2048 Co-Operativo

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Sistemas Distribuídos 2014

Sdis, Turma 6, Grupo 5
Carlos Matias - ei11153
Diogo Vaz - ei11065
Pedro Silva - ei11061
Rui Grandão - ei11010

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

1 de Junho de 2014

Indice

1 Introdução

O principal objetivo deste trabalho seria implementar uma versão do popular jogo 2048, sendo que nesta versão qualquer utilizador que aceda so site estará a jogar a mesma versão do jogo com outros que também lá se encontram, criando assim uma sala de jogo onde todos jogam em conjunto. A jogabilidade é, na integra, igual à do jogo original (2048 Original), com uma modificação, dois tipos de jogo diferentes. Assim como o jogo original, a versão implementada pelo grupo está presente também em Web.

O projeto apresentado é muito diferente daquele referido na especificação. O grupo conclui que uma versão Co-Operativa do jogo em vez de uma de Batalha seria mais interessante, não só porque já existia uma versão do jogo parecida à proposta, mas também porque ter vários jogadores a jogar no mesmo tabuleiro pode originar jogadas interessantes. O grupo não fez uso das tecnologias *libGDX*, as quais froam referidas na especificação do projeto.

Este relatório começa por introduzir os funcionalidades da aplicação, de seguida as bibliotecas utilizadas no projeto, seguido da descrição da implementação dividida pelas secções do código de jogo, comunicação e o serviço de *login*. O relatório termina com um pequeno exemplo de utilização e com a conclusão em relação ao projeto feito.

2 Especificação do Projeto

Como referido na introdução, o grupo implementou o jogo 2048 numa versão cooperativa, através de uma aplicação Web. Esta aplicação apresenta uma página inicial, que apresenta alguma informação sobre o projeto e permite ao utilizador registar-se ou fazer *login* na aplicação. O registo pode ser feito de uma maneira tradicional ou via a API do *Facebook*. De referir que, para conseguir que o Login fosse acessível a qualquer utilizador, o *Facebook* teria que aprovar a aplicação e isso envolveria fornecer dados privados do servidor, algo que o grupo não estaria disposto, num projeto deste ambiente, a realizar.

Uma vez na página inicial, estando o utilizador registado, ou não, e carregando no botão para jogar, o utilizador entrará na sala de jogo. Como referido o jogo é mecânicamente igual ao 2048 mas com dois modos de jogo.

Estes modos de jogo são o modo Anarquia e o modo Democracia. No primeiro modo qualquer comando introduzido por um utilizador é executado, enquanto no segundo modo existe um período de cinco segundos entre cada execução de movimento. Nesse período existe uma votação para o movimento a seguir, ou seja, durante cinco segundos os utilizadores introduzem movimentos e, ao fim desses cinco segundos, é executado o movimento com maior votação. A escolha do modo jogo é feito em tempo real sendo possível ao utilizador votar no modo que quer jogar, a cada dez segundos o modo com mais votos será aplicado.

O ecrã de jogo, assim como o ecrã do 2048 original, apresenta ainda dois indicadores de pontuação, um para a atual e outro para a melhor de sempre. Existe ainda um botão de novo jogo que é clicável sempre que um jogo termina, quer seja vitória ou derrota.

3 Desenvolvimento

Nesta secção do relatório estão descritas as bibliotecas usadas pelo grupo e os motivos que as levaram a ser utilizadas.

3.1 Bibliotecas Utilizadas

3.1.1 Twitter Bootstrap 3

Para permitir que o aspeto visual da aplicação Web não ocupasse grande parte do tempo de desenvolvimento do projeto, sem, no entanto, descurar o relevo que um site visualmente apelativo possui para o utilizador, e permitindo o foco nas funcionalidades em rede do projeto, o grupo optou por utilizar a biblioteca *Bootstrap 3*, fornecida pelo *Twitter*. Esta biblioteca fornece uma série de elementos de *HTML* e *CSS*, assim como funcionalidades *Javascript* para esses mesmos elementos.

3.1.2 Node.js

A plataforma *Node.js* permite correr código *Javascript* do lado do servidor de forma assíncrona. Esta plataforma facilita a comunicação em tempo real, a qual é um requesito necessário da nossa aplicação, e é um requesito na biblioteca de *Sockets* que a aplicação usa. Estes levaram então à implementação de *Node.js* no projeto.

3.1.3 Socket.io

Socket.io é uma biblioteca de *Sockets* para *Node.js* que permite, com facilidade, implementar trocas de mensagens em tempo real e mensagens de *broadcast*. A biblioteca permite ainda correr *Websockets* em browsers mais antigos usando *Flash* e outros métodos permitindo uma comunicação mais fiável e maior compatibilidade.

