Trabalho Prático P2

José Rúben Silva Freitas[1] and Leonardo Tadeu Nunes Abreu[1]

Universidade da Madeira, Portugal

Abstract. Com este artigo pretendemos aplicar os nossos conhecimentos teóricos adquiridos ao longos das aulas através de uma aplicação *web* utilizando os recursos fornecidos pelo docente. O jogo da Batalha Naval possibilita aos utilizadores fazerem registo de jogos públicos e privados de modo a interagir com diversos utilizadores na plataforma.

Keywords: Full Stack Development · JavaScript · NodeJs · Vue.js · socket.io · Embed.js.

1 Introdução

O principal objetivo deste projeto é continuar o trabalho desenvolvido no trabalho prático anterior, ou seja, criar uma aplicação *web* com auxílio de ferramentas alternativas ao desenvolvimento tradicional. Desta forma é possível continuar a desenvolver as nossas capacidades de *full stack development* de forma consistente, de modo a alcançar os objetivos propostos foi implementado em tempo real o jogo da Batalha Naval através da framework de Node.js express e da framework de javascript Vue.js.

Com este relatório pretendemos apresentar de forma clara e breve todo o trabalho desenvolvido na criação da plataforma. Iremos realizar uma breve descrição da abordagem utilizada para implementação dos requisitos necessários, referir singularmente como cada um deste recursos foi desenvolvido, caraterizar as vertentes de utilização da plataforma e finalmente apresentar os comandos necessários para instalação e consequente execução do projeto. Para concluir realizamos uma pequena discussão relativamente à qualidade, viabilidade e aspetos de desenvolvimento futuro do produto.

2 Implementação

O nosso projeto prático P2, foi desenvolvido baseado numa estrutura API, ou seja o lado do cliente consegue aceder aos dados do lado do servidor através de pedidos axios, exceto em casos de comunicação em tempo real, como por exemplo o processo de um jogo.

De modo a implementar o comportamento descrito anteriormente, foram utilizadas portas diferentes para o cliente e o servidor, desta forma é possível criar um ambiente onde ambas as estrutura estão hospedadas separadamente. Esta opção de implementação é benéfica no ambiente de desenvolvimento, pois permite tratar o *frontend* e o *backend* individualmente, dependendo unicamente dos pedidos Http e da comunicação via *socket*.

2.1 Requisitos tecnológicos

De modo a satisfazer todos os requisitos tecnológicos propostos pelo docente, foi necessária a utilização de algumas tecnologias referidas nas aulas, tal como algumas outras, nomeadamente *socket.io*, *Embed.js*, *Vue.js*, t*Express*, *Mongoose*, *axios*, *jquery* e como já esperado HTML, CSS e JavaScript.

O ***socket.io*** é responsável por toda a comunicação existente durante um jogo, este permite a inicialização, troca de dados e finalização dos comportamentos associados a esta mesma secção. Esta biblioteca javascript é dividida em duas partes, uma biblioteca do lado do cliente que é executada no *browser* e uma biblioteca do lado do servidor para o Node.js, onde para o nosso caso específico possuímos um *frontend* hospedado na porta 8080, o *backend* na porta 8000 e a *socket* na porta 3000 de modo a criar a ligação entre ambos. Além dos comportamentos associados ao jogo, esta é também utilizada para o chat de comunicação entre dois jogadores, ou seja é responsável por toda a comunicação, entre *backend* e *frontend* existente no *template* designado por "Room".

Para o desenvolvimento da interface utilizou-se a framework JavaScript ***Vue.js***, onde foi criado um ficheiro principal e um ficheiro de roteamento que controlam todo o fluxo do *frontend*. Assim é possível criar uma estrutura dinâmica que consome menos recursos de navegação. Através da utilização de componentes foi possível obter níveis elevados de abstração e reutilização que beneficiam a manutenção, performance e esforços de desenvolvimento.

