

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE ENGENHARIA DE SISTEMAS E TECNOLOGIA
DA INFORMAÇÃO

ANDRÉ LUIS AFFONSO DE MELO - 31174
KELVIN BRUNO ALFREDO AMANCIO - 30467

Towerfall

Itajubá

2017

ANDRÉ LUIS AFFONSO DE MELO - 31174
KELVIN BRUNO ALFREDO AMANCIO - 30467

Towerfall

Monografia apresentado ao professor Bruno Tardiole Kuehne como requisito parcial para a aprovação na disciplina de Sistema Distribuídos do curso de Engenharia da Computação da Universidade Federal de Itajubá.,

Itajubá

2017

Resumo

Esta monografia apresenta as tecnologias, métodos e as etapas para o desenvolvimento de um jogo multiplayer baseado no jogo já existente Towerfall Ascension. Utilizou-se para o desenvolvimento do jogo os conhecimentos adquiridos durante as aulas práticas de Sistemas Distribuídos, além de conhecimentos adicionais de tecnologias que fizeram-se necessárias para a realização do jogo conforme o mesmo foi pensado e estruturado desde o início do seu desenvolvimento.

Palavras-chave: Jogo Multiplayer. Towerfall Ascension. Sistemas Distribuídos

Lista de ilustrações

Sumário

1	INTRODUÇÃO	5
2	O JOGO	6
2.1	Tecnologias Utilizadas	6
2.1.1	HTML	6
2.1.2	CSS	7
2.1.3	Bootstrap	7
2.1.4	Javascript	7
2.1.5	React	8
2.1.6	Meteor	8
2.1.7	Node	9
2.1.8	Banco de Dados NoSQL	9
2.1.9	Socket.IO	10
2.1.10	MongoDB	10
3	DESENVOLVIMENTO DO JOGO	12
4	CONCLUSÃO	14
	REFERÊNCIAS	15

1 Introdução

Desde que foi proposto o desenvolvimento de um jogo multiplayer no início do semestre, deu-se início entre os membros do grupo uma discussão acerca de qual jogo seria desenvolvido e de quais tecnologias seriam utilizadas.

Pensou-se em desenvolver um jogo que tivesse certo grau de popularidade, você dinâmico e interativo, para que o desenvolvimento não ficasse muito trivial, e que tivesse fácil jogabilidade. Diante de tais fatores, escolheu-se o jogo TowerFall Ascension, jogo multiplayer, *player × player* que pode ser jogado em até quatro jogadores. A ideia não inicial não era desenvolver uma cópia fidedigna do jogo, mas sim tê-lo como de base e inspiração para a estrutura de *gameplay* do nosso jogo.

Para a escolha de qual linguagem e tecnologias utilizar no projeto levou-se em conta a nível de popularidade de tais tecnologias e quais linguagens, bibliotecas e *frameworks* vão ao encontro do que está sendo mais requisitado no mercado de trabalho atualmente. Dessa forma, o trabalho teve também a utilidade de preparar e atualizar os membros em relação as tecnologias mais comumente utilizadas nas grandes empresas.

2 O Jogo

TowerFall é um *indie* desenvolvido por Thorson em junho de 2012. Thorson testou o jogo em seus amigos particulares e se inspirou nos jogos que jogou na infância. Sua mecânica foi inspirada nos jogos da juventude de Thorson, como Bushido Blade e GoldenEye 007, e influenciada pelo feedback demo no torneio de jogos da série Evolution Championship.

O jogo foi muito bem recebido pela crítica, porém a falta de um modo *online* decepcionou muito dos jogadores que até hoje espera uma versão que possa ser jogada em rede.

TowerFall é um jogo de arena de combate com arco e flecha, onde os jogadores se matam até restar apenas um jogador. [3] No modo multiplayer, até quatro jogadores lutam em uma batalha real usando um suprimento limitado de flechas. Jogadores reabastecem o suprimento de flechas com as flechas jogadas na arenas. Os jogadores também podem pegar flechas de outros jogadores. *power-ups "Treasure"* dão aos jogadores escudos, asas e flechas com maior poder. As regras do jogo podem ser personalizadas e salvas para uso futuro.

Existem quatro modos de jogo. No modo single-player, o jogador deve atingir alvos ao redor da arena antes que o tempo se esgote. Os outros modos se baseiam em combate onde o último que sobreviver vence.

