma jogada. Para implementar essa funcionalidade, foi adicionada uma lógica que configura um temporizador assim que uma jogada é iniciada. Durante esse período de tempo, o jogador que possui o tabuleiro terá 30 segundos para realizar sua jogada, enquanto o outro jogador não terá limite de tempo.

Caso os 30 segundos se esgotem e nenhuma jogada seja registrada, a exceção de tempo limite será capturada e uma mensagem será enviada aos jogadores para informá-los sobre o ocorrido. Isso garante que os jogadores sejam notificados quando excede o tempo limite para realizar uma jogada e permite que a partida prossiga adequadamente.

Jogador

Login

A página de Login é implementada através de uma JavaServer Page (JSP), onde os utilizadores podem inserir o seu nome de utilizador e a respetiva senha para iniciar a sessão.

A autenticação do utilizador é processada redirecionando a requisição para a LoginServlet, que por sua vez, recorre à classe auxiliar UserDataStore para validação dos dados inseridos pelo utilizador. Importante destacar que a classe UserDataStore implementa o padrão de design Singleton, assegurando que apenas uma instância da mesma seja criada ao longo da execução do programa. Ao ser inicializada, esta classe lê todos os dados armazenados na base de dados, especificamente no ficheiro XML, e carrega os nomes de utilizador (usernames) e as respetivas senhas (passwords) para efetuar a verificação durante o processo de autenticação.

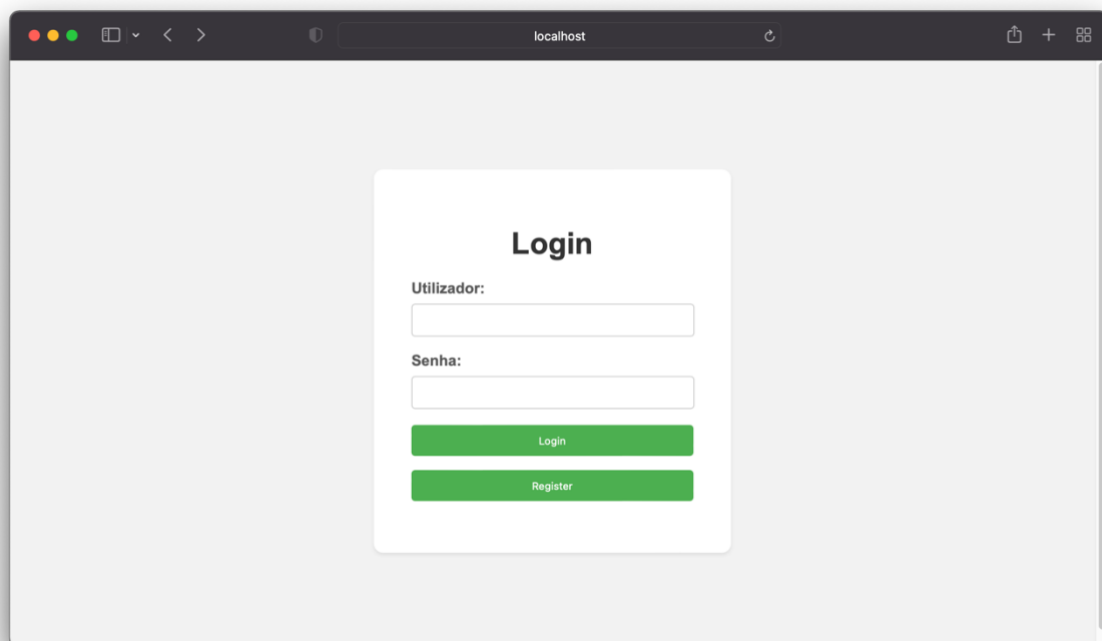


Figura 2 — Página de Login

Esta interface oferece uma opção de criar uma nova conta. Ao clicar no botão "Register", o utilizador é redirecionado para a página de perfil.jsp, conforme ilustrado na Figura 3. Esta página de perfil permite ao utilizador fornecer as informações necessárias para concluir o processo de registo.

Perfil Privado

Registo

Na página de perfil privado que se pode ver na figura 3, o utilizador tem a possibilidade de criar um novo registo através do preenchimento dos campos necessários e confirmando em "Save Changes".

