UNIVERSIDADE DO SUL DE SANTA CATARINA

PAULO RICARDO JANSEN

FAWE:
JOGO MULTIPLAYER ONLINE USANDO HTML5 E WEBSOCKET

Florianópolis

2020

PAULO RICARDO JANSEN

FAWE:
JOGO MULTIPLAYER ONLINE USANDO HTML5 E WEBSOCKET

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Saulo Popov Zambiasi, Dr.

Florianópolis

2020

PAULO RICARDO JANSEN

FAWE:
JOGO MULTIPLAYER ONLINE USANDO HTML5 E WEBSOCKET

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina.

Florianópolis, 02 de Dezembro de 2020.

_____

Professor e orientador Saulo Popov Zambiasi, Dr..
Universidade do Sul de Santa Catarina

_____

Prof. Vera Rejane Niedersberg Schuhmacher, Dra..
Universidade do Sul de Santa Catarina

_____

Prof. Richard Henrique de Souza, Dr.
Universidade do Sul de Santa Catarina

RESUMO

Os jogos online permitem que as pessoas se divirtam juntas de diferentes lugares. Para que estes jogos sejam criados é necessário o conhecimento de diferentes tecnologias. O presente trabalho aborda o desenvolvimento do jogo Fawe, o qual utiliza as tecnologias HTML5, Canvas e Websockets. Em cima disso, realizou-se uma pesquisa bibliográfica para compreender melhor estas tecnologias da web. O projeto foi desenvolvido junto com a criação de um Game Design Document detalhando todas as informações e conceitos do jogo. O jogo foi modelado utilizando conceitos de engenharia de software com diagramas de classe e atividades. Ao final do projeto, 17 pessoas avaliaram o jogo através de um questionário online e os resultados foram comparados.

Palavras-chave: GDD. Game Design Document. Websocket. HTML5. Canvas. Jogos Digitais. Jogos multiplataforma. Desenvolvimento de Jogos. Comunicação em Tempo Real. Programação para Web. Internet. MMORPG. RPG.

ABSTRACT

Online games allow people to have fun together from different places. For these games to be created it is necessary to know different technologies. In order to demonstrate the technologies of HTML5 Canvas and Websocket, this work focuses on the development of the Fawe game. From that, a bibliographic search was carried out to better understand these web technologies. The project was developed together with the creation of a Game Design Document detailing all the information and concepts of the game. The game was modeled using software engineering concepts with class and activity diagrams. At the end of the project, 17 people evaluated the game using an online questionnaire and the results were compared.

Keywords: GDD. Game Design Document. Websocket. HTML5. Canvas. Games. Game Development. Real Time Applications. Web Programming. Internet. MMORPG. RPG.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Segundo Huizinga (2008), os jogos podem ser definidos como uma atividade repleta de significados, que podem favorecer a aprendizagem, o desenvolvimento físico e mental, a socialização e as mais variadas áreas, ou simplesmente como atividades lúdicas de apenas entretenimento. São encontrados na história da humanidade desde 3500 a.C e trouxeram contribuições importantes na formação da cultura e outras áreas do conhecimento.

Em seu livro, Huizinga (2008) relata que até mesmo os animais também praticam esta atividade:

> Bastará que observemos os cachorrinhos para constatar que, em suas alegres evoluções, encontram-se presentes todos os elementos essenciais do jogo humano. Convidam-se uns aos outros para brincar mediante um certo ritual de atitudes e gestos. Respeitam a regra que os proíbe morderem, ou pelo menos com violência, a orelha do próximo. Fingem ficar zangados e, o que é mais importante, eles, em tudo isto, experimentam evidentemente imenso prazer e divertimento.

Com a chegada dos primeiros computadores, não demorou muito até que alguém tivesse a ideia de tornar possível um jogo dentro deste novo universo. Segundo Azevedo (2003), uma equipe de estudantes do MIT (Instituto de Tecnologia de Massachusetts) desenvolveram em 1961, o primeiro jogo de computador, que se chamava Spacewars. O jogo simula uma batalha espacial com naves e estrelas, com gráficos bidimensionais muito simples em um computador que tinha o tamanho de uma mesa.

Hoje, de acordo com Chikhani (2015), os jogos evoluíram através de diversas gerações e condizem com a atualidade, se desenvolvendo junto com avanços tecnológicos, em poderosos computadores. Exploram diferentes tipos de interações sociais de forma marcante até mesmo no mundo virtual. Além disso, estão incluídos no dia a dia e no lazer de um público cada vez mais exigente.

Ainda conforme Chikhani (2015), o surgimento da internet e dos protocolos de comunicação possibilitaram que jogos de diferentes dispositivos se comuniquem. Esta comunicação trouxe diversos novos desafios para o desenvolvimento de jogos se transformando em um novo tipo de jogo que hoje é chamado de jogo multijogador online.

1.1 PROBLEMÁTICA

Os jogos online podem levar muito tempo para serem desenvolvidos e custar muito dinheiro. Segundo Wells (2020), quando um jogo conecta diversos jogadores simultaneamente, pertence a modalidade MMO, que é a sigla em inglês para Multiplayer Massive Online (Multijogadores Massivos Online).

Crowfall é um jogo MMO que está em desenvolvimento desde 2015. Ele foi anunciado através de uma campanha no Kickstarter, ArtCraft (2015), ganhando um total de quase US$1,8 milhões para ser desenvolvido. Depois, passou por diversas novas rodadas de investimentos chegando a mais de US$40 milhões de acordo com Kerr (2020). Hoje, em 2020, já se passaram cinco anos e o jogo ainda está em fase beta.

Pekanov (2020) descreve os desafios durante o desenvolvimento de um jogo online que trabalhou no período de dois anos. Uma das maiores dificuldades está na complexidade em lidar com um ambiente virtualizado e muitas conexões ao mesmo tempo, visto que cada dispositivo conectado pode aumentar o uso dos recursos necessários para a execução do jogo. Outro desafio está em manter clientes e servidores atualizados e sincronizados.

Para Kozovits e Feijó (2003), estes jogos podem ser desenvolvidos com diferentes tipos de arquiteturas em sistemas distribuídos. Estas são escolhidas através das necessidades de troca de informações que forem definidas para o funcionamento do jogo. A mais comum delas é a arquitetura cliente-servidor, mas existem outras opções como por exemplo os sistemas ponto-a-ponto.

1.2 OBJETIVOS

A seguir são apresentados os objetivos do trabalho que podem ser classificados em geral e específicos.

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo o desenvolvimento de um jogo multijogador online com enfoque no uso do HTML5 e WebSockets.

1.2.2 Objetivos Específicos

O presente trabalho tem como objetivos específicos:

- Pesquisa de jogos multiplayers e tecnologias de comunicação.
- Pesquisa das tecnologias HTML5 e WebSockets.
- Criação de um Game Design Document.
- Modelagem do Jogo.
- Desenvolvimento do Jogo.
- Implementação e Testes para avaliação do protótipo.

1.3 JUSTIFICATIVA

Segundo Rouget (2012), para promover as tecnologias Canvas e Websockets no HTML 5, que são as tecnologias utilizadas neste trabalho, a Mozilla desenvolveu, em 2012 o jogo Browser Quests. O jogo, que foi desenvolvido por apenas duas pessoas, um programador e um artista, provou que o HTML5 estava pronto para ser utilizado para o desenvolvimento de jogos de maior complexidade.

Muitos jogos que utilizam estas tecnologias obtiveram sucesso. Um dos que mais ficou conhecido foi o Agario. Desenvolvido pela famosa empresa de jogos para web Miniclip, em 2016, segundo Cowley (2016), Agario atingiu 116 milhões de downloads em dispositivos móveis desde seu lançamento em 2015. O jogo fez tanto sucesso que acabou influenciando o desenvolvimento de outros jogos como Slither.io, que também utiliza as mesmas tecnologias.

Além disso, a indústria de games é uma das que mais cresce no Brasil e no mundo, e vem se consolidando como a mais rentável no ramo do entretenimento mundial, movimentando apenas no ano de 2019, segundo a empresa de análise de mercado SuperData (2020), a quantia de US$ 120 bilhões. Deste total, US$29,6 bilhões são derivados do nicho de jogos para computador. No Brasil, o dado mais recente é do ano de 2018, Valor Investe (2019), onde o país faturou 1,5 bilhões de dólares, possuindo 75,7 milhões de jogadores e sendo o líder latino-americano e 13º na classificação global.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Este trabalho está organizado nos seguintes capítulos:

Capítulo 1: Introdução ao ecossistema de jogos de computador, objetivos e justificativa do trabalho.

Capítulo 2: Pesquisa bibliográfica sobre a o desenvolvimento de jogos e suas tecnologias com foco em HTML5.

Capítulo 3: Apresenta as metodologias científicas utilizadas para o desenvolvimento deste trabalho.

Capítulo 4: Apresenta o Game Design Document (Documento de Design do Jogo) do jogo a ser desenvolvido.

Capítulo 5: Apresenta a modelagem do jogo.

Capítulo 6: Apresenta o desenvolvimento do jogo.

Capítulo 7: Aplicação de questionários

Capítulo 8: Conclusões e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo aborda uma revisão sobre o desenvolvimento de jogos multiplayer online.

2.1 JOGOS MULTIPLAYER ONLINE

Segundo Chikhani (2015), quando os primeiros videogames surgiram, ainda não existia rede ou internet. Para que os jogos pudessem ser jogados em grupo, era preciso que os jogadores utilizassem controles diferentes e compartilhassem a mesma tela. Esse modo de jogo existe até hoje e é adotado principalmente pelos consoles da atualidade.

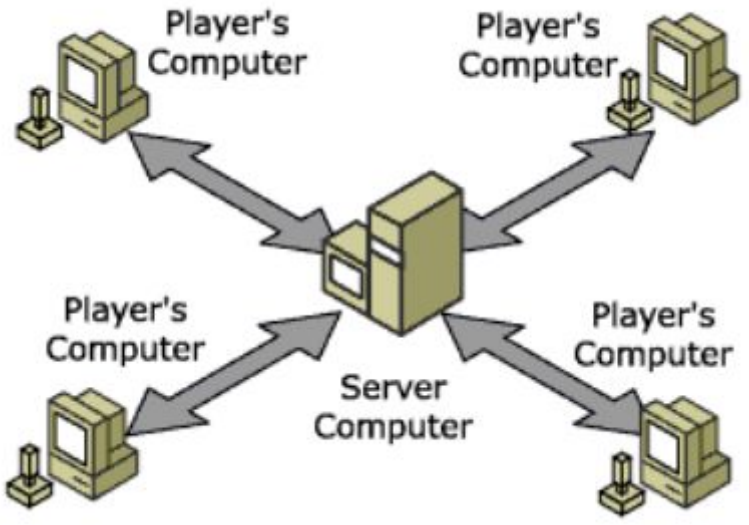
Ainda antes da popularização da internet, de acordo com Chikhani (2015), alguns videogames como o Atari 2600 traziam uma opção de comunicação via telefone. Era possível baixar jogos gratuitamente pela linha telefônica que se conectava junto com o console do videogame. Muitos desses jogos eram desenvolvidos por programadores independentes que disponibilizavam seus códigos para download.

Os primeiros jogos online começaram a surgir juntamente com a disseminação da internet pelo mundo. Um dos primeiros jogos de sucesso, que possibilitou milhares de jogadores se conectarem simultaneamente, foi Ultima Online (UO). Segundo Edwards (2007), lançado em 1997, UO é um jogo de mundo aberto onde o jogador cria um personagem que pode fazer diversas atividades como explorar, construir ou lutar. Como o jogo não tem um objetivo específico ou um roteiro a ser seguido, as pessoas podem jogar por horas todos os dias.

2.2 ARQUITETURAS PARA JOGOS ONLINE

Existem diversas arquiteturas possíveis para se desenvolver um jogo multijogador online. A alternativa mais versátil delas é, segundo Kozovits e Feijó (2003), através de sistemas cliente-servidor. Neste tipo de arquitetura, é necessário o desenvolvimento de um software para o cliente e outro para o servidor. O software cliente é executado nos dispositivos dos usuários e se conecta com um único servidor. Este, é executado por um computador responsável por manter todos os clientes conectados atualizados através de uma troca constante de mensagens.

Figura 1 - Arquitetura cliente-servidor típica



Fonte: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/03_36_kozovits.pdf

Um único servidor muitas vezes não possui recursos suficientes para atender uma grande quantidade de jogadores. Segundo Ignatchenko (2015), para resolver isso e aumentar a capacidade do jogo, é necessário encontrar alternativas de distribuir os acessos em mais servidores. É possível criar uma infraestrutura que faça com que um mesmo servidor seja executado mais vezes por mais computadores. Esta solução é utilizada pela maioria dos jogos online.

2.3 TECNOLOGIAS DE COMUNICAÇÃO E IMPLEMENTAÇÃO

É notável que o desenvolvimento de jogos demande diversos tipos de conhecimento, como descreve Tonéis (2015, p10):

> Entre as diversas ciências que compõem o núcleo de um profissional em jogos digitais certamente a interdisciplinaridade é um fator comum e presente em todas. Todo o processo criativo pode ser comprometido se o profissional em jogos não souber se comunicar com os responsáveis pela implementação ou programação(código do jogo). Não basta ser um exímio desenhista, ou roteirista, ou ainda um brilhante level design, torna-se necessário conhecer o processo em sua totalidade.

O próximo tópico apresenta uma pesquisa sobre a história do HTML5 e suas aplicações em jogos.

2.3.1 HTML5

De acordo com Lemay (2002), a Web tornou-se acessível para a maioria das pessoas principalmente por conta da facilidade de se trabalhar com URLs, um endereço pequeno e universal que aponta para algo na internet. Até mesmo a publicação de conteúdo ficou acessível para elas, facilitando o compartilhamento de conteúdo online em nível mundial.

Segundo Lemay (2002), as páginas da Web são desenvolvidas com a linguagem HTML (Linguagem de Marcação de Hipertexto). O HTML foi criado para descrever, de maneira simples, a estrutura de uma página, para que depois ela fosse interpretada de forma gráfica por um navegador de internet.

Nos primeiros anos do surgimento do HTML, existiram muitos problemas durante o processo de desenvolvimento, como relata Eis (2012, p16):

> O código ficava enorme, fazendo com que o download da página demorasse, a manutenção era terrível e o know-how de como o desenvolvimento foi feito era restrito a alguns profissionais da equipe. Era tudo muito inflexível. Isso encarecia a mão de obra e por isso os projetos ficavam cada vez mais caros.