3.1.4 Restify

A biblioteca *Restify* permite uma implementação simples de um servidor emphRESTful em *Node.js*. Esta biblioteca foi utilizada para permitir a partilha de variáveis entre os dois servidores implementados, algo que não aconteceria se fosse usado um tradicional servidor *PHP*.

3.2 Implementação

Nesta secção do relatório é descrita a forma como o grupo implementou cada uma das fases relevantes do desenvolvimento do projeto.

3.2.1 Jogo

Para evitar ter que desenvolver um jogo já implementado, o grupo utilizou o código fonte do jogo feito por Gabriele Cirulli disponível no *Github*. A utilização que fizemos desse código encontra-se ao abrigo da licença do MIT do jogo. Para o jogo funcionar foram introduzidas algumas alterações no ficheiro (original) *gameManager.js*, nomeadamente a inicialização do *Socket* e dos seus *handlers*, e modificações na função que executa movimentos.

De forma a melhorar a experiência do utilizador foram também adicionados botões para votar no estado de jogo, um *log* dos movimentos mais recentes, e um contador de votos e de votos de movimentos caso o modo seja democracia. Todos estes objetos são atualizados em tempo real.

3.2.2 Comunicação

Como era necessário usar um servidor *RESTful* e como era também importante usar comunicação em tempo real, célere e bidirecional, para ter uma aplicação fluida, foi feita a escolha de usar dois tipos de servidores, ambos implementados em *Node.js*, com a diferença de um usar a *framework Restify* e o outro a *framework* de *Sockets Socket.io*.

3.2.3 Servidor Socket.io

Neste servidor é feita a comunicação, em tempo real, dos movimentos dos utilizadores ao servidor, o início de um novo jogo, a contagem dos votos e a contagem dos votos de movimentos caso o modo seja a democracia.

Quando o servidor recebe uma mensagem dum pedido de movimento é verificado se o *timestamp* do pedido, que é criado pelo cliente quando este é gerado, é inferior ao último movimento efetuado. Esta verificação é efetuada para garantir que o movimento que o utilizador fez é efetuado quando ele tem a versão em que o movimento será aplicado, ou seja, um movimento criado antes de outro, mas que é recebido pelo servidor num momento posterior, é ignorado. É também verificado se o cliente que pretende executar o movimento fez algum movimento nos últimos 100 milissegundos. Se o tiver feito o utilizador é bloqueado por 300 milissegundos para prevenir que alguém torne o jogo impossível de jogar estando sempre a fazer movimentos.

Caso o pedido de movimento seja validado é enviada uma mensagem para todos os utilizadores conectados, incluindo o que enviou o pedido, para executarem o movimento. Se o modo for democracia a diferença será que o voto não irá resultar num movimento mas num voto a favor do pedido de movimento efetuado sendo a execução feita por uma função periódica tendo em conta os votos.

O voto no modo atual do jogo é também feito através de uma mensagem ao servidor, a única validação feita é se o utilizador votou no último segundo, se o tiver feito o voto não é contabilizado.

3.2.4 Servidor Restify

Este servidor implementa pedidos *GET* e *PUT* no *url gameState*. Fazendo um pedido *GET*, o utilizador recebe o estado de jogo atual ou *404*, se não existir estado de jogo. Este pedido é feito na inicialização do cliente, para começar no estado de jogo correto, e periodicamente para verificar se houve alguma dessincronização. O pedido *PUT* é feito pelos clientes para atualizar o estado do jogo quando é feito um movimento.

3.2.5 Login

À partida, e para evitar a utilização de ferramentas diferentes, o grupo tentou implementar o *login* na aplicação usando *Javascript* e *Node.js*. O fato desta comunicação ser assíncrona criou alguns problemas na comunicação o que inviabilizou a sua realização no que toca a inquirir a base de dados. Para conseguir na mesma implementar uma funcionalidade de *login* o grupo optou então por tecnologias mais familiares como o *Php*.

4 Exemplos de Utilização

A aplicação desenvolvida tem uma interface simplista e funcional. Nesta secção o grupo apresenta alguns *Screenshots* da aplicação e, através dos mesmos, indica como interagir com a aplicação.

