José Carlos Costa Milito

LearnCraft - Uma *Engine* para Criação de Jogos RPG Construcionistas

Maceió-AL
2018

José Carlos Costa Milito

LearnCraft - Uma *Engine* para Criação de Jogos RPG Construcionistas

Dissertação apresentada ao Curso de Mestrado em Modelagem Computacional do Conhecimento da Universidade Federal de Alagoas como requisito parcial para obtenção do grau de Mestre em Modelagem Computacional de Conhecimento

Universidade Federal de Alagoas – UFAL
Instituto de Computação
Programa de Pós-Graduação Mestrado em Modelagem Computacional de Conhecimento

Orientador: Arturo Hernández Domínguez

Maceió-AL
2018

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecária Responsável: Helena Cristina Pimentel do Vale – CRB4 - 661

M644l Milito, José Carlos Costa.

Learncraft : uma engine para criação do jogos RPG construcionistas / José Carlos Costa Milito. – 2018.

94 f. : il.

Orientador: Arturo Hernández Domínguez.

Dissertação (mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2018.

Bibliografia: f. 87-92.

Apêndices: f. 93-94.

1. Jogos digitais RPG. 2. Ensino e aprendizagem. 3. Construcionismo. 4. Role playing games. 5. Engines. I. Título.

CDU: 004.4:37

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Modelagem Computacional de Conhecimento
Avenida Lourival Melo Mota, Km 14, Bloco 12, Cidade Universitária
57.072-900 Maceió AL Brasil CGC: 24.464.109/0001-48
Telefone: (082) 3214-1364/1825

MODELAGEM COMPUTACIONAL DE CONHECIMENTO

Ata da defesa de dissertação do aluno
José Carlos Costa Milito

Realizou-se no dia 21 de maio de 2018, a partir das 17h00min, na sala de reuniões do Instituto de Computação da Universidade Federal de Alagoas, a defesa de dissertação de Mestrado em Modelagem Computacional de Conhecimento, intitulada "LearnCraft – Uma Engine para Criação de Jogos RPG Construcionistas", apresentada por José Carlos Costa Milito, graduado em Ciência da Computação, como requisito parcial para a obtenção do grau de Mestre em Modelagem Computacional de Conhecimento, perante a seguinte comissão examinadora:

Professor e orientador Arturo Hernández-Domínguez
Instituto de Computação – Ufal

Professor Marcus de Melo Braga
Instituto de Computação – Ufal

Professor Leandro Dias da Silva
Instituto de Computação – Ufal

A dissertação foi considerada aprovada pela Comissão. Finalizados os trabalhos, às 19h30min, lavrou-se a presente ata, que vai assinada pelos membros da Comissão.

Maceió, 21 de maio de 2018.

.
Arturo Hernández-Domínguez
Doutor em Informática – Université Toulouse III Paul Sabatier/França

.
Marcus de Melo Braga
Doutor em Engenharia e Gestão do Conhecimento – Universidade Federal de Santa Catarina/Brasil

.
Leandro Dias da Silva
Doutor em Engenharia Elétrica – Universidade Federal de Campina Grande/Brasil

Agradecimentos

Aos meus pais, por dar início a tudo isso
A Aline por manter tudo isso inteiro e pela inspiração de todas as horas de Ragnarok Online
Ao Arturo, pela paciência em mostrar o caminho das pedras até aqui, e pelo Jarabe Tapatio, as coronas, as margaritas e os burritos mais à frente. ¡Viva México!
Ao Marcus, eterno guru, pelas pinceladas de mestre nesta tela. PQLPFC!
A Leandro, velho parceiro de copo e metal, por outras tantas pinceladas. E não saio daqui antes de tomar uma grade
A Nailton e demais toscos pela inspiração de todas as madrugadas de World of Warcraft. Lok'tar Ogar friends!
A Dirk, Lerxst e Pratt pela metade da trilha sonora de minha vida. And the meek shall inherit the earth
A Michi, Kai, Weiki, Markus e Ingo pela outra metade. Our only hope's your victory, so follow the sign
E agora, para algo completamente diferente, a John, Eric, Graham, Michael e os dois Terries, pelo cálice sagrado e consequente inspiração nas skills do mago e naqueles coelhos vis, cruéis e mal humorados que habitam os campos de LearnCraft. Always look on the bright side of life!

Resumo

A evasão escolar constitui um problema global, também se verificando no Brasil, com altas taxas de abandono e reprovação nos ensinos fundamental e médio; dentre os motivos para tal, predomina a falta de interesse por parte do alunado. Jogos, ao se configurar como ofertantes de experiências de aprendizado, ao mesmo tempo, motivantes, engajadoras e prazerosas para os aprendizes, aparecem como um elemento significativo no auxílio à equação destes problemas. Dentro desse contexto, o construcionismo aparece como uma preciosa fonte de subsídios teóricos para a elaboração de jogos educacionais, com pesquisas demonstrando que o processo de aprendizado apresenta o potencial de ocorrer de forma mais natural e satisfatória quando o aprendiz está engajado na construção de um artefato. Dentre os gêneros de jogos, destacam-se hoje os Role Playing Games, com grande apelo dentre as novas gerações de alunos. Tendo em vista tal potencial educativo, e como uma contribuição à equação dos problemas supracitados, neste trabalho é exposta a autoria de uma ferramenta capaz de gerar jogos construcionistas dentro do gênero RPG, sem a necessidade de conhecimento de linguagem de programação por parte do autor.

Palavras-chave: Educação, construcionismo, jogos, *Role Playing Games, engines*

Abstract

School dropout is a global problem, also occurring in Brazil, with high rates of dropout and disapproval in elementary and high school; among the reasons for this, the lack of interest on the part of the student predminates. Games, as providers of learning experiences, at the same time, motivating, engaging and pleasurable for the apprentices, are an important solution of these problems. Within this context, constructionism appears as a valuable source of theoretical subsidies for the elaboration of educational games, with research demonstrating that the learning process has the potential to occur more naturally and satisfactorily when the learner is engaged in the construction of an artifact . Among the gaming genres, the Role Playing Games nowadays stand out, with great appeal among the new generations of students. Given this educational potential, and as a contribution to the solution of the above mentioned problems, in this work is exposed the authorship of a tool capable of generating constructor games within the RPG genre, without the need of knowledge of programming language by the author.

Keywords: Education, constructionism, games, Role Playing Games, engines

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADL	Architecture Description Language
API	Application Programming Interface
CSS	Cascading Style Sheets
FPS	First Person Shooter
JSON	JavaScript Object Notation
HP	Health Points
HTML	Hypertext Markup Language
MMORPG	Massively Multiplayer Online Role Playing Game
MP	Mana Points
NPC	Non-Player Character
PBL	Problem Based Learning
RPG	Role Playing Game
RTS	Real Time Strategy
SASS	Syntactically Awesome Stylesheets
SPL	Software Product Lines
XP	Experience Points

Sumário

1 INTRODUÇÃO

1.1 Motivação

De acordo com (HAMARI et al., 2016), a evasão escolar constitui um problema global, com 20% a 25% do alunado de vinte e oito países pertencentes à OECD(Organização para a Cooperação e Desenvolvimento Econômico) sendo classificados no grupo de baixo engajamento nas atividades acadêmicas.

Como expõe (MORA, 2011), o tédio pode ser um fator influenciador no comportamento do alunado, levando seus membros à tendência de se tornarem dispersivos, evitar assistir aulas, e, em última instância, abandonar a escola; dessa forma, muitos alunos se sentem academicamente desengajados, rejeitando, assim, experiências por eles consideradas aborrecidas e não significativas, como as atividades em sala de aula.

Tal problema também se verifica no Brasil. Segundo (MEC, 2016), a taxa de abandono do ensino fundamental, no ano de 2011, foi de 2,8%; o número cresce substancialmente no contexto do ensino médio, para 9,5%. As taxas de reprovação, segundo a mesma fonte, chegaram, no citado ano, a 9,6% para o ensino fundamental e 13,1% para o ensino médio.

Como descrito por (NERI et al., 2015), os motivos para a evasão escolar incluem: dificuldade de acesso à escola (10,9%); necessidade de trabalhar, a fim de gerar renda (27,1%); falta de interesse (40,3%), tendo os restantes entrevistados respondido com outras causas.

Depreende-se dos números apresentados motivos para preocupação com os níveis de engajamento, assim como do desempenho apresentados pelo corpo discente do Brasil. As causas para tais níveis, segundo os mesmos números, apontam para a falta de interesse na forma como o conteúdo acadêmico é apresentado.

São descritos por (MORA, 2011) os resultados de uma pesquisa realizada na *Romero Elementary and Middle School*, uma escola bilíngue localizada na área metropolitana de uma cidade do nordeste norte-americano, acerca de fatores que influenciavam o grau de motivação de seus alunos nas atividades acadêmicas. Concluiu-se que o desinteresse do alunado advinha da desconexão entre o ato de aprender e as atividades empreendidas na escola. Ao indagar por que eles tinham de aprender determinado assunto, a eles era respondido por seus professores que aquela matéria seria parte da prova, o que se mostraria insuficiente para motivá-los a estudar aquele assunto em particular; nesse contexto, aulas de ciências e matemáticas eram particularmente rejeitadas. Ao contrário, eles se mostraram um interesse renovado ao se engajar com atividades mais interativas, como conduzir experimentos ou construir artefatos; tais atividades eram classificadas por eles como “não entediantes”, por dar a eles oportunidades de se expressar e compartilhar, o que os fazia se voluntariar entusiasticamente quando eram requeridos alunos para esses

eventos.

Jogos se configuram como um relevante componente dessa proposta, como afirma (BARZILAI; BLAU, 2014), ao defender que estes oferecem experiências de aprendizado, ao mesmo tempo, motivantes, engajadoras e prazerosas para os aprendizes; essas características advêm de fatores psicológicos, afetivos e cognitivos. Corroborando esta afirmação, segundo (HWANG; CHIU; CHEN, 2015), estudos indicam que o aprendizado baseado em jogos constitui uma abordagem efetiva para a construção do conhecimento, promovendo a motivação e habilidades de resolução de problemas. Entretanto, é necessário desenvolver jogos cujos objetivos se apliquem ao contexto do assunto a ser estudado, ao invés de meramente servirem como interfaces para a apresentação de material relativo ao conteúdo acadêmico em estudo.

Tais afirmações são ainda reforçadas por (MCGONIGAL, 2011), ao afirmar que os jogos educacionais constituem uma importante e crescente indústria, e vêm sendo desenvolvidos para ajudar a ensinar basicamente qualquer tópico imaginável, desde história até matemática, ciências e línguas estrangeiras. Quando aliados a um bom design e um significativo conteúdo educacional, é fornecido um poderoso alento a alunos desmotivados.

Neste contexto, o construcionismo aparece como uma fonte de subsídios teóricos para a elaboração de jogos educacionais. É afirmado por (HAY; BARAB, 2001) que Seymour Papert (criador desta corrente filosófica educacional) defende que não apenas o conhecimento deve ser construído pelo aprendiz, mas que o processo de aprendizado apresenta o potencial de ocorrer de forma mais natural e satisfatória quando este está engajado na construção de um artefato; portanto, o construcionismo permite ao alunado desenvolver sua própria interpretação de sua interação com o mundo, permitindo ainda que tal artefato trabalhe como um instrumento de partilha de suas descobertas.

Soluções que unam os benefícios de jogos e construcionismo são defendidas por (KAFAI, 2006), ao afimar que ao invés de incluir aulas previamente elaboradas em jogos, como propõe a abordagem instrucionista, jogos construcionistas fornecem ao aprendiz a possibilidade de, além de jogar, construir seus próprios jogos, como uma forma de compartilhar o conhecimento adquirido. Jogos que apresentem ao jogador o desafio de construir um artefato, dentro da premissa básica do construcionismo, portanto, constituem dispositivos com o potencial de, em primeira instância, mitigar os efeitos causados pela falta de interesse por parte do alunado, como descrito anteriormente nesta sessão; a prazos mais dilatados, as evidências apontam que conforme advenha o avanço dos recursos tecnológicos disponíveis, aliado ao refinamento das técnicas de produção de jogos construcionistas, estes contribuirão de forma significativa para o equacionamento dos problemas supracitados.

