

Eng. Informática	3º Ano	1º Semestre	2016-17	Av. Periódica
2º Projeto	Data Limite Divulgação Resultados: 25 Jan 2017			
Data: 3 Nov 2016	Data Entrega: 9 Jan 2016			

Projeto II – Batalha Naval

OBJECTIVO

O objetivo deste projeto consiste em implementar o jogo da Batalha Naval utilizando **Angular**, **NodeJS**, **MongoDB** e **WebSockets**.

JOGO DA BATALHA NAVAL ORIGINAL

A Batalha Naval é um jogo de tabuleiro, originalmente desenvolvido para dois jogadores, cujo o objetivo consiste em descobrir em que posições, ou células, estão localizados os navios do oponente.

O jogo original é jogado em dois tabuleiros de 10x10 – um dos tabuleiros é utilizado para representar os navios do jogador (tabuleiro de defesa), o outro é utilizado para marcar as posições descobertas no tabuleiro do oponente (tabuleiro de ataque). Os tabuleiros são identificados horizontalmente por números e verticalmente por letras. No início do jogo, cada jogador coloca os seus navios, dispondo-os horizontalmente ou verticalmente (não é permitido colocar navios em diagonais), sendo que os navios não se podem sobrepor ou tocar. Ambos os jogadores dispõem do mesmo número de navios: 1 porta-aviões, 1 couraçado, 2 cruzadores, 3 contratorpedeiros e 4 submarinos.

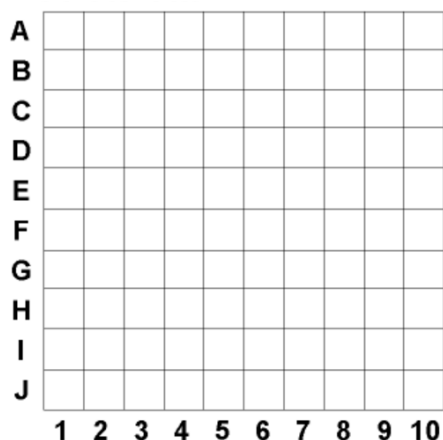


Figura 1: Tabuleiro de jogo

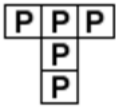



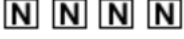
	1 Porta-Aviões
	1 couraçado (navio de 4 células)
	2 cruzadores (navios de 3 células)
	3 contratorpedeiros (navios de 2 células)
	4 submarinos (navios de 1 célula)

Tabela 1: Peças do jogo

O jogo é jogado por turnos. Cada jogador pode “disparar” três tiros contra o tabuleiro do adversário. No final dos três tiros, o adversário deve informar se acertou em algum navio e, em caso afirmativo, em que tipo de navio acertou. Os tiros certos devem ser assinalados como “Tiro”, os restantes deverão ser assinalados como “Água”. O jogador deve então marcar no seu tabuleiro de ataque as posições escolhidas, distinguindo os tiros certos dos tiros em água, de forma a poder adivinhar onde se encontram as restantes posições dos navios por descobrir. Se conseguir afundar um navio, deverá ser informado do mesmo. Ganha o jogador que conseguir afundar todos os navios do adversário primeiro.

CENÁRIO

O jogo a implementar é baseado no jogo da Batalha Naval descrito na secção anterior, mas com algumas diferenças consideráveis, nomeadamente:

- O jogo deve poder ser jogado por vários jogadores (mínimo dois e máximo quatro), todos contra todos, sendo que cada jogador deverá ter disponível um tabuleiro de defesa e um tabuleiro de ataque por cada oponente;
- O número de tiros por turno deverá ser igual a 2 vezes o número de oponentes;
- Em cada turno, cada jogador pode disparar contra qualquer um dos oponentes (e.g. disparar dois tiros para o primeiro adversário e quatro para o terceiro, ou disparar todos os tiros para o primeiro adversário);
- Os tiros certos devem ser marcados no tabuleiro, e deverão ser reportados através de texto, indicando a posição e qual o tipo de navio atingido (e.g. Posição A3 - Tiro no Porta-Aviões);
- Os tiros em água devem também ser marcados no tabuleiro (e.g. com uma cruz);
- Todos os jogadores devem visualizar todos os tiros, certos ou na água, mesmo que não estejam envolvidos na jogada. Esta informação deverá estar claramente visível na área do jogo;
- Sempre que um navio é afundado deve ser destacado no tabuleiro de jogo (e.g. destacar limites do navio ou todo o navio a uma cor diferente), e deverá ser reportado através de texto, qual o tipo de navio afundado e quem o afundou (e.g. Porta-Aviões afundado pelo “João Silva”);
- Ganha o jogador que mantenha pelo menos uma parte de um dos seus navios por afundar.