Além disso, naquela época, o HTML ainda não tinha recursos para o desenvolvimento de aplicações gráficas de maior complexidade. Para isso, era necessário o uso de outras ferramentas ou extensões. A mais conhecida delas, que dominou a internet por muito tempo, foi o Flash. Macromedia, relata em uma apostila Lite (2002, p 1) como o Flash era revolucionário:

> Nenhuma outra ferramenta foi, até agora, capaz de produzir uma página da Web de alto desempenho em tela inteira, animada e interativa tão bem ou com tanta facilidade quanto o Flash. o Flash MX simplesmente torna esse fato mais real do que nunca.

O uso do Flash só começaria a diminuir quando a empresa Apple, publicou uma carta aberta em seu site (STEVE JOBS, 2010), que seus aparelhos iPhone e iPads não teriam a tecnologia. O principal motivo mencionado foi que o Flash é uma tecnologia proprietária e eles acreditavam que as tecnologias da web deveriam ser abertas. Mas, outros pontos importantes, como o fato do Flash ter sido criado para ser utilizado com mouse e não com telas touchscreen também foram levados em conta.

Outra gigante que também foi abandonando o Flash com o tempo foi o Google. Em uma publicação em seu blog (ANTHONY LAFORGE, 2017), relata que os sites da Web estão mudando para tecnologias abertas e deixando de usar o Flash. O autor demonstra em números, que a quantidade de usuários que acessavam algum site com Flash em seu navegador Chrome, caiu de 80 para 17 por cento entre 2014 e 2017.

Ao mesmo tempo que o Flash foi deixando de ser utilizado, o HTML foi evoluindo. Novas versões foram surgindo até chegar no HTML5, que é a versão atual. tornando possível até mesmo o desenvolvimento de jogos com processamento gráfico. Roque (2018, p9), que utilizou Flash durante uma década e depois migrou para o HTML5, relata o impacto da nova tecnologia: “[...] saiba que o HTML5 possui ferramentas suficientes para produzir quaisquer resultados que o Flash produza e que não há motivos para manter animações, sistemas ou quaisquer conteúdos na web em formato Flash.”

2.3.2 WEBSOCKETS

De acordo com Dionisio (2015), o desenvolvimento de aplicações que precisam de comunicação em tempo real, como os jogos online, pode ser feito com o uso da tecnologia Websocket. Esta permite uma comunicação bidirecional de baixa latência entre cliente e servidor, ou seja, uma conexão rápida e constante, sendo possível enviar e receber mensagens a qualquer instante.

Ainda segundo Dionisio (2015), o Websocket, ou WS, é um protocolo e sua especificação foi projetada para funcionar tanto em navegadores da web como em dispositivos móveis e tablets. Sendo assim, ele pode ser implementado com diferentes linguagens que consigam interpretá-lo corretamente, tanto em clientes como servidores.

Para que um cliente e um servidor estabeleçam uma conexão utilizando Websocket é necessário uma requisição inicial utilizando o protocolo padrão da web HTTP. Segundo Meenakshi (2019), este procedimento é chamado de handshake. Com ele, o servidor consegue identificar cada cliente como uma conexão individualmente.

No desenvolvimento de jogos online, é comum que cada conexão de websocket represente um jogador conectado a um ambiente simulado por um servidor. As conexões são gerenciadas de acordo com as necessidades do jogo. Os ambientes simulados podem ser, por exemplo, partidas em jogos de tiro em primeira pessoa, ou diferentes mundos em jogos de fantasia.

2.3.3 CANVAS

De acordo com Roque (2018), Canvas é um elemento de marcação da linguagem HTML. Ele torna possível a criação de desenhos gráficos de alto desempenhos que são interpretados através do javascript. É importante ressaltar que o elemento canvas sozinho não tem nenhuma função. Ele precisa de um código javascript para poder executar as funcionalidades gráficas.

Roque (2018, p18), descreve as indicações do Canvas:

> O CANVAS é indicado para projetos multimídia, jogos, composição gráfica e efeitos visuais sofisticados. As versões modernas dos navegadores utilizam recursos de aceleração gráfica (GPU - Graphics Process Unit) para interpretar e renderizar o CANVAS. Essa prática acelera o processamento dos gráficos via GPU, diminuindo o processo da CPU do usuário.

No desenvolvimento de jogos, o Canvas é utilizado para o desenvolvimento das animações que ocorrem durante a execução do jogo. Segundo Malone (2020), essas animações, chamadas de sprites, são uma sequência de desenhos reunidas em um único arquivo de imagem. Através de código javascript, esse arquivo é acessado parcialmente de forma sequencial formando uma animação.

Figura 2 - Arquivo imagem de uma sprite.

Fonte: http://www.williammalone.com/articles/create-html5-canvas-javascript-sprite-animation/

Todas as animações podem ser reunidas em um único contexto, que seria a tela do jogo. São desenhadas em coordenadas específicas que podem ser modificadas por ações de um jogador ou pelo algoritmo que controla o jogo.

A maioria dos jogos utiliza como base de desenvolvimento uma técnica chamada de game loop (ciclo de jogo). De acordo com Malone (2020), com essa técnica é possível controlar tudo que será desenhado na tela através de iterações a cada quadro. Basicamente, é separado o código do que controla os objetos do jogo com o código que renderiza de fato o jogo. Esses códigos são organizados em funções geralmente chamadas de update (atualizar) e render (renderizar) dentro de uma função de game loop. A função game loop deve garantir que essas duas funções sejam executadas infinitamente uma após a outra durante todo tempo de jogo.

Figura 3 - Exemplo em código javascrpit de um game loop.

```
function gameLoop () {

  window.requestAnimationFrame(gameLoop);

  coin.update();
  coin.render();
}

// Start the game loop as soon as the sprite sheet is loaded
coinImage.addEventListener("load", gameLoop);
```

Fonte: http://www.williammalone.com/articles/create-html5-canvas-javascript-sprite-animation/

O domínio dessa técnica possibilita o desenvolvimento de muitos tipos de jogos bidimensionais. Os jogos são apenas imagens desenhadas em uma tela em posições determinadas por uma lógica de programação e por ações de um jogador. Evoluindo estes algoritmos é possível obter diversos tipos de efeitos visuais diferentes.

Segundo Roque (2018), além de imagens, também é possível desenhar linhas, retângulos e texto com Canvas. Existe uma série de opções disponíveis para serem configuradas como cor, tamanho, tipo de fonte, entre outros. A lógica de desenho segue a mesma das imagens, é preciso definir as coordenadas desejadas para os objetos.

3 MÉTODO

Este capítulo aborda a metodologia proposta neste trabalho com uma descrição da caracterização do tipo de pesquisa, das atividades metodológicas e da solução proposta.

3.1 CARACTERIZAÇÃO DO TIPO DE PESQUISA

Segundo Cervo (2002), existem inúmeros tipos de pesquisas com abordagens distintas, cada uma com diferentes procedimentos e peculiaridades. Elas não necessariamente se opõem uma à outra, e sim, se complementam, sendo indispensáveis para o avanço da ciência.

A pesquisa bibliográfica é, segundo Gil (1996), desenvolvida utilizando materiais já existentes, tendo como principais fontes, os livros e os artigos científicos. Sua principal vantagem é permitir que o autor tenha como base uma infinidade de recursos disponíveis sem que ele tenha que pesquisá-los diretamente. Este trabalho utiliza diferentes tipos de fontes, tendo como principais os livros e artigos científicos voltados para a área de desenvolvimento de jogos.

Em contrapartida, ainda segundo Gil (1996), é muito importante avaliar a qualidade do material utilizado na pesquisa. Determinadas fontes com dados e informações incorretas podem prejudicar muito o trabalho a ser desenvolvido.

A metodologia de desenvolvimento deste trabalho é a de pesquisa aplicada, que para Cervo (2002), possui fins práticos. Por meio de pesquisa bibliográfica com foco em compreender as tecnologias necessárias para se atingir o objetivo proposto, que, no caso deste trabalho, é o desenvolvimento do jogo Fawe.

Após o desenvolvimento, se utiliza pesquisa qualitativa para validação do protótipo entrevistando usuários voluntários. Por meio de um questionário enviado a eles, são realizadas perguntas que avaliam se os objetivos do projeto desenvolvido foram atingidos.

3.2 ATIVIDADES METODOLÓGICAS

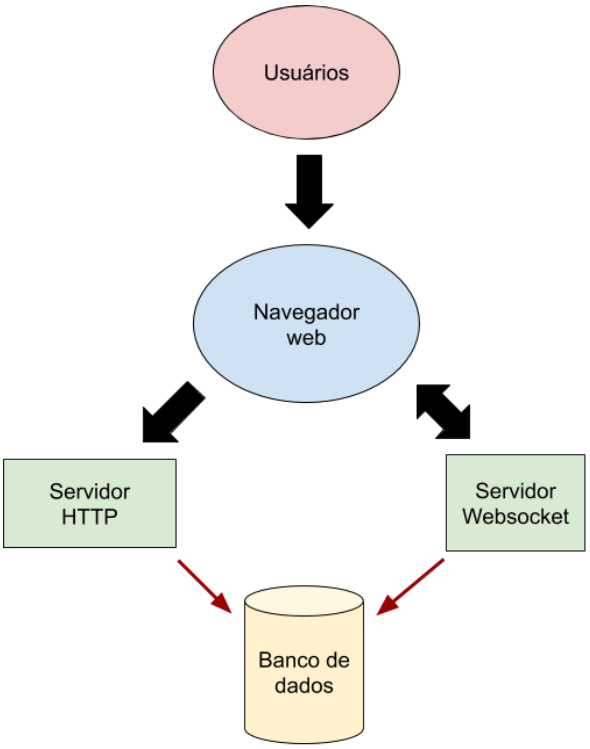
O desenvolvimento do jogo definido neste trabalho, terá as seguintes etapas:

- Levantamento bibliográfico das tecnologias utilizadas para o desenvolvimento do jogo.
- Definição da arquitetura a ser utilizada, detalhando a comunicação entre os sistemas a serem criados.
- Desenvolvimento do Game Design Document do jogo.
- Modelagem do jogo.
- Implementação da arquitetura definida com suas funcionalidades.
- Avaliação do jogo através de entrevistas com usuários.
- Apresentação dos resultados obtidos dos usuários .

3.3 SOLUÇÃO PROPOSTA

Este trabalho tem como objetivo o desenvolvimento de um protótipo de jogo online em que jogadores consigam interagir entre si conectados em diferentes dispositivos. Dentro do jogo, deve ser possível, em tempo real, realizar ações como andar, se comunicar ou atacar inimigos em um mundo virtual. A figura a seguir apresenta a arquitetura da solução proposta:

Figura 4 - Arquitetura proposta.



Fonte: Autoral, 2020.

Os usuários terão acesso ao jogo por meio de um navegador de internet compatível. O servidor HTTP será responsável por fornecer o código do jogo para o navegador, enquanto o servidor Websocket será responsável por fazer a comunicação em tempo real com os jogadores. O banco de dados será utilizado para armazenar dados do jogo.

3.4 DELIMITAÇÕES

Este trabalho conta com as seguintes delimitações:

- Desenvolvimento um jogo com gráficos bidimensionais sem o uso de nenhum motor gráfico (engine).
- Não é explorado o desenvolvimento de jogos tridimensionais.
- O jogo desenvolvido funcionará em um navegador web.

4 GAME DESIGN DOCUMENT

Neste capítulo é apresentado o Game Design Document (Documento de Design do Jogo).

De acordo com Rogers (2010), O GDD é um documento que contém todas as informações de um jogo. Deve detalhar conceitos como ideia inicial, história, jogabilidade, interface gráfica, personagens, fases e tudo mais que estiver relacionado ao design do jogo.

Todo o conteúdo gráfico deste capítulo foi criado por Douglas Jansen. Uma autorização foi anexada ao final da monografia.

4.1 FAWE

O jogo a ser desenvolvido chama-se Fawe. Ele é do gênero MMORPG (Massively Multiplayer Online Role-Playing Game) onde o jogador cria um personagem e pode seguir o seu próprio caminho dentro de um mundo virtual com outros jogadores.

Figura 5 - Logo do jogo.

Fonte: Douglas Jansen, 2020

O nome Fawe surgiu a partir das iniciais dos quatro elementos em inglês organizados de maneira que tenha um sentido fonético. Os elementos são ordenadamente Fire

(fogo), Air (ar), Water (água) e Earth (terra). Estes quatro elementos serão utilizados como base para toda a história do jogo.

4.2 ESPECIFICAÇÕES TÉCNICAS

O jogo deve funcionar como um site, sendo compatível com navegadores que suportem as tecnologias do HTML5, Canvas e Websocket. Por se tratar de um jogo bidimensional com gráficos simples, pode ser jogado em celulares e computadores sem muito poder de processamento.

4.3 HISTÓRIA DO JOGO

A história do jogo se passa em um mundo em constante destruição causado pela raça humana. Para evitar a destruição completa, os Deuses criaram um mecanismo. De todos os 12 continentes existentes no mundo, há sempre 3 ocultos. De tempos em tempos, os 3 continentes mais destruídos pela raça humana são engolidos pela terra (ou pelo mar, ou pelo fogo), e três novos continentes emergem, renovados e livres da influência humana.

Os jogadores começam o jogo num ambiente destruído, prestes a ser engolido chamado de Finraza. As primeiras missões visam salvar os remanescentes e seguir para as maiores cidades dos 3 continentes mais seguros e desenvolvidos.

4.4 GAMEPLAY

Para jogar Fawe é necessário primeiramente criar um personagem e vinculá-lo a uma conta. É muito importante que os jogadores tenham a possibilidade de criar mais de um personagem e que tenham o progresso de todos eles salvos.

4.5 AUTENTICAÇÃO COM GOOGLE

O processo de criação de contas e preenchimento de formulários muitas vezes é cansativo para usuários. Por isso, a única maneira de entrar no jogo deve ser através de uma conta Google. Assim, se o usuário já estiver conectado a uma conta Google em seu dispositivo, pode entrar no jogo sem a necessidade de preencher nenhum cadastro, com apenas um botão.

Além da praticidade para os usuários, usar este tipo de autenticação possui outros benefícios. Não é preciso desenvolver e nem dar manutenção em nada relacionado à registro e autenticação, tornando o processo de desenvolvimento simplificado. A segurança das contas como por exemplo proteção e recuperação de senha fica fora da responsabilidade do jogo.

O maior problema de usar apenas este tipo de autenticação é a dependência que ele causa. Se algum jogador não possuir uma conta Google e não estiver disposto a registrar uma, não terá acesso ao jogo.

4.6 CRIAÇÃO DE PERSONAGEM

Assim que o jogador estiver autenticado, pode criar um personagem. Na tela de criação de personagem são definidos o nome e as características físicas iniciais. Também podem ser definidos a tonalidade de pele e o estilo de cabelo.

Figura 6 - Tela de criação de personagem.

