INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS SÃO JOÃO EVANGELISTA
Curso Bacharelado em Sistemas de Informação

Henrique Gonçalves Rocha

DESENVOLVIMENTO DE JOGO MULTIPLAYER PARA PLATAFORMA WEB UTILIZANDO PHP E JAVASCRIPT

São João Evangelista
2013

Henrique Gonçalves Rocha

DESENVOLVIMENTO DE JOGO MULTIPLAYER PARA PLATAFORMA WEB UTILIZANDO PHP E JAVASCRIPT

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Campus São João Evangelista, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Rosinei Soares de Figueiredo
Coorientador: Ricardo Bittencourt Pimentel

São João Evangelista
2013

FICHA CATALOGRÁFICA
Elaborada pelo Serviço Técnico da Biblioteca do
Instituto Federal Minas Gerais – Campus São João Evangelista

V672s ROCHA, Henrique Gonçalves, 1987 -

Desenvolvimento de jogo multiplayer para plataforma Web utilizando PHP e JavaScript./ Henrique Gonçalves Rocha. São João Evangelista, MG: IFMG – Campus São João Evangelista, 2013.
44 p.: il.

Trabalho de Conclusão de Curso - TCC (graduação) apresentado ao Instituto Federal Minas Gerais – Campus São João Evangelista – IFMG, Curso de Bacharelado em Sistemas de Informação, 2013.
Orientador: Prof. Me. Rosinei Soares de Figueiredo
Coorientador: Prof. Esp. Ricardo Bittencourt Pimentel

1. Computação. 2. Linguagem de programação . 3. Software . I. Instituto Federal Minas Gerais – Campus São João Evangelista. Curso de Bacharelado em Sistemas de Informação. II. Título.

CDD 005.133

Henrique Gonçalves Rocha

DESENVOLVIMENTO DE JOGO MULTIPLAYER PARA PLATAFORMA WEB UTILIZANDO PHP E JAVASCRIPT

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Campus São João Evangelista, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Rosinei Soares de Figueiredo (Orientador)

Ricardo Bittencourt Pimentel (Coorientador)

Ronan Dutra Mendonça (Professor Convidado)

São João Evangelista, 12 de novembro de 2013.

DEDICATÓRIA

Dedico este trabalho aos meus avós...

À minha linda avó Hercília, que não tive oportunidade de conhecer, mas pude ver através de fotografias a alegria dela ao me carregar no colo.

Ao meu avô Geraldo, um ótimo pai de família e um exemplo a se seguir. Com quem aprendi muito e tenho muita saudade.

Ao meu avô Cláudio, que tive a oportunidade de passar uma pequena parte da minha infância. Homem de coração enorme que me faz muita falta.

À minha querida e amada avó Nazinha, mulher virtuosa e temente a Deus, a minha segunda mãe. Ela não sabe a falta que me faz.

AGRADECIMENTOS

Aos meus amigos, colegas e professores, que me apoiaram durante a caminhada.

Ao meu orientador Rosinei, que é um excelente profissional e teve um papel fundamental no sucesso deste trabalho.

À minha mãe Rita e meu pai Cláudio, por todo carinho e apoio recebido, convertidos em alegria e satisfação.

Ao meu irmão Guilherme, pela ajuda no desenho das cartas.

Em especial à minha namorada Laiza, pelo apoio e ânimo que me proporcionou durante os momentos mais difíceis deste projeto. E também pela paciência que teve em escutar várias vezes meus comentários sobre esse assunto.

E agradeço principalmente a Deus, por ter me dado sabedoria e também o privilégio de conhecer e conviver com essas pessoas extraordinárias.

RESUMO

Com o crescente mercado de jogos e o desejo do autor de aplicar na prática os conhecimentos obtidos em sala de aula, optou-se em desenvolver um jogo de cartas WEB *Multiplayer*. O objetivo do trabalho foi desenvolver a versão inicial desse jogo de cartas no intuito de ampliar os conhecimentos teóricos e práticos do autor em relação às tecnologias utilizadas. O software foi desenvolvido de acordo com a metodologia do Desenvolvimento de Software Adaptativo (ASD) que possui três fases que se completam, são elas: Especulação, Colaboração e Aprendizagem. O banco de dados utilizado que armazena os dados sobre os jogadores e as informações referentes à suas cartas no jogo foi o Mysql. Adotou-se para esta versão inicial do jogo de cartas o pacote WampServer que é composto pelo servidor de banco de dados Mysql, pelo servidor PHP e pelo servidor Apache. As páginas do site do jogo foram desenvolvidas em HTML e formatadas em CSS. As linguagens de programação utilizadas foram as linguagens PHP e JavaScript. Através da linguagem PHP foi feita a comunicação com o banco de dados e a linguagem JavaScript foi utilizada na implementação dos eventos do jogo. A comunicação entre cliente e servidor foi implementada com auxílio do framework NodeJS, através de programação orientada a eventos na linguagem JavaScript, tanto do lado cliente, quanto do lado servidor. O servidor do jogo é executado pelo NodeJS através do prompt de comando, ele responde aos eventos solicitados pelos clientes. No início do jogo, os dois oponentes que estão se enfrentando em uma batalha escolhem em qual parte do campo colocar cada um de seus cinco guerreiros. Depois de prepararem o campo com seus guerreiros, ambos os jogadores precisam clicar no botão iniciar para então começar a partida. O primeiro jogador a atacar é escolhido pela sorte. Após a realização de um ataque, a vez de atacar é passada para o oponente. Vence a partida o primeiro jogador que fizer com que os pontos de vida de três guerreiros oponentes cheguem à zero. O trabalho foi concluído com sucesso, pois a versão inicial está funcionando perfeitamente e o autor obteve várias experiências com todas as tecnologias utilizadas, elevando seus conhecimentos.

Palavras-Chave: Desenvolvimento de jogos. Programação orientada a eventos. Comunicação cliente/servidor.

ABSTRACT

With the growing gaming market and the desire of the author to apply in practice the knowledge obtained in the classroom, it was decided to develop a card game Multiplayer WEB. The objective was to develop the initial version of this card game in order to extend the theoretical and practical knowledge of the author in relation to the technologies used. The software was developed according to the methodology of Adaptive Software Development (ASD), which has three phases that complete each other, they are: Speculation, Collaboration and Learning. The database used to store the player's date and the information of their cards in the game was the Mysql. It Was adopted for this initial version of the card game WampServer package which consists of the server database Mysql, by the PHP server and the server Apache. The pages of the game site were developed in HTML and CSS formatted. The programming language used was the language PHP and JavaScript. By language PHP was made a communication with the database, and the JavaScript language was used in the implementation of the events of the game. The communication between client and server was implemented with the support of framework's NodeJS, by event-driven programming language JavaScript, both client-side, the server side. The game server run by NodeJS by prompt command, it responds to the events requested by the players. At the beginning of the game, two opponents who are facing each other at a battle, can choose which part of the field to put each of their five warriors. After making the field with his warriors, both players need to click on the Start button and then start the game. The first attacking player is chosen by luck. After carrying out an attack, the attack turn is changed to the opponent. Win the game the first player who causes the hit points of three warriors opponents reach until zero. The work was completed successfully because the initial version is perfectly running and the author obtained several experiences with all the technologies used, increasing his knowledge.

Keywords: Game development. Event-driven programming. Client/server communication.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente o mercado de jogos eletrônicos tem se destacado fortemente na economia nacional (Mercado..., 2012) tanto na venda de consoles, cartuchos e discos de jogos, controles e acessórios, quanto em jogos *online*.

Os consoles podem ser portáteis ou não portáteis. Os portáteis são os que podem ser jogados em qualquer local sem a necessidade de estarem ligados a uma tomada de energia elétrica, por possuírem baterias ou pilhas. Já os nãos portáteis precisam estar conectados a rede de energia elétrica e geralmente utilizam um aparelho de televisão para reproduzirem a imagem.

Os *smartphones* também estão sendo bastante utilizados para essa atividade de entretenimento, por serem portáteis e terem ótimas configurações de *hardwares*.

Muitos jogos estão sendo desenvolvidos para a plataforma Android. O Android é o sistema operacional utilizado em aproximadamente 80% dos *smartphones* no Brasil. (BRASIL..., 2012).

Existem também os jogos *online,* que são bastante jogados por permitirem a interação entre diversos jogadores de várias partes do mundo. Eles são subdivididos em jogos *desktop* (jogos que necessitam ser baixados e instalados no computador) e jogos de *browser* (jogados através do navegador) que rodam na plataforma WEB.

Este trabalho aborda o desenvolvimento de um jogo para *browser*. O referido jogo é um jogo de cartas, no qual as cartas representam os guerreiros dos jogadores que batalharão entre si em busca da vitória.

O tema foi escolhido por ser bastante versátil, envolvendo diversas tecnologias, como: HTML (Linguagem de Marcação de Hipertexto), linguagem de programação WEB, banco de dados, etc. Essas e outras tecnologias foram estudadas e aplicadas na construção do projeto. Além da utilização de um dos modelos de desenvolvimento de software aprendidos durante a disciplina Engenharia de Software. Ao término do projeto, o autor pode adquirir experiências envolvendo as tecnologias citadas acima e obteve novos conhecimentos teóricos e práticos.

O foco do trabalho foi a integração de diversas tecnologias conhecidas e ainda não conhecidas pelo graduando, proporcionando novos conhecimentos para o mesmo e a aplicação prática de todos esses conhecimentos durante o desenvolvimento do jogo. Fez parte do trabalho o estudo dessas tecnologias, o

desenho e modelagem dos personagens e a definição das regras e cálculos de combate do jogo. As tecnologias estudadas foram aplicadas no desenvolvimento do banco de dados, na construção e modelagem do site, na modelagem dos personagens e na construção da aplicação.

O objetivo principal desse trabalho foi desenvolver a versão inicial de um jogo de cartas para *browser* com o intuito de ampliar os conhecimentos teóricos e práticos do autor em relação às tecnologias utilizadas. Essa versão inicial foi a parte de batalha das cartas, cuja comunicação entre os clientes foi feita através da arquitetura cliente/servidor. Para atingir esse objetivo, primeiramente foi necessário atingir uma série de objetivos que estão listados a seguir: a) Definir qual a linguagem de programação Web que seria utilizada; b) Desenhar e trabalhar as imagens que foram inseridas no site e no jogo; c) Desenvolver a estrutura do site em HTML (linguagem de marcação de hipertexto), com as páginas formatadas com CSS (folha de estilo em cascata); d) Desenvolver o jogo em si: regras, funcionamento, cálculo de dano, personagens, condições de vitória e derrota, etc. (ideia geral do jogo); e) Escolher um processo de desenvolvimento de software que se adequasse com maior compatibilidade com o projeto; f) Desenvolver o banco de dados do jogo; g) Aprender um modo de programação cliente/servidor que possibilitasse a implementação do jogo.