2.1 Tecnologias Utilizadas

Optou-se por desenvolver o jogo utilizando a linguagem Javascript, que nos últimos três anos vem estando sempre entre as três mais utilizadas no mercado de trabalho. Utilizou-se a *library de websockets Socket IO*, a biblioteca de interfaces *React JS*, a *framework* de *CSS Bootstrap*, a linguagem de desenvolvimento *BackEnd Node JS* e o banco de dados *NoSQL MongoDB*.

2.1.1 HTML

HTML é uma linguagem de marcação de hipertexto usada na criação de páginas *web*. A linguagem HTML foi desenvolvida por alguns físicos do CERN, o laboratório europeu de partículas físicas, e que provia inicialmente páginas estáticas usadas somente para a distribuição de artigos científicos(MUSCIANO; KENNEDY et al., 1996). Porém, com a popularização dos *websites*, os documentos HTML passaram a ser customizados através do CSS e JavaScript, deixando as páginas *web* com uma aparência mais amigável. Atualmente, toda página *web* é desenvolvida utilizando o HTML em conjunto com outras

ferramentas. No projeto desenvolvido, o HTML foi utilizado na criação da plataforma com o auxílio do CSS e JavaScript.

2.1.2 CSS

O CSS é uma linguagem de folhas de estilo utilizada para a estilização (cor da página, posicionamento dos elementos, fontes, entre outras funcionalidades) de documentos HTML. O CSS permite a separação entre a formatação e o conteúdo do documento, dessa maneira, quando é necessário alterar a aparência de uma página basta modificar um único arquivo. Na maioria das vezes o CSS é utilizado em sincronia com um framework que já possui vários estilos prontos e facilita muita o trabalho do desenvolvedor. Utilizou-se durante o desenvolvimento da plataforma a framework Bootstrap.

2.1.3 Bootstrap

Bootstrap é um *framework open source* de desenvolvimento *web* utilizado para a criação de *designs de websites*. O *framework* é baseado em CSS e fornece uma enorme variedade de templates, formulários, botões, menus de navegação, entre outras interfaces estilizadas. O Bootstrap é bastante utilizado para a criação de interfaces responsivas, onde um único código é escrito para a criação de interfaces para diferentes dispositivos (computadores *desktops*, *smartphones* e *tablets*). Para a utilização do *framework* basta adicionar o *CDN (Content Delivery Network)* do Bootstrap no documento HTML e utilizar as classes disponíveis no site oficial para realizar as customizações.

2.1.4 Javascript

Primeiramente, é importante destacar que o JavaScript não tem nenhuma relação com a linguagem de programação Java. De acordo Flanagan (2006), é senso comum que o JavaScript seja uma versão simplificada do Java, a linguagem de programação desenvolvida pela *Sun Microsystems*. A similaridade dos nomes foi puramente uma estratégia de *marketing* feita pelos criados do JavaScript, já que o Java fazia sucesso na época (a linguagem foi originalmente chamada de *LiveScript* e o nome foi mudado de última hora).

JavaScript é uma linguagem interpretada pelos navegadores, não necessitando ser compilada, o que favorece a disseminação de que JavaScript seja um linguagem simplificada e que seja uma linguagem usada por não-programadores. Na verdade JavaScript é uma linguagem complexa que segue o paradigma de orientação a objetos e a eventos. O que acontece é que a maioria das pessoas desconhecem muitos dos seus recursos e acabam utilizando-a para realizar tarefas básicas.

Assim como as outras linguagens, JavaScript está em crescente evolução. A linguagem segue o padrão ECMA (*European Computer Manufacturer's Association*) e atualmente

sobre atualizações a cada seis meses. Oficialmente, a linguagem tem o nome de *ECMAScript*, porém esse nome nunca é utilizado na prática, somente quando refere-se aos padrões da linguagem.

A mais popular utilização do JavaScript é como uma linguagem *client-side*, ou seja, quando ela é interpretada e embarcada em um *web browser*. *Client-side* em JavaScript combina a linguagem interpretada com o modelo de documento DOM definido pelo navegador. Os documentos podem conter *scripts*, e esses *scripts* podem controlar o que é apresentado nos navegadores. O JavaScript basicamente controla o comportamento da página.

Os navegadores com interpretador JavaScript são capazes de executar os *scripts* presentes nos documentos HTML, deixando as páginas mais dinâmicas e interativas.

2.1.5 React

React é uma biblioteca Javascript para criação de interfaces baseada em *web components*. React foi criada pela equipe de desenvolvimento do *Instagram* e desde então ganhou grande notoriedade, sendo utilizada hoje por grandes plataformas como *Facebook*, *Instagram*, *NetFlix* e *Airbnb*.

