2048 CO-OP

Universidade do Porto

Faculdade de Engenharia

FEUP

Rui Grandão - ei11010
Pedro Silva - ei11061
Carlos Matias - ei11153
Diogo Nunes - ei11065

Introdução

O principal objetivo deste trabalho consiste em implementar uma versão do popular jogo *2048* sendo que nesta versão qualquer pessoa que ligue ao *url* disponibilizado estará a jogar a mesma versão do jogo criando assim uma sala de jogo onde todos jogam em conjunto.

Existem dois modos de jogo, a anarquia e a democracia. No primeiro modo qualquer comando introduzido por um dos utilizadores é executado enquanto no segundo modo existe um período de cinco segundos entre cada execução de movimento e nesse período existe uma votação para o movimento a seguir. 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.

Este relatório começa por introduzir as bibliotecas utilizadas no projeto, seguido da descrição da implementação dividida pelas secções do código de jogo, comunicação e o serviço de login e no fim é feita a conclusão em relação ao projeto feito.

Bibliotecas utilizadas

Node.js

Node.js permite correr código *javascript* do lado do servidor de forma assíncrona. Os benefícios que trazia para comunicação em tempo real e o facto de ser um requisito para a biblioteca de *sockets* que escolhemos usar levaram-nos a usa-la.

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.

Restify

A biblioteca permite uma implementação simples de um servidor *RESTful* 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.*

Implementação

Jogo

Para evitar ter que desenvolver um jogo já implementado foi utilizado o código fonte do jogo feito por Gabriele Cirulli disponível no *Github*. A utilização que fizemos desse código encontra-se ao abrigo da licença *MIT* do jogo. Para o jogo funcionar foram introduzidas algumas alterações no ficheiro *gameManager.js*, nomeadamente a inicialização da *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.

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*.

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 depois é ignorado.

É também verificado se o cliente que pretende executar o movimento fez algum movimento nos últimos 100 milissegundos, se 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.

Servidor *restify*

Este servidor implementa pedidos *GET e PUT* no *uri* 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.

Login

Conclusão

Após o desenvolvimento podemos concluir que com este projeto tivemos um grande contacto com os problemas da comunicação em tempo real de jogos e as várias maneiras de os remediar. Tivemos também uma primeira experiência de desenvolvimento *web* em tempo real com *node.js* e a biblioteca *socket.io*.

Em relação ao produto final concluímos que, tendo em conta o objetivo final, tivemos sucesso na nossa implementação, o jogo encontra-se jogável e com uma interface agradável.