O jogo foi desenvolvido utilizando não apenas uma e sim duas linguagens de programação. A linguagem PHP para fazer as interações com o Banco de Dados e a linguagem JavaScript para a escrita do código do jogo, tanto do lado cliente quanto do lado do servidor.

A comunicação entre cliente e servidor foi feita através de eventos. O servidor do jogo é executado através do *framework* NodeJS que é explicado com detalhes no capítulo Metodologia. Também foi desenvolvido um *chat* (janela de bate papo em tempo real) para possibilitar a comunicação entre os usuários no momento da batalha.

A pesquisa possui três principais contribuições: a) A contribuição individual que foi o objetivo principal do trabalho; b) A contribuição social que é a parte do entretenimento que o jogo pode proporcionar para a sociedade; c) E a contribuição acadêmica para futuros alunos que desejam desenvolver algo parecido.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir serão apresentados alguns conceitos que deram fundamento ao desenvolvimento do trabalho proposto em relação à Engenharia de Software, ao processo de desenvolvimento utilizado, o que vem a ser esse processo de desenvolvimento e um pouco sobre o funcionamento da web.

2.1 Engenharia de Software

Primeiramente é necessário definir o que é software para ser possível entender o que é Engenharia de Software. Paula Filho define software como "[...] a parte programável de um sistema de informática. Ele é um elemento central: realiza as estruturas complexas e flexíveis que trazem funções, utilidade e valor ao sistema.'' (PAULA FILHO, 2009, p. 4).

Pressman afirma que uma definição formal para software poderia ser escrita da seguinte forma:

> "Software é: (1) instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados; (2) estruturas de dados que possibilitam que os programas manipulem adequadamente a informação; e (3) documentos que descrevem a operação e o uso dos programas." (PRESSMAN, 2010, p. 12).

Segundo Sommerville (2007, p. 5), "A Engenharia de Software é uma disciplina de engenharia relacionada com todos os aspectos de produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção".

Para que se desenvolva um software com qualidade e dentro dos prazos estabelecidos é necessário à utilização de ferramentas, métodos e processos da Engenharia de Software. Processos são conjuntos de ferramentas e métodos utilizados para a confecção de um produto de software de qualidade dentro dos prazos de entrega. Métodos são as diferentes maneiras que podem ser usadas para concluir as diversas partes dos processos, ou seja, cada método é o caminho escolhido para solucionar alguma parte do processo. E as ferramentas são todos os softwares e tecnologias utilizadas pelos métodos no desenvolvimento de um

processo e na construção de um software. Esses três elementos são fundamentais para permitir que o gerente tenha controle do processo de desenvolvimento de software. (PRESSMAN, 2010).

Os sistemas desenvolvidos utilizando as técnicas da engenharia de software permitem de forma fácil a identificação e correção de falhas nos mesmos, pois os sistemas são bem planejados e os códigos documentados. Esta documentação também diminui riscos em caso de perca ou alterações da equipe de desenvolvimento. Permitindo que a nova equipe ou os novos membros saibam o que foi feito e possam continuar o trabalho sem perderem muito tempo “decifrando” o código. (SOMMERVILLE, 2007; PAULA FILHO, 2009).

Logo a seguir será explicado com mais detalhes o que vem a ser um processo de desenvolvimento de software.

2.1.1 Processo de Software

Para Sommerville (2007, p. 42), “Um processo de software é um conjunto de atividades que leva à produção de um produto de software.”. Apesar das diferenças entre os processos de software, todos compartilham das mesmas atividades essenciais. São elas: a) Especificação de software, que é a definição das funcionalidades e restrições que o sistema possuirá; b) Projeto e implementação de software, que é o desenvolvimento do projeto propriamente dito, de acordo com os requisitos levantados; c) Validação de software que é a fase de avaliação, para saber se o software faz realmente aquilo que o cliente deseja; d) Evolução de software, que é a possibilidade de alterações futuras, caso mude as necessidades do cliente.

Essas atividades essenciais são à base de todos os processos de software. Alguns processos são estruturados, mais burocráticos e outros já são mais flexíveis, como os métodos ágeis. (SOMMERVILLE, 2007).

Os processos da Engenharia de Software definem o modo como será desenvolvido o software, envolvendo um grande número de tarefas como: “[...] planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção.” (PRESSMAN, 2010, p. 31).

Quando não é utilizado nenhum processo de desenvolvimento de software, o processo é chamado de “codifica-remenda”, havendo apenas uma especificação ou até mesmo nenhuma. Os desenvolvedores começam imediatamente a programar, corrigindo e remendando o código assim que erros vão aparecendo, sem nenhuma documentação. (PAULA FILHO, 2009).

Softwares desenvolvidos sem utilizar um processo de desenvolvimento são altamente susceptíveis a falhas. Essas falhas podem acarretar em longos períodos de tempo perdido na sua identificação e no retrabalho de corrigi-las. Mesmo que se encontre e corrija a falha, é possível que ela afete ainda alguma outra parte integrada do sistema. Sendo assim, uma falha pode gerar outras falhas. (SOMMERVILLE, 2007; PAULA FILHO, 2009).

Utilizar um processo de software não quer dizer que o sistema estará totalmente livre de falhas, mas diminui a probabilidade da ocorrência das mesmas. E caso ocorram, torna-se fácil identificar onde estão localizadas essas falhas para efetuar as devidas correções. Pois, todos os requisitos do sistema estarão documentados e desenvolvidos de acordo com as técnicas e padrões do modelo de processo. (PAULA FILHO, 2009).

2.1.2 Modelo de Processo de Software

Os modelos de processos de software são uma representação abstrata de um processo de software. Segundo Sommerville (2007), eles são divididos em três modelos genéricos: a) O modelo de processo em cascata, que representa as atividades fundamentais do processo de software de forma separada; b) O desenvolvimento evolucionário, que intercala as atividades essenciais, permitindo assim um desenvolvimento rápido de um sistema inicial, que será refinado conforme os requisitos do cliente para suprir suas necessidades; c) A Engenharia de Software baseada em componentes, que utiliza-se de um número significativo de componentes reusáveis, sem a necessidade de se criar tudo do zero e sim fazer a integração das partes de sistemas já desenvolvidos para compor o sistema que está sendo implementado.

Esses três modelos de processo de software são amplamente usados na Engenharia de Software. Eles não são apenas usados separadamente, pois podem se integrar na composição de um novo modelo de processo de desenvolvimento de

software. Como é o caso do RUP (*Rational Unified Process*) que possui elementos dos três modelos essenciais. (SOMMERVILLE, 2007).

Não serão citados todos os modelos de processo de desenvolvimento existentes, pelo fato de não ser o foco do trabalho. Apenas foram identificados os modelos genéricos e afirmado o conceito de que eles podem se unir para comporem um novo modelo de processo.

As abordagens de desenvolvimento tradicionais, também conhecidas por abordagens de desenvolvimento orientadas a planos são essenciais para grandes sistemas, que necessitam do trabalho de muitos profissionais e para sistemas críticos. Por exigirem que as especificações sejam documentadas detalhadamente, diminuindo assim, o risco de falhas gravíssimas. Sistemas críticos são sistemas cujo um erro pode causar consequências drásticas, como a perda de vidas ou financeira. Contudo, quando essa abordagem de desenvolvimento orientada a planos foi utilizada em pequenos e médios sistemas, chegou-se a conclusão que o tempo gasto para o planejamento e documentação era superior ao da soma do tempo de implementação com o tempo de testes, o que gerou a necessidade da criação de metodologias mais rápidas, que não perdessem muito tempo em documentação e focassem principalmente na implementação e testes. (SOMMERVILLE, 2007).

2.2 Desenvolvimento Ágil

Utilizando um modelo de processo ágil, torna-se possível criar softwares úteis de forma rápida. Os processos ágeis geralmente são interativos, de maneira que as fases de especificação, projeto, desenvolvimento e testes são intercaladas. Na maioria das vezes a entrega não acontece de uma só vez e sim em incrementos que integrarão o sistema com novas funcionalidades. (SOMMERVILLE, 2007).

Paula Filho explica que: “Processos ágeis ou métodos ágeis constituem um grupo de metodologias diferentes entre si, mas caracterizadas por princípios em comuns, mais baseado no trabalho cooperativo do que no formalismo e na documentação escrita.” (PAULA FILHO, 2009, p. 102).

Os métodos ágeis são focados na parte do desenvolvimento e na entrega incremental, mas utiliza-se de processos distintos para atingir esses objetivos. Contudo, possuem alguns princípios comuns característicos de todos os métodos ágeis, são eles: a) Envolvimento do cliente: Para avaliar as interações do sistema e

fornecer novos requisitos para o mesmo; b) Entrega incremental: A entrega é feita por partes e o cliente ajuda a definir quais partes serão entregues em cada incremento; c) Pessoas, não processos: Significa que as pessoas evolvidas no projeto não estarão seguindo processos prescritivos e sim desenvolvendo suas próprias maneiras de trabalhar, explorando suas próprias habilidades; d) Aceitação de mudanças: Saber que requisitos vão mudar. Então desenvolver o sistema de maneira que facilite a acomodação das mudanças de requisitos; e) Mantendo a simplicidade: Trabalhar de forma simples, tentando diminuir a complexidade do sistema. (SOMMERVILLE, 2007).

Processos de desenvolvimento ágeis são ideais para sistemas WEB como afirma Sommerville: “O desenvolvimento menos formal é particularmente apropriado para sistemas baseados na WEB, que requerem uma combinação de habilidades em projeto gráfico e em software.” (SOMMERVILLE, 2007, p. 5).

Como o sistema que será desenvolvido é um jogo para a plataforma WEB, que será explicada mais a frente, o modelo de processo de software escolhido para o desenvolvimento do sistema foi o Desenvolvimento de Software Adaptativo (ASD), por ser um método ágil e adaptável a possíveis mudanças. O principal motivo da escolha de um método ágil foi o curto prazo de tempo para o desenvolvimento do sistema.

2.2.1 Desenvolvimento de Software Adaptativo (ASD)

O Desenvolvimento de Software adaptativo (ASD) foi criado pelo fundador da *Agile Alliance,* Highsmith e consiste basicamente em três partes que se sobrepõem. São elas: especulação, colaboração e aprendizado. O método ASD estimula que os programas sejam desenvolvidos de forma interativa e incremental, fazendo uso de protótipos constantemente. (SANTOS; SOUZA; NASCIMENTO, 2011; COSTA, 2011).

Os nomes das três fases enfatizam a diferença entre desenvolvimento de softwares ágeis e tradicionais. “Especulação” no lugar de “planejamento”, porque no desenvolvimento de software há incertezas e a palavra planejamento indica o contrário. A fase de “colaboração” identifica a importância do trabalho em equipe. Fazendo com que ao haver comunicação entre a equipe, reduz o número de incertezas a respeito do projeto. E a fase de “aprendizado” é o conhecimento que se

deve ter que os requisitos podem mudar a qualquer momento, exigindo que a equipe aprenda como se comportar diante dessas mudanças. (HIGHSMITH apud COSTA, 2011).