Como descrito por (MACHADO; SANTOS; DIAS, 2017), *Role Playing Games* (RPGs) constituem um gênero de jogos onde um jogador personifica um personagem, sozinho ou em grupos, conhecidos como parties, recebendo a tarefa de vencer uma série de desafios, com o propósito de atingir um objetivo, na forma da conclusão de uma história. À

medida em que o jogador vai avançando no sentido de concluir sua meta, vai evoluindo seu personagem, em termos de atributos, como força, inteligência e destreza, capacitando-o a enfrentar obstáculos menos acessíveis no decorrer do jogo. O ambiente onde os jogos se desenrolam caracterizam-se como micromundos, que incluem a interação com personagens controlados pelo mestre do jogo (a máquina, em caso de jogos eletrônicos), conhecidos como *nonplayable characters* (NPCs).
Portanto, RPGs, como exposto, surgem como provedores da possibilidade da construção de artefatos, como preconizado pela definição de construcionismo, tanto do ponto de vista do criador do jogo, que tem diante de si o objetivo de construir um micromundo, com missões e personagens, que podem constituir obstáculos ou auxílios ao jogador; quanto do ponto de vista do jogador, a quem cabe criar e evoluir um personagem, com o qual escreverá uma história dentro do ambiente do jogo.
Some-se aos desafios e ao potencial apresentado por jogos educacionais do gênero RPG para equacioná-los, o crescente interesse mercadológico por jogos, em particular por RPGs, como *Priston Tale, Ragnarok Online* e *World of Warcraft*; bem como o consequente potencial de se tornarem parte da solução para os desafios expostos neste item, para elucidar a motivação para a escolha do tema proposto.

1.2 Problema

Como uma ferramenta de criação de jogos RPG poderia ajudar a solucionar os problemas causantes da evasão escolar verificada globalmente?

1.3 Hipótese

Uma ferramenta pode ser útil caso consiga unir os conceitos de jogos RPG e construcionismo, a fim de capitalizar o interesse do alunado pelo primeiro, mantendo, assim, seu interesse; e o potencial didático do segundo, a fim de garantir um desempenho acadêmico satisfatório por parte dos aprendizes. Tal ferramenta, a fim de permitir seu uso por membros dos corpos discente e docente sem conhecimentos de linguagem de programação, deve ainda permitir a construção de jogos a partir de modelos gráficos.

1.4 Objetivo geral

Projetar e implementar a engine *LearnCraft*, direcionada à agilização na criação de jogos RPG de cunho construcionista.

1.5 Objetivos específicos

- Realizar um levantamento teórico e prático acerca dos assuntos contidos no trabalho proposto
- Apresentar o construcionismo, com o seu histórico, sua aplicabilidade em sala de aula e a implementação de suas premissas em jogos
- Apresentar o conceito de *engine*, com exemplos
- Elucidar no que consistem *Role Playing Games*, em suas diferentes formas, com conceitos básicos e exemplos
- Compartilhar das técnicas de programação utilizadas para a elaboração da *engine*
- Demonstrar a usabilidade da engine implementada com um jogo com ela realizado, a ser usado em sala de aula

1.6 Metodologia da pesquisa

As informações acerca dos conceitos básicos, que incluem engines, construcionismo e *Role Playing Games*; bem como dados estatísticos, constantes nesta monografia, vieram de artigos, livros e sites de revistas científicas.
Como referencial para o conceito de jogos construcionistas, foram eleitos, estudados e descritos no capítulo correspondente, *games* com as características defendidas por essa abordagem de ensino, como *Civilization, SimCity, Age of Empires* e *Minecraft*.
Para a eleição dos aspectos componentes da engine em estudo, tais como classes de personagens, obstáculos e mapas, foi feito um levantamento de materiais relativos a RPGs de relevância no mercado, como *Ultima, Final Fantasy, Ragnarok Online* e *World of Warcraft*, jogos esses que foram aqui estudados acerca de sua jogabilidade e enredo.
Para a realização do estudo de caso da ferramenta em foco, que consiste da elaboração de um jogo utilizando a engine LearnCraft, se propondo, assim, a validar sua usabilidade, foram escolhidos materiais relativos a disciplinas de ensino superior, contidas em sites *web*. Os materiais produzidos pelos jogadores, a partir de sua experiência com o referido jogo, foram submetidos a um profissional da área, para apreciação e comparação de resultados. A atividade foi realizada na Faculdade da Cidade de Maceió, e envolveu a participação de dez alunos, todos graduandos do curso de Ciência da Computação da referida instituição, na faixa etária entre os 20 e os 25 anos, sendo 9 representantes do sexo masculino e 1 representante do sexo feminino.
Para a implementação central ao tema do trabalho, foram escolhidas as linguagens JavaScript e TypeScript, com a biblioteca fornecida pelo *framework* Phaser.IO, para a parte cliente; o banco de dados MongoDB, que apresenta a possibilidade de armazenagem de

dados em JSON (JavaScript Object Notation), formato texto e independente de linguagem, para a persistência de dados; e a plataforma Node.js para as atribuições do servidor.

1.7 Estrutura da dissertação

A seguir, serão apresentados, no capítulo 2, aspectos concernentes à fundamentação teórica para a construção deste trabalho, dividida em quatro sessões, enfocando respectivamente: Construcionismo, *Engines*, RPG e reuso de software.
O capítulo 3 fará uma análise dos trabalhos correlatos a este, com comparações entre características como técnica de reuso utilizada, e presença ou ausência de construção de *engine*, RPG e proposta construcionista.
O capítulo 4 trará a implementação central a este trabalho, com explanações acerca das ferramentas práticas e teóricas utilizadas em sua realização, bem como *screenshots*, diagramas e trechos de código.
No capítulo 5 será exposto o estudo de caso realizado em sala de aula. O capítulo 6 concluirá o trabalho, com considerações finais, potenciais contribuições para os problemas expostos e trabalhos futuros a acrescentar potencialidades à ferramenta neste trabalho descrita.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão descritos os elementos da fundamentação teórica do presente trabalho, na forma das características inerentes ao construcionismo, bem como seu histórico, e as apresentações de jogos representativos deste paradigma educacional; o conceito e representações de softwares classificados como engines; a definição, história e exemplificação mercadológica de Role Playing Games; e, finalmente, serão apresentadas técnicas de reuso de software,.

2.1 Construcionismo

Esta seção descreve as origens do construcionismo, as bases filosóficas que regem seus preceitos, e um dos produtos de suas premissas abordado, aqui representados na forma de jogos construcionistas.

2.1.1 Origens

O paradigma behaviorista, concebido pelo psicólogo norte-americano John Broadus Watson, coloca o aprendiz como um receptáculo de informações e reagente a estímulos recebidos, como descrito por (ERTMER; NEWBY, 1993) e corroborado por (JONES; BRADER-ARAJE, 2002). Tal paradigma defende a ideia de que os professores devem fornecer estímulos, a fim de que, os alunos possam não somente aprender, como ter seu conhecimento mensurado por meio de seu comportamento; se o aluno não aprende, torna-se necessário reforçar o comportamento tido como correto, ou desestimular comportamentos indesejáveis.

O psicólogo norte-americano Burrhus Frederic Skinner evoluiu os conceitos acima mencionados, naquilo que seria denominado behaviorismo radical, como descreve (RUTHERFORD, 2000), ao afirmar que a transformação do indivíduo ocorre em virtude de ambientes passados e presentes, internos a ele, excluindo desse processo entidades intangíveis à sua experiência pessoal. Em oposição às ideias supracitadas, surge o construtivismo, criado pelo psicólogo suíço Jean Piaget. Este defende a ideia de que o aluno deve construir sua própria base de conhecimento; como é afirmado por (BODNER, 1986), o construtivismo advoga que aprendizes têm o potencial de construir seu próprio conhecimento; eles vão além de repetir aquilo que lhes foi dito, ou que eles leem; alunos procuram por significado, e buscarão ordenar os eventos e informações que recebem, segundo sua concepção de mundo.

Tais afirmações são reforçadas por (HEIN, 1991), ao afirmar que o construtivismo parte da premissa básica de que o aluno não somente apresenta o potencial de construir sua

própria base de conhecimento, como deve fazê-lo; individualmente ou em grupos, ele deve construir significado, com base na afirmação que não há conhecimento independente do significado, construído por um indivíduo ou uma comunidade.
Ainda de acordo com (HEIN, 1991), o construtivismo pode ser definido pelas seguintes premissas:

- Aprendizado é um processo ativo; o aprendiz usa sua percepção das informações adquiridas como matéria-prima para a elaboração de seu conhecimento.
- Ao aprender, pessoas aprendem a aprender; por exemplo, ao aprender sobre a cronologia de uma série de eventos específica, exercitamos nosso conhecimento sobre o próprio conceito de cronologia.
- A ação de construir conhecimento acontece na mente do aluno; ações físicas podem ser necessárias para muitas atividades de conhecimento, como, por exemplo, marcenaria; entretanto, o senso crítico é fundamental para um aprendizado adequado.
- Aprendizado envolve linguagem; os nomes que damos às coisas são parte indissociável de seu significado particular.
- Aprendizado é uma atividade social, estando intimamente ligado às conexões do aluno com seus colegas, professores e alunos.
- Aprendizado é contextual; não aprendemos coisas independentes de seu contexto; por exemplo, não faz sentido estudar subtração e nunca usá-la para situações diárias, como o controle de gastos.
- Conhecimento é necessário para o aprendizado; qualquer esforço de ensino deve levar em conta os conhecimentos prévios do aprendiz.
- Aprender requer tempo de assimilação; esta é produto de um período de exposição e reflexão em relação a novos conceitos.
- Motivação é um aspecto chave para o aprendizado; a menos que se saiba porque estudar algum assunto, existe a possibilidade do não-envolvimento no processo de assimilação de ideias.

O pensamento construtivista, entretanto, como é afirmado por (ERTMER; NEWBY, 2013), não nega a existência de um mundo real; porém, defende a ideia de que o que sabemos deste vem das interpretações que fazemos de nossas experiências individuais, não havendo, portanto um significado correto das coisas; aprendizes, por conseguinte, não transferem conhecimento para suas memórias, mas o constroem a partir de suas interações, sendo portanto, criado e não adquirido.

2.1.2 Definição

O matemático e professor Seymour Papert, nascido na África do Sul e radicado nos Estados Unidos da América, em (PAPERT, 1986) apresentou o termo construcionismo a partir das teorias construtivistas supracitadas; tais noções foram extendidas com a ideia de que o aprendizado é mais eficiente quando parte de uma atividade em que a construção de um artefato é oportunizada ao aluno. Ele defende ainda que o construcionismo une influências pessoais e sociais no aprendizado, pois o artefato produzido é produto da interação entre a construção de conhecimento pessoal e social, significativa e pública.
Papert afirma ainda, em (PAPERT; HAREL, 1991), que o construcionismo compartilha, do construtivismo, o ponto de vista do aprendizado como "'construir estruturas de conhecimento"', por meio da progressiva internalização das ações. Ele, então, agrega a ideia de que tal processo ocorre de maneira particularmente efetiva, num contexto onde o aprendiz é conscientemente engajado a construir um artefato, seja este um brinquedo, uma monografia, ou um jogo, como é proposto neste trabalho.
As noções de Papert acerca do construcionismo têm sido aplicadas a diversos tipos de ambientes de estudo. No contexto do *Epistemology and Learning Group*, no *Massachusetts Institute of Technology*, foram implementados projetos envolvendo software educativo, que incluem jogos educativos; sistemas complexos; e implementações científicas. Fora do grupo, há projetos para documentos multimídia, sistemas especialistas, modelos computacionais, além de mundos virtuais. O modelo construcionista, nos citados projetos, resignifica a tecnologia como meio para a exploração intelectual. Por exemplo, a realidade virtual tem sido usada para pesquisas relacionadas ao meio ambiente e à medicina, com alunos usando dispositivos de realidade virtual para navegar em mundos criados em sala de aula.