Fonte: Douglas Jansen, 2020

O nome escolhido deve ser único. Se o jogador digitar um nome que já exista no servidor, uma mensagem de erro solicitando que ele escreva outro nome deve aparecer. Caso o nome seja válido, o personagem deve ser criado com as características escolhidas.

4.7 ENTRANDO NO MUNDO

Após criado, o personagem é colocado automaticamente no mundo e o jogador começa a enfrentar seus primeiros desafios.

Figura 7 - Visão do personagem.

Fonte: Douglas Jansen, 2020

Fawe possui um mundo aberto. Os jogadores são livres para fazer missões, coletar recursos, interagir com outros jogadores e enfrentar criaturas inimigas. Essas atividades possuem diferentes tipos de recompensas fazendo com que o jogador evolua em diversos contextos diferentes.

4.8 MECÂNICA DO JOGO

A mecânica do jogo deve ser simples. Todas as ações podem ser realizadas apenas com mouse ou touchscreen. Assim que o personagem entrar no mundo, é possível andar apenas com um toque ou clique na localização desejada e automaticamente o personagem deve ir até o ponto desejado.

Os gráficos são estilo de visão *top down* (visto de cima), onde o jogador vê o seu personagem sempre no centro da tela. Alguns menus para visualizar chat de conversa, mochila e missões devem aparecer pelos cantos da tela. Também deve aparecer um mini-mapa na parte superior da tela para que o jogador possa se localizar no mundo.

4.8.1 NÍVEIS E ATRIBUTOS

O principal sistema de evolução do jogo é através de pontos de experiência adquiridos através de missões concluídas e inimigos derrotados. O personagem ganha um nível sempre que conseguir determinada quantidade de pontos. É necessário uma quantidade de pontos maior a cada nível atingido, tornando o jogo cada vez mais desafiador.

A cada nível atingido, o jogador fica mais forte aumentando seus pontos de vida. Ganha também um ponto de atributo que pode ser distribuído como preferir. Os atributos disponíveis são força, inteligência e destreza.

4.8.2 ITENS

O jogador pode também coletar itens e equipamentos concluindo missões, coletando recursos ou derrotando inimigos. Estes itens ficam armazenados no inventário pessoal de cada personagem. Cada item tem um tipo. Podendo ser um consumível, equipamento ou um recurso.

Os itens do tipo consumível dão algum tipo de bônus e deixam de existir caso sejam utilizados. As poções de cura, por exemplo, aumentam os pontos de vida do jogador quando utilizadas.

Figura 8 - Poções de cura consumíveis.

Fonte: Douglas Jansen, 2020

Os itens do tipo equipamento podem ser equipados e removidos a qualquer momento de acordo com a necessidade do jogador. Cada equipamento tem uma posição e

possui algum tipo de bônus como aumento de defesa ou velocidade. As posições disponíveis são cabeça, corpo, pernas, pés e mãos.

4.8.3 MISSÕES

As missões podem ser realizadas a partir de uma conversa com NPCs espalhados pelo mundo. Estas conversas se dão a partir de caixas de diálogos em que o jogador interage através de botões. Cada missão tem um desafio diferente e dá ao jogador alguma recompensa quando concluída.

Figura 9 - Diálogo com um NPC.

Fonte: Douglas Jansen, 2020

Os diálogos de missões devem ser ricos e interativos. As missões podem ser repetitivas (o jogador pode fazer quantas vezes quiser), ou únicas (só pode ser feita uma vez). Também é possível que missões só possam ser iniciadas a partir do término de outras missões

ou quando o jogador atingir um nível específico. A variedade e complexidade destas missões são importantes para o jogador consiga explorar diferentes locais do jogo.

4.8.4 RECURSOS

Determinados elementos do mundo do jogo são coletáveis. Objetos como árvores, plantas e pedras podem ser coletados se o jogador possuir um item específico para coletá-lo. Estes itens servem de recursos e podem ser utilizados para concluir missões durante o jogo.

Figura 10 - Alguns dos recursos disponíveis para serem coletados.

Fonte: Douglas Jansen, 2020

As árvores, por exemplo, podem ser coletadas se o personagem do jogador possuir um machado equipado. Da mesma forma, as minas podem ser coletadas com uma picareta. Cada objeto tem sua própria animação de coleta e pode deixar alguns recursos no chão.

Os itens a serem deixados no chão tem uma probabilidade configurável. Uma pedra poderá derrubar várias pequenas pedras e ter uma porcentagem rara de derrubar um diamante.

Figura 11 - Spritesheet de pedra sendo coletada.

Fonte: Douglas Jansen, 2020

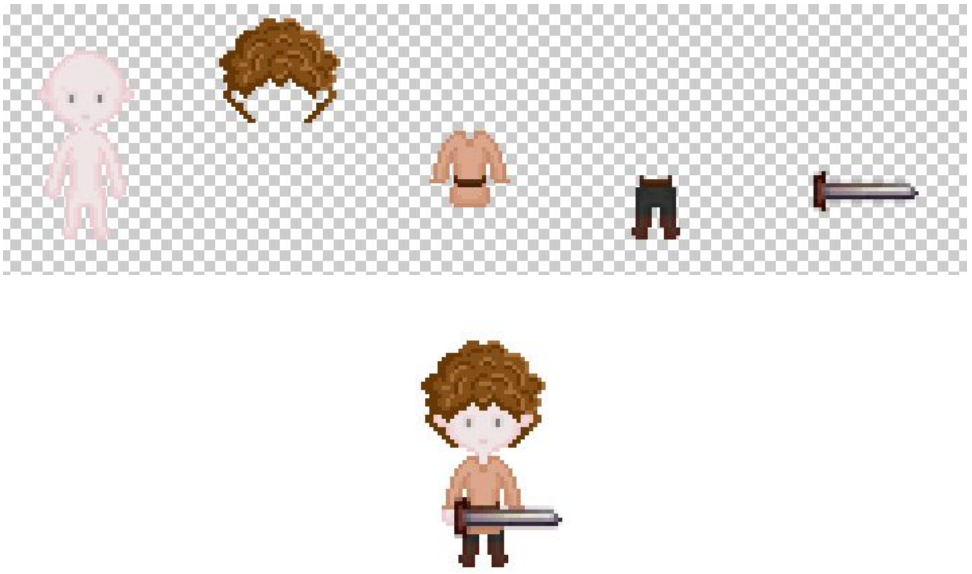
Para que os recursos não se esgotem, o mundo do jogo deve ser responsável por recriar estes recursos de tempo em tempo. Quando uma árvore for coletada, ela deve retornar depois de alguns minutos. O mesmo serve para todos os recursos. Alguns recursos podem ter uma raridade alta e demorar muito tempo para reaparecer.

4.9 PERSONAGENS

Os personagens são construídos no início do jogo. Tem características próprias como cor de cabelo, estilo de roupa e um nome único e ficam armazenados em um banco de dados.

A customização dos personagens se dá a partir da sobreposição de spritesheets. Há uma imagem para cada tipo de customização. A primeira é a de tonalidade de pele, que possui cinco tipos diferentes. Na frente do corpo ficam as spritesheets de cabelo, roupas e armas. As roupas e armas estão disponíveis de diversas maneiras para serem conseguidas durante o jogo.

Figura 12 - Spritesheets formando um personagem.



Fonte: Douglas Jansen, 2020

4.10 NPCS

NPC (em inglês: non-player character) são personagens não jogáveis que habitam o mundo do jogo. No mundo do Fawe, eles são habitantes que empregam diferentes tipos de atividades para jogadores que interajam com eles. A maioria dessas atividades são missões em que o jogador troca itens coletados do mundo por alguma recompensa.

Assim que o jogador entra no mundo pela primeira vez, ele deve encontrar o primeiro NPC. Este deve lhe mostrar uma mensagem de boas vindas e introduzi-lo a história do jogo. Também deve lhe dar uma arma inicial para que ele possa enfrentar seus primeiros inimigos.

Conforme o jogador explorar o mundo, deve encontrar novos NPCs. Cada um deve lhe dar diferentes tipos de tarefas e desafios.

4.11 CONTROLES

As principais ações do jogo como andar, atacar, pegar itens do chão e falar com NPCs podem ser realizadas apenas com mouse ou touchscreen.

Para jogadores que estiverem usando teclado, alguns atalhos de acesso rápido estão disponíveis. Os números de 1 a 9 servem para atalhos de uso de itens e habilidades. A tecla ESC fecha todo tipo diálogo aberto e a tecla Enter deve avançar diálogos com NPCs e enviar mensagem no chat.

4.12 CÂMERAS

A câmera do jogo é sempre fixa com o personagem do jogador no centro da tela. Ela deve se mover de acordo com suas movimentações pelo mundo. Quando o jogador tocar na tela para se mover, a câmera deve acompanhá-lo automaticamente de maneira que continue parado no centro da tela. Este tipo de câmera funciona como um campo de visão, onde o jogador vê o cenário se movendo ao seu redor e tem a sensação de que está explorando.

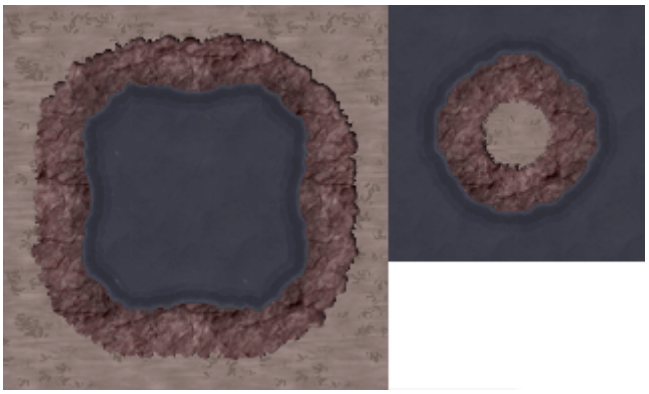
4.13 UNIVERSO DO JOGO

Os mapas do jogo são desenvolvidos utilizando um sistema quadriculado de imagens. Estas imagens também podem ser chamadas de *tiles*. Três matrizes posicionadas uma sobre a outra contém peças de um quebra cabeça que formam o mapa.

4.13.1 TERRENO

A primeira camada contém os *tiles* de terreno, que podem ser basicamente água, terra ou estrada. A água serve apenas como um bloqueio de movimentação para os jogadores. É possível se movimentar apenas pelas estradas e pela terra.

Figura 13 - Spritesheet de água e terra.



Fonte: Douglas Jansen, 2020

4.13.2 UNIDADES

A segunda camada contém os *tiles* de unidade, que podem ser qualquer objeto a ser posicionado no mundo. Estes objetos podem ser pedras, árvores, flores, paredes, cercas etc.

Figura 14 - Alguns elementos do mundo do jogo.

Fonte: Douglas Jansen, 2020

Alguns tipos de *tiles* de unidade podem ter uma estrutura de spritesheet personalizada. A cerca, por exemplo, tem uma estrutura de oito desenhos que se encaixam conforme sua posição no mundo.

Figura 15 - Spritesheet de cerca.

Fonte: Douglas Jansen, 2020

As árvores e alguns outros elementos tem um desenho que ocupam dois quadrados, mas fisicamente no mundo do jogo ocupam apenas um. Ou seja, o jogador pode passar por trás destes elementos, dando uma sensação de maior perspectiva.

4.13.3 TELHADOS

A terceira camada serve como uma camada de telhado que é posicionada na frente dos personagens do mundo. Esta camada deve aparecer e desaparecer automaticamente de acordo com a posição do personagem no mundo. A imagem mostra um exemplo de como deve ser essa funcionalidade:

Figura 16 - Imagem mostrando a camada de telhado aparecendo quando o jogador está fora da casa e desaparecendo quando o jogador está dentro da casa.

Fonte: Douglas Jansen, 2020

A união dessas camadas deve formar o mapa do jogo. Além dos elementos de formação do mapa também existirão os jogadores, NPCs e inimigos.

Figura 17 - Combinação de sprites formando um mapa.

Fonte: Douglas Jansen, 2020

4.14 INIMIGOS

Os inimigos têm uma spritesheet um pouco mais simples que a dos personagens. Eles têm uma inteligência artificial que pode ser configurada para que eles se defendam, sigam jogadores e caminhem pelo mundo. Cada tipo de inimigo possui uma dificuldade e pode dar diferentes tipos de recompensas ao ser derrotado.

Figura 18 - Spritesheet de um dos inimigos do jogo.

Fonte: Douglas Jansen, 2020

Assim como os personagens, os monstros também possuem níveis e atributos e podem ser configurados com velocidade de ataque e de movimento. Algumas criaturas são do tipo agressivas, atacando jogadores que se aproximam.

Quando um monstro do mundo é derrotado, depois de um determinado tempo, nasce um outro num local próximo. Esse mecanismo deve garantir uma população de criaturas equilibrada no mundo, tornando possível a criação de localidades específicas adicionando diferentes níveis de dificuldade.

4.15 INTERFACE

A interface do jogo está dividida entre um mini-mapa na parte superior, alguns botões de utilidades na parte inferior, e os gráficos do jogo ocupando o restante da tela. Cada botão possui uma funcionalidade específica de atalho, informação ou ação.

Figura 19 - Interface do Fawe.

Fonte: Douglas Jansen, 2020

A interface gráfica deve se ajustar automaticamente de acordo com o tipo e tamanho de tela. Nos computadores, por exemplo, onde a tela é horizontal (widescreen), os menus devem acompanhar o formato da tela e se ajustar nos cantos.

5 MODELAGEM DO JOGO

Neste capítulo é apresentado a modelagem do jogo. Primeiro são definidos os requisitos do jogo, que são divididos entre requisitos funcionais e não funcionais. Depois, são elaborados os diagramas de funcionamento do jogo. Por último, é apresentado a modelagem do banco de dados.

5.1 ESPECIFICAÇÃO DE REQUISITOS

Para Sommerville (2011), os requisitos de um software são divididos em requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais, de acordo com Sommerville (2011, p. 59) “São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações”.

Já os requisitos não funcionais, Segundo Teixeira (2018), “Tangem a exigência técnica de um ambiente, como sendo: aspectos de segurança do sistema, desempenho, prevenção de falhas e etc”. Ou seja, não estão ligados diretamente ao detalhamento das funcionalidades de um software. Para Sommerville (2011, p. 60), os requisitos não funcionais “não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários”.

5.1.1 REQUISITOS FUNCIONAIS

O jogo possui funcionalidades que podem ser divididas em requisitos. Todas as ações que o jogador faz durante o jogo são especificadas nesta parte. O quadro a seguir detalha os requisitos funcionais elaborados.

Quadro 1 - Requisitos funcionais do jogo Fawe.