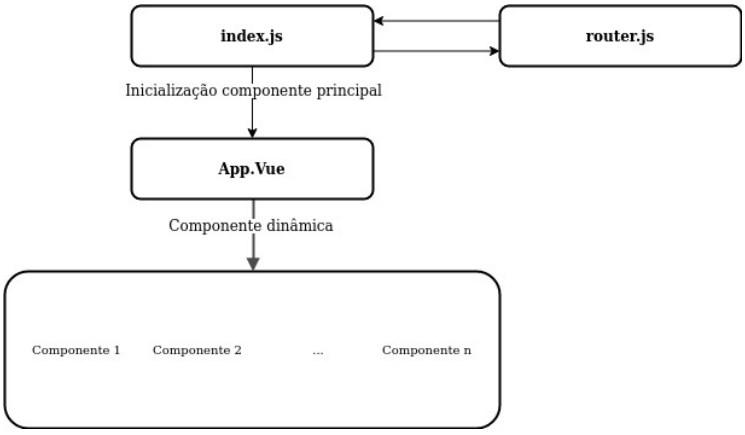

Fig. 1. Estrutura frontend com Vue.js

Relativamente ao ***Express***, este foi organizado numa estrutura Model-Route-Controller, extremamente semelhante à estrutura implementada no projeto prático anterior, desta forma é possível criar uma base estável, mecanismos de manutenção e facilidade de compreensão. Isto foi possível através das funcionalidades de

roteamento que a própria framework nos fornece e da biblioteca de NodeJs, *Mongoose* que proporciona uma solução baseada em esquemas para modelar os dados da aplicação. Devido à organização utilizada a interação com os documentos pertencentes na base de dados, foi em alguns casos realizada através de técnicas baseadas em estruturas / esquemas.

A utilização do ***Embed.js***, devido a limitações na abordagem de *frontend*, teve que ser adaptada de modo a satisfazer os requisitos necessários do projeto. Como referido anteriormente, foram utilizados os *templates* fornecidos pela tecnologia Vue.js o que dificulta exponencialmente a implementação desta biblioteca, pois estes não permitem a inclusão explícita de ficheiros .ejs, uma alternativa seria a declaração dos componentes Vue dentro do ficheiro ejs, ou seja, isto seria o sistema de *template* principal. Visto que a utilização desta tecnologia é um requisito obrigatório, esta foi utilizada para apresentar de forma dinâmica a página de acesso na porta do *backend*, onde foi possível utilizar grande parte das funcionalidades fornecidas, como por exemplo variáveis JavaScript em HTML e *templates* para reutilização de código.

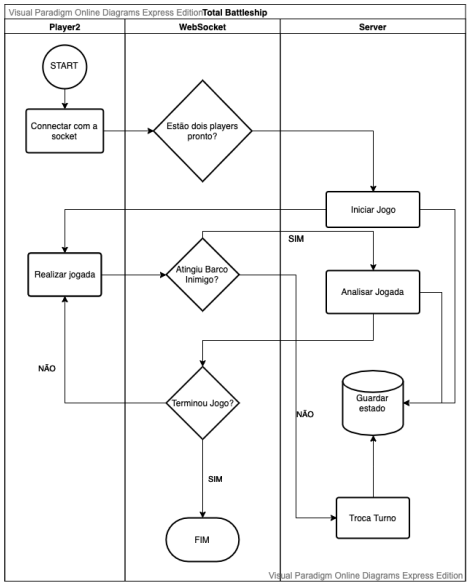

Fig. 2. Estrutura de comportamento da socket

2.2 Requisitos funcionais

Relativamente a funcionalidades o sistema permite executar todos os requisitos inicialmente propostos, como tal possui um sistema de **autenticação** baseado

em tokens que permite aos utilizadores realizarem registos e inícios de sessão de modo a poder participar em jogos da plataforma. Consequente da implementação de utilizadores registados, foram ainda desenvolvidas secções com o objetivo de aceder aos jogos deste mesmo utilizador, nomeadamente jogos finalizados e jogos em execução.