TOP TEN DE VITÓRIAS E DE PONTUAÇÕES

A aplicação deverá ter um *Top 10* com os jogadores com mais vitórias e um *Top 10* com os jogadores mais pontuados. Para calcular a pontuação de cada jogador num jogo, deve considerar o seguinte:

- Cada tiro certo vale 1 ponto;
- Cada navio afundado (último tiro certo) vale:
 - Porta-aviões: 20 pontos;
 - Couraçado (navio de 4 células): 15 pontos;
 - Cruzador (navio de 3 células): 10 pontos;
 - Contratorpedeiro (navio de 2 células): 5 pontos;
 - Submarino (navio de 1 célula): 3 pontos;
- O jogador vencedor recebe 50 pontos.

IDENTIFICAÇÃO E AUTENTICAÇÃO

Só os utilizadores registados e autenticados poderão criar, entrar ou jogar jogos. Cada jogo, bem como os recursos associados ao jogo (e.g. chat), só deverá ser acessível aos jogadores do mesmo.

A aplicação deverá permitir o registo de utilizadores com, pelo menos, a seguinte informação:

- Nome de utilizador – um nome único que será utilizado para fazer a autenticação e para representar o utilizador durante o jogo;
- Senha – deverá ter pelo menos 3 caracteres;
- Confirmação da senha – deverá ser igual à senha;
- E-mail – o utilizador deve inserir um e-mail válido.

HISTÓRICO DE JOGOS

A aplicação deverá manter o histórico de todos os jogos jogados numa base de dados, que deverá conter no mínimo a seguinte informação:

- Criador – jogador responsável pela criação do jogo;
- Terminou – indica se o jogo terminou ou não (um jogo pode não terminar se todos os jogadores desistirem ou se houver um problema técnico que impeça a conclusão do jogo);
- Data Início – data e hora do início do jogo;
- Data Fim – data e hora do fim do jogo;
- Vencedor – jogador que venceu o jogo;
- Jogadores – lista de jogadores que participaram no jogo. Para cada jogador deverá ser possível saber a sua classificação e pontuação no jogo.

Todos os utilizadores (incluindo os utilizadores anónimos) deverão poder pesquisar e consultar o histórico dos jogos (toda a informação referida anteriormente). Para além disso, os utilizadores registados deverão também poder consultar o histórico dos seus jogos (pesquisar e consultar histórico dos jogos, mas limitando o espaço de pesquisa aos jogos em que participou).

Para além de apresentar o histórico dos jogos sem tratamento especial (através de listas simples com os dados), a aplicação deverá trabalhar os dados para obter informação estatística relevante (e.g. média de jogos por dia; média de jogos por jogador; jogador que jogou mais jogos; etc.) e deverá apresentar a informação de forma mais apelativa possível (e.g. gráficos; resumos, etc.).

GAME LOBBY

Esta área permite a um utilizador criar ou entrar (juntar-se) num jogo pendente (por iniciar). Deverá apresentar uma lista de jogos pendentes, indicando o nome do utilizador que o criou e o número de jogadores já em espera. Quando um jogo começa, deixa de estar pendente e é retirado da lista. O jogo deve ter no mínimo 2 jogadores e no máximo 4, sendo um deles o utilizador que o criou. O início do jogo é da exclusiva responsabilidade do seu criador, podendo fazê-lo a qualquer altura desde que haja pelo menos dois jogadores.