Identificação	**Descrição**
RF 001	O jogador pode se autenticar com uma conta Google.
RF 002	O jogador pode criar vários personagem.
RF 003	O jogador pode escolher diferentes opções de cabelo e tonalidades de pele para o personagem.
RF 004	O jogador deve conseguir escrever o nome desejado para o seu personagem. Este nome deve ser único. Ou seja, nenhum outro personagem de nenhum outro jogador poderá utilizá-lo.
RF 005	O jogador pode selecionar um personagem.
RF 006	O jogador pode movimentar seu personagem dentro do jogo.
RF 007	O jogador pode atacar inimigos com seu personagem.
RF 008	O jogador visualiza um mini-mapa no canto superior da tela.
RF 009	O jogador pode coletar itens do chão com seu personagem.
RF 010	O jogador pode coletar recursos como madeira.
RF 011	O jogador pode visualizar seus itens adquiridos no jogo em uma mochila.
RF 012	O jogador pode enviar e receber mensagens de outros jogadores no chat.
RF 013	O jogador pode equipar itens como armas e armaduras no seu personagem.
RF 014	O jogador conversa com NPCs dentro do jogo.
RF 015	O jogador pode visualizar os atributos do seu personagem dentro do jogo.
RF 016	O jogador recebe pontos de experiência a cada inimigo derrotado.
RF 017	O jogador morre toda vez que seus pontos de vida chegarem a zero.
RF 018	O jogador volta ao início do mundo sempre que morrer.
RF 019	O jogador visualiza uma barra de atalhos na tela do jogo.

Fonte: Autoral, 2020.

Estes requisitos foram elaborados com o objetivo de se obter uma experiência de jogabilidade mínima ao estilo clássico de um MMORPG. Ou seja, o jogador cria seu personagem e pode explorar o mundo do jogo e evoluir conforme encontra novos desafios.

5.1.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais do jogo são as partes técnicas que não estão diretamente ligadas à funcionalidades específicas. Envolvem principalmente o ambiente e as tecnologias utilizadas para o desenvolvimento. O quadro a seguir detalha os requisitos não funcionais do jogo a ser desenvolvido.

Quadro 2 - Requisitos não funcionais do jogo Fawe.

Identificação	**Descrição**
RNF 001	O jogo funciona em navegadores de internet atualizados.
RNF 002	O servidor do jogo é compatível com ambiente linux.
RNF 003	É possível jogar Fawe em computadores e celulares
RNF 004	É possível jogar Fawe com mouse ou touchscreen.
RNF 005	É possível disponibilizar o jogo em uma rede.
RNF 006	O jogo utiliza HTML5 Canvas para parte gráfica .
RNF 007	O jogo utiliza Websocket para comunicação em tempo real.
RNF 008	O jogo usa o framework Ruby on Rails para a parte de autenticação e gerenciamento de personagens.

Fonte: Autoral, 2020.

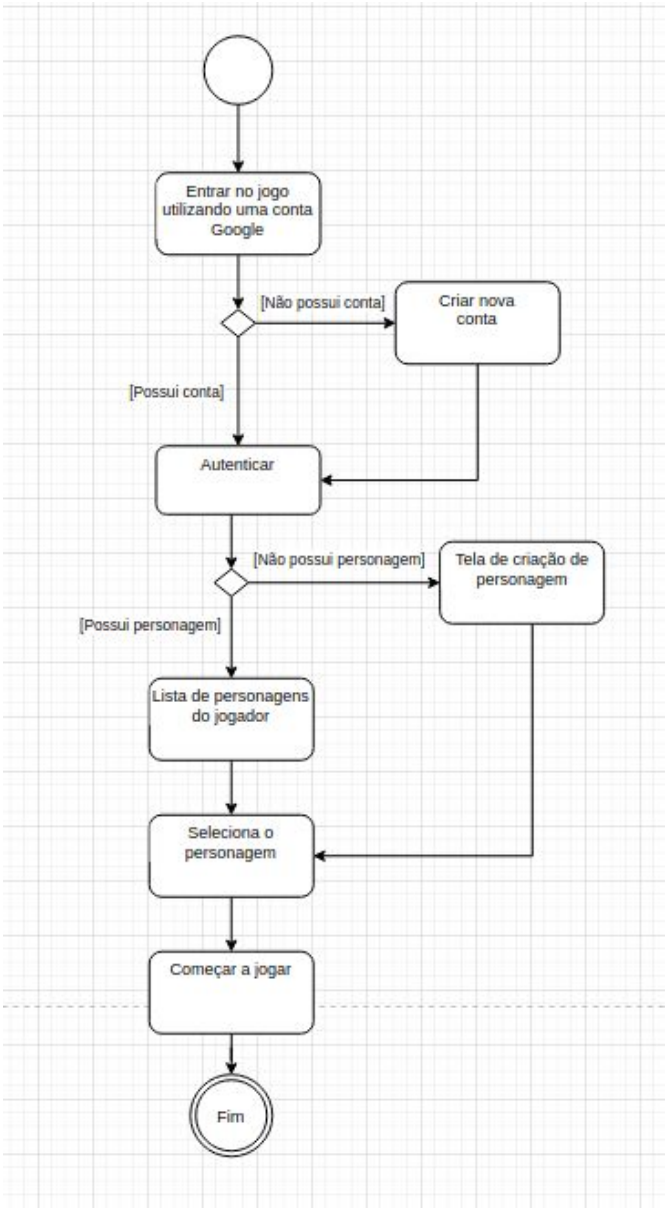
5.2 DIAGRAMA DE ATIVIDADES

Segundo Sommerville (2011, p. 85), “Os diagramas de atividades são destinados a mostrar as atividades que compõem um processo de sistema e o fluxo de controle de uma atividade para a outra.”

5.2.1 AUTENTICAÇÃO, SELEÇÃO E CRIAÇÃO DE PERSONAGEM

Antes do jogador entrar no mundo e iniciar a jogar de fato, é preciso que ele entre se autentique e crie um personagem. É muito importante que estas ações não sejam cansativa para o jogador. O diagrama de atividades a seguir detalha o fluxo até que o jogador comece a jogar.

Figura 20 - Diagrama de atividades do momento da autenticação dos jogadores até o início do gameplay.

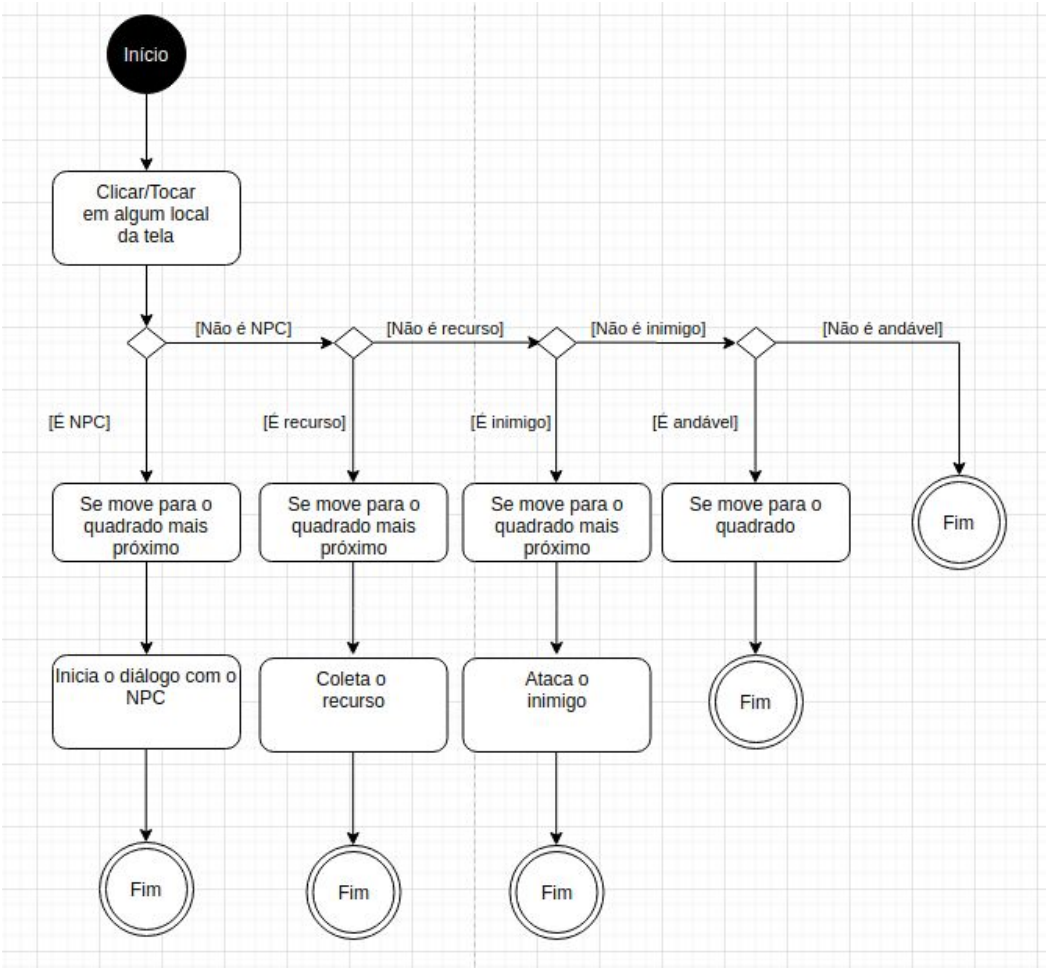


Fonte: Autoral, 2020.

5.2.2 INTERAÇÃO DE CLIQUE/TOQUE NA TELA

A principal ação do jogo acontece através de um clique ou toque em algum local desejado na tela do jogo. Está ação deve reconhecer o objeto selecionado e fazer as interações necessárias.

Figura 21 - Diagrama de atividades da interação de clique na tela.



Fonte: Autoral, 2020.

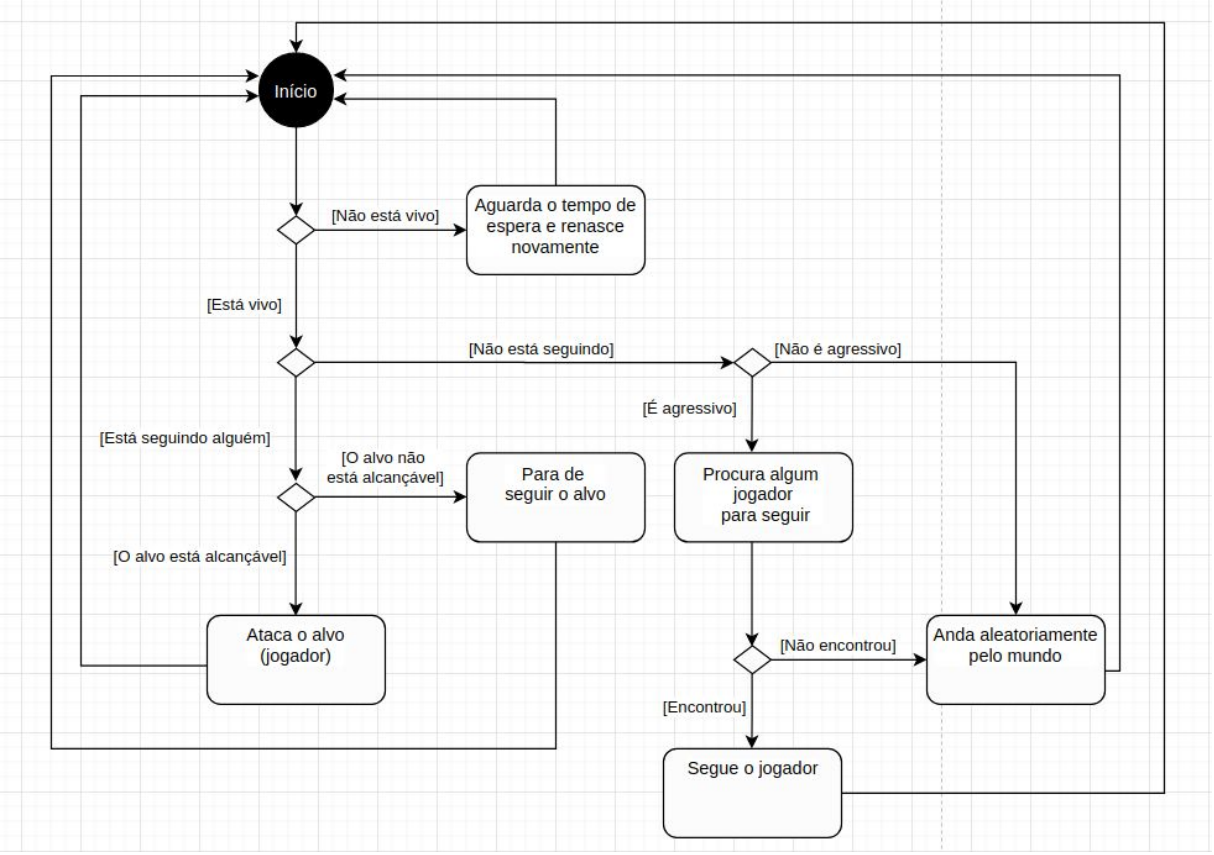
O diagrama mostra como ações de clique ou toque na tela são tratadas dentro do jogo. O personagem é direcionado de acordo com o elemento selecionado. Quando seleciona um NPC (personagem não jogável) , deve se mover para perto dele e iniciar uma conversa. Quando seleciona um recurso, deve se mover para perto dele e coletá-lo. Quando seleciona um inimigo, deve se mover para perto dele e atacá-lo. Em último caso, deve apenas se mover.

5.2.3 INTELIGÊNCIA DOS INIMIGOS

O mundo do jogo é habitado por diferentes inimigos. Eles possuem uma inteligência básica. São capazes de se mover aleatoriamente e se defender quando são

atacados. Alguns tipos de inimigos podem ser configurados para serem agressivos e ataquem jogadores que se aproximem. O diagrama de atividades a seguir demonstra como funciona esta inteligência.

Figura 22 - Diagrama de atividades da inteligência dos inimigos.



Fonte: Autoral, 2020.

Inimigos mortos por jogadores ficam invisíveis aguardando um tempo para renascerem. Este tempo é controlado por um algoritmo que também controla toda a sua inteligência. Os inimigos devem andar aleatoriamente pelo mundo e se defender quando atacados. Devem seguir jogadores até um determinado raio de alcance, caso contrário voltam para sua posição original e novamente andam aleatoriamente.

5.3 BANCO DE DADOS

O sistema de gerenciamento de banco de dados (SGBD) utilizado para o jogo é o MySQL. Sendo um dos bancos de dados mais utilizados no mundo, DB Engines (2020), é um banco relacional que utiliza SQL como linguagem de interface. Foi escolhido por ser uma excelente alternativa gratuita para o desenvolvimento de bancos de dados relacionais.