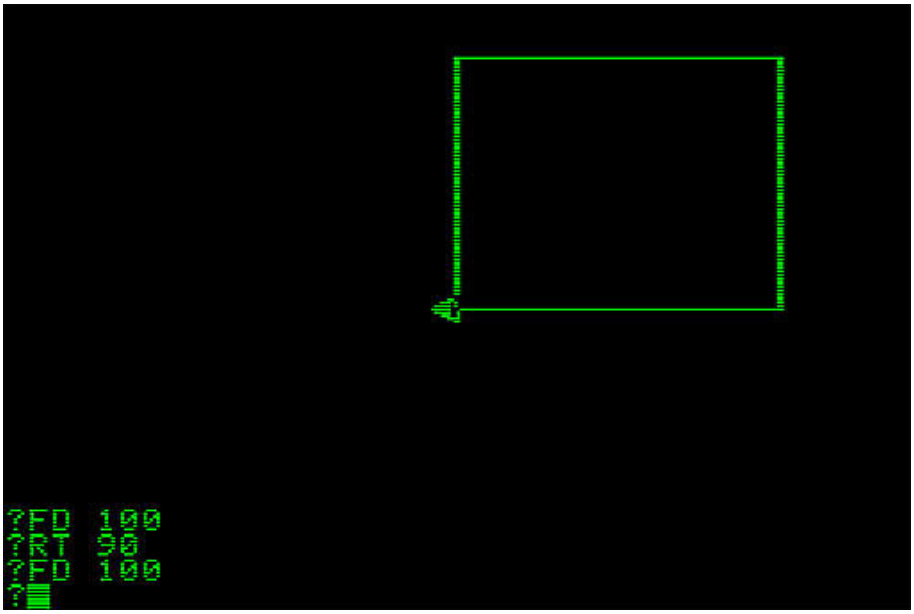
2.1.3 Jogos construcionistas

A elaboração de *video games* construcionistas, como preconizado por esta corrente filosófica de ensino e aprendizado, tem como premissa básica, engajar jogadores na implementação de artefatos pessoalmente significativos. Tal processo de construção facilita o desenvolvimento de estruturas cognitivas internas, além de trazer o aprendiz a uma relação mais estreita com ideias, fenômenos e sistemas previamente explorados. A crescente literatura acerca do uso de videogames os caracteriza como contextos para aprendizado, incluindo seu potencial como ambientes motivacionais, seu alinhamento com identidade e normas juvenis, e seu potencial de permitir um engajamento renovado e interativo com conceitos vistos em sala de aula (HOLBERT et al., 2014).
Em adição ao papel do aprendiz como jogador de *games* construcionistas, este pode aparecer como criador, como é descrito por (KAFAI, 2006), ao afirmar que, no lugar de incluir material de aula diretamente nos jogos, os construcionistas almejam fornecer ao alunado a

possibilidade de construir seus próprios jogos, possibilitando, assim, novas relações com o conhecimento confrontado; estudos realizados indicam significativo engajamento entre os aprendizes no processo de criar jogos. O mesmo autor, em (KAFAI; BURKE, 2015), afirma que Papert via o engajamento com a linguagem de programação Logo (figura 1), como uma forma de agilizar a construção de estruturas de conhecimento; programar nesta linguagem permitiria o elo entre o artefato do mundo físico (neste caso, a tartaruga, usada como ferramenta para a construção de figuras) com concepções mentais, representadas por regras e objetos; Papert afirmava que objetificações como a citada tartaruga, facilitam a identificação pessoal do aprendiz com o objeto, ajudando, assim, a construir examinar e revisar conexões entre o velho e o novo conhecimento.

Assim explicitado, uma ferramenta para elaboração de jogos construcionistas apresenta o

Figura 1 – Logo



Fonte: Theantiroom (2011)

potencial de servir como uma via em mão dupla, que engaje, no processo de construção, criador e jogador, num ciclo que leva à implementação de um artefato que traz, por sua vez, um incremento sensível na base de conhecimentos de todos os envolvidos, à medida que o jogador avance em seu objetivo. Tal processo é facilitado ainda pela identificação do alunado com jogos, que serão exemplificados nas seguintes sessões.

2.1.3.1 Civilization

A série Civilization (FIRAXIS, 2016) (figura 2), iniciada com Civilization em 1991, e tendo como mais recente capítulo Civilization VI, em 2016, apresenta ao jogador o desafio de construir uma civilização à sua escolha, a evoluindo da idade da pedra até a era espacial. Os jogos da série são baseados em turno (como os criados com o auxílio da

ferramenta apresentada no presente trabalho), ou seja, como num jogo de xadrez, onde os jogadores tentam alcançar um dos objetivos propostos pelo jogo - contra adversários humanos ou controlados por máquina - por meio das ações 4X - "eXplore, eXpand, eXploit, eXterminate"(explorar, no sentido de desbravar novas regiões; expandir; explorar, no sentido de usufruir de recursos; erradicar). A cada turno, os jogadores devem decidir que avanços tecnológicos, culturais e cívicos deveriam adotar; tomar medidas diplomáticas em relação aos seus oponentes; configurar o que será produzido em cada uma de suas cidades; e mover suas unidades, tais como colonos e unidades militares.

Os jogos se desenrolam em mapas pré-definidos, tais como o mapa da terra, ou algum continente específico; ou algum gerado pela máquina previamente ao início do jogo, com terrenos, incluindo mares, montanhas, florestas e pradaria. As dimensões da matriz do mapa também pode ser configurada no momento de criação do jogo (esta característica encontra-se presente nos jogos criados com o LearnCraft).

Os jogadores podem escolher uma entre diversas civilizações a seu dispor, incluindo os romanos, gregos, astecas e japoneses, por exemplo, cada uma com seus pontos fortes. Os jogadores iniciam sua partida em algum ponto aleatório do mapa, a partir do qual, deverão escolher um local para a construção de sua primeira cidade, levando em conta variáveis como recursos próximos, que incluem águas abundantes em peixes, ou montanhas ricas em minerais.

As unidades de cada jogador se movem, a cada turno, um determinado número de casas, que variará, por sua vez, da natureza da unidade (um explorador por exemplo, pode andar até duas casas por turno) e do terreno sobre o qual se move (uma montanha por exemplo, tem uma penalização que não ocorrerá numa pradaria). Haverá combate quando duas unidades de civilizações rivais entre si buscam ocupar o mesmo espaço de terreno.

De acordo com (SQIRE, 2005), para obter sucesso em Civilization, os jogadores devem compreender conceitos como anarquia, despotismo e democracia; o significado estratégico de cenários como vales e florestas; e pesquisar acerca de questões como a existência de petróleo em dados terrenos e quais os efeitos do comunismo.

O jogo fornece, portanto, uma valiosa contribuição ao aprendizado de diversas disciplinas, como geografia, economia e ciências sociais, ao engajar o jogador no processo de aprendizado dessas áreas, a fim de cumprir com seus objetivos, sejam estes dominar os adversários ou chegar primeiro ao espaço.

2.1.3.2 SimCity

Na série SimCity (MAXIS, 2015) (figura 3), classificada como um jogo RTS (*Real Time Strategy*), ou seja, um estratégico em tempo real, foi iniciada com SimCity em 1989, e tem como mais recente versão SimCity 4, de 2003; o jogador tem como missão fundar uma cidade a partir de um terreno vazio, onde morarão seus Sims (habitantes) em zonas de

Figura 2 – Civilization V

Fonte: TheGamer (2012)

desenvolvimento, classificadas em:

- Zonas residenciais, onde os Sims morarão em casas e prédios de apartamentos;
- Zonas comerciais, onde os Sims consumirão e trabalharão em escritórios;
- Zonas industriais, que fornecem oportunidades de emprego aos Sims, em fábricas.

Figura 3 – SimCity 4

Fonte: ElectronicArts (2012)

À medida que a cidade progride, o jogador deverá disponibilizar para seus Sims serviços como saúde, educação, segurança e lugares de lazer. Fornecimentos de eletricidade, água

e gestão de lixo também serão vitais. Para manter os serviços funcionando, deverão ser arrecadados impostos, que virão dos Sims, bem como de bases militares e prisões federais, que poderão ser autorizadas pelo jogador. A partir de SimCity3000, o jogador poderá fazer acordos com as cidades vizinhas, para resolução de problemas como queima de lixo. Como afirma (MINNERY; SEARLE, 2014), SimCity fornece um ambiente urbano, onde alunos experimentam, fazendo uso de seus conhecimentos; ao refletir sobre suas cidades, eles aplicam conceitos como orçamento, urbanização, ecologia e disponibilização de serviços, como fazem, por exemplo, ao escolher fontes energéticas, quando devem levar em consideração aspectos como custos, quantidade de energia gerada e impacto no meio ambiente.
Assim posto, fica latente a contribuição de SimCity para a formação de cidadãos conscientes dos aspectos que devem ser levados em consideração na construção de uma coletividade humana, assim como as ações que podem ser tomadas para se atingir o bem estar de seus habitantes, o que constitui, em última análise, material de significativa contribuição no tocante à matriz extra-curricular de um alunado que se pretende a inserção na cultura global.

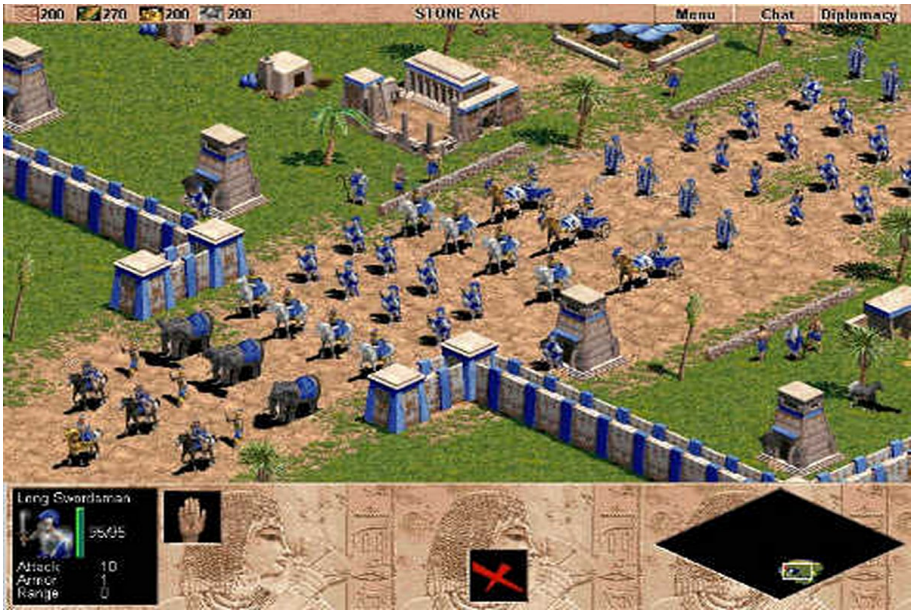
2.1.3.3 Age of Empires

A série Age of Empires (ENSEMBLE, 2013) (figura 4) foi desenvolvida pela Ensemble Studios, tendo sido iniciada em 1997, com o mais recente lançamento tendo sido Age of Empires II: Rise of the Rajas, de 2016. Assim como SimCity, Age of Empires também é classificado como RTS. Nesta série, o jogador tem diante de si o objetivo de construir uma estrutura militar para enfrentar seus inimigos, gerenciando três recursos básicos:

- Alimentos, a princípio obtidos por extração, posteriormente evoluindo para a agropecuária;
- Unidades militares, como arqueiros e catapultas, que evoluem com o desenvolvimento tecnológico;
- Instalações, como estábulos, docas, castelos e fazendas, gradualmente disponíveis.

A série, a princípio, tinha como foco eventos passados na Europa, da idade da pedra à idade do ferro; em Age of Empires II: The Age of Kings, a ação se desenrolava na idade média, indo até a conquista espanhola do México. A partir de Age of Empires III, foi explorada a idade moderna, com a colonização européia das Américas.
É afirmado por (ARRUDA, 2011) que o jogador de Age of Empires tem a possibilidade de reescrever a história, tendo que, para isso, aprender a narrativa histórica dos fatos, além de desenvolver formas de interação social com outros jogadores, como alianças e acordos, a fim de chegar ao objetivo proposto.

Figura 4 – Age of Empires



Fonte: PhoneArena (2002)

Assim colocado, fica patente o potencial da série em ensinar de forma dinâmica ao jogador, fatos históricos, com os quais este terá particular afeição ao se deparar com a possibilidade de alterá-los (por exemplo, ao manter os mouros com o domínio da Península Ibérica), conforme sua habilidade e capacidade de tomada de decisão.