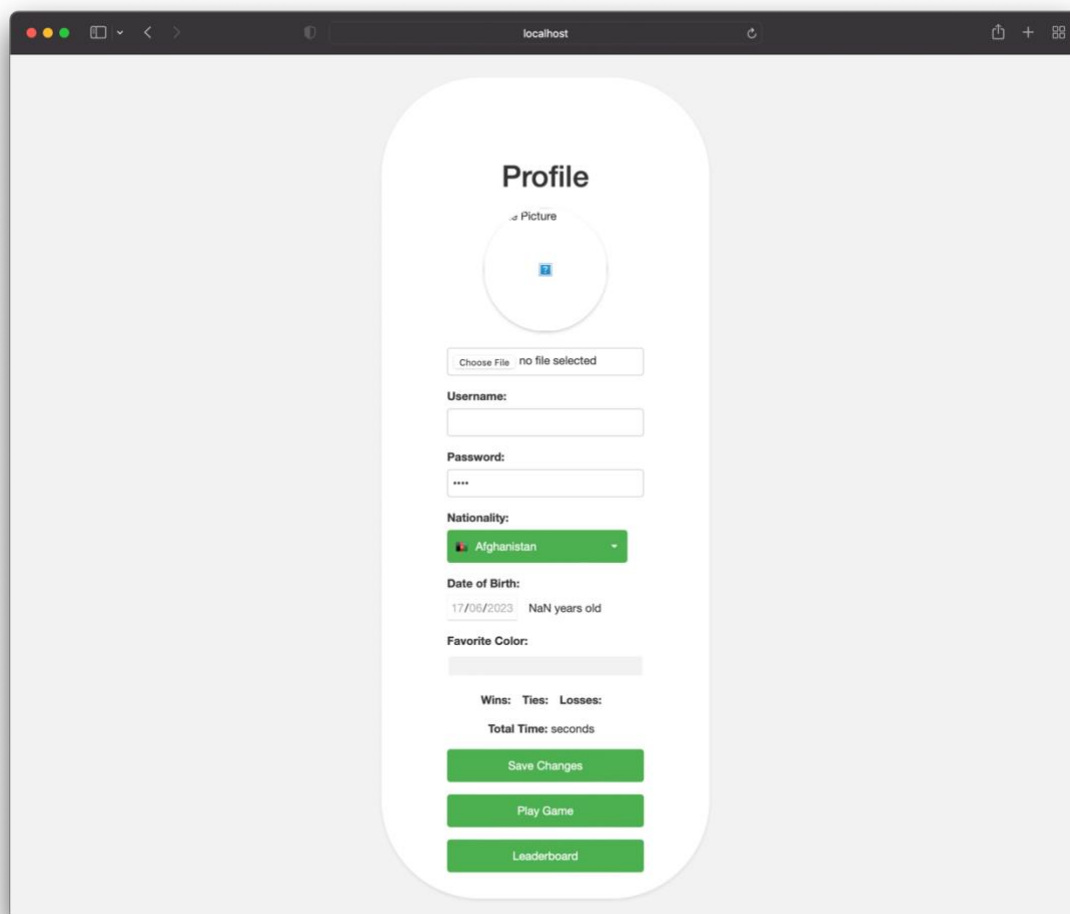


Figura 3 — Página de perfil privado

As operações de registo e atualização dos dados são geridas pela servlet 'RegistrarServlet'. Em termos de resposta aos problemas que possa ocorrer nesta página, desenvolvemos os seguintes mecanismos.

- Caso o nome de usuário já existir, solicita-se que seja fornecido um nome diferente.
- Ao pressionar o botão "Jogar" sem realizar o registo será mostrada uma mensagem de erro que pede para fazer login ou registrar-se.
- Se tentar aceder o Leaderboard, será redirecionado para a página inicial de login.

Consulta do Perfil Privado

A figura 4 mostra como as informações, anteriormente armazenadas na base de dados do servidor, são exibidas na interface gráfica proporcionada pela JSP. A cor de fundo selecionada pelo utilizador, a sua preferida, é refletida na página, proporcionando uma experiência de usuário personalizada. É também possível indicar a nacionalidade do usuário através de uma lista *dropdown*. Além disso, a página exibe informações sobre o desempenho do usuário nos jogos, incluindo o número de vitórias, empates, perdas e o tempo total dedicado aos jogos.