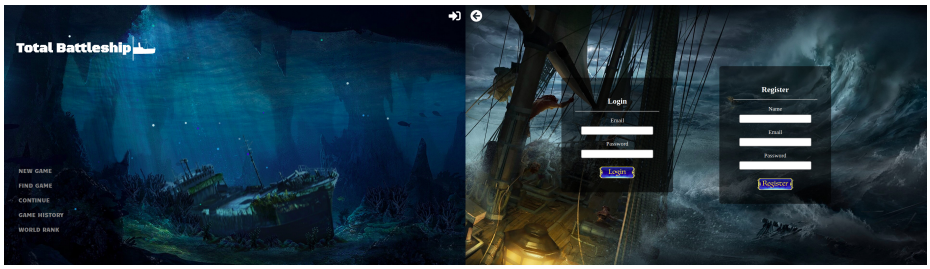

Fig. 3. Secção de menu e autenticação

Todos estes parâmetros são armazenados na base de dados do sistema, ou seja, é possível **registar jogos** para que estes possam ser **carregados e continuados posteriormente**. Isto é possível devido a *arrays* de posições que contêm o estado atual dos tabuleiros, onde estes são atualizados no lado do servidor após receção de eventos específicos da socket.

Estes estados estão todos representados no lado do servidor através de uma interface simples, funcional e apelativa. Aqui é possível aceder a todas as secções mencionadas e visualizar / executar todas as ações possíveis no jogo da Batalha Naval.

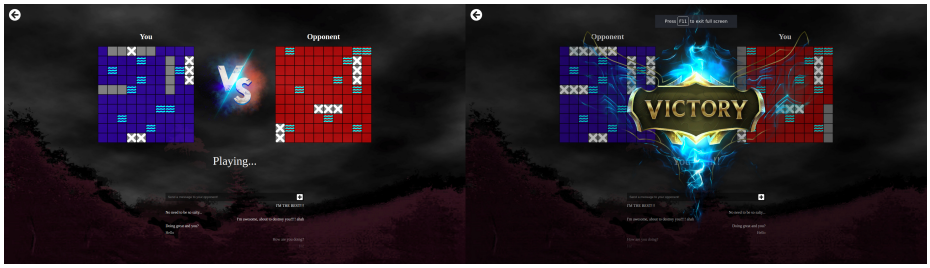

Fig. 4. Interface de um jogo específico

2.3 Requisitos extra

Encriptação Adicionalmente aos requisitos obrigatórios implementou-se neste trabalho **encriptação das mensagens do socket.io**, para tal utilizou-se a *package socket.io-encrypt*, que permite realizar a encriptação e desencriptação de mensagens através de *cryptr* [1].

Rankings Outra *feature* implementada no projeto é a inclusão de um um sistema de *ranks*, esta funcionalidade funciona por ligas, nomeadamente "*unranked*", "*bronze*", "*silver*", "*gold*" e "*diamond*". De forma geral as funções responsáveis por esta secção são exucutadas após o fim de cada jogo e tendo em conta as suas prestações, analisamos se este permanece na sua liga, é promovido ou despromovido, para tal criou-se no menu uma opção em que qualquer jogador pode verificar os ranks e quais os jogadores pertencem aos mesmos. Cada vitória equivale a 4 pontos, enquanto que uma derrota equivale a 2 pontos negativos, desta forma um jogador irá ser promovido após ultrapassar os 20 pontos, comparativamente um jogador é despromovido quando alcança pontos negativos. Relativamente a casos específicos, um jogador na liga "*bronze*" nunca poderá descer, todos os jogadores começam na liga "*unranked*" e após 5 jogos, de acordo com os seu resultado irão ser colocados em "*bronze*", "*silver*" ou "*gold*", após cada promoção de liga os pontos voltam a zero, exceto na liga "*diamond*" onde os pontos sobem infinitamente, pois não existe uma liga superior. Finalmente um jogador nunca poderá descer de liga nos primeiros dois jogos nessa liga, ou seja um jogador recentemente promovido para uma dada liga, caso perca os dois primeiros jogos mantém-se nessa mesma liga, mas posteriormente caso alcance pontos negativos irá descer.

Animações Visto que a vertente deste projeto enquadra-se na criação de um jogo, as animações sonoras e efeitos visuais são importantes, pois estas características cativam os utilizadores na interação com a plataforma. Foram colocados alguns efeitos com partículas nos menus do jogo e ecrãs de carregamento de jogos, mas o maior esforço foi realizado na página de um próprio jogo onde possuímos um fundo de ecrã dinâmico, efeitos sonoros para *feedback* de jogadas e música de fundo para desenvolver o ambiente e experiência do jogador.