2.1.3.4 Minecraft

Figura 5 – Mineraft

Fonte: Mzstatic (2016)

Minecraft (MOJANG, 2013) (figura 5) é um jogo do tipo sandbox (ou seja, não tem objetivos fixos para o jogador), desenvolvido pela empresa sueca Mojang. A ideia é permitir ao jogador construir artefatos a partir de cubos em 3D, num mundo, a princípio, gerado pela máquina. Outras atividades incluem exploração, coleta de recursos e combate, nos modos de jogo survival, que envolve a construção com busca de recursos; creative, onde os jogadores têm acesso a recursos infinitos; ou adventure onde se pode jogar em mapas criados pelo próprio jogador ou outros da comunidade Minecraft.
De acordo com (THORSTEINSSON; NICULESCU, 2016), Minecraft tem se mostrado popular nos níveis fundamental, médio e superior, ao se adequar ao ensino de várias disciplinas, tais como ciência, ao oferecer elementos de eletrônica e eletricidade; bem como se mostra uma ferramenta auxiliar de ensino de arte computacional, storytelling digital e fundamentos de video games, particularmente no ambiente da educação à distância, por suas capacidades multiplayer, que permite a alunos de diferentes localidades trocarem experiências que trarão sensível incremento em seus níveis de conhecimento, assim como em suas capacidades cognitivas.
Por suas características de jogo voltado à construção de artefatos sem objetivos previamente estabelecidos, Minecraft se mostra uma ferramenta de abrangentes possibilidades no tocante à multidisciplinaridade de sua aplicação em sala de aula. O jogo em questão, assim exposto, representa de forma arquetípica o paradigma construcionista, conforme proposto por Seymour Papert.

2.2 Engines

Esta seção descreve o conceito de engine, os fatores que motivaram seu advento, e exemplos desta classe de software, para melhor compreensão acerca de sua relevância e potencial de auxílio à criação de games.

2.2.1 Motivação

De acordo com (DALMAU, 2004), a história dos vídeo games teve início em 1961, quando um estudante do MIT (*Massachusetts Institute of Technology*) criou, num Digital PDP-1 minicomputer, Spacewar!, o primeiro videojogo, que confrontava duas naves duelando entre si; posteriormente, o jogo foi apresentado ao empresário norte-americano Nolan Bushnell, então estudante de engenharia elétrica na Universidade de Utah. Sob sua influência, ele criou *Computer Space*, que podia ser jogado numa máquina ligada a uma TV. Em 1972, ele fundaria a Atari, que viria a popularizar o conceito de videogame.
Como descreve (BATES, 2004), o desenvolvimento de um jogo na citada época até o advento de *game engines*, exigia que o programador se encarregasse dos seguintes aspectos:

- Física
  Gerencia os aspectos físicos do jogo, tais como os efeitos da gravidade, ou resistência da água;

- Gráficos
  Originalmente, envolvia o trabalho artístico de desenhar personagens dos jogos, bem como cenários, pixel por pixel. Mais recentemente, adveio o papel de programador 3D, com domínio em conceitos relativos à álgebra linear, como vetores e matrizes;

- Inteligência Artificial
  Frequentemente conhecida como AI (Artificial Intelligence), envolve a lógica para simular inteligência em inimigos e oponentes, para implementar seus comportamentos;

- Áudio
  A maior parte dos jogos faz uso de sons, e muitos têm uma trilha sonora;

- Script
  É o roteiro do jogo, que motiva o jogador a seguir jogando, para acompanhar seu desenrolar;

- User Interface
  Comumente conhecida como UI, é a *interface* do jogador com o jogo;

- Input
  Gerencia a entrada de ações do jogador na interação com o jogo, utilizando dispositivos como teclado, joystick e mouse;

- Rede
  Em jogos multijogador, é a parte responsável pela comunicação entre máquinas, sobre redes locais ou internet.

Como é possível depreender destas informações, o papel de programador já envolveu, necessariamente, o conhecimento sobre uma série de disciplinas, que, com o avanço da complexidade dos jogos passou a exigir, para seu desenvolvimento, equipes progressivamente mais numerosas; para o gerenciamento dessas equipes, assim como viabilizar o desenvolvimento de jogos por desenvolvedores independentes, fazia-se necessária uma tecnologia que trouxesse consigo uma mudança de paradigma.

2.2.2 Definição

Engines de *video games*, como será exposto na presente sessão, são definidos como programas de software voltados à criação de jogos eletrônicos, facilitando e agilizando sua produção, por meio da separação de atribuições, antes delegadas ao programador, que

agora podia manter seu foco criativo em aspectos relevantes à concepção do jogo em si.
Como afirma (EL-NASR; SMITH, 2006), no final da década de 1990, desenvolvedores de jogos começaram a separar componentes de jogos, tais como regras, comportamentos e personagens, da codificação central destes, permitindo, assim, criar novos comportamentos e gráficos, ao conectar a estes arquiteturas reusáveis, que manipulam renderização de polígonos, controle de câmera, iluminação, física, sons etc.
De tal separação, adveio o conceito de *engine*, que permitia ao criador de jogos manter o foco em aspectos relacionados à jogabilidade, deixando para o software tarefas antes atribuídas ao programador, como exposto na sessão anterior.
Ainda segundo o mesmo texto, várias dessas *engines* oferecem linguagens de *script*, que permitem ao usuário criar novos mundos, ou modificar jogos previamente existentes, a fim de obter experiências significantemente renovadas, num processo conhecido como *modding*. Assim o *modding* aparece como uma alternativa à criação de jogos desde seu início, ao tirar proveito da estrutura básica de games pré-existentes e acrescentar a estes personagens, mapas e comportamentos que justificarão, assim, classificar esses mods como jogos totalmente novos.
Aspectos positivos da reusabilidade de código são reforçados por (LEWIS MICHAEL E JACOBSON, 2002), ao afirmar que os custos do desenvolvimento de simulações cada vez mais realistas, como as que têm sido demandadas pelo mercado, tem crescido de forma tal, que desenvolvedores não se permitem destinar todos os seus recursos a um só jogo. Tal tendência trouxe o advento de *engines* de jogos - aplicações modulares escritas para um jogo específico, mas abrangentes o suficiente para ser usadas para a criação de uma família de jogos similares.
Portanto, a agilização oferecida por uma *engine* de jogos, aliada à democratização provida por suas características, que podem inclusive dispensar o conhecimento de linguagens de programação, dispensando, assim, o criador dos jogos de apresentar capacidades avançadas de programação, permitindo dessa forma, que professores de várias áreas usufruam de suas potencialidades, caracteriza *engines* gráficas como ferramentas de significativa importância no desenvolvimento de soluções personalizadas às necessidades de cada organização ou disciplina. Tais características, unidas ao potencial didático oferecido por jogos construcionistas, abre caminhos promissores no campo do processo de aprendizado.

2.2.3 Engines de jogos

Nesta sessão, serão examinadas game engines populares no mercado, com suas características e funcionalidades oferecidas.

2.2.3.1 Unreal Engine

Escrita e programável em C++, Unreal Engine (UNREAL, 2017) (figura 6) é uma engine de jogos desenvolvida pela empresa norteamericana Epic Games, com sua primeira aparição no jogo FPS (*First Person Shooter*), ou seja um jogo de tiro do ponto de vista do personagem controlado pelo jogador, Unreal, de 1998. Desde então, embora a princípio idealizada para o citado gênero de jogos, ela tem sido utilizada para outros estilos, inclusive RPG. A mais recente versão, é a 4, lançada em 2012, incluindo como plataformas alvo de suas implementações Windows, MacOS, Linux, HTML (Hypertext Markup Language), iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, além de dispositivos de VR (Realidade Virtual), para os quais foram elaboradas populares séries, como Unreal Tournament, Borderlands, Gears of War e Mass Effect.
. Uma experiência acadêmica, construída com o auxílio do Unreal Engine, o Tactical

Figura 6 – Unreal Engine

Fonte: Unreal-Brasil (2016)

Language and Culture Training System (TLCTS), é apresentada por (JOHNSON; WANG; WU, 2007), onde este é descrito como uma plataforma interativa de aprendizado voltada ao desenvolvimento, por parte do alunado, da habilidade de comunicação em línguas estrangeiras. A utilização da engine permite integrar um sistema de tutoria inteligente a tecnologias de jogos. As lições fazem extensivo uso de reconhecimento de fala, fornecendo assim, *feedback* de desempenho aos alunos. Os aprendizes aplicam suas habilidades em jogos interativos, que requerem conhecimento da língua proposta pelo jogo, para ter sucesso em seus objetivos. Assim, as probabilidades de programação 3D desta ferramenta, se configuram como potencialmente auxiliares na agilização do desenvolvimento de games educativos. A obrigatoriedade de uso da programação em linguagem C++, dada sua relativa

complexidade, entretanto, aparece como um fator restritivo nesse processo, excluindo educadores sem conhecimentos de linguagens de programação.

2.2.3.2 Unity

Unity (UNITY, 2017) (figura 7) é uma engine programável em C#, desenvolvida pela empresa norteamericana, de origem dinamarquesa, Unity Technologies.
É direcionada ao desenvolvimento de aplicações tanto 2D quanto 3D, compilando código para as bibliotecas Direct3D, OpenGL Open GL ES e WebGL. São suportadas as plataformas Android, Android TV, Nintendo Wii U (sendo a SDK padrão), Nintendo Switch, Nintendo 3DS, MacOS, iOS, Playstation Vita, Windows, Windows Phone, Xbox One, além das tecnologias de VR, como Oculus Rift, Playstation VR e Microsoft HoloLens. Ela é representada por jogos de significativa relevância no mercado, como Battlestar Galactica Online, Heartstone: Heroes of Warcraft, Angry Birds 2, bem como a série Assassin's Creed.
É descrito em (INDRAPRASTHA; SHINOZAKI, 2009) um estudo, levado a cabo com

Figura 7 – Unity Engine

Fonte: WindowsCentral (2012)

a *engine* Unity, cujo objetivo foi identificar as vantagens e limitações do uso de *game engines* para a implementação de um aplicativo voltado à representação visual de design urbano, sendo esse trabalho realizado no distrito central de Yaesu, em Tokyo, Japão. Foi considerado que o uso de uma *engine* acrescentaria aos aspectos visuais, a interação do usuário com o ambiente. Como conclusão do trabalho, apesar da constatação de algumas limitações, advindas do fato de a *engine* não ter embarcada uma ferramenta CAD (*Computer Aided Design*), suas características de programação orientada a objetos, unidas a suas potencialidades de renderização de gráficos 3D, a caracterizam como uma ferramenta de grandes possibilidades de construção de aplicações para simulação de ambientes.

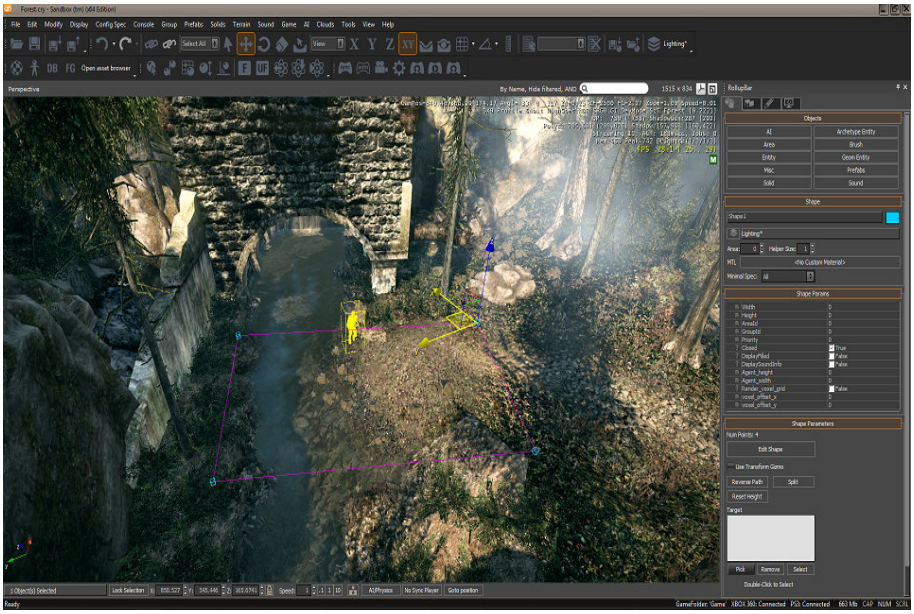
Desta forma, fica patente que o uso desta ferramenta em particular, e de engines 3D em geral, mostra importante versatilidade nas áreas de estudo, aqui ilustrada num trabalho de arquitetura. A automatização de aspectos ligados à física, unidos às potencialidades gráficas e de interação do usuário parecem fornecer consideráveis subsídios a disciplinas como química, física e biologia, pelas possibilidades de construir modelos que simulem de forma efetiva conceitos como ligações químicas e comportamentos de objetos.