É importante ressaltar que a maior parte das informações do jogo ficam na memória do servidor e por questões de performance e simplicidade no desenvolvimento não são gravadas no banco de dados. São essas informações como todos os NPCs, itens, mapas e toda parte de conteúdo do mundo do jogo que são controladas no servidor websocket.

Portanto, o banco de dados do jogo possui apenas 3 tabelas principais: Accounts (Contas), Characters (Personagens) e Character Items (Itens dos personagens). Essas 3 tabelas são suficientes para que os jogadores não percam seu progresso durante o jogo.

Para criação do modelo, foi utilizado a ferramenta Drawio, disponível no site[1].

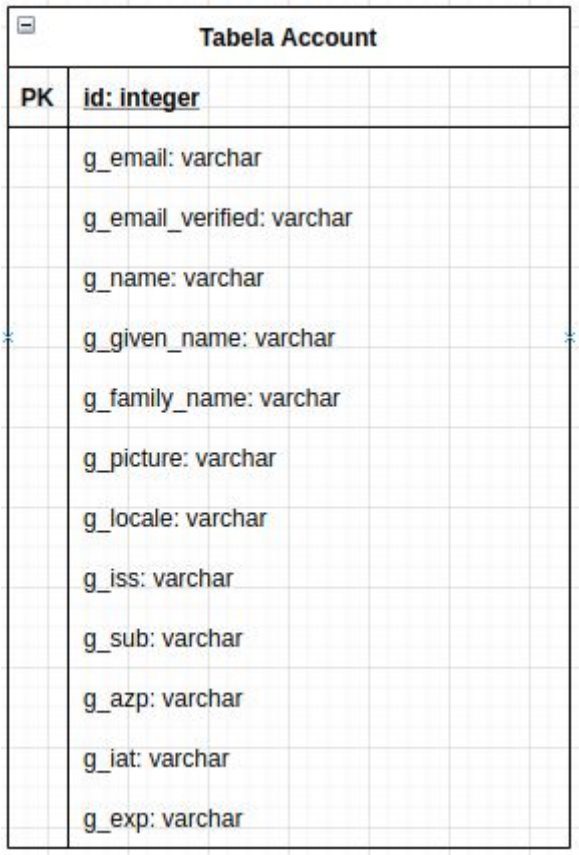
5.3.1 TABELA ACCOUNT

As contas identificam os usuários, que são os jogadores. Precisam de alguns campos de identificação como email e campos necessários para autenticação do Google.

As colunas são nomeadas iniciando com “g_” para representar que são campos referentes a integração com login por Google. Futuramente poderão ser adicionados campos de outros tipos de autenticação ou até mesmo uma criação de senha para login próprio.

[1] https://app.diagrams.net/

Figura 23 - Tabela de Account (contas).



Fonte: Autoral, 2020.

Os dados referentes aos campos mostrados na figura anterior são fornecidos pela API do Google. Os primeiros campos são dados de identificação e exibição como email, se o e-mail está verificado ou não, nome, sobrenome, link de foto pública da conta Google e país. Os campos restantes são referentes a autenticação e identificação. O campo "g_sub", por exemplo, é um id único de identificação com o google.

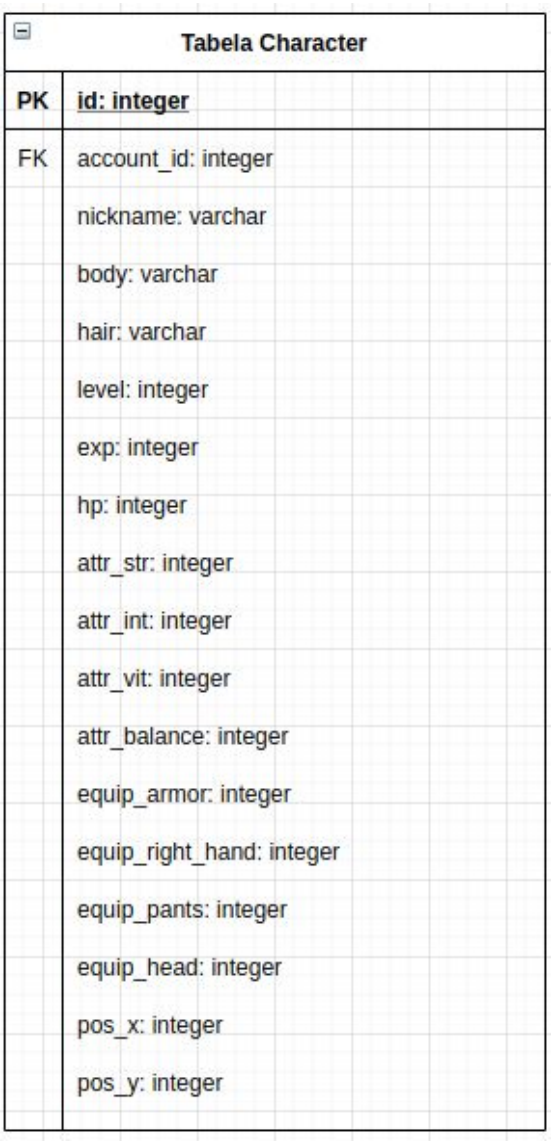
A documentação completa com detalhes de cada um dos campos encontra-se no link[2].

[2] https://developers.google.com/identity/protocols/oauth2/openid-connect

5.3.2 TABELA CHARACTER

A tabela de Character armazena todos os personagens criados pelos usuários. Possui todas informações que precisam ser salvas para que o jogador não perca o progresso do personagem. Os campos dessa tabela envolvem principalmente os dados característicos do personagem como nome, atributos, equipamentos e posição no mundo.

Figura 24 - Tabela de Character (personagens).



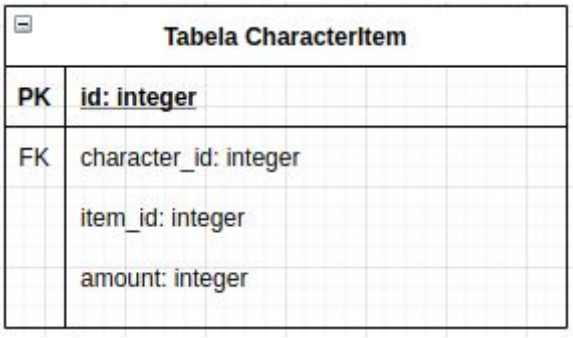
Fonte: Autoral, 2020.

O primeiro campo, *account_id*, liga o personagem a uma conta de um jogador. O campo *nickname* grava o nome do personagem. Os campos *body* e *hair* identificam as características físicas do personagem. Os campos que iniciam com *attr* são referentes aos atributos do personagem. Os campos que iniciam com *equip* são referentes aos equipamentos. Os campos *pos_x* e *pos_y* mostram a última posição do personagem no mundo.

5.3.3 TABELA CHARACTER ITEM

A tabela de Character Item armazena todos os itens dos personagens. Todas as interações com a mochila do personagem do jogador são registradas nessa tabela. A cada vez que um jogador pega um item do chão, é adicionado ou alterado um registro nela.

Figura 25 - Tabela de CharacterItem (itens dos personagens).



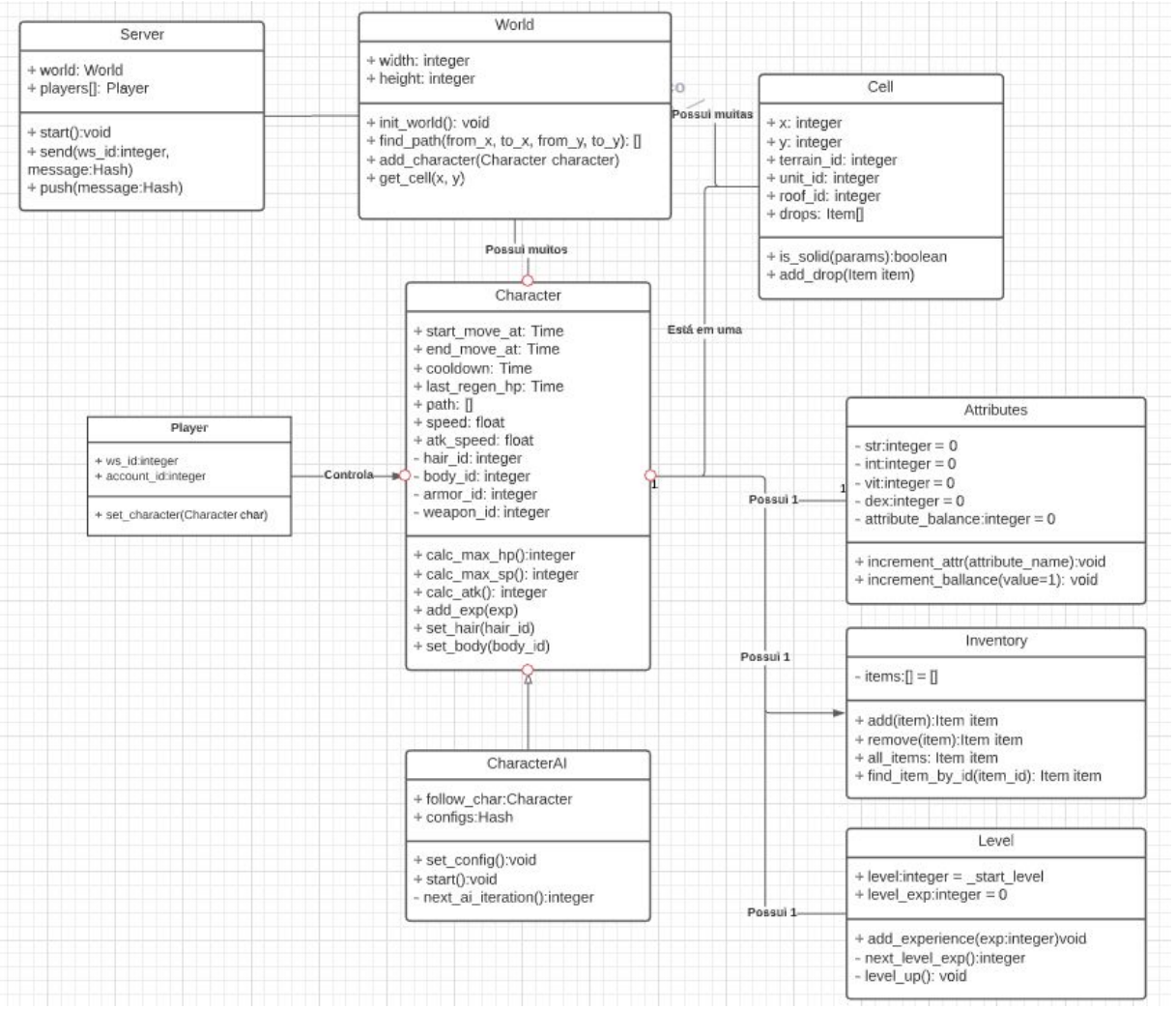
Fonte: Autoral, 2020.

Composta por apenas 4 colunas, a coluna *id* identifica o registro na tabela. A coluna *character_id* identifica a qual personagem pertence os itens deste registro. A coluna *item_id* indica qual é o item. Por último, a coluna *amount* indica a quantidade de itens.

5.4 DIAGRAMA DE CLASSES

O servidor de Fawe é controlado pelo seguinte diagrama de classes:

Figura 26 - Modelagem UML do jogo.



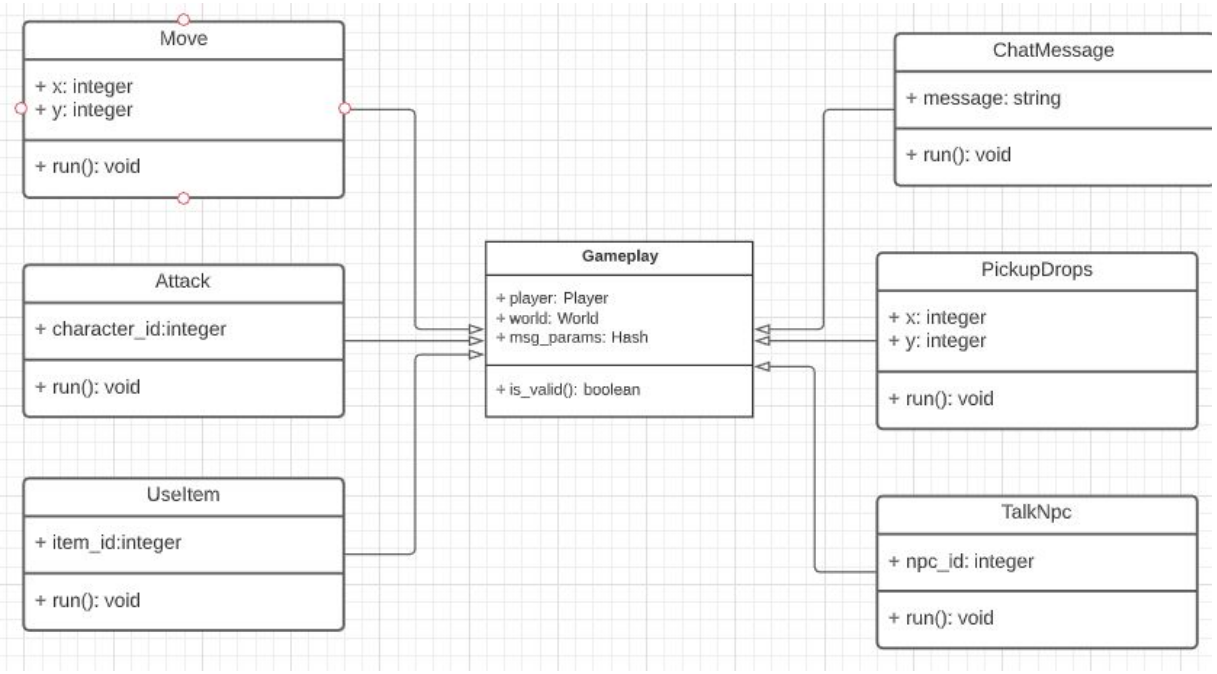
Fonte: Autoral, 2020.

O diagrama demonstra o funcionamento da estrutura de dados do servidor. O servidor websocket é iniciado pelo método *start*. Assim que iniciado, carrega o mundo do jogo para a memória e começa a aguardar mensagens. Os jogadores, depois de autenticados,

entram na lista de personagens do mundo e através de mensagens podem fazer suas interações.

As interações do jogo são chamadas de gameplay. Cada gameplay equivale a uma classe e um tipo de mensagem diferente. As gameplays originam de uma classe abstrata *Gameplay*. Estas classes existem para organizar as ações do jogo em classes únicas. Elas precisam apenas implementar o método *run*.

Figura 27 - Modelagem UML dos tipos de jogadas.



Fonte: Autoral, 2020.