JOGOS MÚLTIPLOS

Esta funcionalidade deve permitir que cada jogador possa jogar mais do que um jogo na mesma página Web. Isto significa que o jogador pode visualizar e interagir com vários jogos, todos a decorrer em simultâneo e de forma independente.

OUTRAS FUNCIONALIDADES

O projeto deve incluir outras funcionalidades que lhe acrescentem valor e que deverão ser definidas pelos estudantes. A avaliação destas funcionalidades depende da complexidade e quantidade das funcionalidades implementadas. Cada funcionalidade é associada a um nível de complexidade com valores de 1 a 4 (1 – mais simples; 4 – mais complexo). Para obter a nota máxima no critério “Outras Funcionalidades”, os estudantes deverão implementar na perfeição (nota=100%) um conjunto de funcionalidades cujo nível total de complexidade seja igual a 4, ou por exemplo, ter uma nota de 50% em todas as funcionalidades de um conjunto com um nível total de complexidade igual a 8.

Exemplo de cálculo:

Funcionalidades implementadas: f1; f2; f3; f4

Níveis de complexidade: f1 = 1; f2 = 2; f3 = 1; f4 = 3

Notas individuais: f1 = 60%; f2 = 30%; f3 = 100%; f4 = 40%

Nota Final: $(60\% * 1 + 30\% * 2 + 100\% * 1 + 40\% * 3) / 4 = 85\%$

De seguida, são apresentadas algumas sugestões de funcionalidades possíveis associadas a níveis de complexidade. Os níveis de complexidade servem apenas de referência e podem variar de acordo com a funcionalidade em concreto.

BOTS (Nível complexidade = 3)

Permitir jogar contra jogadores não humanos (*bots*). Os *bots* poderão, ou não, ser inteligentes (com possível aumento do nível de complexidade para 4). Caso esta funcionalidade seja implementada, deverá ser possível ao criador do jogo adicionar *bots* enquanto o jogo está pendente. Para todos os efeitos, um *bot* substituirá um jogador humano - conta para o número total de jogadores e não deverá afetar a experiência de jogo dos restantes jogadores.

Nota: mesmo nos casos em que o criador do jogo jogue apenas contra *bots*, o jogo deverá manter a arquitetura e comportamento *multiplayer*.

SERVIÇOS DE AUTENTICAÇÃO EXTERNOS (Nível complexidade = 2)

Permitir que a autenticação seja feita por outros serviços, como por exemplo *Facebook Login for the Web*, *Google Identity Platform*, *Twitter Sign In*, etc. Alguma informação sobre estes serviços:

- <https://developers.facebook.com/docs/facebook-login/web>
- <https://developers.google.com/identity/>
- <https://developers.google.com/identity/sign-in/web/sign-in>
- <https://dev.twitter.com/web/sign-in>

AVATARES (Nível complexidade = 1)

Permitir que os utilizadores possam ser representados não apenas pelo seu nome, mas também por um avatar (imagem ou foto).

CHAT ROOMS (Nível complexidade = 1)

Permitir que os utilizadores comuniquem entre si através de salas de *chat*. Poderá existir uma sala de *chat* pública no *lobby* (acessível a todos os utilizadores, incluindo convidados) e uma sala de *chat* privada por cada jogo (acessível apenas aos jogadores desse jogo).

REPETIÇÃO DE JOGO (Nível complexidade = 4)

Permite visualizar uma animação que mostre todas as jogadas de um jogo que já decorreu. Deverá ser perceptível quem é o responsável por cada jogada.

MOBILE APP (Nível complexidade = 4)

Permite aos utilizadores jogar um jogo num cliente *mobile*. Esta funcionalidade não se pode limitar ao uso de desenho responsivo na página Web, implicando sempre a utilização de uma *framework* multiplataforma.

PROJETO

Além dos requisitos funcionais especificados no “Cenário”, o projeto deve obedecer a algumas restrições tecnológicas e requisitos não-funcionais relacionados com a estrutura da aplicação e do código, assim como com a interação entre o cliente Web e o servidor.