2.2.3.3 CryEngine

CryEngine (CRYENGINE, 2017) (figura 8), desenvolvida pela empresa alemã Crytek, é desenvolvida em Lua, C++ e C#, apresentando, embarcadas em sua instalação básica, ferramentas como editor de materiais, criador de rios e estradas, água em 3D, mixagem de som, edição de AI, sistema de animação de personagens, editor de animação facial e ferramentas para análise de desempenho; tendo sido utilizada para a implementação de populares jogos, como as séries FarCry e Crysis.

Uma inovadora experiência é apresentada por (NAKEVSKA et al., 2011), onde é criada

Figura 8 – CryEngine



Fonte: CryTek (2013)

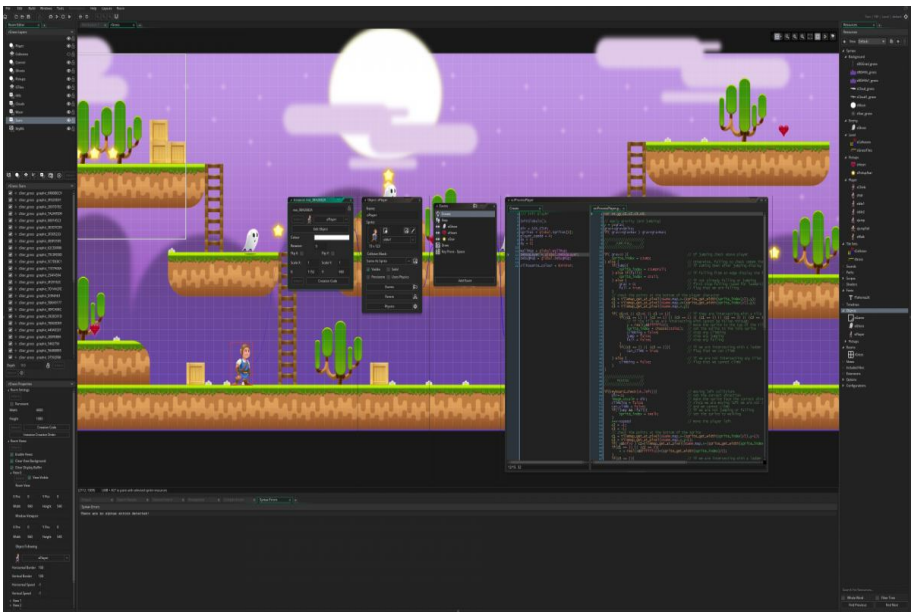
uma instalação de realidade mista, em que são integradas realidade virtual e realidade aumentada, a fim de implementar uma instalação que forneça uma experiência convincente, num ambiente de realidade virtual baseado em projeções, denominado CAVE, que consiste de uma sala, em que projeções nas paredes, piso e teto, envolvam o espectador. Para a realização da instalação, foi eleita a ferramenta CryEngine, por suportar uma variedade de funcionalidades úteis para criar ambientes virtuais imersivos e realistas e, com gráficos de alta qualidade. Foram assim implementadas as instalações: ALICE, que faz a imersão

no capítulo do livro Alice no País das Maravilhas, em que esta experimenta alimentos e bebidas para aumentar e reduzir sua estatura; Virtual Memory Tour, que é ambientado na Holanda do século XVI; e Virtual Garden, que permite ao espectador criar seu próprio jardim virtual.
As características de inclusão de ferramentas para os mais diversos fins, como oferecido por CryEngine, a tornam uma aplicação apropriada para a realização de projetos tecnologicamente desafiantes e inovadores, com o aqui exposto.

2.2.3.4 GameMaker Studio

GameMaker Studio (YOYOGAMES, 2017) (figura 9) é uma engine criada pela empresa escocesa YoYo Games, em 1999. Programada em Delphi e destinada a programadores inexperientes ou pessoas sem conhecimentos de programação, GameMaker baseia suas criações em sequências de ações drag and drop para seus jogos mais básicos, além da codificação numa linguagem de script denominada *GameMaker Language*, para jogos mais complexos. A *engine* se caracteriza ainda por não encriptar o conteúdo dos executáveis dos aplicativos com ela gerados, permitindo a engenharia reversa, e consequentemente, o compartilhamento de forma facilitada do código utilizado para a codificação destes.
Como é revelado pelo professor holandês Mark Overmars, criador da *engine*, em (OVER-

Figura 9 – GameMaker Studio



Fonte: YoYoGames (2017)

MARS, 2004), GameStudio tem sido amplamente usado no campo do ensino da ciência da computação, com organizações o utilizando em cursos de verão, com destaque para o *Children's Technology Workshop*, que oferece uma grade curricular baseada no programa. Várias escolas de ensino fundamental o usam para estimular o interesse na computação, com interessante taxa de sucesso, pois os alunos tendem a preferir programar em Game-

Maker a outras atividades acadêmicas, de cunho mais tradicional. Na Universidade de Utrecht , alunos aprendem conceitos básicos de design de jogos, tais como estabelecer regras, documentá-los, e, em última instância, criar jogos divertidos.
Desta forma exposto, GameMaker Studio, por suas características de possibilidade de criação de jogos com arrastar-e-soltar e linguagem script de alto nível, aliadas à facilidade de se obter o código fonte de uma aplicação previamente codificada, se configura como uma ferramenta democratizadora do processo de criação de jogos educativos, com a inclusão de programadores inexperientes e professores e alunos sem habilidades de codificação de programas de computador, no processo criativo.

2.3 *Role Playing Games*

Esta sessão descreve as características básicas dos jogos RPG (*Role Playing Games*), sua história desde suas origens, sua entrada no mundo computacional e sua representatividade no mercado, na forma de alguns dos mais populares softwares deste gênero.

2.3.1 Definição

Como descrito por (PEREIRA; ANDRADE; FREITAS, 1992), Role Playing Game (jogo de interpretação de papéis), é jogado com papel, lápis e dados, com os participantes se reunindo em volta de uma mesa; por isso, são comumente chamados de "RPGs de mesa". Cada partida, segundo o mesmo texto, é gerenciada por um participante, denominado *dungeon master* (mestre), que descreve aos jogadores os cenários, objetos e inimigos, conforme a história se desenrola, com a participação dos demais participantes, além de servir como um árbitro em relação à aplicação das regras do jogo. Os demais jogadores têm fichas, onde anotam os atributos e estado atual de seus respectivos personagens; logo após definir os valores iniciais de cada atributo, o mestre dá início ao jogo com uma narrativa, que contextualiza os jogadores dentro da história.

2.3.2 Histórico

RPGs, na forma descrita na sessão anterior, têm suas origens no ano de 1974, quando, de acordo com (TINSMAN, 2003), a primeira versão de D&D (Dungeons & Dragons) foi lançada, com a tiragem de mil cópias, produzidas artesanalmente pelo projetista de jogos norteamericano Gary Gygax, e publicadas pela empresa TSR (Tactical Studies Rules), de propriedade do autor.
Tal jogo introduziu, como é descrito por (SCHICK, 1991), conceitos que se tornariam padrões, incluindo atributos (tais como força, inteligência e destreza), classes (guerreiro, mago, sacerdote), níveis, raças (humano, anão, elfo e hobbit, então chamado halfling),

equipamentos, monstros e tesouros, assim como masmorras subterrâneas, com salas e portas com armadilhas e itens mágicos. O jogo inclui ainda regras para viagem por terra e mar, como taxas de movimento em diferentes tipos de terreno.
Como acrescenta (RDUSHAY, 2011), para a geração de personagens, são usados três dados comuns, conhecidos como 3d6 na terminologia RPG; dos resultados, sairiam os valores para os atributos Força, Inteligência, Sabedoria, Constituição, Destreza e Carisma; não era possível alterar os valores dos dados, mas se podia trocar pontos entre atributos. De outro lançe de 3d6, sairia a quantia disponível a cada jogador para a aquisição de equipamento; na ficha de personagem no início do jogo, além dos valores citados, são anotados o nome de cada um, além de sua classe. Anões e halflings são limitados à classe de guerreiro, enquanto elfos podem ser guerreiros ou magos, podendo trocar de classe em diferentes aventuras. Ainda segundo mesmo autor, cada personagem tem um número de habilidades, denominadas *spells*, que podem ser usadas para o combate contra cinquenta e oito monstros diferentes; este é realizado usando um d20 (dado de vinte lados), para determinar a sorte do personagem em cada golpe, à qual são acrescentados os valores de atributos do personagem envolvido, bem como o equipamento usado por este, para calcular o resultado final.
Uma versão atualizada, também de autoria de Gygax, de D&D foi lançada entre 1977 e 1979, AD&D (*Advanced Dungeons and Dragons*). Dentre as atualizações estavam classes como assassino, druida, monge, paladino, ladrão, bardo e ilusionista, como descreve (GUIDE, 1995). Paralelamente, outros RPGs eram lançados no mercado, tais como *Chivalry & Sorcery, RuneQuest*, e saindo da temática tradicional de capa e espada medieval, *Metamorphosis Alpha, Traveller, Gamma World* e *Superhero: 2044*, ambientados numa temática de ficção científica.

2.3.3 Computer Role Playing Games

Inspirados na série D&D, um número de programadores se engajou em criar versões eletrônicas do jogo, com as mesmas características do original, porém com um computador para assumir o papel de mestre. Dentro do jogo, o jogador escolhe uma classe, e sobe de nível, à medida que acumula XP (Experience Points, pontos de experiência, recebidos ao vencer inimigos e completar missões, nesse contexto denominadas *quests*), a fim de adquirir mais pontos de atributos, poder usar equipamentos e armas mais avançados, e assim, poder explorar mapas mais desafiantes. Recursos críticos para seu desempenho são o HP(Health Points),pontos de vida, que uma vez esgotados, levam o personagem à morte, com alguma penalização para sua ressurreição; MP(Mana Points), eventualmente denominados SP(Skill Points), pontos de mana, que são gastos para o lançamento de alguma habilidade especial, por exemplo, a bola de fogo de um mago; e dinheiro, comumente representado pela denominação gold (ouro). Elementos recorrentes nesse gênero de jogos são os NPCs (Non

Figura 10 – The Dungeon

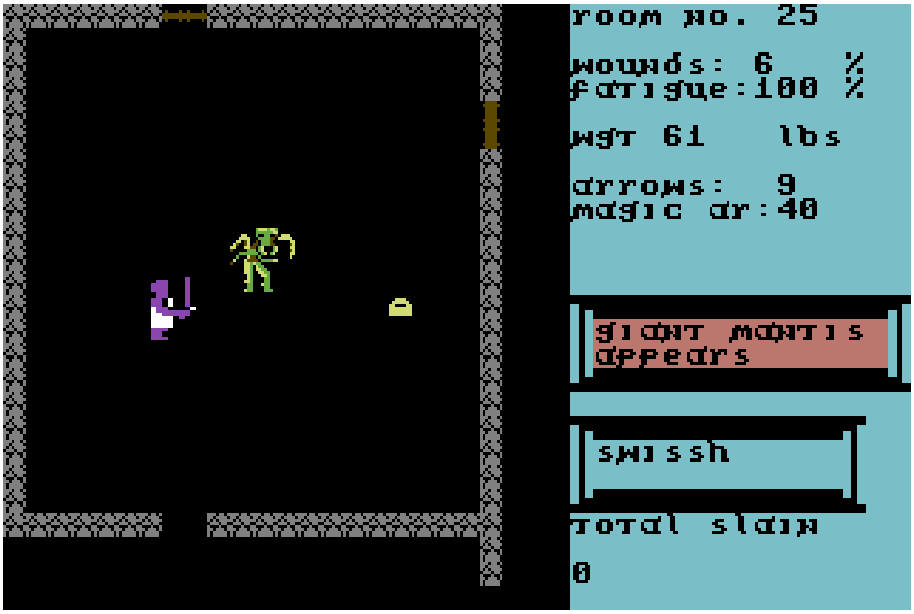
Fonte: crpgaddict (2011)

Player Characters), personagens controlados pela máquina, que realizam tarefas como compra e venda de poções, armas e equipamentos, além de propor missões aos jogadores. De acordo com (BARTON, 2008), representam essa primeira geração de CRPGs (Computer Role Playing Games), dois jogos datados de 1975: Dungeon, criado pelo programador de jogos norteamericano Don Daglow; e Pedit5, também chamado The Dungeon (figura 10), de seu conterrâneo Rusty Rutherford.