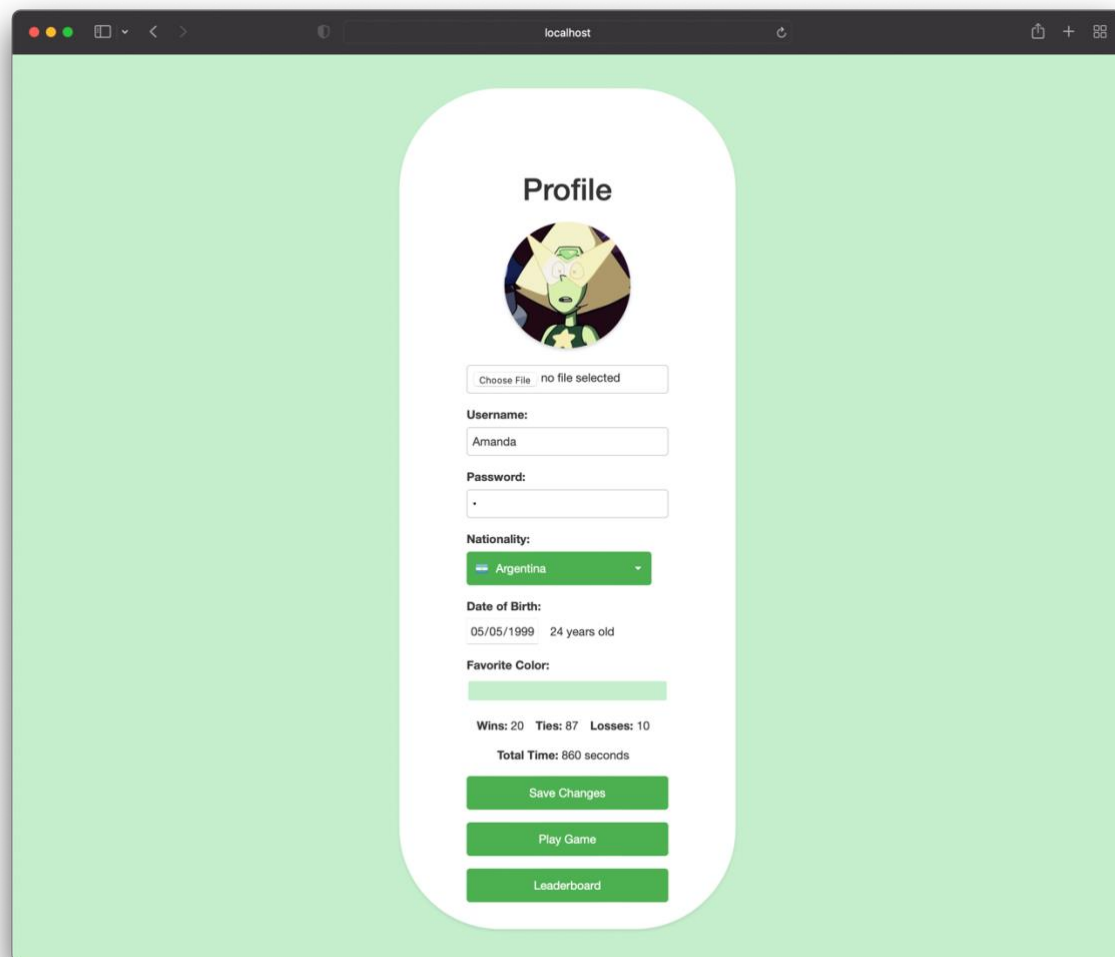


Figura 4 — Perfil Privado Preenchido

A partir desta página de perfil, o utilizador pode optar por entrar no modo de jogo ou consultar o 'Leaderboard'.

Leaderboard

A composição e apresentação do 'Leaderboard' começam com a invocação da servlet 'ScoreboardServlet' que tem a responsabilidade de aceder aos dados no ficheiro xml e extrair as informações de todos os utilizadores. Uma vez recolhidos, os utilizadores são organizados com base no número de vitórias obtidas. No caso de um empate no número de vitórias, a distinção é feita com base no tempo médio gasto por jogo, garantindo assim um ranking justo e equilibrado.