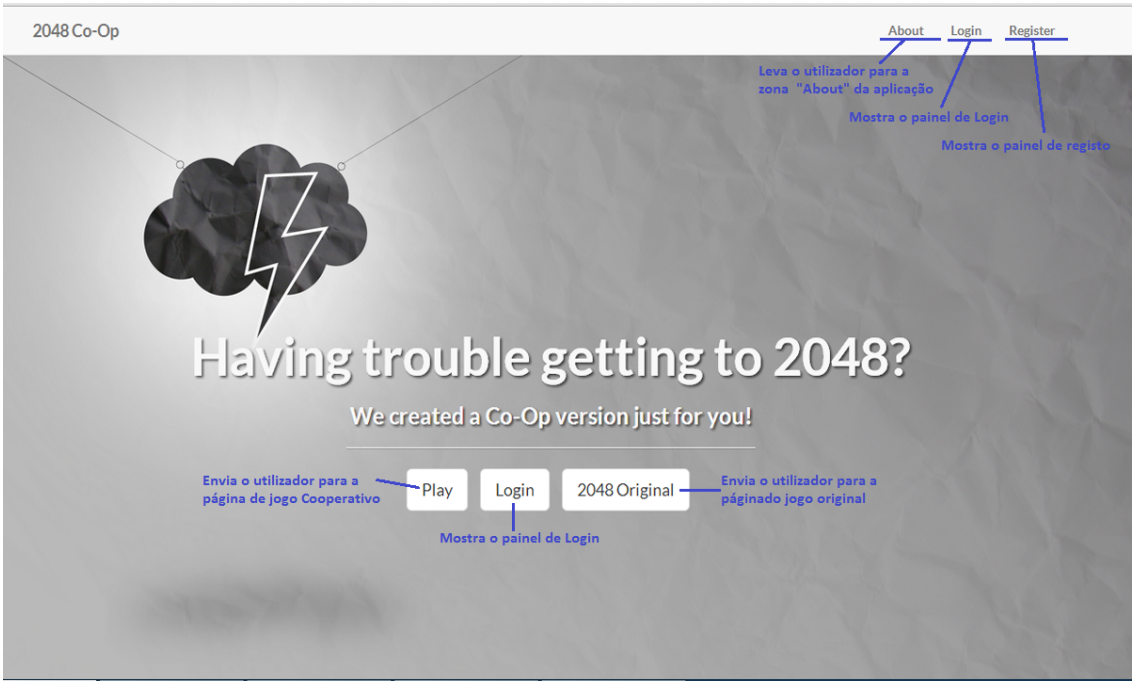

Figura 1: Página inicial da aplicação.

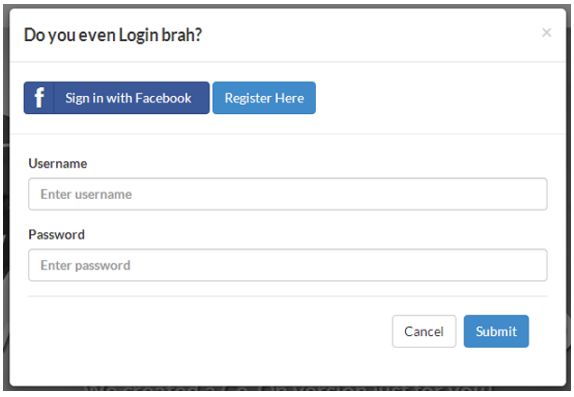

Figura 2: Painel de *login* da aplicação.

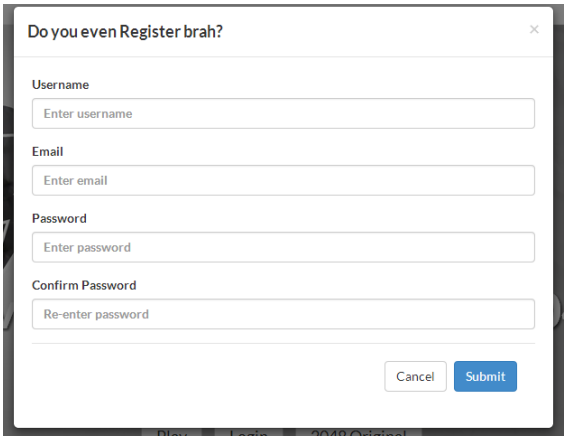

Figura 3: Painel registo da aplicação.