Ainda de acordo com (BARTON, 2008), a seguir, em 1979, veio Temple of Aphshai (figura 11), desenvolvido pela empresa norteamericana Automated Simulations (posteriormente rebatizada Epyx), originalmente para computadores TRS-80 e Commodore PET, com versões posteriores para Apple II, Atari 8-bit, IBM PC, VIC-20, Commodore 64, Atari ST, Amiga, Macintosh e Amstrad CPC; nos anos seguintes, se seguiriam Akalabeth: World of Doom, da California Pacific Computer para Apple II e IBM PC (1980); Wizardry, da Sir-Tech, para Apple II (1981); Sword of Fargoal, da Epyx para VIC-20 e Commodore 64 (1982); e de particular relevância, Ultima, que será comentado com detalhes na próxima sessão deste trabalho.

Da segunda metade dos anos 1980 em diante, um grande número de empresas entraria no ramo do desenvolvimento desse gênero de *games*, tais como New World Computing, Arkane

Figura 11 – Temple of Apshai para Commodore64



Fonte: hardcoregaming101 (2010)

Studios, Limbic Entertainment, Stormfront Studios, Interplay, além das multinacionais japonesas Sega e Nintendo, lançando, para microcomputadores e consoles de videogames, jogos agora considerados clássicos, como as séries *Might and Magic, Neverwinter Nights, Phantasy Star, Chrono Trigger* e *Final Fantasy*. Da década de 1990 em diante, cabe destacar a Blizzard Entertainment, com suas globalmente populares séries *Diablo* e *World of Warcraft*, que representa, por sua vez, uma subdivisão do gênero, hoje com grande relevância no mercado, o MMORPG(*Massively Multiplayer Online Role Playing Game*), que reúne jogadores da internet em um servidor, para uma aventura em comum, num mapa compartilhado. Para enfrentar determinados desafios, os jogadores poderão se reunir em grupos, denominados *parties*, na terminologia RPG.

2.3.4 CRPGs no mercado

A seguir, serão apresentados com um maior aprofundamento em detalhe, alguns dos jogos do gênero Computer Role Playing Games de relevância entre o público nas últimas décadas, que se mostrariam fundamentais para a definição desse gênero.

2.3.4.1 Ultima

Ultima (ULTIMA, 2012) (figura 12) é uma série de CRPGs, originada no supracitado Akalabeth: World of Doom, que teve início em 1981.
De acordo com (ULTIMACODEX, 2017), ao iniciar o jogo, o jogador deverá criar quatro personagens, que formarão uma party, com os campos: nome, sexo, raça (humano, anão,

elfo, fuzzy ou bobbit), classe (sacerdote, druida, mago, alquimista, ilusionista, ladrão, paladino, bárbaro ou guerreiro), e finalmente, distribuir um total de cinquenta pontos entre os atributos: força, destreza, inteligência e sabedoria.
Ainda de acordo com a mesma fonte, a saga é dividida em três eras, subdivididas em três jogos cada:

- Era da Escuridão(I - III)
  Acontece na terra de Sosaria, lar de cidades-estado em guerra entre si, sem uma liderança central. Sosaria é ameaçada por um trio de mágicos do mal: Mondain (Ultima I), Minax (Ultima II) e Exodus(Ultima III).

- Era da Iluminação(IV - VI)
  Sosaria passara a se denominar Britania, em honra a Lord British, primeiro rei de uma terra agora unificada, e que criaria um sistema de moral denominado as oito virtudes: honestidade, compaixão, valor, justiça, sacrifício, honra, espiritualidade e humildade, baseadas em três princípios: o livro da verdade, a vela do amor e o sino da coragem.

- Era do Armagedom(VII - IX) Na última última era, o mundo de Britania está novamente sob ataque, nesta vez por uma entidade do mal chamada O Guardião, que ergueria oito pilares mágicos, que corromperiam o povo de Britania, os levando a repudiar as oito virtudes.

Figura 12 – Ultima Online

Fonte: rockpapershotgun (2015)

A série ganharia em 1997 um representante MMORPG, *Ultima Online*. Se passando nos mundos de Felucca, Trammel, Ilshenar, Malas, Tokuno e Ter Mur, o jogo teve as expansões,

a partir de 1998: The Second Age, Renaissance, Third Dawn, Lord Blackthorn's Revenge, Age of Shadows, Samurai Empire, Mondain's Legacy, Stygian Abyss e Time of Legends, esta última de 2015.

2.3.4.2 Final Fantasy

A série Final Fantasy (SQUAREENIX, 2017) (figura 13) é publicada pela empresa japonesa Square Enix Holdings, e desenvolvida pelo projetista de videogames Hironobu Sakaguchi, tendo sido iniciada em 1987, e tendo como mais recente representante Final Fantasy XV, de 2016.
De acordo com (FINALFANTASY, 2017), nos jogos, o jogador comanda uma *party* de quatro personagens, cada um deles de uma classe a ser escolhida no momento de sua criação, dentre guerreiro, ladrão, monge, mago branco, mago vermelho ou mago negro. O jogador dá a cada personagem sob seu comando ordens, como "lutar", "usar mágica" ou "usar item", através de uma interface baseada em menu de opções, no decorrer do encontro com oponentes. Anteriormente a Final Fantasy XI, os combates eram baseados em turno, com cada lado ocupando sua metade da tela; a citada versão introduziu o combate em tempo real, onde os personagens reagem a estímulos a qualquer momento da batalha.
É descrito pela mesma fonte que o conflito central na série tem foco num grupo de personagens enfrentando um antagonista do mal, frequentemente se desenrolando num estado soberano em rebelião, da qual os citados personagens tomam parte. Outra característica de cada uma das aventuras desta série é conter dois vilões principais, onde o principal se revela subserviente do segundo, levando o grupo de heróis a estender sua luta além do inicialmente concebido. Esferas e cristais mágicos são itens recorrentes, frequentemente conectados ao tema central das aventuras, constituindo um papel central nos enredos; assim, os controle sobre tais cristais compreende o objetivo principal da história. Os elementos clássicos, ou seja, terra, fogo, água e ar também constituem um tema recorrente, relacionado aos heróis vilões e itens, assim como conflitos entre a tecnologia e a natureza.

2.3.4.3 Ragnarok Online

Ragnarok Online (figura 14) é um MMORPG criado pela empresa sul-coreana Gravity, baseada no manhwa (revista em quadrinhos coreana) conterrâneo Ragnarok, lançado em 2002, em seu país de origem, para Windows.
Como descreve (IROWIKI, 2017), o jogador poderá escolher entre as classes, conhecidas neste jogo como job (emprego) de espadachim, mago, acólito, ladrão, arqueiro e mercador, que evoluirão no decorrer das passagens de nível. Por exemplo, o acólito poderá escolher evoluir para sacerdote ou monge, e assim por diante, até se tornar arcebispo (evolução do sacerdote), ou sura (evolução do monge).

Figura 13 – Final Fantasy XV

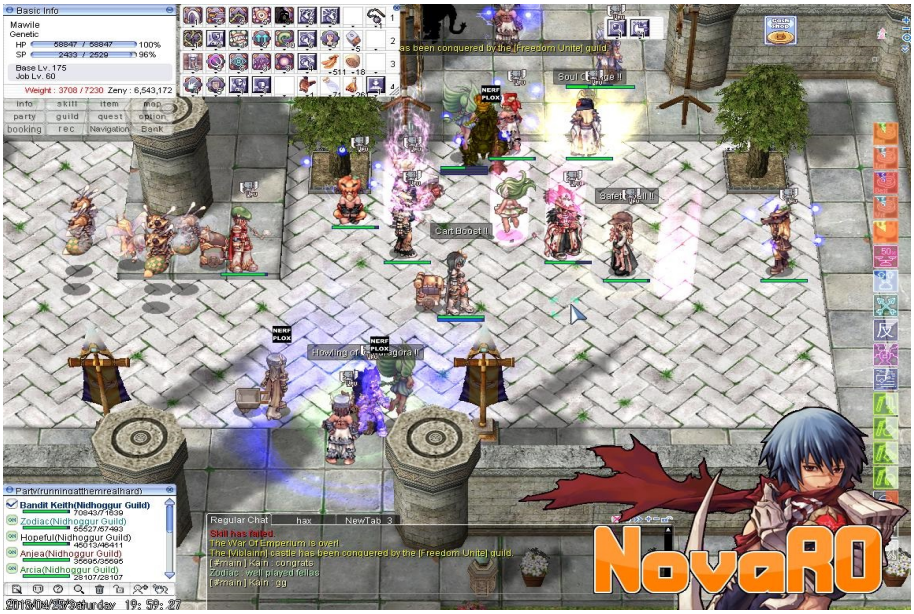
Fonte: FinalFantasy (2017)

Também segundo a mesma fonte, o jogo se passa em um continente separado em três nações: O Reino de Rune Midgard, onde todos os personagens iniciam suas aventuras; a industrializada República de Schwatzvalt; e a religiosa Arunafeltz. Nas várias cidades do jogo, há atividades nas quais os jogadores podem se engajar, a maior parte delas, missões propostas pelo Eden Group, uma organização secreta, que recompensa o jogador com xp, além de equipamento exclusivo pelo cumprimento de suas demandas. Outra instituição presente em todas as cidades é a Kafra Corporation, que fornece aos jogadores os serviços de tele transporte, armazenagem, salvamento de lugar para volta em caso de eventual morte e aluguel de carrinho de transporte. Também há em todas as cidades lojas para compra e venda de itens; os jogadores também têm a opção de usar os espaços das cidades para montar suas próprias lojas, onde vendem seus itens.

2.3.4.4 World of Warcraft

World of Warcraft (WORLDOFWARCRAFT, 2017) (figura 15), comumente conhecido como WoW, é um MMORPG lançado pela Blizzard Entertainment em 2004, baseado na franquia Warcraft, uma série de RTS (Real Time Strategy), publicada pela própria Blizzard, a partir dos anos 1990. Foram disponibilizadas pela empresa, posteriormente, as expansões Burning Crusade (2007), Wrath of the Lich King (2008), Cataclysm (2010), Mists of Pandaria (2012), Warlords of Draenor (2014) e Legion(2016).
Como descreve (WOWHEAD, 2017), o jogo confronta duas facções nos mundos de Azeroth (disponível desde a versão original) e Draenor (iniciada como parte da Burning Crusade), a Aliança (representada por humanos, elfos da noite, anões, gnomos, draenei e worgen)

Figura 14 – Ragnarok Online



Fonte: novaragnarok (2015)

e a Horda (cujas raças são orcs, tauren, trolls, mortos vivos, elfos de sangue e goblins). Há ainda os pandarens, uma raça neutra, que pode se alinhar a qualquer das facções ao chegar a determinado nível.

De acordo com o mesmo texto, ao criar um personagem, o jogador deverá escolher sua raça, dentre as supracitadas; e sua classe, dentre guerreiro, mago, paladino, druida, xamã, ladino, sacerdote, caçador, cavaleiro da morte, bruxo, monge ou caçador de demônios. Posteriormente, poderá ainda escolher uma profissão primária dentre alfaiataria, alquimia, couraria, encantamento, engenharia, escrivania, esfolamento, ferraria, herborismo, joalheria ou mineração; e uma ou mais profissões secundárias, dentre arqueologia, culinária, pesca e primeiros socorros.

World of Warcraft oferece servidores, à escolha do jogador ao iniciar cada um de seus personagens, dedicados a:

- PvE (Player versus Enviroment), onde os jogadores têm como foco combater monstros controlados pela máquina, completar missões, e se aventurar em *dungeons* (ambientes povoados por monstros, que requerem parties de cinco membros, e dão direito a itens especiais, à medida que os jogadores vencem inimigos especiais, chamados bosses) e raids (similares a dungeons, entretanto, exigindo grupos de dez até quarenta jogadores).

- PvP (Player versus Player), que em adição às características do modo acima descrito, é permitido o combate entre membros das duas facções a qualquer momento (à exceção do caso de um dos jogadores se encontrar no território pertencente à sua facção).

Figura 15 – World of Warcraft

Fonte: wowhead (2017)

- RP (Roleplay), onde os jogadores são motivados a jogar orientados à interpretação de papéis.