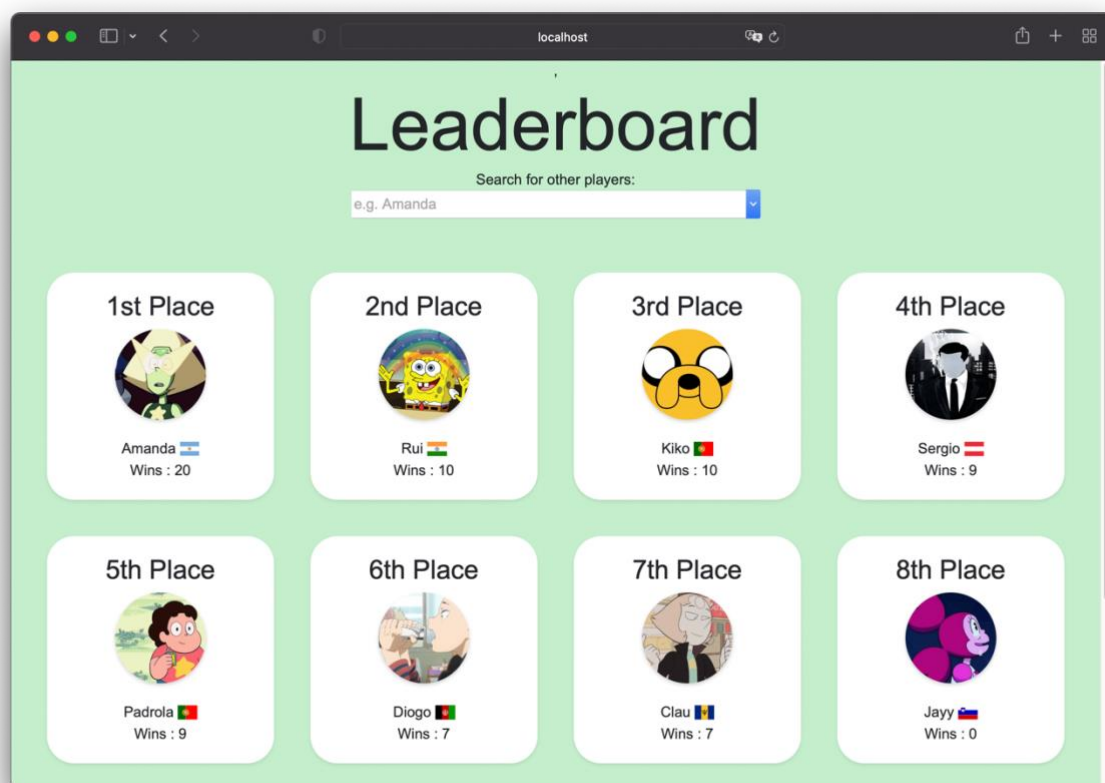


Figura 5 — Leaderboard

Esta estruturação dos dados é facilitada pela classe auxiliar 'Utilizador', que armazena informações relevantes como alcunha, fotografia, nacionalidade, vitórias, empates, derrotas, tempo total de jogo e a sua cor favorita.

Importante notar que o nosso 'Leaderboard' foi concebido para acomodar um máximo de 25 utilizadores, mantendo uma interface limpa e fácil de navegar.

No Leaderboard é possível procurar pelos adversários, a funcionalidade utiliza o *autocomplete* no auxílio e que ao pressionar *enter* abre o perfil correspondente. Em termos a tolerância de erros é verificado que só abre perfis de utilizadores válidos.



Figura 6 — Procura Autocomplete

Na Figura 7 pode-se observar um exemplo de um perfil público, oferece uma visão mais profunda das estatísticas e informações individuais dos adversários e até a cor de fundo da página corresponde à cor favorita do jogador ao qual o perfil corresponde, enriquecendo assim a interatividade e o entendimento dos utilizadores sobre o desempenho dos concorrentes.