Um dos grandes diferenciais do React é a componentização dos itens da interface, deixando-os reutilizáveis e flexíveis e também a utilização do *Virtual DOM* que permite a atualização unitária dos componentes, não interagindo diretamente com o DOM(Document Object Model) a cada modificação, deixando a aplicação muito mais performática. Um algoritmo de *diff* é chamado toda vez que ocorre uma alteração na página e armazena as mudanças no *virtual DOM*. Após isso, o *virtual DOM* envia para o DOM apenas o que foi alterado. A figura ?? exemplifica o funcionamento desse método.

O React foi escolhido para o desenvolvimento da aplicação por estar em alta no mercado, ter alta performance, grande taxa de reutilização de código e pelo fato de que os membros da equipe já possuíam certa familiaridade com a biblioteca.

2.1.6 Meteor

Meteor é uma *framework fullstack* para o desenvolvimento de aplicações tanto *web* quanto *mobile*. Através dele é possível ter um ambiente de desenvolvimento completo para o execução e gerenciamento de uma aplicação desde o banco de dados MongoDB, *backend* Node, *frontend* Javascript e *build mobile* usando Cordova. O Meteor automatiza todas as tarefas do *frontend* e já compila uma versão minificada do código em Javascript.

O envio de dados e informações no Meteor fúciona através de *websockets*, o que garante que as atividades sejam atualizadas em tempo real. Além disso, a *framework* provê

um ambiente de desenvolvimento com *live reloading*, permitindo desenvolver e conferir as alterações no *browser* instantaneamente.

O Meteor foi escolhido pela equipe de desenvolvimento por agrupar todas as tecnologias utilizadas em um único ambiente, ser de simples utilização e ter um baixíssimo período de desenvolvimento em relação as outras tecnologias. Além desses fatos, toda a *stack* do Meteor utiliza a linguagem Javascript, o que facilitou ainda mais o aprendizado, já que o time tinha afinidade com a linguagem.

2.1.7 Node

Durante muito tempo a linguagem JavaScript foi utilizada somente no *client-side*, isto é, na estilização do comportamento das páginas HTML. Porém, com a popularização da linguagem, criou-se o Node Js para que o JavaScript também pudesse ser usado no lado do servidor. Apesar de novo(fundado em 2009), o Node ja é muito popular e empregado em aplicações de grandes empresas como o *Google*, *NetFlix* e *PayPal*. De acordo com Cantelon et al. (2014), o Node é uma plataforma feita em JavaScript para construir de maneira fácil e rápida aplicações network escaláveis. Existem várias vantagens em se utilizar JavaScript no server-side, uma delas é que o JavaScript é uma linguagem usada em vários bancos de dados não relacionais, facilitando assim a integração com o server-side através do Node JS. O Node JS é geralmente usado em conjunto com outros frameworks de rotas, validação de usuários, criptografia, entre outros.

2.1.8 Banco de Dados NoSQL

Todos os bancos de dados que não seguem exclusivamente o clássico padrão relacional são conhecidos como NoSQL(*Not only SQL*). Os bancos NoSQL surgiram para resolver os problemas de escalabilidade dos bancos tradicionais, já que pode se tornar muito caro ou complexo em grandes aplicações em SQL.

Existem atualmente uma grande quantidade de projetos open-source, sendo alguns deles: Cassandra, MongoDB, CouchDB, Neo4JS e Valdemort. Grandes empresas como *Google*, *Amazon*, *Facebook* e *Linkedin* já utilizam os bancos NoSQL, fato que ajuda ainda mais na sua popularização. Os bancos não relacionais são divididos nas seguintes categoriais:

- Bancos chave/valor: Também denominados tabelas *hash*, armazenam valores através de chaves e realiza a busca com o auxílio de suas chaves.
- Banco de Dados orientado a documentos: Armazena os dados em coleções de atributos e valores, onde os atributos podem receber vários valores (*array* de valores). Esse tipo de banco não segue uma estrutura rígida como as tabelas do SQL (*Structured*

Query Language), o que faz desse tipo de banco uma boa opção para o armazenamento de dados semi estruturados.

- Banco de Dados orientado a grafos: Esse banco está relacionado a um modelo de dados estabelecido, o modelo de grafos. O objetivo é representar os dados como grafos direcionados. O banco possui basicamente três elementos: os nós (vértices do grafo), os relacionamentos (arestas), e os atributos dos nós e relacionamentos. Esse tipo de grafo é comumente usado para a criação de rotas e em redes sociais, onde um usuário “a”, tem um relacionamento com vários usuários “b” que se relacionam com outros usuários “c”.