2.4 Reuso de Software

Reuso de software é descrito por (KRUEGER, 1992) como o processo de criar sistemas de software a partir de software existente, ao invés de construir novos sistemas do início. Assim colocado, com o advento do reuso de software, a produção de sistemas que utilizem suas técnicas, apresenta a tendência de redução do número de horas trabalhadas, viabilizando, assim, uma significativa deflação de seu custo, trazendo crescimento em sua competitividade; de forma análoga, organizações de menor porte verão incrementadas suas possibilidades de ingressar no mercado.
Nas sessões seguintes, serão apresentadas técnicas de reuso vigentes no campo do desenvolvimento de software, com as respectivas definições:

2.4.1 *Frameworks*

De acordo com (JOHNSON, 1997), a estrutura de um *framework* pode ser descrita como um design reusável de toda ou uma parte de um sistema, que é representado por um conjunto de classes abstratas e a forma como suas instâncias interagem entre si. De acordo como a mesma fonte, Seu propósito pode ser definido como um esqueleto de uma aplicação, que pode ser personalizado pelo desenvolvedor.
Assim exposto, *frameworks* constituem um código extensível, que permite ao desenvolvedor, por meio do reuso de seus componentes, evitar a reescrita de funções previamente existentes,

simplesmente a elas adicionando funcionalidades particulares.
Segundo (VALUECODERS, 2018), os *frameworks* de maior relevância entre desenvolvedores de software atuais são apresentados a seguir:

- AngularJS: É um *framework front-end* JavaScript, código fonte aberto, concebido particularmente para aplicações web de uma só página.
- Laravel: Realizado com fins de desenvolvimento de aplicações web segundo um padrão arquitetural que prevê a separação de responsabilidades entre os componentes da aplicação.
- React.js: É mantido pelo Facebook, aliado a uma considerável comunidade de desenvolvedores, sendo frequentemente utilizado para o desenvolvimento de interfaces em aplicações web, em particular quando há interação com dados que apresentem grande variância no decorrer do tempo.(VALUECODERS, 2018)
- Node.js: Apresenta características de um ambiente completo de desenvolvimento, que incluem um framework, para auxiliar na criação de aplicações escaláveis e com bom desempenho, sendo capaz de trabalhar com significativo número de conexões simultâneas.

2.4.2 Componentes de Software

De acordo com (CLEMENS; DOMINIK; STEPHAN, 2002), o desenvolvimento baseado em componentes objetiva construir software a partir de componentes pré-existentes; fazer componentes como entidades reusáveis; e aperfeiçoar aplicações, ao substituir componentes. Representantes significativos dos avanços no tocante ao uso desta técnica são as tecnologias MS COM/DCOM, J2EE, Corba, OSGi, e .NET, amplamente utilizadas em vários sistemas distribuídos e aplicações desktop. A última tendência em aplicações baseadas em componentes é representada por sistemas embarcados em tempo real, como, por exemplo, a Automotive Open System Architecture (Autosar), utilizada para a automatização de veículos automotivos.
Segundo (PLASIL; VISNOVSKY, 2002), um componente de software pode ser definido como uma entidade de natureza *black box*, que esconde sua implementação, e requer uma conjunto de entradas de dados, acessados por meio de interfaces, para seu funcionamento. Componentes podem ser combinados, a fim de formar um só componente de nível mais alto, sendo essas composições especificadas numa ADL (Architecture Description Language), ou linguagem de descrição de arquitetura. Ainda segundo a mesma fonte, o comportamento de componentes pode ser descrito com diagramas UML de interação, colaboração e estados. Assim descrito, componentes de software fornecem possibilidades de reuso que dispensam o desenvolvedor de codificar sua aplicação desde o início, podendo utilizar os serviços ofereci-

dos pelos componentes, sem que este necessite conhecer os detalhes de sua implementação. Dentro do padrão MVC, componentes podem ser categorizados em:

- Componentes de modelo se responsabilizam por requisições de consulta ou alteração de informações relevantes às aplicações que os utilizem. Podem ser exemplificados por APIs (Application Programming Interfaces) com implementações de funcionalidades de acesso a bancos de dados.

- Componentes de visão fornecem uma interface gráfica aos usuários de aplicações. Telas de autenticação de usuário com campos de nome de usuário e senha são exemplos de componentes que podem ser compartilhados por várias aplicações.

- Componentes de controle são responsáveis pela interação entre as camadas de visão e modelo, realizando, por exemplo, a verificação da validade de um número de Cadastro de Pessoa Física, antes que os dados de um cliente sejam cadastrados num sistema de *e-commerce*.

2.4.3 Linhas de Produto de Software

Segundo (NORTHROP et al., 2007), uma linha de produtos é um conjunto de produtos que, juntos, preenche uma lacuna de um segmento de mercado, ou cumpre uma missão específica. Linhas de produto não constituem uma novidade na indústria, tendo sido concebidas pelo empresário norte-americano Henry Ford, em 1913. Entretanto, Linhas de Produto de Software, designadas *Software Product Lines* (SPL) na literatura inglesa, são um conceito relativamente novo, que está emergindo rapidamente, como um importante e viável paradigma de desenvolvimento. Flexibilidade é uma palavra-chave no mercado de software, e SPL cumprem a promessa de sistemas personalizados, construídos especificamente para as necessidades de clientes específicos. O conceito é explanado por (MCGREGOR, 2007), ao descrever SPL como uma técnica que dá lugar a conjuntos de sistemas compartilhando um número de características (denominadas *features* no contexto das LPS) em comum, que satisfazem as necessidades específicas de um segmento de atividades. Ainda segundo (MCGREGOR, 2007), as características em comum são designadas comunalidades, e as particularidades de cada produto são denominadas variabilidades; a separação entre estas é definida pela Engenharia de Domínio; os produtos da linha são definidos pela Engenharia de Aplicação. De acordo com a citação, linhas de produto podem ser descritas como uma forma de engenharia de produção que remete à metodologia de linhas de montagem de veículos, onde um determinado modelo é fabricado com uma série de componentes em comum, tais como motor e chassis, que comporão as comunalidades; e outras características são personalizadas dentro da linha, como cor e presença ou ausência de itens acessórios, como aparelho de ar condicionado ou direção hidráulica.
Um significativo esforço em disponibilizar as funcionalidades de SPL para a comunidade

desenvolvedora é implementado pelos programadores brasileiros Marcílio Mendonça e Moisés Branco; e do canadense Donald Cowan, o S.P.L.O.T. (Software Product Lines Online Tools), que, como descrito em (MENDONCA; BRANCO; COWAN, 2009), é um conjunto de ferramentas online para o auxílio ao desenvolvimento de aplicações baseadas em Linhas de Produto de Software. Ainda de acordo com a mesma fonte, S.P.L.O.T[1], disponibiliza as seguintes ferramentas:

- *Feature Model Editor* é um editor de modelos de *features*, com características fixas, facultativas ou variantes dentro da SPL.
- *Product Configuration* é um configurador de *features* de produtos constantes da engenharia de aplicação de uma SPL.
- *Automated Analysis* realiza uma análise automatizada das SPL, detectando inconsistências em suas construções.

2.4.4 Geradores de Aplicação

De acordo com (PRESSLEY; GALI; DOGGETT, 2014), ferramentas geradoras de software oferecem ao desenvolvedor uma interface gráfica, com a qual este poderá criar aplicações sem codificação. A partir do modelo definido na interface, a citada ferramenta é capaz de gerar código em diferentes linguagens, como C++, Java e HTML.
Assim descrito, esta técnica de reuso prevê possibilidades consideráveis em termos de ganho de tempo no ciclo de vida de uma aplicação, bem como uma democratização no tocante às equipes de desenvolvimento, que poderão incluir componentes sem conhecimento de linguagens de programação.
Destacam-se os seguintes empreendimentos nesta área:

- Bobile[2] Gerador de aplicações *mobile*, que se propõe a oferecer aos usuários a possibilidade de criar aplicações para aquisição de produtos e serviços, contatos com clientes e fornecedores, agendamentos e demais funcionalidades que se mostrem necessárias a organizações que as utilizem.
- Kissflow[3] Ferramenta que oferece um número superior a cinquenta *templates*, que poderão ser personalizados pelo desenvolvedor, a partir de uma *interface* do tipo *drag and drop*, para criar formulários com campos, tabelas e conexões com bancos de dados, em processos cujos fluxos de atividade são definidos pelo usuário.
- Bubble[4] Aplicação voltada ao desenvolvimento de aplicativos web, capaz de gerar páginas com elementos como texto, imagens, mapas e entradas de dados com uma

[1] Disponível em www.splot-research.org
[2] Disponível em https://bobile.com
[3] Disponível em https://kissflow.com/
[4] Disponível em https://bubble.is/

interface drag and drop, permitindo ainda a atualização de documentos em tempo real, dispensando, desta forma, do desenvolvedor, o conhecimento de linguagens destinadas a web design, como HTML e CSS (*Cascading Stylesheets*).

3 TRABALHOS CORRELATOS

Nesta sessão serão identificados trabalhos científicos voltados à proposta do uso de jogos com fins educacionais, suportados por alguma técnica de reuso.
Foram identificados, na literatura disponível para consulta na web, trabalhos relativos à construção de engines destinados à produção de jogos; assim como aplicações destinadas ao mesmo fim. Porém, nenhum deles dedicado a *games* engines para elaboração de jogos RPG com conteúdo construcionista.
Serão apresentados, neste capítulo, os trabalhos supracitados.

3.1 *A Framework to Design Educational Mobile-Based Games Across Multiple Spaces*

De acordo com (FERNÁNDEZ-PANADERO et al., 2015), a adoção de dispositivos móveis e tecnologias cientes de contexto oferece a oportunidade de criar jogos educacionais móveis, capazes de ser usados em espaços múltiplos, que enriquecem a experiência dos aprendizes.
O *framework* apresentado neste trabalho oferece a possibilidade de facilitar e acelerar a criação de jogos com essas características, com um conjunto de templates a ser completados pelo usuário, que trazem consigo uma sensível agilização no tocante à autoria da narrativa e o design de jogos.
Como estudo de caso, o framework foi usado para desenvolver um jogo móvel para um museu, sendo este utilizado por dez alunos de idades entre seis e doze anos. Os resultados do estudo mostram que é tecnicamente factível usar o *framework* para criar um jogo móvel através de três espaços: museu, casa e sala de aula, sendo a experiência considerada pelos alunos como uma atividade de aprendizado motivadora.

3.2 *Towards a Service-Oriented Architecture framework for educational serious games*

Como afirmado por (CARVALHO et al., 2015), jogos educacionais sérios são tipicamente concebidos como produtos únicos, totalmente padronizados de acordo com os requisitos de um cliente em particular. Portanto, sua construção pode se mostrar uma tarefa proibitivamente custosa, tanto em termos de tempo quanto de recursos financeiros, pela baixa reusabilidade dos componentes do produto final. A abordagem SOA (Service-Oriented Architecture), ou seja, arquitetura orientada a serviço pode oferecer uma solução

para a redução de custos, trazendo consigo, assim, um fomento ao desenvolvimento de jogos sérios.
Neste trabalho, é aplicado um modelo denominado Activity Theory-based Model of Serious Games (ATMSG), para identificar componentes relevantes existentes que possam ser reusados para múltiplos jogos sérios educacionais. É utilizada a estrutura derivada para classificar os elementos de um jogo existente e identificar como ele pode ser refatorado e expandido segundo o paradigma SOA.

3.3 *The RAGE Game Software Components Repository Supporting Applied Game Development*

Como descrito por (GEORGIEV et al., 2017), O projeto RAGE visa estimular a indústria de jogos sérios, ao disponibilizar um grupo de componentes reusáveis de tecnologia de jogos, que estúdios de jogos sérios possam facilmente integrar a seus projetos, facilitando e acelerando seu desenvolvimento, além de reduzir os custos totais. Os componentes incluem funcionalidades como aprendizado personalizado, balanceamento de jogo, tecnologias de linguagem, *storytelling* interativo e gamificação social.
Ainda de acordo com (GEORGIEV et al., 2017), este trabalho apresenta a arquitetura do repositório de componentes e a metodologia de desenvolvimento com estes, com detalhes de projeto e implementação da arquitetura de back-end, bem como das ferramentas de front-end correspondentes, com uma descrição dos experimentos iniciais com a infraestrutura, análise e identificação de desenvolvimento e pesquisa futuros.