A abordagem da metodologia é totalmente focada a adaptação a possíveis mudanças que deverão ser aceitas de forma contínua, ou seja, o tempo todo o projeto corre risco de sofrer novas mudanças. Para ser possível que o projeto seja adaptável a mudanças, o processo de desenvolvimento define seis princípios básicos que serão utilizados durante todo o ciclo de vida do projeto. São eles: a) Foco na missão: Que define o que será feito no projeto e não a forma que será feito; b) Desenvolvimento baseado em componentes: Deve-se focar no produto, desenvolvendo as partes que integrarão o produto final sem se preocupar com as tarefas que serão realizadas; c) Interação: Tendo o foco em etapas, sem se preocupar em fazer certo da primeira vez, caso necessário é refeita alguma parte; d) Tolerância às mudanças: Frequentemente é necessário fazer mudanças nos softwares, por isso é essencial que o sistema seja capaz de se adaptar a prováveis mudanças; e) Incremento de produtos: Entregar as partes efetivamente dentro dos prazos fixos de entrega; f) Orientação a riscos: Desenvolver de forma prioritária os incrementos de maiores riscos para o projeto. (HIGHSMITH apud COELHO, 2011).

A próxima seção tratará o conceito de WEB, por ser a plataforma onde o jogo será executado. Tornando-se essencial a compreensão do que venha a ser a WEB para a compreensão completa deste trabalho.

2.3 World Wide Web

Para facilitar o conhecimento da WEB, primeiramente deve-se conhecer o conceito de Internet. Que é uma rede mundial que conecta inúmeras redes e permite o funcionamento de vários serviços de compartilhamento de informações. (BEAL, Vangie, 2010).

Para Hahn e Stout (1995, p. 2), a Internet “[...] é muito mais do que uma rede de computadores ou um serviço de informação.”, pois “[...] as redes de computadores são simplesmente um meio de transporte de informação. A beleza e a utilidade da Internet estão na informação em si.”.

A World Wide Web, também conhecida como WEB, ou WWW, ou simplesmente W3 é um serviço que organiza vários documentos da internet em

forma de hipertexto, que são lidos através de um browser. Esses documentos podem conter textos, imagens, sons, vídeos e links que servem para acessar outros documentos. Basicamente “[...] hipertexto é um dado ligado a outros dados.”, ou seja, esses dados possuem links que podem ou não ser seguidos de acordo com o desejo do usuário da internet. Também é possível a pesquisa de informação na WEB através de palavras-chave. (HAHN; STOUT, 1995, p. 494).

A WEB pode ser simplesmente descrita como um serviço de compartilhamento de informações criado para ser executado através da internet. Ela utiliza o protocolo HTTP (Protocolo de Transferência de Hipertexto), cuja função é transmitir informação pela internet através da WEB. Essas informações são interpretadas por navegadores, conhecidos como *browsers*. (BEAL, Vangie, 2010).

Browser é um programa que serve como ferramenta para leitura de hipertextos das páginas WEB. Através dele é possível navegar através da WEB digitando o endereço das páginas ou navegando através de links contidos nas páginas WEB. (HAHN; STOUT, 1995).

O próximo capítulo relata o desenvolvimento de alguns trabalhos que possuem temas relacionados com o tema do presente trabalho. Fazendo com que algumas experiências vividas pelos autores contribuam para o desenvolvimento do jogo de cartas proposto.

3 REVISÃO LITERÁRIA

Este capítulo apresenta alguns trabalhos que contribuem para o desenvolvimento desse projeto. Quatro trabalhos referentes ao desenvolvimento de jogos para WEB e o quinto fazendo um comparativo entre três plataformas bastante utilizadas no desenvolvimento de jogos WEB.

O título do primeiro trabalho é o “Desenvolvimento do Jogo Robogol para navegadores web utilizando o motor de jogos Unity 3D”, cujo autor é Diego Eduardo Carvalho. O segundo trabalho foi escrito por dois autores, Emiliano Carlos M. Firmino e Jucimar Maia da Silva Junior, com o título de ”Desenvolvimento de Jogos em HTML5”. O terceiro trabalho, chamado “Desenvolvimento de Jogos para a Terceira Idade” foi desenvolvido por três autores, são eles: Anibal Lopes Guedes, Fernanda Lopes Guedes e Naiara Sebben. O quarto trabalho também foi escrito por dois autores, Daniel Chaves Toscano Dantas e Eliseu Paz e Silva de Guimarães com o título de “Sistema Especialista de Inteligência Artificial Integrado a Plataforma WEB para Entretenimento em um Ambiente Interativo Multiusuário”. O quinto trabalho foi escrito pelo autor Mário Barbosa de Araújo Junior e seu título é “Um Estudo Comparativo de Tecnologias WEB para Games”.

3.1 Trabalhos Relacionados

No trabalho “Desenvolvimento do Jogo Robogol para navegadores web utilizando o motor de jogos Unity 3D” foi desenvolvido um jogo de Robogol para rodar em navegadores WEB com o objetivo de divulgar o jogo real desenvolvido pela embresa XBOT. O jogo consiste em quatro robôs, cada robô controlado por um jogador, compondo dois times de dois robôs cada, uma bola de golfe, dois gols e a mesa de jogo. Os robôs são controlados por controle remoto e possuem quatro movimentos: Girar sentido horário, girar sentido anti-horário, andar pra frente e andar para trás. Como no futebol, ganha quem fizer o maior número de gols durante o tempo de partida. O sistema desenvolvido é um simulador deste jogo, que permite a divulgação e o acesso para qualquer pessoa conectada à internet. (CARVALHO, 2012).

O simulador foi desenvolvido com o auxílio do Unity 3D versão 3.0(CORRAL, 2009), que é um software motor de jogos multiplataforma. Um motor de jogos é um

programa que ajuda na produção específica de jogos, facilitando tanto o desenvolvimento da parte gráfica quanto da parte lógica do sistema. A principal vantagem do Unity é dar suporte às colisões dos objetos, facilitando a vida do programador. A linguagem de programação utilizada na codificação do simulador foi a C# e o banco de dados foi o Postgre SQL versão 8.4. (CARVALHO, 2012).

O sistema não é multiusuário, o time adversário é controlado por robôs que não possuem inteligência artificial implementada. Eles simplesmente seguem uma sequência de comandos pré-estabelecidos. (CARVALHO, 2012).

Apesar de ser simples, ele faz o que se deseja. Que é mostrar ao público como é o funcionamento desse jogo, possibilitando aos interessados adquirirem o produto real.

No segundo trabalho, ”Desenvolvimento de Jogos em HTML5”, foi desenvolvido dois jogos simples, o jogo da velha para testar as funcionalidades do Canvas, que é um elemento do HTML5 e o jogo Sukuri para testar o comportamento do HTML5 em frente à necessidade do controle de eventos e renderização constantes. Ambos os jogos foram desenvolvidos utilizando a linguagem de programação JavaScript. (FIRMINO; SILVA, 2010).

O jogo da velha, também conhecido como Tic-Tac-Toe é bastante conhecido e simples de jogar. Nesse trabalho, ele foi desenvolvido através do *Canvas* e não foi implementado para multijogadores. O jogo é jogado entre um jogador e um computador, mas também é possível colocar dois computadores para jogarem uma partida. Não foi implementada inteligência artificial. Os computadores jogam de forma aleatória, não se importando com o resultado da jogada, ou seja, não se importa se a jogada é vantajosa ou não de acordo com as regras do jogo. O tabuleiro foi desenhado através do *Canvas* e os marcadores de jogadas são as letras “O” e “X” em fonte Arial 65. (FIRMINO; SILVA, 2010).

O jogo Sukuri é um jogo onde o usuário controla uma cobra que se movimenta por um labirinto enquanto os computadores também controlam outras cobras no mesmo labirinto. O objetivo do jogo é comer o maior número de frutas que aparecem em lugares aleatórios do labirinto. Assim que uma fruta é comida por uma cobra, esta cobra aumenta de tamanho. Caso uma cobra colida com outra cobra ou contra a parede do labirinto ela morre. Com o tempo de jogo, a velocidade de movimento das cobras vai aumentando, ficando mais difícil de controlá-la. (FIRMINO; SILVA, 2010).

No terceiro trabalho que é o “Desenvolvimento de Jogos para a Terceira Idade”, foi desenvolvido três jogos em Flash utilizando a linguagem de programação Action Script com os objetivos de estimular o raciocínio lógico de idosos, de ampliar seus horizontes culturais através da utilização da WEB e de disponibilizar através destes jogos uma nova opção de entretenimento. (GUEDES; GUEDES; SEBBEN, 2010).

O banco de dados utilizado para armazenar as regras dos jogos foi o Mysql. Foi implementada uma tela inicial com a opção de escolha entre os três jogos desenvolvidos. O sistema também armazena os recordes de cada jogo. (GUEDES; GUEDES; SEBBEN, 2010).

O primeiro jogo é um jogo para estimular o raciocínio lógico, que funciona preenchendo a matriz com números de 1 a 4 sem repetir o mesmo número em uma mesma linha, ou em uma mesma coluna, ou em um mesmo quadrante. Esse jogo é conhecido como Sudoku e foi adaptado nesta versão para trabalhar com matriz de ordem quatro. O segundo envolve lógica com resolução de problemas. O idoso tem o desafio de descobrir qual o personagem que é culpado e qual é o inocente através de frases de lógica. A tela de jogo é bastante interativa. E o objetivo do terceiro jogo é montar uma pirâmide de animais, onde os animais mais leves devem ficar no topo da pirâmide e os mais pesados na base. (GUEDES; GUEDES; SEBBEN, 2010).

Nesse outro trabalho, “Sistema Especialista de Inteligência Artificial Integrado a Plataforma WEB para Entretenimento em um Ambiente Interativo Multiusuário”, os autores desenvolveram um jogo para a plataforma WEB, que simula times de futebol, onde os usuários são donos dos times e ao mesmo tempo técnicos. Cada usuário monta sua equipe contratando e vendendo jogadores virtuais e escalam os jogadores para as partidas. As partidas são executadas através de inteligência artificial e os usuários podem acompanhá-las em tempo real. Portanto esse jogo é multiusuário. (DANTAS; GUIMARÃES, 2010).