A figura mostra todas as jogadas do jogo com seus respectivos parâmetros. A jogada Move por exemplo, requer dois parâmetros do tipo inteiro, *x* e *y*, que determinam o local no mundo em que o jogador deseja se mover. A jogada Attack requer um parâmetro *character_id* que identifica qual personagem o jogador quer atacar. A jogada UseItem requer um parâmetro *item_id* que identifica qual item o jogador quer usar. A jogada Chat Message recebe um parâmetro de texto que envia uma mensagem no chat do jogo.

Todas estas jogadas são executadas pelo método *run* que já terá acesso ao mundo do jogo e todos com todos os jogadores. Assim, poderá fazer todas as validações e alterações necessárias e enviar as mensagens para quem for necessário.

6 DESENVOLVIMENTO

A primeira parte do desenvolvimento são as telas necessárias para que os jogadores se autentiquem e entrem no mundo do jogo. São elas: autenticação, criação e seleção de personagem. Essa etapa é muito similar ao desenvolvimento de uma aplicação web contendo formulários e rotas de navegação no navegador de internet.

Em seguida vem a parte gráfica do jogo. O jogador só tem acesso a ela depois de passar pela autenticação e criação de personagem. Os gráficos do jogo são controlados em tempo real por um servidor.

6.1 TECNOLOGIAS UTILIZADAS

O projeto foi desenvolvido em ambiente linux com tecnologias gratuitas e de código aberto.

Fawe possui todas as características básicas de uma aplicação web. Páginas, preenchimento de formulários, cookies, navegação e outros detalhes. Para lidar com esse tipo de ambiente foi utilizado o framework Ruby on Rails.

Ruby on Rails é muito prático para criar aplicações web robustas de um jeito rápido. Depois de instalado, é possível gerar uma aplicação completa com apenas algumas linhas de comando.

Para parte gráfica do jogo foi utilizado Javascript. Com as bibliotecas padrões, é possível mostrar todos os gráficos e se comunicar com o servidor websocket.

Por último, para lidar com a comunicação em tempo real foi utilizado um servidor Websocket para a linguagem Ruby chamado EM-Websocket.

6.2 AUTENTICAÇÃO E CRIAÇÃO DE PERSONAGEM

A primeira etapa de desenvolvimento consiste em iniciar uma aplicação web para lidar com o gerenciamento de personagens do jogador. Para criar um projeto em Ruby on Rails basta executar o comando *rails new* no terminal. Um diretório será criado contendo arquivos iniciais de uma aplicação web.

As seguintes rotas foram definidas para o servidor web:

Quadro 3 - Rotas de navegação do servidor web.

ROTA	MÉTODO	DESCRIÇÃO
/	GET	Tela inicial com botão para entrar no jogo.
/auth/signin	POST	Autenticação com google.
/auth/signout	GET	Logout
/new_character	GET	Tela de criação de personagem.
/new_character	POST	Envio dos dados para criação de personagem.
/characters	GET	Lista de personagens vinculados a conta do usuário logado.
/play	GET	Rota onde o jogo será executado.

Fonte: Autoral, 2020.

No Ruby on Rails não é necessário utilizar linguagem SQL para trabalhar com banco de dados. Validações de campos e relacionamentos entre tabelas podem ser feitas de maneira simples utilizando métodos do Ruby on Rails.

Figura 28 - Código na linguagem Ruby mostrando o modelo personagem no servidor web.

```ruby
class Character < ApplicationRecord
  has_many :char_items
  validates :nickname, uniqueness: true
end
```

Fonte: Autoral, 2020.

O código mostra o modelo do personagem (Character) herdando a classe *ApplicationRecord* que é o modelo base do Ruby on Rails. Ela possui algumas funções que facilitam a validação de parâmetros e associação entre tabelas. A função *has_many* conecta a um outro modelo utilizando a regra relacional um para muitos.

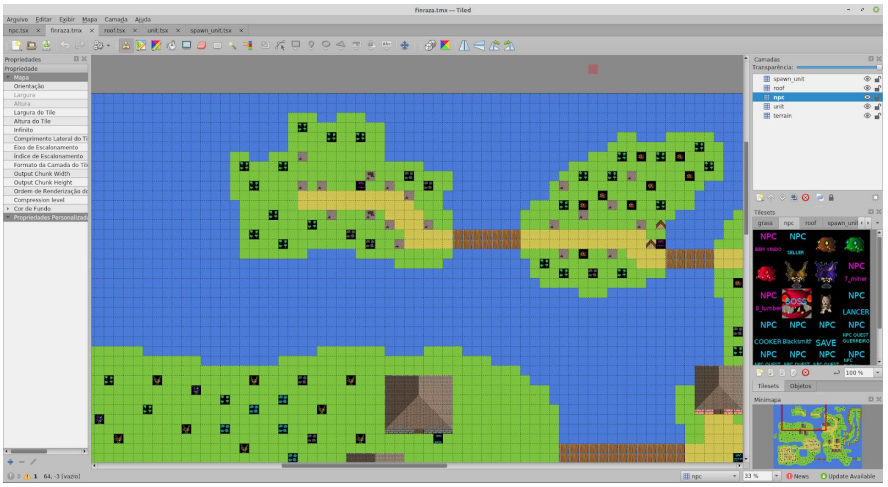
A etapa de autenticação foi desenvolvida com a biblioteca Google Sign-In que faz todo o processo de login ou registro e retorna o usuário conectado. Com os dados do usuário conectado é criada uma sessão para identificá-lo.

Depois de autenticado, o personagem vai para a tela de criação de personagem. Está tela (representada na Figura 6), é um formulário HTML com Javascript para fazer a navegação entre os botões de troca de características do personagem.

6.3 CONSTRUINDO O MUNDO DO JOGO

O Tiled é uma ferramenta para edição de mapas em jogos 2D. Nele é possível criar e organizar um mapa de imagens com diversas camadas depois exportá-lo em formato json para que ele possa ser interpretado. O arquivo exportado possui uma lista de matrizes onde todos os valores são números. Cada item da lista representa uma camada do mapa. Cada camada é uma matriz de imagens com um número de identificação.

Figura 29 - Captura de tela do mapa do Fawe na ferramenta Tiled.



Fonte: Autoral, 2020.

O mapa criado no Tiled contém todos os elementos do jogo com suas posições. Estes elementos são divididos em três camadas principais: terrenos, unidades e telhados. Quando exportado, ele vira um arquivo em formato *json* e pode ser lido pelo jogo.

Em seguida, esse arquivo em *json* é carregado pelo servidor do jogo durante a sua inicialização. Este *json* é transformado em uma estrutura de objetos na linguagem Ruby para que possa ser manipulado e depois retornado em *json* para os jogadores. Esta estrutura é basicamente uma lista de objetos do tipo Cell (célula). Cada célula equivale a um quadrado do mundo do jogo. A imagem a seguir mostra uma parte da classe desenvolvida:

Figura 30 - Parte da classe Cell (célula).

```ruby
class Cell

  attr_reader :unit_id, :terrain_id, :drops, :x, :y

  def initialize(terrain_id, unit_id, x, y, roof_id = nil)
    @terrain_id = terrain_id
    @unit_id = unit_id
    @roof_id = roof_id
    @drops = []
    @x = x
    @y = y
  end

  def distance_to(cell_to)
    Math.sqrt(((cell_to.x-x)**2) + ((cell_to.y-y)**2))
  end
```

Fonte: Autoral, 2020.

Cada célula possui um terreno (terrain id) que pode ser água ou grama, unidade (unit id) que é algo acima deste terreno como árvores, pedras ou paredes e um telhado que serve apenas para criar uma camada gráfica de telhado nas casas do mundo do jogo. É através dessa estrutura que o servidor consegue definir se os jogadores podem ou não executar suas movimentações. Também consegue modificar itens do cenário como remover árvores.

6.4 INICIANDO O SERVIDOR

O servidor é iniciado como um script Ruby no terminal com o comando *ruby server.rb*. Este script fica no arquivo *server.rb* e inicia o servidor Websocket. Para iniciar o servidor é necessário determinar a porta e o IP desejados. Depois, é possível implementar os callbacks de novas mensagens, novas conexões ou conexões finalizadas.

Figura 31 - Código em Ruby do servidor websocket sendo iniciado.

```ruby
EventMachine::WebSocket.start(
  host: '0.0.0.0',
  port: 5000
) do |ws|

  ws.onmessage { |msg|
    begin
      begin
        json_msg = JSON.parse(msg)
      rescue
        Log.alert "invalid json: #{msg}"
      end

      case json_msg['message']
      when 'auth'
        if @players[ws.object_id].nil?
          Authentication.auth(json_msg, self, ws, @world, @players)
        end
      when 'gameplay'
        unless @players[ws.object_id].nil?
          @world.gameplay json_msg, @players[ws.object_id], @world, ws
        end
      else
        Log.alert "message not found: #{msg}"
      end
    rescue => exception
```

Fonte: Autoral, 2020.

Depois de iniciado, o servidor fica aguardando novas conexões e novas mensagens. Conexões estabelecidas podem enviar mensagens que são recebidas em texto e convertidas para o formato *json*. Estas mensagens contêm informações que são enviadas para o objeto *world* que gerencia todas as jogadas.

6.5 DESENHANDO OS GRÁFICOS

Os gráficos do jogo são desenhados dentro de um game loop em javascript com um elemento HTML5 Canvas. Todas as spritesheets do jogo possuem tamanho múltiplo de 64

pixels. Com isso é possível desenhar e encaixar matematicamente os elementos na tela como se fossem peças de um quebra-cabeças.

Estes elementos são desenhados a partir de dados recebidos pelo servidor. Os principais dados são o mundo, representado pela variável *world* e os personagens, representados pela variável *characters*. O mundo (variável *world*), possui uma lista com todas as células a serem desenhadas. Os personagens (variável *characters*), possuem uma lista com todos os jogadores, NPCs e inimigos do mundo.

Figura 32 - Código em Javascript desenhando os terrenos do mundo do jogo.

```
function draw_terrains(fx, tx, fy, ty) {
    for(var x = fx; x < tx; x++) {
        for(var y = fy; y < ty; y++) {
            cell = find_cell(x, y);
            if(cell) {
                var pos_x = parseInt(cell['x'] * cell_size + scroll_x);
                var pos_y = parseInt(cell['y'] * cell_size + scroll_y);
                var terrain = TERRAINS[cell['t']];
                ctx.drawImage(terrain['sprite'], cell['pos_x_t'], cell['pos_y_t'],
                    cell_size, cell_size, pos_x, pos_y, cell_size, cell_size);
            }
        }
    }
}
```

Fonte: Autoral, 2020.

O código da Figura 32 mostra como são desenhados os terrenos (água, grama e estradas) do jogo. As variáveis *fx, tx, fy e ty* servem para que o jogo desenhe apenas os quadrados da tela do jogador que fica ao redor do personagem, e não o mundo inteiro. As variáveis *scroll_x* e *scroll_y* definem a posição do personagem do jogador. Com isso, a visão da tela fica sempre centralizada no personagem e só é desenhado o que é necessário.

É muito importante que seja desenhado na tela apenas o necessário. Em geral os jogos utilizam muito processamento e uma das chamadas que mais causam pico de processamento é a de desenho. No caso do Canvas, ela é chamada pela função *drawImage*.

6.6 IMPLEMENTANDO A COMUNICAÇÃO EM TEMPO REAL

A mensagem mais simples para ser a implementada é a de chat. O servidor deve apenas repassar para todos os jogadores conectados a mensagem digitada pelo jogador em um campo de texto *input* na interface gráfica.

Figura 33 - Código em Ruby mostrando a implementação da mensagem de chat.

```ruby
module Gameplay
  class ChatMessage < Gameplay
    def run
      if params['message'].is_a? String
        world.server.channel_push('all', {
          message: 'chat',
          nickname: player.character.nickname,
          message: params['message']
        })
      end
    end
  end
end
```

Fonte: Autoral, 2020.

As mensagens são enviadas para os clientes através do método *channel_push* mostrado na figura. Este método dispara uma mensagem para cada cliente conectado ao servidor. Em seguida, a mensagem é recebida por todos os clientes também em formato *json* e pode ser tratada.

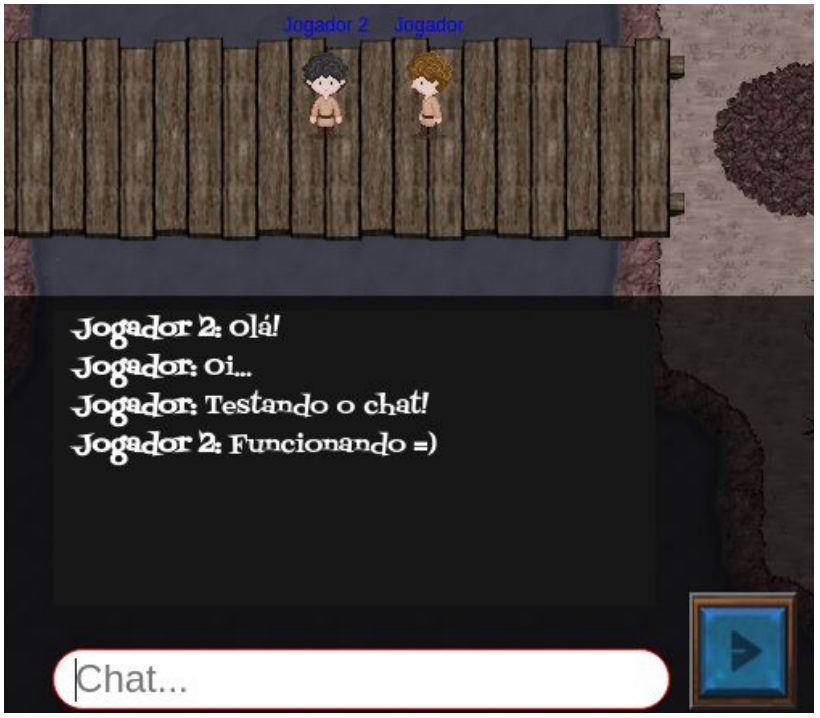
Figura 34 - Código em Javascript recebendo a mensagem de chat do servidor.

```javascript
case 'chat':
    var messages = document.getElementById("chat");
    messages.innerHTML += "" +
        "<div><b> " + msg['nickname'] + ": </b>" + msg['message'] + "</div>";
    messages.scrollTop = messages.scrollHeight;
    break;
```

Fonte: Autoral, 2020.

No código da figura 34, caso a mensagem seja do tipo *chat*, o Javascript irá adicionar uma nova mensagem no elemento HTML que ficam as mensagens do jogador. Esta mensagem conta com os campos *nickname* e *message* que identificam o jogador que enviou a mensagem e o conteúdo dela. A figura a seguir mostra o chat funcionando:

Figura 35 - Captura de tela mostrando dois jogadores conversando.



Fonte: Autoral, 2020.