2.1.9 Socket.IO

Socket.IO é uma biblioteca JavaScript para aplicativos da Web em tempo real. Permite comunicação bidirecional em tempo real entre clientes da web e servidores. Ele tem duas partes: uma biblioteca do lado do cliente que é executada no navegador e uma biblioteca do lado do servidor para o Node.js. Ambos os componentes têm uma API quase idêntica. Como o Node.js, ele é orientado a eventos.

O Socket.IO usa principalmente o protocolo WebSocket com polling como opção de fallback, enquanto fornece a mesma interface. Embora possa ser usado simplesmente como um wrapper para WebSocket, ele fornece muitos mais recursos, incluindo transmissão para vários soquetes, armazenamento de dados associados a cada cliente e entrada e saída de dados assíncrona. Pode ser instalado com a ferramenta de gerenciamento de pacotes npm.

O Socket.IO fornece a capacidade de implementar análises em tempo real, fluxo binário, mensagens instantâneas e colaboração de documentos. Entre os usuários notáveis estão o Microsoft Office, o Yammer e o Zendesk.

Socket.IO lida com a conexão de forma transparente. Ele será atualizado automaticamente para o WebSocket, se possível. Socket.IO não é uma biblioteca WebSocket com opções de fallback para outros protocolos em tempo real. É uma implementação personalizada de protocolo de transporte em tempo real, além de outros protocolos em tempo real. Suas partes de negociação de protocolo fazem com que um cliente que suporta o padrão WebSocket não consiga contatar um servidor Socket.IO. E um cliente de implementação do Socket.IO não pode falar com um servidor WebSocket ou Long Polling Comet baseado em não-socket.IO. Portanto, o Socket.IO requer o uso das bibliotecas Socket.IO no lado do cliente e do servidor.

2.1.10 MongoDB

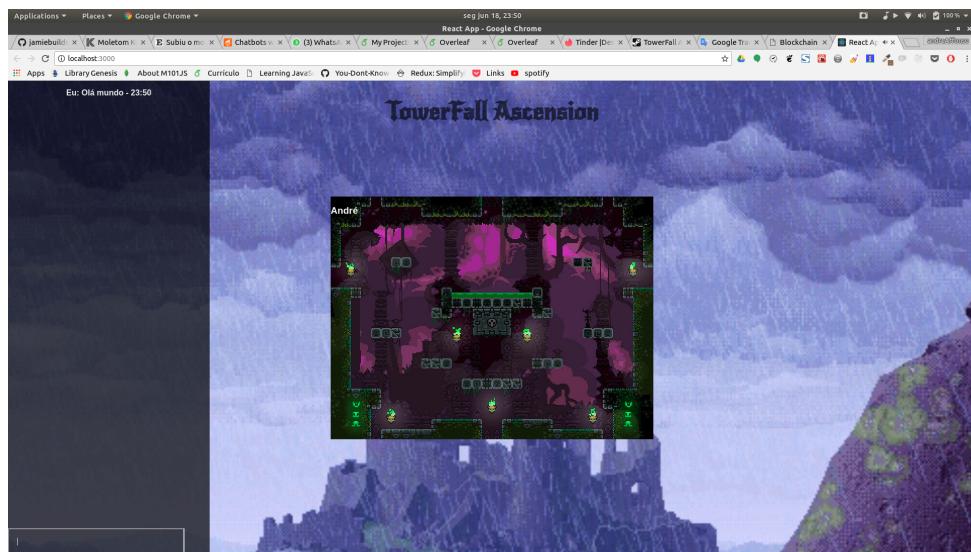
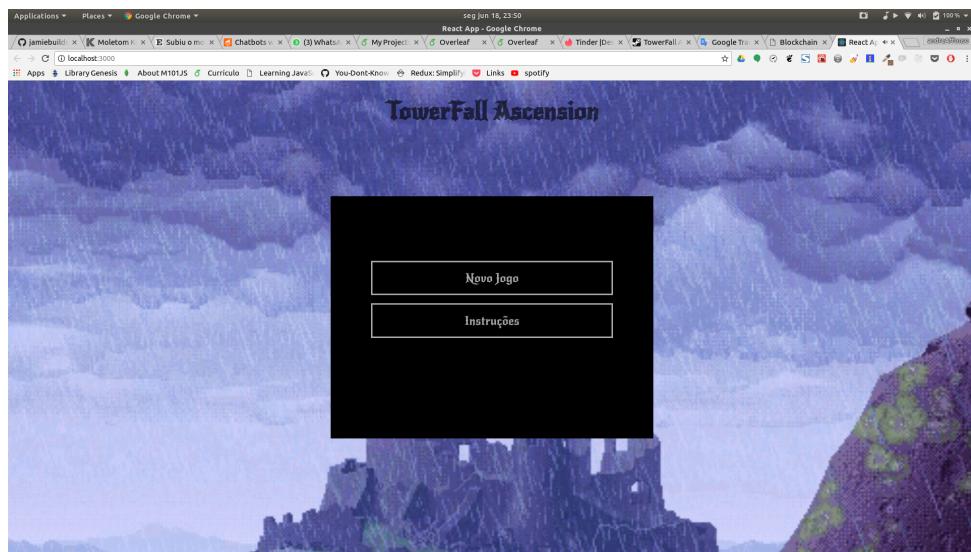
MongoDB é uma é um banco de dados *open-source*, de alta performance, orientado a documentos. Foi desenvolvido na linguagem de programação C++ e sua primeira versão