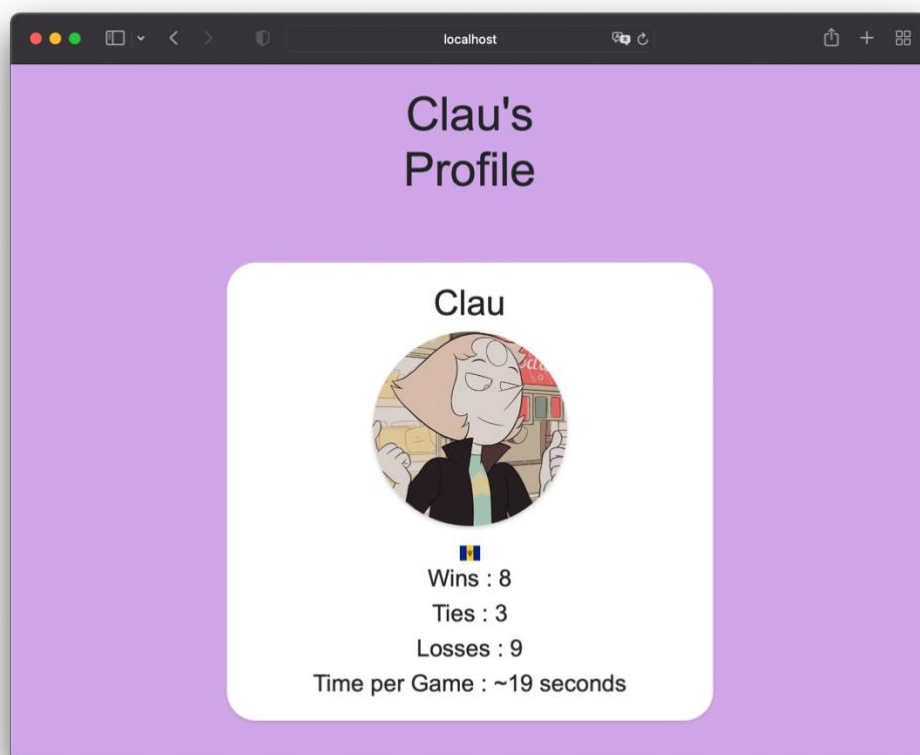


Figura 7 — Perfil Publico

A implementação dessa funcionalidade é suportada pela 'PublicProfileServlet', que fornece a ponte necessária para a apresentação da JSP 'publicProfile' do jogador escolhido. Demonstrando como as Servlets permitem facilitar a comunicação e a transferência de informações entre a interface do utilizador e a base de dados do servidor

Jogo

Compatibilidade

À implementação do jogo 4 em linha que tínhamos na parte um foi melhorada adicionando a funcionalidade de jogar com uma interface gráfica no navegador. Essa adição permite uma experiência de jogo mais intuitiva e visualmente atraente para os jogadores.

É permitido que os jogadores utilizem ambos apenas a consola, apenas a interface gráfica do navegador ou uma combinação dos dois..

Em termos de compatibilidade, foram realizados testes bem-sucedidos nos seguintes navegadores: Microsoft Edge, Chrome, Safari, Opera e Firefox. Isso garante que os jogadores possam desfrutar do jogo em uma ampla variedade de plataformas e navegadores. É também possível que ambos os jogadores estejam no mesmo browser a jogar um contra o outro.

Jogabilidade

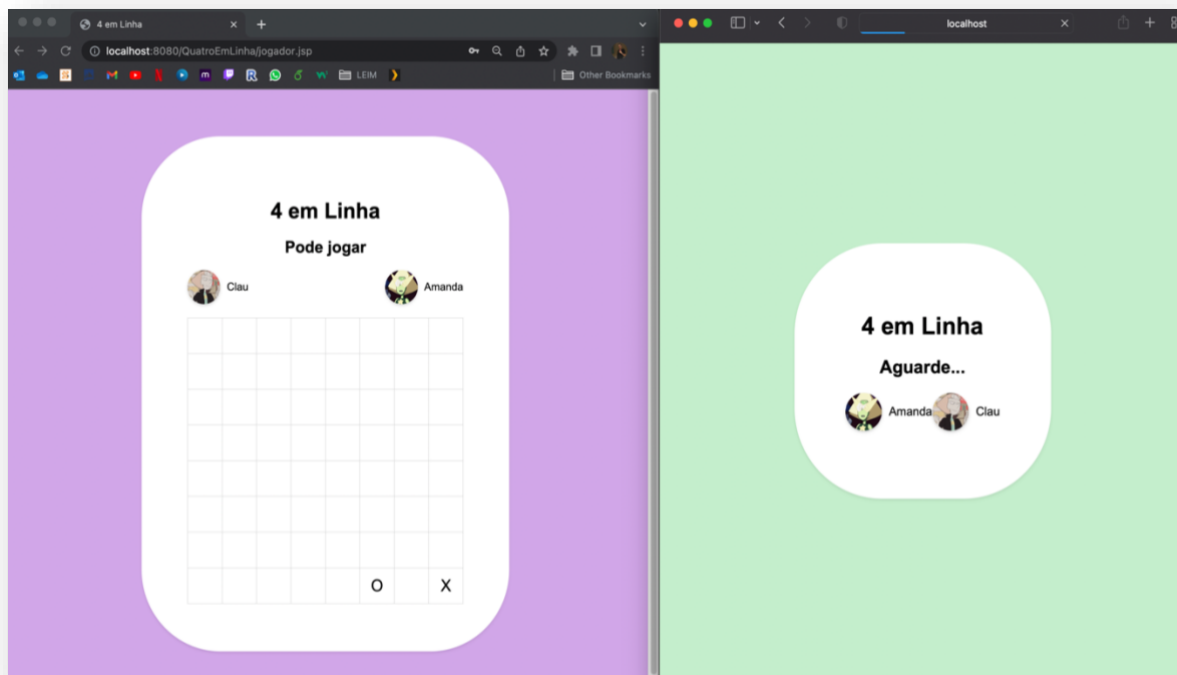


Figura 8 — Interface Gráfica do Jogo

O jogo foi desenvolvido usando JSP com a página 'Jogador', a qual estabelece a comunicação com o servidor utilizando uma instância da classe 'meuJogador' e extrai as informações do jogador a partir da Servlet 'JogoServlet'.