O fluxo mostrado no desenvolvimento do chat segue para todas as mensagens do jogo. O que vai mudando são as ações nas estruturas de dados do jogo. Cada mensagem tem seus próprios parâmetros e ações.

A mensagem de movimentação requer dois parâmetros principais: *x* e *y*. Eles determinam a posição na qual o jogador deseja ir no mundo. O servidor já sabe qual o jogador que está se movimentando porque ele foi anteriormente autenticado. Assim, sabendo qual

jogador deseja se movimentar e para onde ele quer ir, é possível traçar uma rota com um algoritmo de pathfinding[3] e enviá-la para todos os clientes conectados. Os clientes conectados recebem essa rota e aplicam no personagem.

O combate do jogo é feito apenas com o parâmetro *character_id*. O servidor valida quem o personagem que o jogador deseja atacar através de algumas condições: O inimigo deve estar a uma distância de até um quadrado e não pode ser o próprio jogador. Validando estas condições, o servidor pode reduzir a vida do inimigo de acordo com a força do jogador e disparar mensagens avisando a todos os clientes que um ataque foi realizado. Os clientes recebem a mensagem de ataque e aplicam as animações na tela do jogo.

6.7 CRIANDO A INTELIGÊNCIA DOS INIMIGOS

Os inimigos, como mostra a Figura 22, possuem uma inteligência simples que pode ser controlada por um algoritmo em loop. No cliente, todos os personagens e inimigos são armazenados na mesma variável *characters*. Isso facilita a atualização de todas as ações de ataque e movimento, pois são as mesmas para todos.

Assim, para criar uma inteligência para um inimigo basta instanciar um personagem a mais no mundo com uma sprite de inimigo e aplicar uma lógica com chamadas para as próprias jogadas do servidor.

[3] Algoritmo que busca um caminho entre dois pontos. O algoritmo utilizado pelo Fawe encontra-se no github no link: https://github.com/Pommespanzer/astar-ruby

7 APLICAÇÃO DE QUESTIONÁRIOS

Este capítulo aborda uma aplicação de questionários utilizando a escala de Likert com jogadores voluntários.

7.1 SOBRE A DISPOSIÇÃO DO QUESTIONÁRIO

Para a criação das perguntas, foi utilizado critérios com base em Gil (2008). As perguntas então, devem ser claras, concretas e precisas. Não devendo possuir múltiplas interpretações que confundam os entrevistados. Os enunciados devem tratar de um assunto por vez.

O questionário foi desenvolvido utilizando como base a escala de Likert que segundo Gil (2008), é simples e de caráter ordinal. A construção das respostas deve medir a opinião dos entrevistados em forma de uma questão elaborada com respostas em forma de múltipla escolha. As opções disponíveis são: Não sei opinar, Discordo fortemente, Discordo, Indiferente, Concordo e Concordo fortemente.

7.2 APLICAÇÃO DO QUESTIONÁRIO

Foram entrevistadas um total de 17 pessoas. As perguntas foram enviadas através de um formulário online. Antes das perguntas estavam instruções e um link para que os entrevistados pudessem acessar o jogo. As seguintes perguntas estavam presentes no questionário:

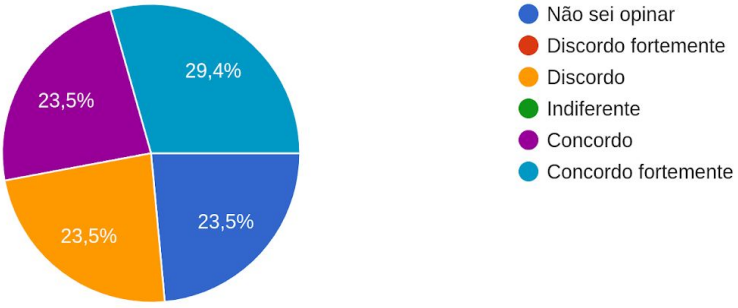
Quadro 4 - Perguntas do questionário.

1 - Você possui experiência com outros jogos no estilo MMORPG?
2 - Fawe possui uma interface amigável e fácil de usar?
3 - Você achou que os gráficos do jogo estão bonitos?
4 - O jogo foi executado sem lentidões ou travamentos no dispositivo que você utilizou para jogar?
5 - Você achou o jogo intuitivo?
6 - A movimentação do personagem é fluída?
7 - Em relação a dificuldade apresentada durante os desafios do jogo, você achou que está equilibrada?
8 - Você se divertiu jogando?
9 - Jogaria Fawe outras vezes?
10 - Gostaria de acrescentar alguma crítica ou sugestão?

Fonte: Autoral, 2020.

A primeira pergunta busca compreender o tipo de jogador que está respondendo o questionário. Ela é importante porque o jogo pode se tornar mais difícil para jogadores que nunca tenham jogado jogos no estilo MMORPG. Nas respostas foi identificado que o público estava bem dividido. Enquanto 5 pessoas (29,4%) concordaram fortemente e 4 pessoas (23,5%) concordaram que já tinham alguma experiência com jogos parecidos, 4 pessoas (23,5%) discordaram e 4 pessoas (23,5%) não sabiam opinar a respeito.

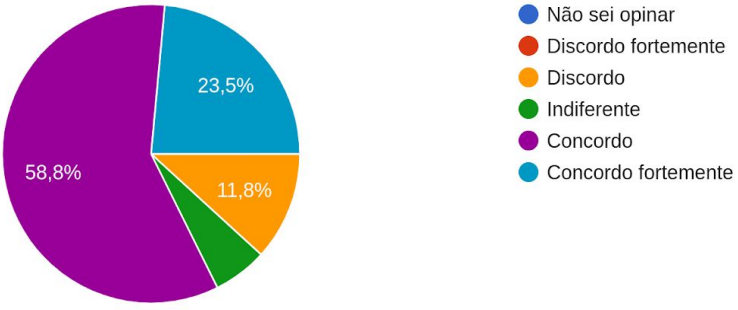
Gráfico 1 - Pergunta 1: Você possui experiência com outros jogos no estilo MMORPG?



Fonte: Autoral, 2020.

A segunda pergunta trata da qualidade da interface do jogo. As respostas foram bem positivas. 4 pessoas (23,5%) responderam concordo fortemente e 10 pessoas (58,8%) responderam concordo. Houveram algumas respostas negativas: 2 pessoas (11,8%) responderam discordo e 1 pessoa respondeu indiferente.

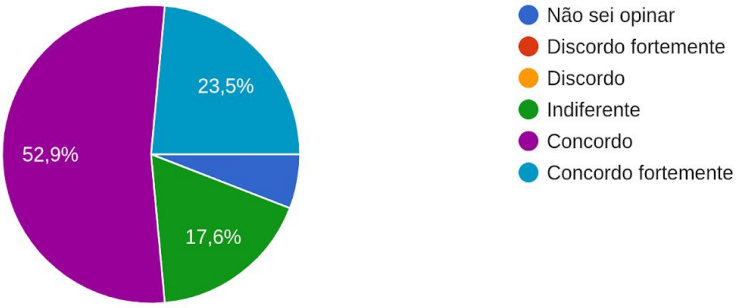
Gráfico 2 - Pergunta 2: Fawe possui uma interface amigável e fácil de usar?



Fonte: Autoral, 2020.

A terceira pergunta também tratou da parte visual do jogo, mas com foco nos gráficos. As respostas novamente foram bem positivas. No total, 9 pessoas (58,8%) concordaram e 4 pessoas (23,5%) concordaram fortemente. Não houve nenhuma resposta discordando. Apenas 3 (17,6%) pessoas foram indiferentes e 1 pessoa não soube opinar.

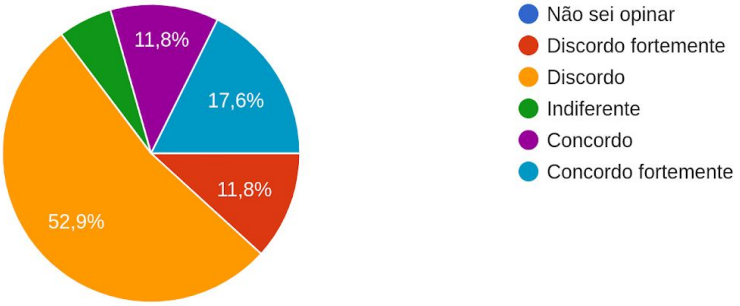
Gráfico 3 - Pergunta 3: Você achou que os gráficos do jogo estão bonitos?



Fonte: Autoral, 2020.

Na quarta pergunta as respostas não foram muito positivas. O objetivo era compreender se o jogo foi executado sem nenhum tipo de lentidão ou travamento. Do total, 9 pessoas (52,9%) discordaram e 2 pessoas (11,8%) discordaram fortemente. Mesmo assim, houve um número considerável de respostas positivas. 3 pessoas (17,6%) responderam concordo fortemente e 2 pessoas (11,8) responderam concordo. 1 pessoa foi indiferente.

Gráfico 4 - Pergunta 4: O jogo foi executado sem lentidões ou travamentos no dispositivo que você utilizou para jogar?

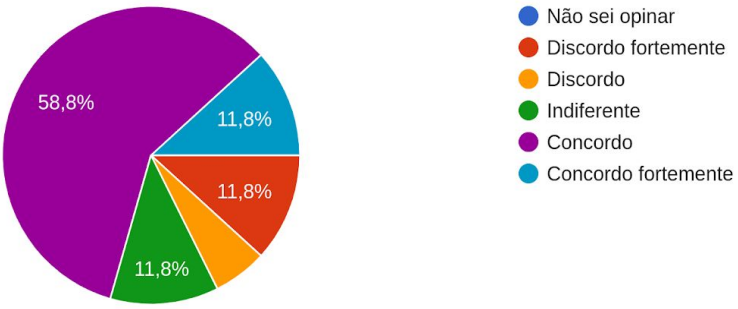


Fonte: Autoral, 2020.

A quinta pergunta buscou entender se os entrevistados acharam o jogo intuitivo. Mesmo o jogo não possuindo nenhum tipo de tutorial ou explicação, as respostas foram bem positivas. 10 pessoas (58,8%) concordaram e 2 pessoas (11,8%) concordaram fortemente.

Algumas respostas foram negativas: 2 pessoas (11,8%) discordaram fortemente e 1 pessoa discordou. 2 pessoas foram indiferentes.

Gráfico 5 - Pergunta 5: Você achou o jogo intuitivo?

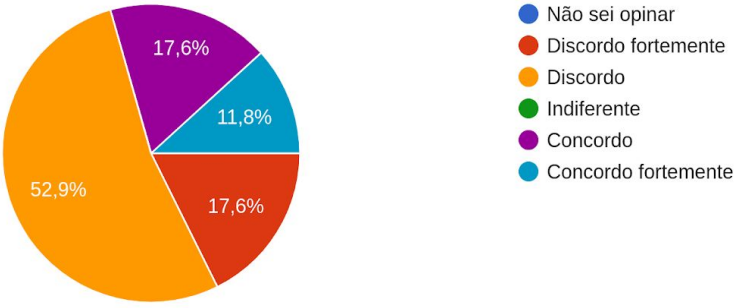


Fonte: Autoral, 2020

A sexta pergunta foi a respeito da movimentação do personagem. O objetivo era entender o que os jogadores acharam da movimentação do personagem. A maioria das respostas foi negativa. 9 pessoas (52,9%) responderam que discordam e 3 pessoas (17,6%) responderam que discordam fortemente da fluidez da movimentação do personagem no jogo. Houve algumas respostas positivas: 3 pessoas (17,6%) responderam concordo e 2 pessoas (11,8%) responderam que concordam fortemente.

Gráfico 6 - Pergunta 6: A movimentação do personagem é fluída?

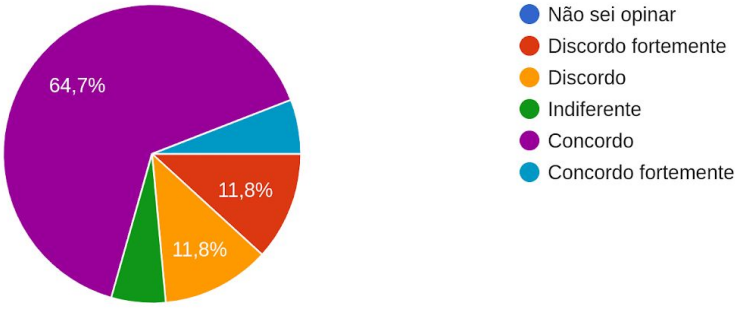
.



Fonte: Autoral, 2020.

A sétima pergunta era sobre a dificuldade apresentada nos desafios dentro do jogo. Em geral, a maioria das pessoas achou as respostas do jogo equilibradas. 11 pessoas (64,7%) concordaram e 1 pessoa concordou fortemente com a dificuldade do jogo. 2 pessoas (11,8%) discordaram e 2 pessoas (11,8%) discordaram fortemente. 1 pessoa foi indiferente.

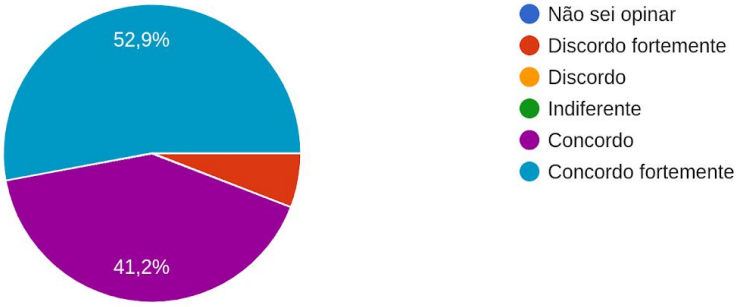
Gráfico 7 - Pergunta 7: Em relação a dificuldade apresentada durante os desafios do jogo, você achou que está equilibrada?



Fonte: Autoral, 2020.

A oitava buscou entender se as pessoas se divertiram com o jogo. Praticamente todas as respostas foram positivas. 9 pessoas (52,9%) responderam concordo fortemente e 7 pessoas (41,2%) responderam concordo. Entre as respostas negativas, apenas uma pessoa respondeu que discordava fortemente.

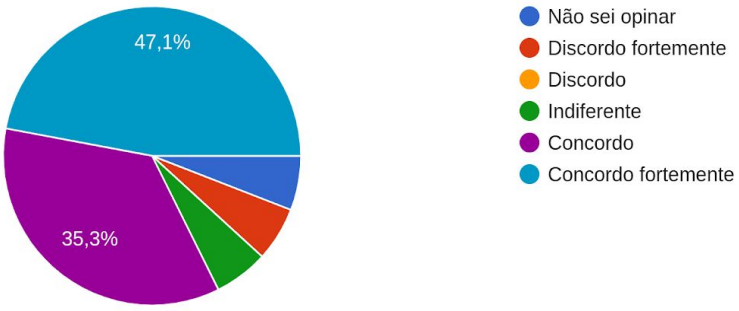
Gráfico 8 - Pergunta 8: Você se divertiu jogando?