foi lançada em 2009. Classificado com um banco NoSQL, o MongoDB usa os documentos JSON(*JavaScript Object Notation*) para armazenar os objetos. Segundo Chodorow (2013), a principal razão para se adotar esse tipo de banco é sua fácil escalabilidade. O banco de dados orientado a documentos substitui o conceito de “tabelas” e “coluna” por um modelo mais flexível, o “documento”. Alocando os documentos em vetores, é possível representar complexas hierarquias relacionais facilmente. O banco também não define padrões para os documentos (os esquemas em SQL), o que facilita a adição e remoção de novos campos. A não definição de padrões é útil quando não se sabe qual o tipo de dado que irá se receber. Diferentemente dos bancos relacionais, o MongoDB não possui *queries* SQL, mas possuem métodos JavaScript responsáveis por realizar as tarefas de criação, leitura, atualização e remoção dos dados. O MongoDB pode ser iniciado e utilizado através do terminal do computador, porém para poder usá-lo junto ao servidor é necessário uma integração com o Node JS através da API(Application programming interface) Mongoose. Através do Node JS é possível a criação de esquemas, métodos de validação de campos, réplica de dados, *backup*, entre outras funcionalidades.

3 Desenvolvimento do jogo

A interface do jogo foi desenvolvida utilizando-se React, e o jogo roda dentro de um CANVAS definido com o tamanho da tela do usuário. Criou-se tmb uma sala de bate papo entre os jogadores com as conversas sendo armazenadas no banco de dados mongoDB. Todas as informações são armazenadas nos estados dos componentes e transmitidas por websockets entre os participantes da rede. O servidor foi configurado para rodar na porta 3001 enquanto o cliente roda na porta 3000.

As imagens abaixo ilustram a estrutura do projeto.



As imagens abaixo ilustram algumas telas do jogo.

```
andreaffonso@andreaffonso:~/Projects/towerfall$ tree -L 1
.
├── client
├── models
└── node_modules
    └── Stack Overflow
        ├── package-lock.json
        └── package.json
├── package.json
├── routes
└── server.js

5 directories, 3 files
```

andreaffonso@andreaffonso:~/Projects/towerfall\$

```
andreaffonso@andreaffonso:~/Projects/towerfall/client/src$ tree -L 1
.
├── App.test.js
├── components
├── containers
└── img
    └── index.css
    └── index.js
    └── registerServiceWorker.js
    └── utils

5 directories, 4 files
```

andreaffonso@andreaffonso:~/Projects/towerfall/client/src\$

4 Conclusão

Durante o desenvolvimento do jogo encontrou-se algumas dificuldades, principalmente em relação a sincronização de informações de um jogo bastante dinâmico.

Muitos conhecimentos de desenvolvimento web foram adquiridos e acredita-se que serão de suma importância para o futuro ingresso dos participantes no mercado de trabalho

Referências

CANTELON, M. et al. *Node.js in Action*. [S.l.]: Manning, 2014. Citado na página 9.

CHODOROW, K. *MongoDB: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2013. Citado na página 11.

FLANAGAN, D. *JavaScript: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2006. Citado na página 7.

MUSCIANO, C.; KENNEDY, B. et al. *HTML, the definitive Guide*. [S.l.]: O'Reilly & Associates, 1996. Citado na página 6.