O banco de dados utilizado foi o Postgre SQL e a linguagem utilizada para a codificação foi a JAVA. O sistema foi dividido por partes separadas e a integração de todas essas partes do sistema foi feita por meio de WEB Service. WEB Service é uma solução que permite a integração e consequentemente a comunicação entre diferentes tecnologias. O sistema é muito complexo, por integrar diferentes tipos de tecnologia. A parte mais complexa é a Inteligência Artificial. O projeto obteve os

resultados esperados pelos desenvolvedores, servindo de aprendizado para os mesmos e de entretenimento para os usuários. (DANTAS; GUIMARÃES, 2010).

Esse trabalho é o que mais se assemelha com o projeto que foi desenvolvido no presente trabalho de conclusão de curso, exceto pela parte de Inteligência Artificial que não foi implementada, pois não haverá batalhas entre um jogador e um robô. Apenas batalhas entre jogadores.

O quinto trabalho, ”Um Estudo Comparativo de Tecnologias WEB para Games”, faz um estudo comparativo entre as principais plataformas de desenvolvimento WEB. Nesse estudo foram apresentadas a plataforma Java da Sun, o Silverlight da Microsoft e o Flash da Adobe. Foram apresentadas suas vantagens e desvantagens com foco no desenvolvimento de jogos WEB. Os quesitos avaliados foram o reuso, curva de aprendizado, integração com outras ferramentas, testes, disponibilização do produto, gráficos, interatividade, dificuldade de instalação e desempenho. Esses quesitos foram avaliados individualmente para cada uma das plataformas, recebendo uma dentre as três notas: Fraco, Médio e Excelente. Das três plataformas estudas a única que se mostrou excelente em todos os quesitos foi à plataforma Flash Player. (ARAÚJO JÚNIOR, 2009).

O Action Script, que dá um excelente suporte a orientação a objetos é a linguagem de programação WEB do Flash Player. O Flash dá suporte tanto no desenvolvimento do código, quanto na manipulação de imagens, conseguindo trabalhar praticamente com todos os formatos de imagem existentes atualmente. Aprender a desenvolver em Flash é muito fácil. Com poucas horas de trabalho o desenvolvedor já consegue manipular eventos de tecla, sons e imagens de forma bastante eficiente. (JUNIOR, 2009).

Com o auxílio deste capítulo, chegou-se a conclusão que o Action Script seria a melhor linguagem de programação para trabalhar com jogos. Porém o jogo proposto por esse trabalho tem características diferentes dos jogos em Action Script apresentados neste capítulo. O jogo proposto é um jogo *multiplayer*, cuja essência é a comunicação cliente/servidor. A princípio pensou-se em adotar a linguagem ActionScript, mas deparou-se com este problema sobre a arquitetura Cliente/Servidor, que até então a implementação não era conhecida pelo autor.

Através de várias pesquisas, encontrou-se o framework NodeJS, que executa servidores implementados em JavaScript. Seu funcionamento será explicado em detalhes no próximo capítulo. Com o estudo sobre o NodeJS e o conhecimento de

suas vantagens, não houve dúvidas de sua importância para o trabalho. Portanto definiu-se o PHP e o JavaScript como as linguagens de programação do jogo de cartas proposto nesse trabalho. A linguagem PHP é a responsável pela interação com o banco de dados e a linguagem JavaScript é a linguagem utilizada para a programação do restante do código, tanto do lado cliente, como do lado servidor.

O próximo capítulo apresenta a metodologia aplicada no desenvolvimento desse trabalho. Mostrando ao leitor o caminho percorrido para alcançar os objetivos propostos.

4 METODOLOGIA

A seguir está descrita a metodologia empregada no desenvolvimento do trabalho. Informando quais são as ferramentas e tecnologias utilizadas e como foram empregadas no sucesso do projeto.

4.1 Pesquisas Importantes

Para dar suporte ao desenvolvimento deste trabalho foram feitas pesquisas em livros e artigos de diversos autores referentes aos principais conceitos que são a base para o desenvolvimento em questão. Foi necessária a pesquisa referente à Engenharia de Software para que o processo ou método de desenvolvimento que foi escolhido sirva para facilitar o desenvolvimento do jogo e elevar a qualidade do mesmo. Outra pesquisa importante é sobre a WEB, que é a plataforma onde o cliente do jogo é executado. Fazendo com que os leitores entendam o que é a WEB e como ela funciona.

Foi importante também a pesquisa de alguns trabalhos relacionados, possibilitando a visão de como foi feita a integração das partes de cada um deles, sendo possível absorver parte da experiência de outros autores. Contribuindo na hora de fazer algumas escolhas, como por exemplo, definir a utilização de alguma ferramenta ou tecnologia em vez de outra.

Alguns conhecimentos já foram adquiridos em sala de aula, sendo interessante uma revisão dos mesmos, para garantir que algum conceito que possa contribuir de maneira significativa no desenvolvimento do trabalho não passe despercebido. Entre esses conhecimentos estão às tecnologias: HTML, CSS, PHP, JavaScript, etc. E os conhecimentos obtidos através das disciplinas: Algoritmos, Redes de Computadores, Engenharia de Software e Banco de Dados.

A pesquisa de maior importância para o autor deste trabalho é a relacionada ao desenvolvimento cliente-servidor. Sendo totalmente necessária nesse projeto por ser um jogo *multiplayer*, havendo a comunicação entre os jogadores. Por isso é essencial seu estudo para o desenvolvimento desse trabalho.

4.2 Ferramentas e Métodos Utilizados

Após a realização das pesquisas definiu-se como seria montado o projeto e quais tecnologias seriam utilizadas. Essa foi a fase de Especulação do método ágil ASD adotado para o desenvolvimento desse projeto.

Como a fase de Especulação é o planejamento do projeto propriamente dito, ela envolve as pesquisas realizadas com o intuito de encontrar ferramentas e tecnologias eficientes e eficazes para o desenvolvimento do projeto e a metodologia de desenvolvimento descrita no presente capítulo.

A codificação do projeto foi realizada de acordo com a fase de Aprendizagem do método ágil ASD, onde o código é escrito de maneira que possa acomodar facilmente possíveis mudanças. A codificação foi escrita utilizando as linguagens de programação PHP e JavaScript. A programação foi efetuada na arquitetura cliente/servidor.

A programação orientada a eventos contribui de maneira bem significativa para a fase de Aprendizagem do método ágil ASD, pois é bem mais fácil fazer alterações no código fonte, quando já sabe onde se deve alterar. Dessa maneira, basta localizar no código o evento que se deseja alterar e então efetuar as alterações necessárias.

Como o projeto possui apenas um desenvolvedor, a fase de Colaboração é impossível de ser executada. Mas ela continua sendo de grande importância, pois se tratando de trabalho em equipe, é essencial que haja comprometimento e uma perfeita comunicação entre seus membros.

O banco de dados utilizado para armazenar as informações dos jogadores e das suas cartas foi o MYSQL, por ser gratuito, de fácil manipulação e por atender perfeitamente as necessidades do software proposto. Optou-se por não armazenar as imagens no banco de dados e sim em uma pasta do projeto chamada imagens. Em seu lugar no banco de dados ficou armazenado o nome das imagens, de maneira que a consulta retorna esse nome que é concatenado com o endereço onde se localiza as imagens. Isso foi feito para aumentar a velocidade das consultas no banco de dados e consequentemente a velocidade do software.

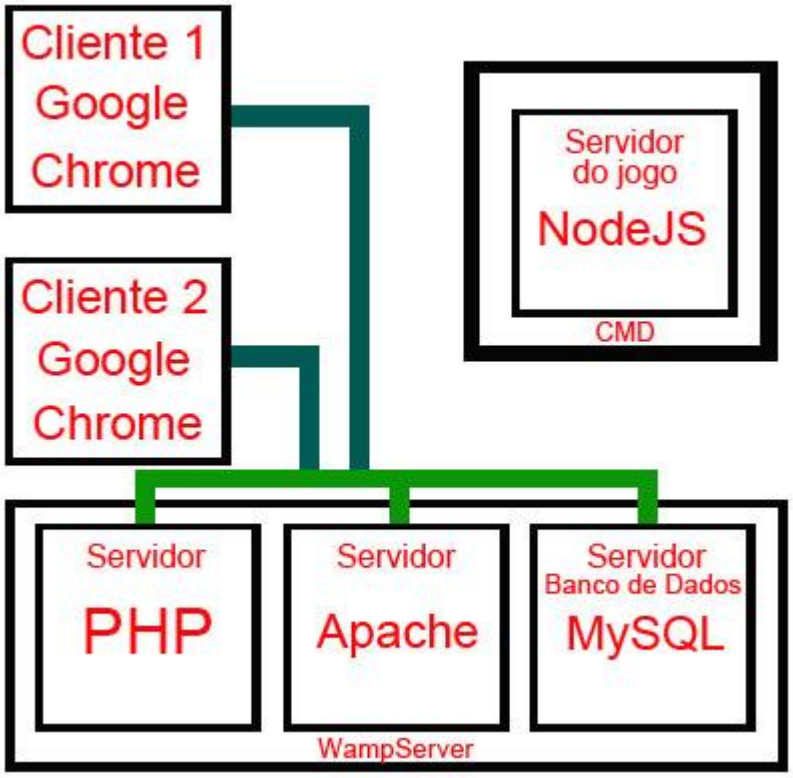
As imagens dos personagens utilizados no jogo foram desenhadas a mão, digitalizadas e trabalhadas através do editor de imagens Gimp que é uma excelente ferramenta de edição de imagens gratuita.

Foi utilizado o IDE (Ambiente de Desenvolvimento Integrado) NetBeans 7.0.1 como ferramenta de desenvolvimento do sistema. Desde as páginas em HTML, as formatações das páginas em CSS até o desenvolvimento do cliente em PHP e JavaScript e do servidor do jogo em JavaScript.

O servidor do jogo se comunica com os clientes através de conexões via *sockets* que são possíveis graças ao NodeJS. Devido a grande importância do framework NodeJS para o sucesso do projeto é extremamente necessário sua explicação mais detalhada. Como são muitas informações, cabe uma seção para descrevê-las. Então as informações sobre o NodeJS estão descritas na próxima seção.

O funcionamento da arquitetura local é mostrado pelas duas figuras a seguir. A Figura 1 mostra como é feita a integração do sistema na máquina local antes de iniciar uma partida.

Figura 1 – Arquitetura local antes da batalha



Fonte: Elaborada pelo autor

Antes de executar qualquer cliente, primeiro é preciso ligar o servidor do jogo. Isto é feito através do *prompt* de comando do sistema operacional.