Chat entre os dois jogadores Tendo em conta outros projetos e alguns sites de *livestream* achou-se por bem a implementação de um chat durante o decorrer do jogo, permitindo assim a interação entre os dois jogadores de forma a que partilhem as suas experiências de jogo como também táticas.

Históricos de jogos Outra *feature* implementada é permitir ao jogador aceder ao seu histórico de jogos realizados, fornecendo assim a informação de quem foi o seu adversário o vencedor do jogo.

JsonWebtoken Foi utilizado o pacote JWT de modo a realizar a transferência de dados de forma segura, especificamente na secção de autenticação na plataforma.

Fig. 5. Chat na página de jogo

É gerada uma nova token para um utilizador aquando de um início de sessão, baseada numa chave secreta definida no backend, desta forma esta pode ser retornada para o lado do cliente e posteriormente armazenada no armazenamento do local do *browser* evitando um grande número de interações entre cliente e servidor com dados com dados sensíveis. O procesos para fim de sessão é realizado de forma semelhante, onde a vriável do token no armazenamento local é removida e o utilizador terá que inicar sessão novamente.

3 Utilização

A interface desenvolvida pretende simplificar ao máximo a interação com a plataforma, desta forma todas as secções são acessíveis através do ecrã inicial.

De modo a poder interagir com o global das funcionalidades, o utilizador terá que estar registado e consequentemente com sessão iniciada. Esta opção está disponível no canto superior direito, de forma semelhante ao fim de sessão.

Caso queira criar um novo jogo, o utilizador terá que escolher entre jogo público ou privado, estes divergem na forma como são acedidos. Caso seja um jogo público, o adversário terá que entrar através da opção "*Find Game*" que devolve o jogo disponível mais antigo, caso contrário na página do jogo é visível um *url* que poderá ser partilhado pelo próprio utilizador.

Um utilizador está limitado pelo número de jogos que tem em execução, ou seja este poder ser impedido de criar novos jogos até finalizar os jogos a que está associado.

Outra funcionalidade está presente no menu, esta permite aceder ao histórico de jogos, ou seja o total de jogos finalizados por este específico utilizador. Aqui é possível verificar o adversário e o resultado deste mesmo jogo.

De forma semelhante à secção anterior, a página "*Current*" apresenta todos os jogos atualmente em execução deste utilizador. Nesta página é possível aceder aos jogos pendentes com o objetivo de carregá-los e continuar.

Finalmente a última opção do menu, apresenta um sistema de classificação dos 10 melhores jogadores de cada rank. Inicialmente é apresentado os 10 mel-

hores jogadores na plataforma, mas é possível filtrar os resultados por rank através da barra de navegação no topo da tabela.

4 Conclusão

Como inicialmente proposto o Total Battleship é uma aplicação *web* que permite aos utilizadores jogarem Batalha Naval online usando comunicação em tempo real. Como todos os produtos presentes no mercado o Total Battleship tem os seus pontos fortes e os seus pontos menos fortes, estes foram descritos ao longo do relatório.

Tendo em conta todos os aspetos anteriormente mencionados achamos que a implementação deste projeto foi uma mais valia para o nosso desenvolvimento como *full stack developers*, pois este projeto permitiu desenvolver novas capacidades adicionais ao desenvolvimento *web* tradicional.

Tendo em conta as limitações temporais de implementação não foi possível implementar todas as ideias que tínhamos previsto inicialmente, como por exemplo, gostaríamos de permitir os jogadores definirem a sua tática posicionando os barcos no mapa conforme a sua vontade, também seria importante implementar um sistema de torneios, implementação de jogos baseados em equipas, dar prioridade a partidas entre jogadores na mesma liga e finalmente permitir a visualização de jogos por utilizadores externos, isto é uma espécie de *livestream*.

References

1. cryptr, https://github.com/MauriceButler/cryptr/. Last accessed 14 Jan 2020