No que diz respeito à tolerância a falhas, foi implementada uma solução para lidar com o caso em que um jogador excede o tempo limite para fazer uma jogada. Nesse cenário, o jogador que atingiu o tempo limite será desconectado automaticamente e redirecionado para a página de login. O outro jogador, será apresentada o ecrã de vitória. Essa abordagem garante que, em caso de falha de um jogador em cumprir o tempo limite, o jogo seja interrompido para esse jogador específico, permitindo que o outro jogador seja declarado vencedor.

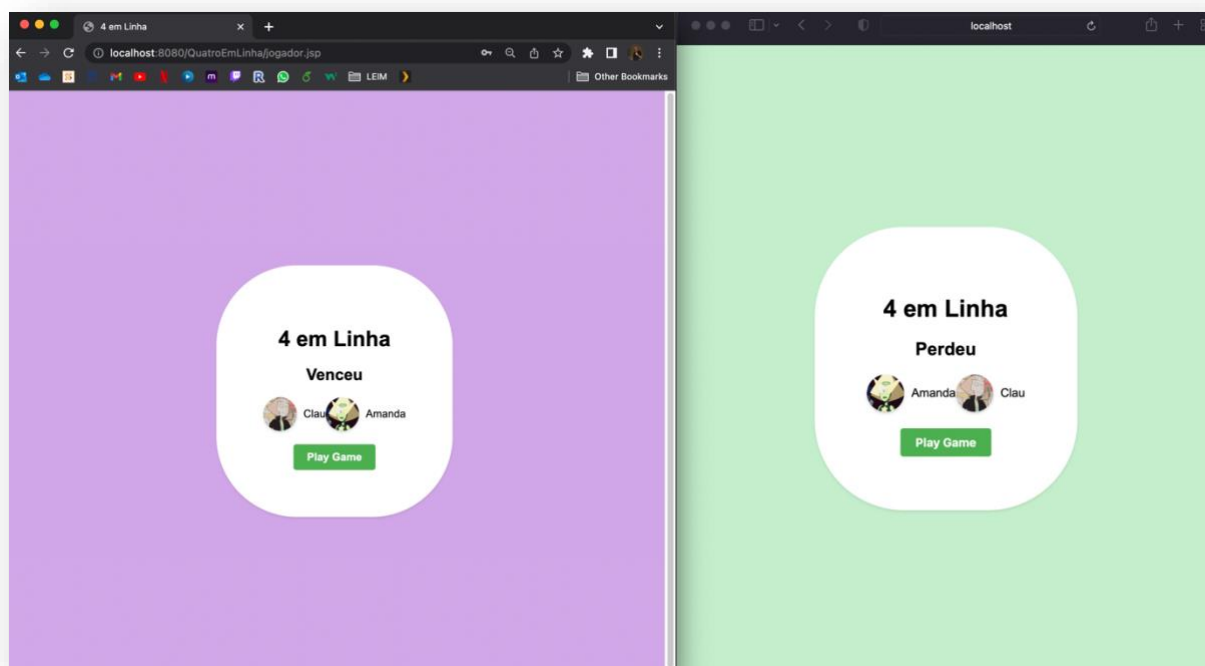


Figura 9 — Final do Jogo

Na figura 9 podemos observar o ecrã de final de jogo indicado se houve uma vitória, derrota ou empate.

Código

As **Servlets** são programas do lado do servidor que lidam com pedidos de clientes e retornam uma resposta personalizada ou dinâmica para cada solicitação. São executadas dentro da Java Virtual Machine (JVM), o que fornece um alto grau de eficiência e as torna escaláveis.

A Interface Servlet declara, mas não implementa, métodos que gerenciam a servlet e as suas comunicações com os clientes. Os dois métodos primários nesta interface são `doGet()` e `doPost()`, que correspondem a requisições HTTP GET e POST, respectivamente.

- **doGet():** é chamado pelo servidor para permitir que uma servlet manipule um pedido GET, num pedido HTTP GET os parâmetros são enviados como uma string de consulta na URL. Geralmente é usado para pedidos de leitura para o servidor em que o URL do pedido corresponde a um recurso específico.
- 2. **doPost():** O método é utilizado pelo servidor para permitir que uma servlet manipule uma solicitação POST, ou seja um HTTP POST onde os parâmetros são enviados no corpo do pedido. É geralmente é usado ao publicar dados que devem ser processados pelo servidor, como o envio de um formulário.