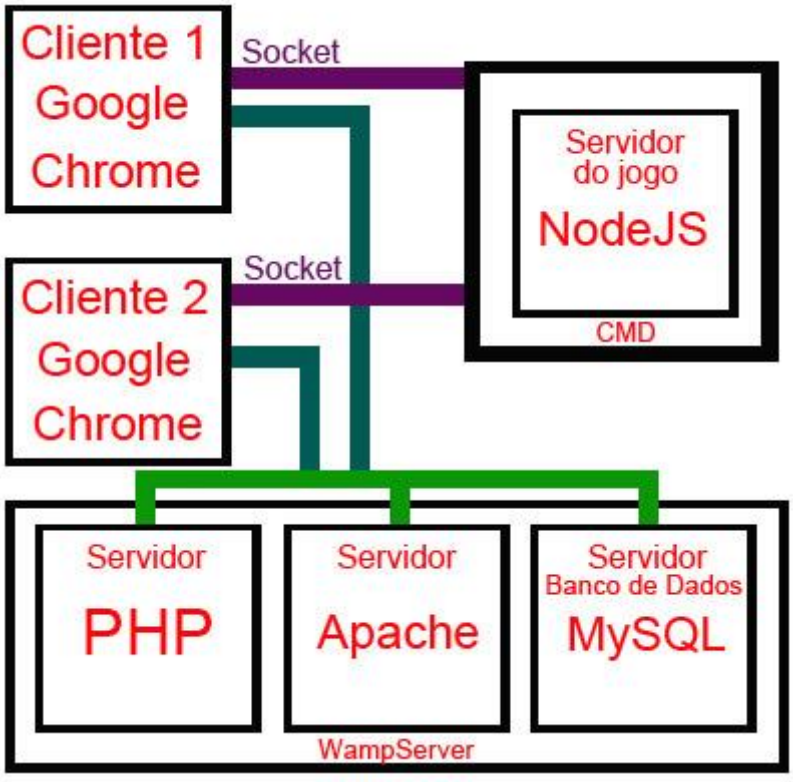
A comunicação é feita através de eventos disparados pelos clientes que acionam eventos no servidor que por sua vez emite as respostas solicitadas.

Assim que um jogador clica no *link* Jogar da página inventário, é feita a conexão via *socket* com o servidor do jogo executado pelo *prompt* de comando do Windows (CMD).

Conexão via *socket* é uma forma de comunicação entre processos que precisam se comunicar remotamente. (ALVES, 2008).

A Figura 2 retrata a conexão via *socket* estabelecida entre os clientes e o servidor do jogo.

Figura 2 – Arquitetura local durante a batalha



Fonte: Elaborada pelo autor

Depois que a partida é finalizada, a conexão via *socket* de ambos os clientes com o servidor é encerrada. Então a arquitetura volta a ser como na Figura 1. Lembrando que esse exemplo demonstra apenas dois clientes, mas é possível conectar vários clientes simultaneamente com o servidor de jogo, cada dupla em

suas respectivas batalhas. Sendo que pode haver clientes conectados via *socket* com o servidor e clientes desconectados.

A estrutura da página de batalhas foi codificada através de *tags* div HTML e essas divs foram formatadas através do CSS. Criaram-se três divs principais chamadas de esquerda, meio e direita. Dentro delas criaram-se as demais divs que fazem parte da página de batalhas. Permitindo que tudo fosse bem organizado e formatado de forma agradável e interativa ao usuário.

A JQuery, que é uma biblioteca de código aberto JavaScript lançada em 2006, foi usada no desenvolvimento do projeto como auxílio para facilitar a codificação de algumas tarefas e diminuir a quantidade de código escrito. (SILVA, 2010).

Outra ferramenta utilizada é o WampServer que é um pacote contendo alguns servidores necessários para o funcionamento da aplicação. Ele possui o servidor Apache, o servidor de banco de dados Mysql e o servidor PHP, para a interpretação dos comandos PHP. Como a aplicação é a versão inicial do jogo de cartas, não faz parte do trabalho sua hospedagem em um servidor. Por enquanto todos os testes foram executados localmente a partir da máquina de desenvolvimento que é de propriedade do próprio desenvolvedor.

4.3 NodeJS

Para possibilitar a comunicação entre cliente e servidor, foi usado o framework NodeJS, que trabalha com programação orientada a eventos. O NodeJS foi desenvolvido por Ryan Dahl em 2009, sendo um framework bem recente. Ele foi escrito nas linguagens de programação C/C++ e sua função é executar servidores implementados em JavaScript. (TIKOV; VINOSKI apud RIBEIRO JUNIOR, 2012).

A programação orientada a eventos é diferente da programação convencional, pois tudo é orientado a eventos emitidos pelos clientes nos momentos convenientes. O cliente não fica parado aguardando as respostas do servidor, sendo que pode continuar emitindo eventos antes mesmo de receber alguma resposta. (WELSH; CULLER; BREWER apud RIBEIRO JUNIOR, 2012).

Na programação convencional, a execução do cliente fica parada, aguardando as respostas do servidor toda vez que envia uma requisição. Não

podendo efetuar nenhuma outra requisição, enquanto a requisição já enviada não for respondida. (CANTELON; HOLOWAYCHUK apud RIBEIRO JUNIOR, 2012).

Essa forma convencional utiliza-se de threads (fluxos de controle sequenciais de um programa) para gerenciar as requisições concorrentes de vários clientes. Ela tem se mostrado menos eficiente quando o número de clientes simultâneos vai aumentando, já que são abertas novas threads para cada cliente. Essas threads são concorrentes em relação à resposta do servidor, que acaba ficando sobrecarregado com o alto número de requisições. Fazendo com que a comunicação cliente/servidor fique lenta. (WELSH; CULLER; BREWER apud RIBEIRO JUNIOR, 2012).

Na programação orientada a eventos, esta questão de concorrência é trabalhada de forma bem mais eficiente. São feitas através de várias instâncias do processo. Possibilitando a concorrência eficiente de um número bem maior de clientes. (TIKOV; VINOSKI apud RIBEIRO JUNIOR, 2012).

5 RESULTADOS

Como o trabalho proposto é a versão inicial de um jogo de cartas, cujos clientes são executados pelo navegador WEB e o autor não ser um profissional em artes gráficas, elaborou-se os desenhos de acordo com a capacidade do mesmo. As imagens das cartas foram desenhadas a mão, digitalizadas e pintadas com o auxílio da ferramenta Gimp. Para as versões futuras deseja-se trabalhar e melhorar as imagens, que é um trabalho independente da estrutura do jogo. Já que ao fazer as alterações nas imagens, basta deixá-las no mesmo diretório e com o mesmo nome, sem necessidade de mexer na estrutura do projeto.

A figura a seguir é uma carta que representa um dos guerreiros do jogo.

Figura 3 – Carta guerreiro

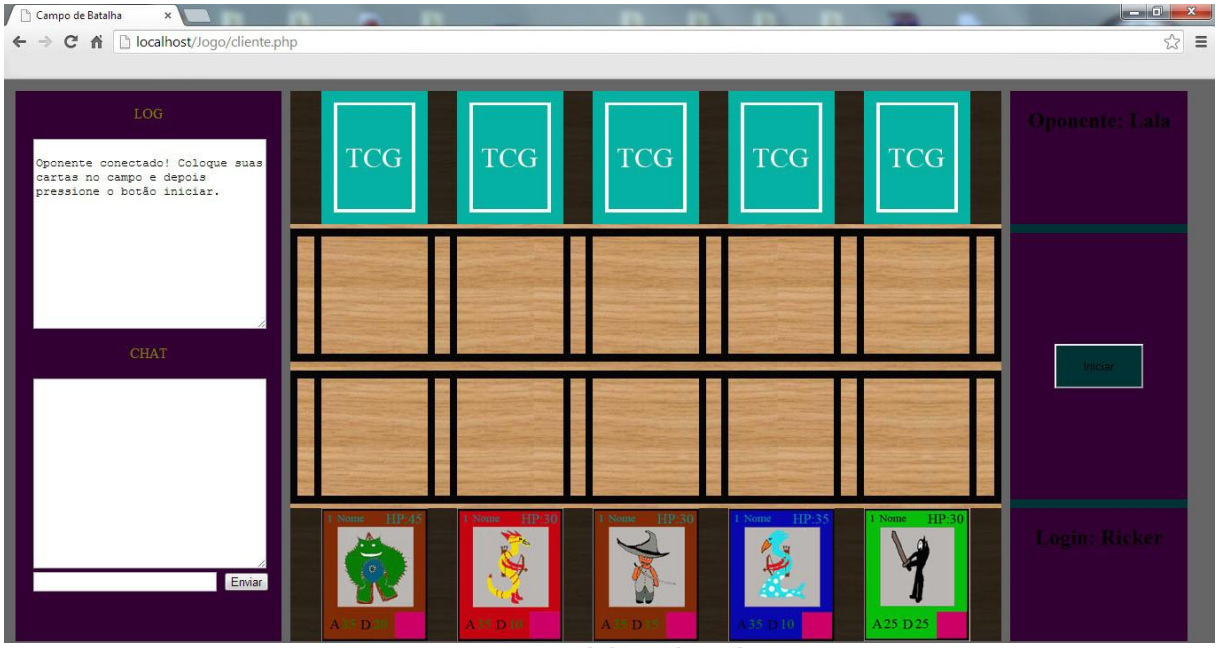
Fonte: Elaborada pelo autor

O número 1 no canto superior esquerdo representa o nível da carta. Para esta versão inicial, todas as cartas estão no nível um. Em seguida vem o nome da carta, as cartas não foram nomeadas, no lugar de seus futuros nomes, deixou-se a palavra nome. No canto superior direito estão os pontos de vida totais do guerreiro, são os pontos de vida máximos que ele pode ter. Já no canto inferior direito são representados os pontos de vida que o guerreiro tem no momento. E quando se chega à zero, o guerreiro é derrotado. O valor a frente da letra A representa o poder de ataque do guerreiro e o valor a frente da letra D, seu poder de defesa.

A tela de batalha é dividida em três seções: esquerda, direita e meio. A seção da esquerda é onde existe o *Chat* para a comunicação com o oponente e o campo

de texto chamado *Log*, onde são escritas as informações sobre o jogo vindas do servidor. A seção da direita mostra informações sobre os competidores e contém o botão iniciar. Já a seção do meio é onde ocorre a batalha das cartas. Esta seção é dividida em quatro partes. Cortadas horizontalmente e listadas de cima para baixo temos: mão do oponente, campo do oponente, campo do usuário e mão do usuário. Isso pode ser observado através da Figura 4.