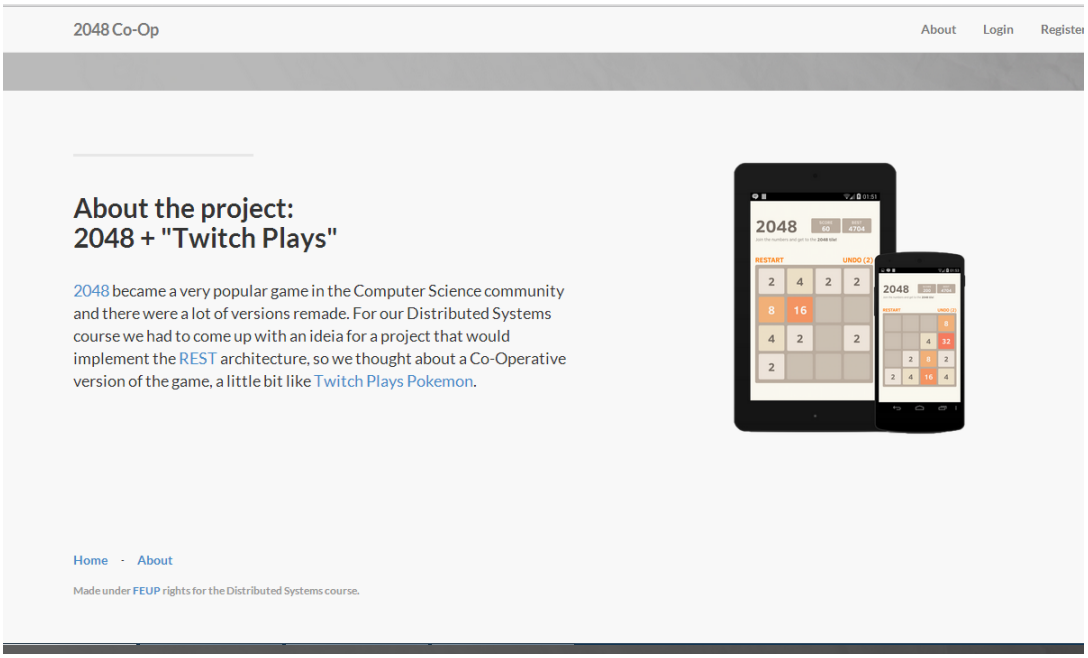

Figura 4: Zona "Sobre a aplicação"da aplicação.

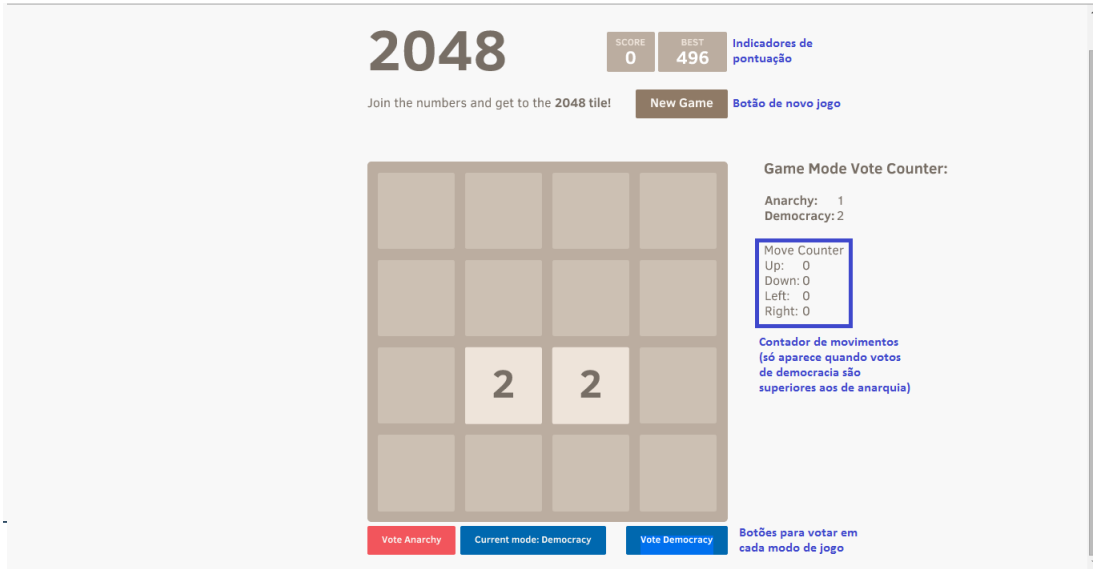

Figura 5: Página de jogo da aplicação.

5 Conclusão e Comentários

Antes de iniciar a conclusão, seria de relevo referir que a liberdade de escolha de um tema por parte dos docentes da unidade curricular, restringindo apenas alguns fatores de desenvolvimento (neste caso a arquitetura *REST*) dá aso a, não só uma maior satisfação com o tema por parte dos estudante, e, derivada desta satisfação, uma maior motivação, como também dá asas à criatividade dos estudantes, podendo deste projeto sair resultados mais interessantes.

Após o desenvolvimento podemos concluir que com este projeto o grupo teve uma maior exposição aos problemas da comunicação em tempo real e as várias maneiras de os remediar. O grupo teve também uma primeira experiência de desenvolvimento web em tempo real com *Node.js* e a biblioteca *Socket.io*, ferramentas que podem ser relevantes para futuros projetos. Em relação ao produto final o grupo concluiu que, tendo em conta o objetivo final, tivemos sucesso na nossa implementação, o jogo encontra-se jogável e com uma interface funcional e, na opinião do grupo, apelativa ao utilizador.

6 Bibliografia

- Twitter Bootstrap 3
- Landing-Page Template
- Node.js
- Socket.io
- Restify
- 2048
- Facebook Javascript SDK