Os métodos `doGet()` e `doPost()` recebem dois argumentos:

- `request HttpServletRequest`: Este é o objeto de pedido que contém informações sobre a solicitação do cliente, como informações de cabeçalho, nomes e valores de parâmetros, etc.
- `response HttpServletResponse`: Este é o objeto de resposta que permite a servlet formular uma resposta para o cliente. A servlet pode gravar dados no corpo da resposta (como conteúdo HTML), definir cabeçalhos de resposta e assim por diante.

JogoServlet

A Servlet "JogoServlet" é responsável por processar as requisições POST relacionadas ao jogo. Ela recebe os dados do jogador, verifica sua existência e realiza a jogada no jogo.

Funcionamento da Servlet:

1. A Servlet **recebe os parâmetros do request**, incluindo o identificador único do usuário (uuid), a alcunha nome do jogador e a senha. E é feita uma verificação se o jogador está registrado na sessão. Caso não esteja, uma mensagem de erro é definida e o fluxo é redirecionado para a página de perfil (/perfil.jsp).
2. O jogador **realiza uma jogada** no jogo através da chamada do método "jogar" do objeto "jogador" da classe "meuJogador".
3. Em seguida, é invocado o método "getData" para **obter informações adicionais do jogador** a partir do arquivo XML de dados. Os valores são armazenados como atributos na requisição HTTP.
4. **O request é encaminhado para a página do jogador** (/jogador.jsp) para exibir as informações e continuar o jogo.

Em resumo, a Servlet "JogoServlet" processa as jogadas do jogador, verifica suas informações no arquivo XML e encaminha as informações necessárias para a página do jogador.

PlayerServlet

A 'PlayerServlet' é a classe responsável por exibir as informações do jogador no seu perfil. Ela implementa diferentes métodos para lidar com várias situações do jogador, tendo em conta a comunicação com o servidor utilizando o protocolo implementado.

Além disso, a 'PlayerServlet' também trata de situações específicas, como quando o jogador regressa ao perfil após um jogo concluído ou quando atualiza os seus dados. Nestas situações, a servlet comunica-se com o servidor utilizando o protocolo implementado.

RegisterServlet

A 'RegisterServlet' desempenha um papel essencial na gestão do registo de novos jogadores e na atualização dos dados dos jogadores existentes. Ela interage com o ficheiro XML, utilizando o protocolo implementado, para garantir a correta manipulação e armazenamento das informações dos jogadores na base de dados.

Em relação às **JavaServer Pages (JSP)**, são uma tecnologia utilizada para desenvolver páginas web que suportam conteúdo dinâmico. Esta tecnologia permite que se insira código Java diretamente no ficheiro HTML usando tags especiais do JSP. Assim como as Servlets, as JSPs também são executadas no servidor e geradas no formato HTML para serem mostradas no navegador do cliente. A principal diferença é que, enquanto as Servlets são classes Java que geram HTML, as JSPs são páginas HTML que contêm código Java.

As tags das JSP podem ser divididas em quatro categorias principais: Diretivas, Declarações, Scriptlets e Expressões.

- **Diretivas:** As diretivas fornecem instruções para o servidor que indicam como a JSP deve ser processada. As diretivas mais comuns incluem a 'page', que define várias propriedades da página, como a linguagem de script utilizada, o tipo de conteúdo da resposta, a codificação de caracteres da página, e assim por diante.
- **Declarações:** Uma declaração pode ser usada para declarar variáveis ou métodos que serão usados mais tarde no código, são definidas entre `<%!` e `%>`.
- **Scriptlets:** Os Scriptlets permitem que sejam inseridas linhas de código Java no arquivo HTML. (`<% ... %>`)
- **Expressões:** As expressões são usadas para produzir uma única saída para o cliente. As expressões são avaliadas e convertidas em uma string, que é então inserida na posição exata em que a expressão aparece no arquivo JSP. (`<%= ... %>`)

O uso de JSP permitiu-nos separar a lógica da apresentação, o que torna o código mais limpo, mais fácil de entender e manter. As páginas JSP geram o HTML com base no estado atual do jogo, permitindo que a página seja atualizada dinamicamente conforme o jogo avança.

Jogador.jsp

JSP Jogador é responsável por implementar a interface gráfica do jogo no navegador, inclui html java e javascript de modo a fornecer uma jogabilidade interativa e intuitiva. As suas principais funções são:

1. **Armazenar informações dos jogadores**, como nome, foto, cor e senha de modo a poder exibi-los.
2. Implementar a **detecção de cliques nas células** do tabuleiro com a função 'handlePlayerMove()' onde é registrada a jogada e atualizado o estado do jogo.
3. **Controlar o tempo de jogada**, redirecionando os jogadores caso ultrapassem o limite de tempo para fazer uma jogada com a função 'startTimer()' que inicia um temporizador de 30 segundos para o jogador fazer sua jogada.