Figura 4 – Campo de batalha no início



Fonte: Elaborada pelo autor

5.1 Principais Regras do Jogo

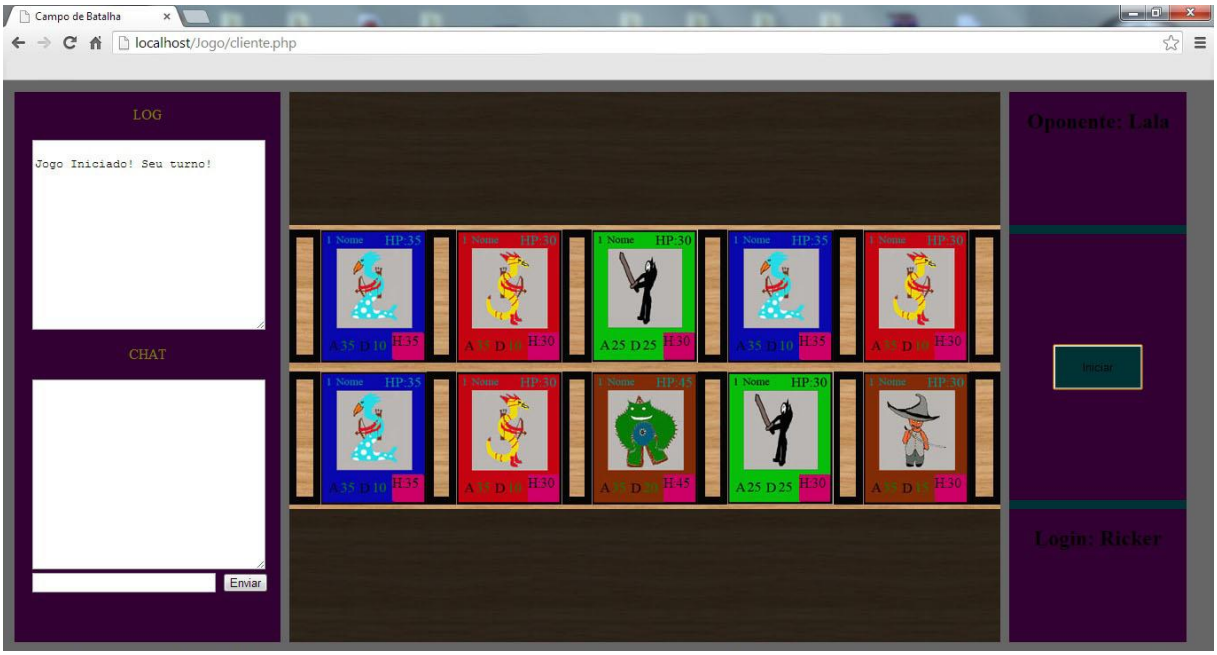
Nesta versão inicial, cada jogador possui cinco guerreiros e ao início da batalha precisa escolher em qual posição do campo ocupará cada um de seus guerreiros. Assim que o campo for preparado o jogador deve pressionar o botão iniciar que se encontra no canto direito da tela de batalha e aguardar até que seu oponente também faça o mesmo. Após a confirmação de ambos os jogadores, é possível ver o campo do oponente com seus respectivos guerreiros na posição escolhida. E na janela de *Log* localizada no canto superior esquerdo da tela de batalha sempre será mostrado de quem é a vez de jogar.

O primeiro jogador a atacar é o jogador que criou a partida. Como não se sabe se existe ou não partida criada, não há como os jogadores manipularem quem

vai criar a partida. Dessa maneira, o que influenciará na decisão de quem será o primeiro jogador a jogar é a sorte de clicar no *link* jogar em um momento que não exista um jogo criado a espera de um oponente. Por enquanto, nesta versão inicial, quem começa a jogar tem vantagem em relação ao seu oponente, pois o jogador que atacar primeiro, estará um ataque à frente.

A Figura 5 retrata o campo de batalha já preenchido por ambos os jogadores, com as cartas nas posições escolhidas por eles.

Figura 5 – Campo de batalha preenchido



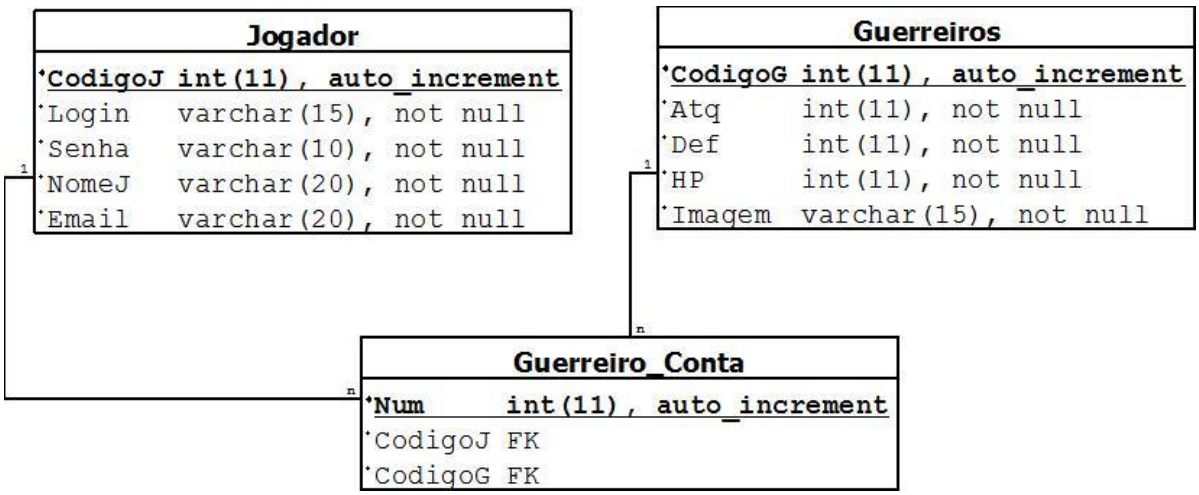
Fonte: Elaborada pelo autor

Para realizar um ataque, o jogador precisa selecionar um de seus guerreiros que será o atacante e depois selecionar o guerreiro do oponente que ele deseja atacar. O cálculo do dano é bastante simples, subtrai-se do HP (pontos de vida) do guerreiro alvo a diferença entre o ataque do guerreiro atacante e a defesa do guerreiro alvo. Se esta diferença resultar em um valor negativo então será aplicado o dano mínimo que será de dois pontos de vida. Quando o HP de um guerreiro chega a zero o guerreiro é derrotado. Quando um guerreiro é derrotado, sua imagem desaparece do campo de batalha. Vence a batalha o primeiro jogador a derrotar três guerreiros. Ao concluir um ataque, a vez de atacar é passada para o adversário e assim sucessivamente até que um dos jogadores vença a batalha.

5.2 Banco de Dados

O banco de dados criado para esta versão inicial possui três tabelas: tabela Guerreiros, tabela Guerreiro_Conta e tabela Jogador. A estrutura do banco de dados pode ser vista com detalhes através do MER (Modelo de Entidade e Relacionamento) demonstrado pela figura a seguir:

Figura 6 – Modelo de Entidade e Relacionamento (MER)



Fonte: Elaborada pelo autor

A tabela Guerreiros possui os seguintes atributos: CódigoG, Atq, Def, Hp e Imagem. O CódigoG é a chave primária da tabela e o atributo Imagem armazena o nome da imagem da carta. Todas as imagens das cartas estão localizadas dentro do diretório chamado imagens, localizado dentro do diretório raiz do projeto. Os demais atributos desta tabela não precisam ser explicados, pois seus nomes já são autoexplicativos.

A tabela Jogador possui os seguintes atributos: CódigoJ, Login, Senha, NomeJ e Email. O CódigoJ é a chave primária da tabela Jogador. O atributo Login é o nome que o usuário usará para se conectar a sessão do jogo e o atributo senha em conjunto com o Login fazem a validação do usuário. O atributo NomeJ é usado para armazenar o nome completo do jogador e o atributo Email para armazenar o email cadastrado pelo jogador.

Por fim, a tabela Guerreiro_Conta serve para identificar quais guerreiros os jogadores possuem em suas respectivas contas. Essa tabela possui os seguintes atributos: Num, CódigoJ e CódigoG. O Num é a chave primária da tabela

Guerreiro_Conta, O CódigoJ é a chave estrangeira que referencia a tabela Jogador e o CódigoG é a chave estrangeira que referencia a tabela Guerreiros.

Para essa versão inicial, cada jogador possui somente cinco guerreiros em sua conta, que é a quantidade necessária para se jogar uma batalha. Então são somente cinco instâncias na tabela Guerreiro_Conta para cada jogador.

5.3 Cliente/Servidor

Como a aplicação faz comunicação entre usuários, foi necessário seu desenvolvimento através da arquitetura cliente/servidor. Tanto o cliente quanto o servidor foram implementados utilizando a linguagem de programação WEB JavaScript. A comunicação entre cliente e servidor nesse trabalho é orientada a eventos. O servidor responde a eventos disparados pelos clientes, com outros eventos que serão executados no cliente. Os cálculos referentes ao jogo são processados pelo servidor do jogo e enviados para os clientes.

5.3.1 Cliente

O cliente é o próprio site do jogo, onde acontece a interação com o usuário. Ao abrir a página inicial do jogo, o jogador terá acesso à tela de *login* para a validação do mesmo. Caso não possua um cadastro, ele tem a opção de clicar no *link* Cadastrar para preencher o formulário de cadastro e então efetuar o *login*.

Figura 7 – Tela de login

Fonte: Elaborada pelo autor

Ao iniciar a sessão, o usuário é redirecionado à página chamada inventário que não será totalmente implementada nessa versão inicial do projeto, porque o foco nessa versão é a parte de batalha, por ser a parte essencial do jogo. Futuramente será implementada na página inventário a função de manuseio das cartas, onde os usuários terão a opção de escolher com quais cartas irão para cada batalha. Por enquanto a página inventário possui apenas a função Jogar. Ao clicar no *link* Jogar o usuário é redirecionado para a página de batalha, onde aguardará pelo seu oponente. Esta conexão é feita de forma aleatória, os dois primeiros a clicarem no *link* Jogar se enfrentarão. Os próximos também se enfrentarão e assim sucessivamente.

A qualquer momento os jogadores poderão trocar mensagens particulares pelo *Chat,* localizado no canto inferior esquerdo da tela de batalha.

A comunicação com o banco de dados foi feita através do PHP. Pegam-se os atributos HP, Atq, Def e Imagem da tabela Guerreiros para cada guerreiro e os armazena em variáveis PHP. São usados quatro vetores PHP de cinco posições cada, para armazenar esses dados.

O atributo imagem, concatenado com o endereço da pasta imagens, onde se localiza as imagens de todos os guerreiros do jogo serve para inserir a imagem das cartas dos jogadores em suas respectivas mãos.

À medida que os jogadores vão escolhendo em qual campo colocar seus guerreiros, os atributos HP, Atq e Def referentes a esses guerreiros são passados dos vetores PHP para vetores JavaScript na posição referente ao campo escolhido. Logo após o campo estiver preenchido por todos os guerreiros, os vetores HP, Atq e Def JavaScript também estarão totalmente preenchidos. Possibilitando agora o envio dos valores contidos nessas variáveis para o servidor do jogo.

5.3.2 Servidor

O servidor do jogo é o responsável pela criação de batalhas, pela conexão entre os jogadores, pelo armazenamento de informações dos clientes, pelo processamento dos dados e pela resposta a eventos solicitados pelos clientes.