Fonte: Autoral, 2020.

A nona pergunta foi para saber se os entrevistados jogariam Fawe outras vezes. Novamente quase todas as respostas foram positivas. 8 pessoas (47,1%) responderam concordo fortemente e 6 pessoas (35,3%) responderam concordo. Entre as respostas negativas, apenas uma pessoa respondeu que discordava fortemente. Uma pessoa foi indiferente.

Gráfico 9 - Pergunta 9: Jogaria Fawe outras vezes?



Fonte: Autoral, 2020.

Na décima e última pergunta a resposta era opcional e do tipo texto. Foi perguntado aos jogadores se eles gostariam de acrescentar críticas ou sugestões. Dos 17 entrevistados, 6 escreveram algo. Entre as melhores sugestões estavam ideias para novos itens no jogo e novas funcionalidades como opção de teleporte. Também foi sugerido algumas melhorias na interface para que fosse possível visualizar a quantidade de itens na barra de atalhos.

8 CONCLUSÕES E TRABALHOS FUTUROS

O principal objetivo do trabalho foi desenvolver o protótipo do jogo Fawe e apresentá-lo aos voluntários.

O desenvolvimento se iniciou a partir da criação um Game Design Document, que detalhou todos os conceitos e ilustrações do jogo. Depois, seguiu para etapa de modelagem UML com a criação de diagramas de classe e diagramas de atividades. Com a modelagem e o GDD criados foi possível iniciar a codificação do jogo. No fim, Fawe foi avaliado através de um questionário online com 17 voluntários.

Desenvolver jogos requer muito esforço e dedicação. Ainda mais quando se trata de um jogo do gênero MMORPG. São muitos as áreas dentro de um jogo deste gênero que podem ser aprofundadas. O conteúdo do jogo pode ser enriquecido, assim como se pode trabalhar em uma infraestrutura robusta para suportar uma grande capacidade de jogadores simultâneos.

Durante o desenvolvimento muitos desafios encontrados foram lidar com questões avançadas de programação. Multithreading, alto consumo de rede e processamento foram assuntos que desde o início do projeto precisaram de atenção e ainda podem ser melhorados.

Nos questionários aplicados, foi identificado que a maioria das pessoas gostou e se divertiu com o jogo. Porém, também foi relatado que o jogo precisa de melhorias na performance e movimentação dos personagens. Muitas pessoas relataram algum tipo de lentidão ou travamento.

8.1 TRABALHOS FUTUROS

Com o projeto desenvolvido é possível pensar em muitos trabalhos futuros. Para que o jogo deixe de ser apenas um protótipo é preciso aprofundar assuntos mais técnicos como segurança de redes, performance e escalabilidade. Também é preciso ampliar quantidade de funcionalidades para que o jogo tenha mais horas jogáveis.

Como produto, é possível pensar em alternativas de gerar algum tipo de receita. A maioria dos jogos do gênero criam algum recurso premium e vendem online por cartão de

crédito. Seria algo muito fácil de aplicar ao jogo tendo em vista que ele foi feito em Ruby on Rails que é um framework com inúmeras bibliotecas prontas para isso.

Fawe é um projeto que pode ser evoluído e ter um alcance de nível internacional. É possível criar ferramentas que traduzam o jogo para várias línguas. O login com Google usado pelo projeto já é compatível com a maioria dos países.

Uma outra alternativa para colocar o jogo no mercado seria através de financiamento coletivo. Existem inúmeras plataformas de crowdfunding nacionais e internacionais.

REFERÊNCIAS

ARTCRAFT ENTERTAINMENT, **Victory!**, Disponível em: <https://www.kickstarter.com/projects/crowfall/crowfall-throne-war-pc-mmo/posts/1178257 /> Aceso em: 26 abr. 2020

AZEVEDO, Eduardo, **Computação gráfica - Volume 1:** teoria e prática. 1. ed. Rio de Janeiro: Elsevier, 2003.

CERVO, A. L.; BERVIAN, P. A. Metodologia Científica. São Paulo: Prentice Hall, 2002.

CHIKHANI, Riad, **The History Of Gaming:** An Evolving Community Disponível em: <https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/ /> Acesso em: 02 mai. 2020

COWLEY, Ric, **Agar.io amasses 113 million mobile downloads in 20 months** Disponível em: <https://www.pocketgamer.biz/news/64583/agario-113-million-downloads/ /> Acesso em: 11 mai. 2020

DIONISIO, Edson, **Uso de Websockets e HTML5**, disponível em: <https://www.devmedia.com.br/uso-de-websockets-e-html5/32267 /> Acesso em: 20 abr. 2020.

EDWARDS, Benj, **10 Years of Ultima Online.** Disponível em: <http://ithare.com/chapter-via-server-side-mmo-architecture-naive-and-classical-deployment-architectures/ > Acesso em: 10 jul. 2020.

EIS, Diego, **HTML5 e CSS3**: com farinha e pimenta. 1. ed. São Paulo: Tableless, 2012

ROQUE, Fernando Marcos Sousa. **Canvas HTML 5**: composição gráfica e interatividade na web. 1. ed. Rio de Janeiro: Brasport, 2018.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. São Paulo: Atlas, 1996.

GIL, Antônio Carlos. **Métodos e Técnicas de Pesquisa Social**. 6ª ed. São Paulo: Atlas,

2008.

JOBS, Steve, **Thoughts on Flash**. 2010. Disponível em: <https://www.apple.com/hotnews/thoughts-on-flash/> Acesso em: 28 mar. 2020.

KERR, Chris, **ArtCraft nets $11.7 million to boost Crowfall development ahead of launch**, Disponível em: <https://www.gamasutra.com/view/news/357273/ArtCraft_nets_117_million_to_boost_Crowfall_development_ahead_of_launch.php /> Aceso em: 26 abr. 2020

KOZOVITS E FEIJÓ, Lauro, **Arquiteturas para Jogos Massive Multiplayer**, Disponível em: <ftp://ftp.inf.puc-rio.br/pub/docs/techreports/03_36_kozovits.pdf /> Acesso em: 23 mai. 2020

HUIZINGA, Johan. **Homo Ludens:** o Jogo como Elemento na Cultura (1938). São Paulo: Perspectiva, 2008.

IGNATCHENKO, Sergey, **Server-Side MMO Architecture. Naïve, Web-Based, and Classical Deployment Architectures.** Disponível em: <http://ithare.com/chapter-via-server-side-mmo-architecture-naive-and-classical-deployment-architectures/> Acesso em: 10 jul. 2020.

LAFORGE, Anthony, **Saying goodbye to Flash in Chrome** <https://www.blog.google/products/chrome/saying-goodbye-flash-chrome/> Acesso em: 29 mar. 2020.

LEMAY, Laura. **Aprenda a Criar Páginas Web com HTML e XHTML em 21 Dias**. 1. ed. [S. l.]: Pearson, 2002.

LITE (ed.). **Macromedia Flash Mx**: Passo A Passo Lite. 1. ed. São Paulo: Pearson, 2002. 254 p.

MALONE, Willian, **Create a Sprite Animation With HTML5 Canvas and Javascript,** Disponível em: <http://www.williammalone.com/articles/create-html5-canvas-javascript-sprite-animation/> Acesso em: 02 mai. 2020.

MEENAKSHI, Avanthika, **WebSockets tutorial: How to go real-time with Node and React**. Disponível em: <https://blog.logrocket.com/websockets-tutorial-how-to-go-real-time-with-node-and-react-8e4693fbf843/ > Acesso em: 10 jul. 2020.

PEKANOV, Pavel, **Is It Hard to Develop a MMORPG?,** Disponível em: <https://hackernoon.com/is-it-hard-to-develop-a-mmorpg-5g623ydy /> Acesso em: 02 mai. 2020

ROGERS, Scott. **Level Up!:** The Guide to Great Video: Game Design. Wiley, 2010

ROUGET, Paul, **BrowserQuest** – a massively multiplayer HTML5 (WebSocket + Canvas) game experiment Disponível em: <https://hacks.mozilla.org/2012/03/browserquest/ /> Acesso em: 17 abr. 2020.

SOMMERVILLE, Ian, **Engenharia de software.** Pearson, 9ª edição São Paulo, 2011

SUPERDATA, **2019 Year In Review.** Disponível em: <https://www.superdataresearch.com/2019-year-in-review/> Acesso em: 16 abr. 2020.

TEIXEIRA, Daniela, **Como escrever requisitos de software de forma simples e garantir o mínimo de erros no sistema/app??.** Disponível em: <https://medium.com/lfdev-blog/como-escrever-requisitos-de-software-de-forma-simples-e-g

arantir-o-m%C3%ADnimo-de-erros-no-sistema-app-74df2ee241cc /> Acesso em: 7 out. 2020.

TONÉIS, Cristiano N, **Matemática Aplicada aos Games.** Uma abordagem teórica e prática para desenvolvedores. Clube dos Autores: São Paulo, 2015

VALOR INVESTE, **Brasil é o 13º maior mercado de games do mundo e o maior da América Latina**, Disponível em: <https://valorinveste.globo.com/objetivo/empreenda-se/noticia/2019/07/30/brasil-e-o-13o-maior-mercado-de-games-do-mundo-e-o-maior-da-america-latina.ghtml/> Acesso em: 16 abr. 2020.

WELLS, Robert Earl, **What Is an MMO?** Disponível em: <https://www.lifewire.com/what-is-an-mmo-4687003 > Acesso em: 10 jul. 2020.

APÊNDICES

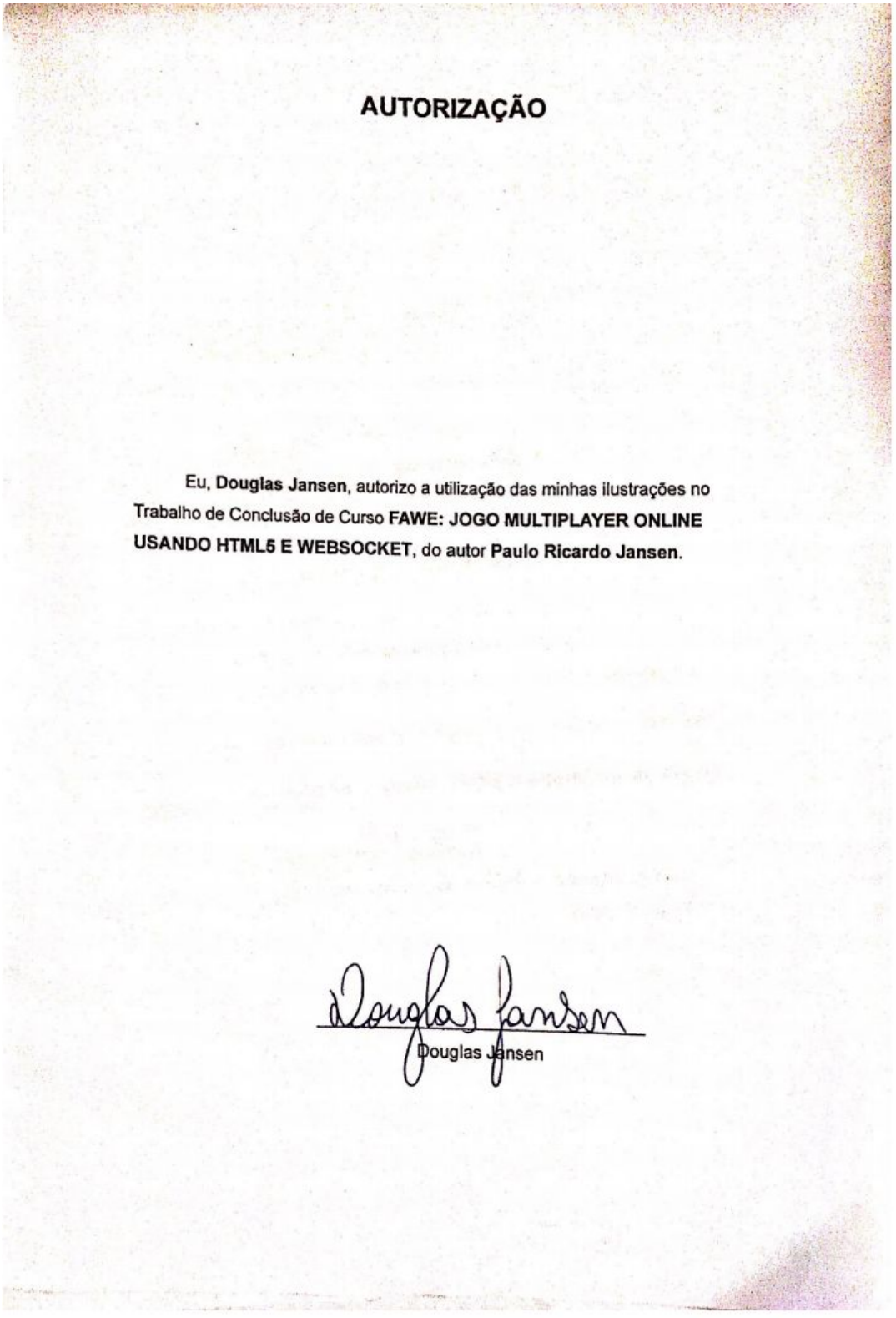
APÊNDICE A – Cronograma

Cronograma TCC – Aluno: Paulo Ricardo Jansen – Orientador: Prof. Saulo Popov Zambiasi, Dr.

ATIVIDADE	Abril				Maio				Junho				Julho				Agosto				Setembro				Outubro				Novembro				Dez.	
Semanas	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Leitura bibliografia	X	X	X	X	X																													
Elaboração Introdução						X	X	X	X																									
Redação Revisão bibliográgica										X	X	X																						
Entrega Cap2													X																					
Elaboração método														X	X																			
Entrega Cap3																X																		
Entrega Av Externo																	X																	
Codificação do jogo									X	X	X	X	X	X	X	X									X	X								
Cap 4. GDD																				X	X													
Cap 5. Modelagem do jogo																						X	X											
Cap 6. Desenvolvimento																								X	X	X								
Questionário																											X							
Conclusões/ Resumo																												X						
Entrega Monog.																													X					
Apresentação																														X	X			
Defesa																																X		
Correções																																	X	
Entrega versão final																																		X

ANEXOS

ANEXO A – Autorização de uso de desenhos



AUTORIZAÇÃO

Eu, **Douglas Jansen**, autorizo a utilização das minhas ilustrações no Trabalho de Conclusão de Curso **FAWE: JOGO MULTIPLAYER ONLINE USANDO HTML5 E WEBSOCKET**, do autor **Paulo Ricardo Jansen**.

Douglas Jansen