Enviar as jogadas para o servidor usando um formulário HTML e o método POST.

Perfil.jsp

O JSP "Player Profile" é responsável por mostrar o perfil privado do jogador, permite que os jogadores visualizem e atualizem as suas informações pessoais e estatísticas de jogo, incluindo vitórias, empates, derrotas e tempo total de jogo.

1. Por meio de scripts JavaScript, o JSP realiza ações como **carregar a imagem de perfil** selecionada, **atualizar a cor de fundo da página**, calcular a idade **a partir da data de nascimento** e inicializar o perfil com os dados do jogador.

meuJogador.java

A classe atua como intermediária entre o cliente web e o servidor, ou seja é responsável por gerir a comunicação entre o jogador que utiliza um navegador e o servidor utilizando o protocolo estabelecido anteriormente. A comunicação é realizada por utilizando strings, da mesma forma que na primeira parte do projeto.

A classe possui vários métodos que são responsáveis por ler e enviar diferentes tipos de mensagens possíveis, desempenha um papel fundamental na troca de informações e define a estrutura e a lógica para processar mensagens recebidas do cliente e enviar mensagens de volta para o cliente.

Conclusões

Ao longo da disciplina de Infraestruturas Computacionais Distribuídas tivemos a oportunidade de aprofundar e aplicar os conhecimentos em arquiteturas e protocolos de comunicação. Na primeira parte do projeto, focamos na consolidação desses conhecimentos, o que nos permitiu criar uma base sólida para que fosse possível a expansibilidade para uma aplicação web.

Na segunda parte do projeto, tivemos a oportunidade de aplicar conhecimentos sobre as tecnologias de desenvolvimento para a World Wide Web, com especial foco em JavaServer Pages (JSP). Nesta fase, desenvolvemos uma aplicação web que envolveu funcionalidades como o login, registo e alteração de perfil de jogador, procura de oponentes pelo nome, gestão de vários jogos simultâneos, e apresentação de um quadro de honra.

Concluimos que o uso de JSP e Servlets, ambos baseados em Java, proporcionam uma excelente heterogeneidade, pois o Java é uma linguagem de programação independente de plataforma, o que significa que as aplicações que utilizamos podem ser desenvolvidas e implementadas em diferentes tipos de sistemas operativos e arquiteturas de hardware. Esta característica favorece a interoperabilidade entre diferentes sistemas.

A expansibilidade da aplicação também foi uma consideração importante, com o uso de JSP e Servlets permite-nos um desenvolvimento expansível e em termos de concorrência, fomos capazes de lidar com vários jogadores simultaneamente.

Entendemos que, embora a nossa implementação seja funcional e eficaz para a finalidade proposta, há sempre espaço para otimizações e melhorias. Uma área que identificamos para potencial melhoria é o nosso uso de JavaScript.

Na nossa implementação atual, utilizamos uma quantidade significativa de JavaScript que apesar de ser uma ferramenta poderosa para tornar as páginas web mais dinâmicas e interativas, o tem a característica de ser uma linguagem de script do lado do cliente, o que significa que o nosso código é completamente exposto aos utilizadores. Isto pode representar um risco de segurança, pois pode ser visualizado por utilizadores mal-intencionados. Estamos cientes dessas possíveis melhorias e consideraremos a sua implementação em trabalhos futuros.

Em conclusão, a combinação de JSP e Servlets forneceu uma base sólida para a construção de um sistema distribuído robusto e eficiente. Ao longo do projeto, fomos capazes de resolver os desafios propostos e implementar uma solução que consideramos eficiente e capaz de lidar com a concorrência. A experiência reforçou a nossa compreensão das vantagens e desvantagens dessas tecnologias e foi essencial para o nosso futuro como engenheiros.

Bibliografia

Para realizar este projeto foi essencial a consulta dos materiais pedagógicos disponibilizados pelo nosso docente, os quais foram indispensáveis para a consolidação dos nossos conhecimentos teóricos. Os diapositivos mais imprescindíveis foram os seguintes:

- Diapositivos "09-WEB-Servlets.pdf"
- Diapositivos "10-WEB-JSPs.pdf"

Além disso, os exemplos de projetos compartilhados através da plataforma Moodle foram extremamente valiosos. Esses exemplos foram cruciais para a nossa compreensão e implementação prática dos conceitos abordados em aula, fornecendo uma base sólida sobre a qual construímos o nosso código.