Cada batalha possui dois jogadores e o servidor permite que aconteçam várias batalhas simultâneas.

Um novo jogo é criado sempre que um usuário busca por um jogo e não encontra. Ao criar o jogo é preciso aguardar até que outro usuário o encontre para fazer a conexão e iniciar a partida.

Todos os jogos criados são armazenados em uma lista de objetos chamada *games* e o índice de cada jogo é gerado de acordo com a data e hora no momento da criação associado ao *login* do jogador que o criou. Dessa maneira todos os índices serão únicos, não havendo repetições. Assim que o jogo é criado, seu respectivo índice é enviado para ambos os clientes conectados, ficando armazenados nos mesmos. Isso é feito para que o servidor saiba para quem responder os eventos solicitados. Por este motivo, todos os eventos emitidos pelos clientes possuem o parâmetro game_id. Além do game_id, cada jogo armazena as informações de ambos os jogadores conectados através dos seus *sockets* de comunicação e todas as informações referentes ao jogo, obtidas através dos eventos emitidos pelos clientes. Essas informações são: O HP, Atq e Def das cartas de ambos os jogadores. É extremamente necessário o armazenamento dessas informações porque o jogo é executado no servidor que recebe as informações dos clientes, processa essas informações e envia as respostas para os mesmos.

Esse mesmo servidor do jogo também é responsável pelo serviço de Chat. Sendo que ao ser enviada uma mensagem, dispara um evento para o servidor que faz com que ele responda emitindo um evento para os dois clientes, informando o nome de quem enviou a mensagem juntamente com a própria mensagem.

Foi preciso bloquear o evento de chat enquanto o jogador fica a espera de seu oponente para que não ocorresse um erro no servidor do jogo. Assim que ambos os jogadores se conectarem na batalha, o evento de chat é liberado, possibilitando eventuais conversas entre os mesmos.

5.4 Programação dos Eventos do Jogo

Toda a comunicação entre cliente e servidor é orientada a eventos. A comunicação é possível graças à conexão via *socket*. Dentre os principais eventos estão à conexão com o servidor, a criação de um novo jogo, a conexão em um jogo já existente, preparar o campo, o evento atacar, o evento começar e o evento de chat.

A seguir, as Figuras 8 e 9 apresentam fotos de partes do código tanto do lado cliente como do lado servidor que demonstram como é feita a comunicação de chat entre os jogadores. Dessa maneira é possível entender um pouco sobre a comunicação orientada a eventos. Foi escolhido esse trecho do código para demonstração por ser uma parte independente do jogo e de fácil compreensão, já que o objetivo do *chat* é somente enviar e receber mensagens.

Figura 8 – Chat do lado cliente

```
//Função que envia mensagens pelo chat.
function enviarMSG(){
    if(chavechat){
        msg = login+" disse: "+$('#textchat').val();
        socket.emit('msgparticular',{msg: msg,game_id: game_id});
        $('#textchat').val('');
    }
    else {
        var messageE1 = document.getElementById('textareachat');
        messageE1.innerHTML +="\nNinguém conectado para receber a mensagem!";
    }
}

//Trata o evento chamado retornoMSG vindo do servidor.
socket.on('retornoMSG', function(data){
       var messageE2 = document.getElementById('textareachat');
        // Exibindo a mensagem enviada
        messageE2.innerHTML +="\n"+ data.data;
 });
```

Fonte: Elaborada pelo autor

Para envio de mensagens, o usuário precisa digitar a mensagem em um campo de texto chamado textchat que se encontra abaixo da janela de *chat* da página de batalha e clicar no botão enviar. Ao clicar nesse botão, a função enviarMSG é executada.

A função é demonstrada pela figura 8. Ao executá-la é testada a condição da chavechat, que se torna verdadeira somente depois que ambos os jogadores estejam conectados no jogo. Se for verdadeira, então é gerada uma mensagem e armazenada na variável msg. Essa mensagem é composta pelo *login* do jogador que enviou a mensagem em conjunto com a palavra “disse:”, mais a mensagem escrita no campo textchat.

A partir deste momento é emitido um evento para o servidor através do comando socket.emit, onde msgparticular é o nome dado para esse evento e os parâmetros enviados para o servidor são a variável msg e o game_id para que o servidor possa saber em qual jogo foi enviada a mensagem. Em seguida, os dados escritos referentes à mensagem no campo textchat são apagados.

Caso a chavechat possua valor falso, quer dizer que o segundo jogador ainda não se conectou ao jogo, então é mostrada a mensagem: “Ninguém conectado para receber a mensagem!”, na área de texto do *chat* chamada textareachat.

A Figura 9 demonstra o que acontece no lado do servidor.

Figura 9 – Chat do lado servidor

```
//Trata o evento chamado msgparticular.
socket.on('msgparticular', function(data){
    var game = games[data.game_id];
            game.player1.emit('retornoMSG',{data: data.msg});
            game.player2.emit('retornoMSG',{data: data.msg});

});
```

Fonte: Elaborada pelo autor

Como o evento que foi enviado é chamado msgparticular, o servidor executará o código de resposta referente a esse evento. Os eventos são identificados pelo nome que se encontra dentro do código socket.on. Foram enviados dois parâmetros pelo cliente, msg e game_id. Para acessá-los é preciso usar o nome data que está definido como parâmetro desta função, seguido de ponto final e o nome dos parâmetros enviados pelo cliente.

Em seguida, o servidor busca em sua lista de games, o game cujo game_id é o mesmo do cliente que foi enviado por parâmetro. E emite para ambos os jogadores conectados neste game um evento com a mensagem enviada. Para esse evento foi dado o nome de retornoMSG e o único parâmetro enviado para os clientes é a mensagem.

Agora de volta a figura 8. O socket.on em ambos os clientes de nome retornoMSG receberão esse evento. E faz com que essa mensagem apareça na janela de *chat*, chamada textareachat. Pode ser visto um exemplo de uma conversa através da Figura 10.

Figura 10 – Exemplo de conversa pelo chat

Fonte: Elaborada pelo autor

Foram implementados vários eventos no desenvolvimento do jogo, cada evento possui um nome e um modo de ativação para disparar as ações implementadas nele. É possível enviar dados via parâmetro pelos eventos, para isso basta escrever entre colchetes o nome do parâmetro, seguido do sinal de pontuação dois pontos (:) e o parâmetro que se quer enviar. Para enviar mais de um parâmetro, basta separá-los por vírgula dentro dos colchetes seguindo a mesma regra descrita anteriormente. Já no local onde será tratado o evento emitido, para ter acesso às informações dos parâmetros, usa-se apenas um nome dentro da função, que eventualmente é chamado data, mas pode ser utilizado qualquer nome. Esse nome usado é escrito antes do nome dos parâmetros que foram enviados da origem, separados pelo sinal de ponto final. Por exemplo: data.nome_do_parametro.

Deu para perceber que a programação orientada a eventos não é tão difícil. Esta demonstração simples contribuiu um pouco para que o leitor que nunca teve contato com esse tipo de programação possa entender um pouco sobre o seu funcionamento.

A seguir, são apresentados dois quadros contendo informações referentes a todos os eventos do jogo. O Quadro 1 contém os eventos ativados pelo lado do cliente e o Quadro 2, contém os eventos ativados pelo lado do servidor. Ambos os

quadros demonstram o nome dos eventos, como são ativados e a descrição de suas ações.

Quadro 1 - Eventos do lado cliente

Eventos Cliente	Ativação	Descrição
buscarop	Ao abrir a página de batalha.	Busca no servidor a existência de um jogo criado que esteja aguardando um oponente.
createGame	Evento noFreeGames emitido pelo servidor.	Solicita ao servidor a criação de um novo jogo.
nome	Evento gameStart emitido pelo servidor.	Envia para o servidor o Login do usuário conectado. Envia também o ID do jogo para que o servidor saiba em qual jogo realizar a operação.
comecar	Evento retorno_campo emitido pelo servidor.	Envia para o servidor o ID do jogo que será iniciado.
msgparticular	Ao clicar no botão enviar do chat.	Envia para o servidor a mensagem desejada. Envia também o ID do jogo para que o servidor saiba para quem encaminhar a mensagem.
prep_campo	Ao clicar no botão iniciar.	Envia o nome, ataque, defesa e HP de todas as cartas do usuário. Informando suas respectivas posições no campo de batalha. Envia também o ID do jogo para que o servidor saiba em qual jogo armazenar os dados.
ataque	Ao clicar na carta alvo.	Envia o ID da carta atacante e da carta alvo para o servidor aplicar a fórmula de dano. Envia também o ID do jogo para que o servidor saiba em qual jogo realizar os cálculos.
fim	Evento vitoria emitido pelo servidor.	Envia para o servidor o ID do jogo que será finalizado.