RESTRIÇÕES TECNOLÓGICAS

A aplicação cliente deve utilizar a *framework* Angular 2. O servidor da aplicação deve ser implementado utilizando NodeJS, WebSockets e MongoDB. Outras tecnologias, bibliotecas ou *frameworks*, desde que complementares às anteriores, poderão ser utilizadas no projeto.

REQUISITOS NÃO-FUNCIONAIS

Devem ser considerados alguns requisitos não-funcionais, nomeadamente:

- A arquitetura da aplicação deve seguir as boas práticas de *design* e as tecnologias utilizadas para cada um dos seus componentes devem ser escolhidas de forma adequada, de acordo com a arquitetura e o contexto de utilização;
- A estrutura e o código da aplicação devem seguir o princípio da separação de responsabilidades, onde diferentes unidades de software (ficheiros, módulos, objetos, funções, etc.) devem ser o mais independente e desacopladas possível;
- O desempenho da aplicação deve ser otimizado. Por exemplo, os movimentos nos tabuleiros de jogo e as mensagens de *chat* devem ser propagados quase instantaneamente para todas as aplicações cliente;
- A aplicação deve ser o mais segura possível, assegurando não apenas que os recursos e características funcionais estão disponíveis apenas para os utilizadores apropriados, mas também que todos os movimentos no jogo feitos por um jogador são mesmo executados por esse jogador – elementos externos não devem ser capazes de interferir com o jogo;
- A aplicação não deve permitir que jogadores, ou outras entidades, possam enganar o jogo analisando o seu estado interno no cliente, ou por qualquer outro meio.

ENTREGA E PUBLICAÇÃO

A aplicação deverá ser publicada num serviço público acessível a partir de qualquer dispositivo na Internet. Os estudantes são livres de escolher o serviço a utilizar, no entanto são apresentadas algumas sugestões:

- Amazon Free Tier (<https://aws.amazon.com/free/>)
- Heroku (<https://devcenter.heroku.com/articles/mean-apps-restful-api>)
- Digital Ocean com o GitHub Education Pack (<https://education.github.com/pack>)
- Openshift da RedHat (<https://www.openshift.com/>)

Para além disso, todo o código fonte do projeto deve ser entregue num **ficheiro zip**, utilizando o *link* disponibilizado na página da Unidade Curricular no Moodle. O ficheiro zip deve conter:

- **info.txt**: ficheiro com informação sobre o grupo de projeto, nomeadamente, número de grupo de projeto, nome e número dos estudantes, turno Prático-Laboratorial, URL de acesso à aplicação publicada e credenciais para teste da aplicação.

Este ficheiro deve incluir também informação sobre quais as funcionalidades implementadas. Esta informação é obrigatória e é utilizada para auxiliar a correção do projeto. Se uma determinada funcionalidade não for registada como “OK” ou “PARCIAL”, será avaliada a 0%. A informação deve assumir a seguinte forma (exemplo indicativo):

```
Jogo - OK
Top Ten - PARCIAL
Identificação e Autorização - OK
Histórico de Jogos - NÃO implementado
Game Lobby - OK
Jogos Múltiplos - NÃO implementado
Outras Funcionalidades - Bots; Avatares; Vídeo em Real-time
dos jogadores (webcams); Chat Rooms
```

- **Pasta “client”**: Pasta que contém todos os ficheiros necessários para correr o projeto no cliente, com código fonte legível.
- **Pasta “server”**: Pasta que contém todos os ficheiros necessários para correr o projeto no servidor, com código fonte legível.
- **Pasta “Dump”**: Pasta criada pelo utilitário mongodump com o backup da base de dados mongoDB.

AVALIAÇÃO

A tabela seguinte sumaria os critérios de correção para avaliação do projeto. Cada critério será avaliado do ponto de vista de utilização da aplicação (funcionalidades, usabilidade, fiabilidade, desempenho, segurança, etc) e de implementação interna (arquitetura, padrões de desenho, qualidade do código, etc)

Nº	Peso	Critério
1	80%	Aplicação Resultante <i>Jogo, Top Ten, Identificação e autorização, Histórico, Lobby, Jogos Múltiplos</i>
2	15 %	Outras Funcionalidades
3	5 %	Publicação