Desenvolvimento de Aplicações Distribuídas

Eng. Informática	3º Ano	1º Semestre	2016-17	Exame Normal Prático
Projeto	Data Limite Divulgação Resultados: 7 Fev 2017			
Data: 10 Jan 2017	**Data Entrega: 31 Jan 2017**			

Projeto – Exame Normal

OBJECTIVO

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

JOGO DA SUECA

O jogo da sueca é um jogo para 4 jogadores em duplas, ou seja, é jogado em equipas de dois jogadores com os parceiros dispostos em lugares opostos (de frente um para o outro). Apesar de existirem várias variantes do jogo, pretende-se com este projeto implementar um jogo da sueca que siga as regras descritas neste enunciado.

O baralho a utilizar deve ter 40 cartas, sendo removidos os 8s, 9s, 10s e jokers existentes nos baralhos padrão. Desta forma, as cartas a utilizar serão: ás, 7s, rei, valete, rainha, 6s, 5s, 4s, 3s e 2s.

O objetivo do jogo é ganhar cartas que valem pontos, de forma a atingir mais de 60 pontos. No total, um baralho tem 120 pontos e o valor das cartas é:

- Ás: 11 pontos;
- 7: 10 pontos;
- rei: 4 pontos;
- valete: 3 pontos;
- rainha: 2 pontos;
- 6, 5, 4, 3, 2: 0 pontos.

Sendo assim, a equipa que contabilize mais do que 60 pontos de acordo com as pontuações anteriormente exibidas, vence o jogo. Caso ambas as equipas tenham 60 pontos, considera-se que há um empate.

O jogo joga-se no sentido horário e cada jogador terá 10 cartas para jogar. As cartas são baralhadas e distribuídas no sentido horário. O primeiro jogador a receber jogo será escolhido aleatoriamente e a primeira carta que recebe deverá ficar visível para todos os jogadores, uma vez

que o seu naipe será o trunfo. As restantes 9 cartas só deverão ficar visíveis para o jogador. De seguida, são distribuídas 10 cartas aos restantes jogadores, no sentido dos ponteiros do relógio, devendo as mesmas ficar visíveis apenas ao respetivo jogador.

O jogo começa a ser jogado pelo jogador que está imediatamente à esquerda do jogador que recebeu primeiro a sua mão. Os outros jogadores devem seguir o naipe jogado pelo primeiro jogador. Caso um jogador não tenha cartas desse naipe, pode jogar uma carta de qualquer outro naipe, incluindo o trunfo. A carta mais alta do naipe em jogo, ou o trunfo mais alto (o trunfo mais alto ganha sempre aos restantes naipes), ganha a rodada. O jogador que ganhou será o primeiro a jogar a próxima jogada, escolhendo o naipe. Caso um jogador minta sobre a ausência de um naipe na sua mão, e seja detetado, estará a fazer "renúncia" ao jogo. Neste caso, a dupla oponente ganha o jogo automaticamente.

CENÁRIO

O jogo a implementar é baseado no jogo da Sueca descrito na secção anterior, mas com algumas considerações adicionais, nomeadamente:

- O jogo deve poder ser jogado por quatro jogadores, dois contra dois, sendo que cada jogador deverá poder ver o seu jogo e as cartas jogadas pelo parceiro e pelos oponentes em cada ronda;
- A escolha da equipa pode ser feita de forma aleatória ou, em alternativa, poderá ser feita por escolha dos participantes antes do início do jogo;
- Pode haver comunicação entre todos os jogadores (por exemplo, através de um chat), mas não é permitida nenhuma comunicação exclusiva entre os parceiros de uma equipa num jogo;
- A pontuação de uma equipa num jogo corresponde ao total de pontos obtidos pelos dois elementos da equipa;
- Para além da pontuação de cada jogo, a aplicação deverá implementar um sistema de pontuação global em que no fim de cada jogo serão atribuídas estrelas aos jogadores conforme a pontuação da sua equipa no jogo, e de acordo com as seguintes regras:
    - 5 estrelas se a equipa ganhar todos os pontos do jogo (120);
    - 3 estrelas se a equipa ganhar 91 ou mais pontos (91 a 119);
    - 2 estrelas se a equipa ganhar com 61 pontos (61 a 90);
    - 1 estrela para as duas equipas em caso de empate (60 pontos);
    - 0 estrelas se a equipa perder o jogo;
- Caso um jogador de uma equipa detete que a equipa oponente fez "renúncia" pode clicar num botão para a denunciar. Caso a "renúncia" se verifique, a equipa que a fez perde o jogo e a equipa oponente ganha 5 estrelas. Caso a "renúncia" não tenha ocorrido, a equipa

que denunciou a renúncia perde e a equipa oponente ganha imediatamente o jogo e 5 estrelas.

TOP TEN DE VITÓRIAS E DE PONTUAÇÕES

A aplicação deverá ter um *Top 10* de estrelas e um *Top 10* de pontos. O *Top 10* de estrelas, deverá contabilizar o total de estrelas relativos ao sistema de pontuação global. O *Top 10* de pontos, deverá contabilizar todos os pontos (relativos às cartas) obtidos em todos os jogos.

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;
- Vencedor1 – jogador 1 da equipa que venceu o jogo;
- Vencedor2 – jogador 2 da equipa que venceu o jogo;
- Jogadores – lista de jogadores que participaram no jogo. Para cada jogador deverá ser possível saber a pontuação no jogo e o total de estrelas obtidas.

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 4 jogadores para começar sendo um deles o utilizador que o criou. O jogo deverá iniciar logo que tenha os 4 jogadores presentes.

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; f 4
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). Atenção que, durante um jogo, a comunicação deverá ser visível a todos os jogadores, de forma a que seja possível verificar que os jogadores de uma mesma equipa, não trocam informações sobre o jogo.

REPETIÇÃO DE JOGO (*Nível complexidade = 4*)

Permite visualizar uma animação que mostre todas as jogadas de um jogo que já decorreu. Deverá ser percetível 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 *Jogo, Top Ten, Identificação e autorização, Histórico, Lobby, Jogos Múltiplos*
2	**15 %**	Outras Funcionalidades
3	**5 %**	Publicação