Fonte: Dados da pesquisa

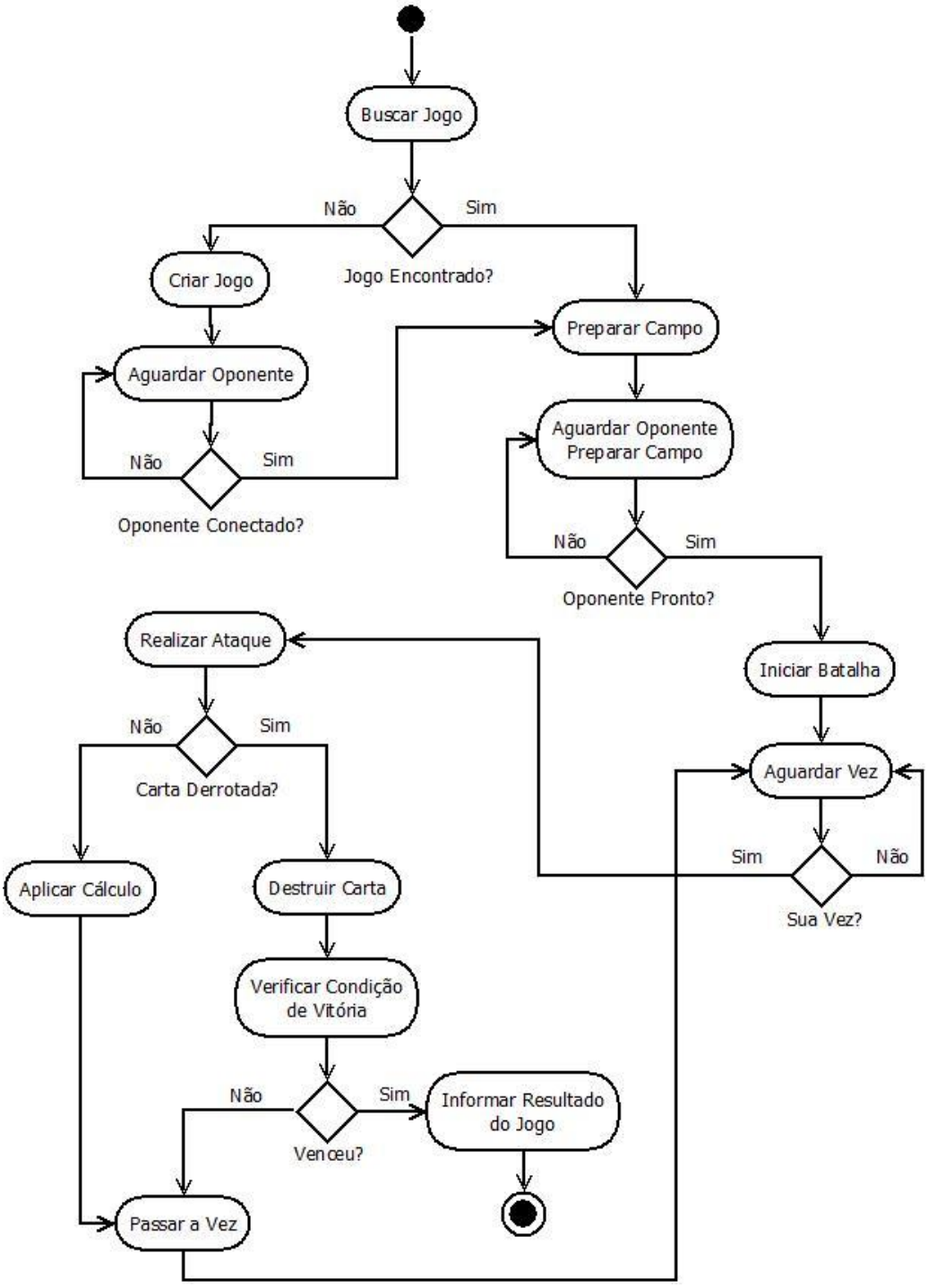
Quadro 2 - Eventos do lado servidor

Eventos Servidor	Ativação	Descrição
noFreeGames	Evento buscarop emitido pelo cliente, caso não encontre um jogo.	Informa ao cliente que não foi encontrado um jogo disponível.
gameCreated	Evento createGame emitido pelo cliente.	Envia para o cliente que criou o jogo, o ID do jogo criado.
gameStart	Evento buscarop emitido pelo cliente, caso encontre um jogo.	Envia para o cliente o ID do jogo encontrado.
nomeop	Evento nome emitido pelo cliente.	Envia para o cliente o Login do oponente.
messagep	Evento msgparticular emitido pelo cliente.	Encaminha para o destinatário a mensagem enviada.
retorno_campo	Evento prep_campo emitido pelo cliente.	Envia para o cliente o nome e a posição das cartas do oponente.
comecando	Evento comecar emitido pelo cliente.	Informa para ambos os clientes que a partida começou.
RAtq	Evento ataque emitido pelo cliente.	Retorna para o cliente o resultado de seu ataque, informando qual o HP resultante da carta alvo e qual é a carta alvo.
RAtqOp	Evento ataque emitido pelo cliente.	Retorna para o cliente o resultado do ataque de seu oponente, informando qual o HP resultante da carta alvo e qual é a carta alvo.
Matou	Evento ataque emitido pelo cliente, quando uma carta do oponente é derrotada.	Informa para o cliente a carta do oponente que foi derrotada.
Morto	Evento ataque emitido pelo cliente, quando uma de suas cartas é derrotada.	Informa para o cliente qual de suas cartas foi derrotada.
Vitoria	Evento ataque emitido pelo cliente quando atinge a condição de vitória.	Envia para o cliente vitorioso uma mensagem informando sua vitória.
Derrota	Evento ataque emitido pelo cliente quando atinge a condição de derrota.	Envia para o cliente derrotado uma mensagem informando sua derrota.

Fonte: Dados da Pesquisa

Por fim, no intuito de melhorar a compreensão do leitor a respeito do funcionamento do jogo desenvolvido. É apresentado um Diagrama de Atividades por meio da Figura 11:

Figura 11 – Diagrama de Atividades do Jogo



Fonte: Elaborada pelo autor

6 CONCLUSÃO

O presente trabalho contribuiu significativamente para o desenvolvimento intelectual do autor, que absorveu novos conhecimentos a respeito de diferentes tecnologias e aplicou na prática todos esses conhecimentos.

Dentre tais tecnologias podem-se citar as linguagens de programação PHP e JavaScript, a Linguagem de Marcação de Hipertexto (HTML), CSS, banco de dados MySQL e o *framework* NodeJS, através de programação orientada a eventos. Também fez parte do trabalho elaborar as regras do jogo e o desenho das cartas.

O principal desafio encontrado, que desde o inicio já era previsto, foi à comunicação cliente/servidor. Ela foi realizada utilizando o *framework* NodeJS que se mostrou muito eficiente na construção do servidor do jogo. A programação orientada a eventos que até então era desconhecida para o autor, foi uma surpresa que enriqueceu seus conhecimentos, facilitando bastante à programação do jogo.

Foi muito gratificante para o autor o sucesso deste trabalho. Que além de contribuir para o aprendizado do mesmo, serve como referência acadêmica para quem deseja desenvolver algo semelhante.

É de interesse do autor a continuidade do projeto. Desenvolvendo as demais funcionalidades do jogo não citadas nesse trabalho, a implementação das funções da página inventário e o aperfeiçoamento das imagens das cartas.

REFERÊNCIAS

ALVES, Maicon Melo. **Sockets Linux.** Brasport, 2008.

ARAÚJO JÚNIOR, Mário Barbosa. **Um Estudo Comparativo de Tecnologias WEB para Games.** Monografia - Universidade Federal de Pernambuco, Graduação em Ciência da Computação, Recife. Disponível em: < http://www.cin.ufpe.br/~tg/2009-1/mbaj.pdf> Acesso em: 14 mai. 2013.

BRASIL fechará o ano com 59 milhões de celulares vendidos. **Prevê IDC.** 19 dezembro 2012. Disponível em: <http://www.mobiletime.com.br/19/12/2012/brasil-fechara-o-ano-com-59-milhoes-de-celulares-vendidos-preve-idc/317660/news.aspx> Acesso em: 11 abr. 2013.

BEAL, Vangie. **The Difference Between the Internet and World Wide Web**. Disponível em: <http://www.webopedia.com/DidYouKnow/Internet/2002/Web_vs_Internet.asp> Acesso em: 26 set. 2013.

CARVALHO, Diego Eduardo. **Desenvolvimento do Jogo Robogol para Navegadores WEB Utilizando o Motor de Jogos Unity 3D.** Monografia - Universidade de São Paulo, Graduação em Ciência da Computação, São Carlos. Disponível em: <http://www.xbot.com.br/wp-content/uploads/2012/11/Monografia_EstagioSupervionadoII_DiegoEduardoCarvalho.pdf> Acesso em: 14 mai. 2013.

COELHO, Cristiane dos Santos. **Relato de Experiência na Implantação de Um Método Ágil em Uma Equipe de Desenvolvimento de Software.** Monografia - Universidade Federal de Lavras, Graduação em Ciência da Computação, Lavras. Disponível em: < http://www.bsi.ufla.br/monografias/2011/CristianeCoelho.pdf> Acesso em: 03 mai. 2013.

CORRAL, Edson**; Conhecendo a Unity 3D.** Disponível em: <http://www.sharpgames.net/Artigos/Artigo/tabid/58/selectmoduleid/376/ArticleID/1661/reftab/36/Default.aspx> Acesso em: 03 mai. 2013.

COSTA, Gustavo Henrique de Carvalho. **Engenharia de Requisitos no Desenvolvimento de Software Ágil.** Monografia - Universidade Federal de Pernambuco, Centro de Informática, Pós-Graduação em Ciência da Computação, Recife. Disponível em: < http://www.cin.ufpe.br/~in1020/arquivos/monografias/2010_2/Monografia_Gustavo_Carvalho.pdf> Acesso em: 03 mai. 2013.

DANTAS, Daniel Chaves Toscano; GUIMARÃES, Eliseu Paz e Silva de. **Sistema Especialista de Inteligência Artificial Integrado a Plataforma WEB para Entretenimento em um Ambiente Interativo Multiusuário.** Universidade Federal do Rio de Janeiro, Escola Politécnica, Departamento de Eletrônica e Computação, Rio de Janeiro. Disponível em: <http://monografias.poli.ufrj.br/monografias/monopoli10002174.pdf> Acesso em: 14 mai. 2013.

FIRMINO, Emiliano Carlo M.; SILVA, Jucimar Maia Jr. **Desenvolvimento de Jogos em HTML5.** Artigo - Universidade do Estado do Amazonas, Coordenação de Engenharia da Computação. Disponível em: <http://sbgames.org/sbgames2010/proceedings/computing/short/Computing_short13.pdf> Acesso em: 25 set. 2013.

GUEDES, Anibal Lopes; GUEDES, Fernanda Lopes; SEBBEN, Naiara. **Desenvolvimento de Jogos para a Terceira Idade.** Artigo - Universidade do Oeste de Santa Catarina. Disponível em: <http://www.sbgames.org/papers/sbgames07/short/17.pdf> Acesso em: 25 set. 2013.

HAHN, Harley; STOUT, Rick. **Dominando a Internet.** Makron Books, 1995.

MERCADO brasileiro de games já é o quarto maior do mundo e deve continuar a crescer. **Folha de S.Paulo,** São Paulo, 08 Outubro, 2012. Disponível em: <http://www1.folha.uol.com.br/tec/1165034-mercado-brasileiro-de-games-ja-e-o-quarto-maior-do-mundo-e-deve-continuar-a-crescer.shtml> Acesso em: 11 abr. 2013.

PAULA FILHO, Wilson de Pádua**. Engenharia de Software:** Fundamentos, Métodos e Padrões. 3ª ed. São Paulo: LTC, 2009.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Pearson, 2010.

RIBEIRO JUNIOR, Francisco de Assis. **Programação Orientada a Eventos no Lado do Servidor Utilizando Node.JS.** Artigo – Ifactory Solutions, Fortaleza - CE. Disponível em: < http://www.infobrasil.inf.br/userfiles/16-S3-3-97136-Programa%C3%A7%C3%A3o%20Orientada___.pdf> Acesso em: 25 set. 2013.

SANTOS, André O.; SOUZA, Cleumir dos S.; NASCIMENTO, Tiago L. **Taskboard Digital para Equipes Scrum.** Monografia – Universidade Anhembi Morumbi, Graduação em Ciência da Computação, São Paulo. Disponível em: <http://engenharia.anhembi.br/tcc-11/cco-07.pdf> Acesso em: 03 mai. 2013.

SILVA, Maurício Samy. **JQuery:** A Biblioteca do Programador JavaScript. 2ª ed. Novatec, 2010.

SOMMERVILLE, I. **Engenharia de Software.** 8ª ed. Pearson, 2007.