3.4 Modelando Ambientes de Aprendizagem Virtuais utilizando *Role-Playing Games*

Segundo (BOAVENTURA; SARINHO, 2017), ainda que *engines* sejam reusáveis para muitos projetos, elas geram alta dependência de um jogo em recursos de implementação fornecidos pela engine eleita para o desenvolvimento, o que envolve a obrigatoriedade em utilizar aspectos indesejáveis destas.
Como uma tentativa de identificar comunalidades e variabilidades no domínio de jogos digitais, surge NESI (*Narrative, Entertainment, Simulation, and Interaction*), como um modelo de *features* capaz de separar aspectos concernentes aos aspectos de narrativa, entretenimento, simulação e entretenimento de um projeto de jogo.
Numa abordagem similar, aparece o *GameSystem, DecisionSupport, and SceneView*(GDS), outro modelo de *features*, que descreve configurações genéricas, como interface com o usuário, além de aspectos comportamentais encontrados na implementação de jogos.
Este artigo apresenta o *Minimal Engine for Digital Games* (MEnDiGa), uma coleção

extensível de classes representativas baseada num conjunto simplificado de features NESI e GDS, que pode ser utilizado como uma fundação para jogos de pequeno porte, dispensando maiores modificações.

3.5 JIndie: Uma Linha de Produto de Software para Jogos Educativos com Foco no Construcionismo

Segundo (FILHO; HERNÁNDEZ-DOMÍNGUEZ, 2016), o envolvimento de jogos no processo de ensino tem se mostrado um fator de motivação e aprendizado de significativa importância. Este artigo tem como objetivo facilitar o desenvolvimento de jogos educacionais, por meio de uma SPL baseada em jogos construcionistas, a qual foi utilizada para o desenvolvimento de quatro jogos, para sua validação. Com base nos resultados, é possível vislumbrar as vantagens do uso de linhas de produto de software para o desenvolvimento de jogos educacionais, que incluem redução no tempo de desenvolvimento, simplificação no tocante à complexidade do software e direcionamento à criação de jogos.

3.6 *Designing a Game Generator as an Educational Technology for the Deaf Learners*

De acordo com (BOUZID; KHENISSI; JEMNI, 2015), são raros os jogos educacionais voltados ao aprendizado de crianças com surdez total ou parcial, devido ao fato de que a maioria desses jogos utiliza como ferramenta de comunicação a linguagem falada, não apresentando, portanto, o uso de linguagens de sinais, necessárias ao aprendizado deste grupo alvo.
De acordo com a mesma fonte, a fim de trazer uma contribuição para o aprendizado de crianças com deficiência auditiva, e fornecer novos meios para educadores que almejem desenvolver materiais educacionais baseados em computador para essas crianças, é apresentado nesse trabalho um projeto para um gerador de software para o desenvolvimento de jogos, chamado MemoSign. O gerador em questão faz uso de conceitos de jogos de memória com uma linguagem de sinais, que permite a pais e professores criar instâncias de jogos sem conhecimento de linguagem de programação, com fins de promoção de aquisição de vocabulário por parte dos jogadores.

3.7 *Learning basic programming concepts with Game Maker*

Segundo (JOHNSON, 2017), um currículo de curso de computação adequadamente projetado inclui programação em mais de um contexto; além de linguagens textuais, como Java, os aprendizes devem também absorver linguagens visuais para projetos como jogos,

a fim de oportunizar o aprendizado do processo de design. Aspectos como planejamento e teste também devem ser enfatizados.
Ainda de acordo com o autor, a proposta desse trabalho é explorar a introdução de uma nova unidade de trabalho como parte da grade curricular na qual alunos de oitava série criaram jogos utilizando Game Maker, uma ferramenta visual de geração de jogos. A atividade consistiu de uma implementação de atividades de aprendizado de natureza construcionista, caracterizada por um padrão colaborativo. Os alunos trabalharam em pares, durante um período de oito semanas, eventualmente se associando a outros pares, revisando os trabalhos alheios e compartilhando conhecimento, e identificando dificuldades e erros com os quais os alunos se deparam, quais práticas são mais adequadas e até que ponto uma abordagem construcionista é viável para esse tipo de trabalho.

3.8 Análise dos trabalhos correlatos

Esta sessão compara os trabalhos descritos neste capítulo, em relação ao presente trabalho.

Tabela 1 – Contribuições dos trabalhos correlatos

Trabalho	RPG	Construcionismo	Engine	Técnica de Reuso
3.1				*Framework*
3.2				*Framework*
3.3				Componentes de Software
3.4	x			SPL
3.5		x		SPL
3.6			x	Gerador de Software
3.7		x		Gerador de Software

Fonte: Elaborada pelo autor

Como se verifica na tabela 1, não foi identificado, dentre os trabalhos correlatos à presente monografia, algum que reunisse as três características centrais ao tema proposto, a saber: RPG, construcionismo e engine. O trabalho exposto na sessão 3.7 apresenta características mais próximas ao tema deste trabalho, não incluindo, entretanto, a implementação de uma engine, se resumindo a propor o uso de uma ferramenta desta natureza em sala de aula.

4 O JOGO CONSTRUCIONISTA NO CONTEXTO DE RPG

Neste capítulo será apresentada LearnCraft, a aplicação que é o principal objetivo desta dissertação.
A origem etimológica de seu título vem de uma combinação das palavras inglesas Learn (Aprender) e Craft (Arte); tal designação referencia ainda, a título de homenagem, a série de jogos RTS Warcraft.
No que concerne à técnica de reuso utilizada em sua implementação, esta pode ser classificada como um gerador de software, técnica descrita na sessão 2.4.4, tendo em vista a democratização do uso da ferramenta, que deverá incluir em sua comunidade criadores de jogos sem conhecimento de linguagens de programação.
A apresentação será feita em dois passos, a saber: a modelagem, onde serão expostos os requisitos funcionais do jogo, com aspectos como cadastro, criação de mapas e o jogo em si; e a implementação, onde será descrito o ferramental teórico e prático utilizado na criação do software.

4.1 Modelagem

A proposta de desenvolvimento constitui uma *engine* para criação de mods, como descrito na sessão 2.2.2, em formato RPG baseado em turnos, com conteúdo construcionista, capaz de funcionar em ambientes web e *mobile*.
Cada um desses mods incluirá: um enigma proposto pelo criador; e um mapa a ser construído pelo mesmo membro da comunidade, com diferentes cenários, onde o personagem do jogador encontrará uma fauna hostil, a qual ele deverá enfrentar, a fim de viajar entre os diferentes pontos do mapa, além de colecionar itens para posterior venda; e cidades com NPCs (Non-Player Characters), com os quais o jogador poderá interagir, para: venda dos itens anteriormente mencionados; e compra de poções de vida e mana, além de pistas que lhe serão úteis para a resolução do enigma proposto. O objetivo do jogo é construir um artefato no formato de um texto, que responda o enigma proposto no início da aventura, o que o caracteriza como uma ferramenta de apoio a metodologias de ensino embasadas no paradigma construcionista, conforme definido na sessão 2.1.2.

4.1.1 Cadastro

Para fazer parte da comunidade LearnCraft, deverá ser feito cadastro, que incluirá:

- Nome

- E-Mail
- Senha

Uma vez feito o cadastro, o aplicativo credenciará o novo membro a participar, tanto como criador de mods, quanto como jogador destes, uma vez criados. Sua identificação deverá ser feita usando para tal um *logon*, constante de seu e-mail e senha, conforme preenchidos no momento do cadastro.

4.1.2 Criação de mods

Para criar um *mod*, deverão ser entradas as seguintes informações:

- Nome
- Dimensões do mapa (linhas e colunas, que deverão ser limitadas a um mínimo de 3 e um máximo de 15)
- Um enigma, a ser resolvido pelo jogador

Após preenchidas essas informações, o criador deverá preencher uma matriz, de dimensões eleitas no passo acima, com os terrenos:

- Água, por onde o jogador não poderá trafegar
- Deserto, cuja fauna é representada por cobras douradas e escorpiões; e flora, que constitui os obstáculos aos movimentos, tanto do jogador, quanto de seus oponentes, é representada por cactos.
- Campo, habitado por coelhos e sapos.
- Floresta, dá moradia a cobras verdes e aranhas; tem arbustos como obstáculos aos movimentos.
- Cidade (com nome obrigatório)

Uma vez preenchida a matriz, o criador deverá adicionar a cada cidade do mapa artefatos com informações que permitirão ao jogador elaborar uma resposta ao enigma proposto. Serão entradas também as coordenadas da posição inicial do jogador, ao entrar no jogo.

4.1.3 Autenticação

O logon no aplicativo é feito com a digitação de e-mail e senha de um usuário pré-cadastrado, como descrito na sessão imediatamente anterior.
Uma vez autenticado, o participante poderá eleger como atividade a ser realizada, dentre as opções: criar um *mod* ou jogar um *mod* previamente criado.

4.1.4 O jogo

Os jogos criados com a ferramenta serão *turn-based*, ou seja, baseados em turno, no estilo de jogos de tabuleiro, como xadrez ou damas, onde cada jogador deverá esperar pela jogada do adversário para empreender seu próximo movimento. O jogador deverá viajar entre as cidades incluídas no mapa do *mod* escolhido, a fim de comprar pergaminhos que lhe darão pistas da resolução do enigma proposto.
Para tal, deverá atravessar terrenos, cada um correspondendo a uma célula do mapa, onde enfrentará elementos da fauna local, usando as habilidades de seu personagem. Cada inimigo derrotado dará ao jogador um item, que poderá ser vendido a um NPC em qualquer cidade do jogo.
Cada célula é constituída por um tabuleiro, configurado como uma matriz de 15x15 células, por onde o jogador se deslocará segundo os pontos cardeais. Os itens ganhos com a vitória sobre os inimigos ficarão armazenados num inventário com capacidade para dezesseis itens. As poções, tanto de vida como mana, assim como o ouro, não ocuparão espaço neste inventário.

4.1.4.1 NPCs (Non-Player Characters)

Cada cidade conterá três NPCs:

- Um comprador de itens, que pagará um valor de dez moedas de ouro por cada item constante do inventário do jogador.
- Um vendedor de pergaminhos, que, ao preço de cem moedas de ouro cada, conterão pistas a ser usadas pelo jogador para compor sua resposta ao enigma proposto no início do jogo.
- Um vendedor de poções, que venderá poções de vida e mana, pelo valor de trinta moedas de ouro cada, para ser utilizadas pelo jogador em suas aventuras pelo mapa do jogo.

4.1.4.2 Atributos

As características básicas de cada personagem é representada por quatro atributos:

- Força: Representa a capacidade do personagem em causar e suportar dano físico
- Inteligência: Incrementa os pontos de mana disponíveis ao personagem, bem como a taxa de sua recuperação
- Destreza: Representa a capacidade do personagem em evitar danos físicos

- Constituição: Incrementa os pontos de vida disponíveis ao personagem, bem como a taxa de sua recuperação

4.1.4.3 Classes

Como descrito na sessão 2.3 deste trabalho, dentro da definição de *Role Playing Game*, os mods criados com o LearnCraft permitirão que, no papel de jogador, o participante atue com uma classe à sua escolha, com respectivos valores iniciais dentre os diferentes atributos, além de habilidades especiais, exclusivas de cada classe.

4.1.4.3.1 Mago

É uma classe que apresenta como especialidade o combate dos inimigos à distância, apresentando reduzido poder de ataque físico e pouca resistência a agressões vindas de seus oponentes.
Inicia o jogo com os valores de atributos:

- Força: 10
- Inteligência: 30
- Destreza: 10
- Constituição: 10

Para realizar suas atribuições, conta com as seguintes habilidades especiais:

- Bola de Fogo: uma chama vinda do firmamento, que atinge uma casa do tabuleiro à escolha do jogador, dentro do campo de visão de seu personagem. Apresenta o poder de causar dano ao oponente que se encontrar na casa escolhida, sendo este proporcional ao intelecto do personagem controlado pelo jogador. Consome 80 pontos de mana.
- Bola de Gelo: uma bola gelo, que, de forma análoga à habilidade supracitada, atinge um oponente, com o poder de mantê-lo inoperante por duas rodadas. Consome 80 pontos de mana.
- Teletransporte: permite ao personagem se deslocar automaticamente até uma casa, dentro do campo de visão do personagem, à escolha do jogador. Consome 100 pontos de mana.

4.1.4.3.2 Cavaleiro

É uma classe que apresenta como especialidade o combate próximo aos seres que venham a obstaculizar seu caminho, com importante poder de ataque físico e resistência a golpes vindos de seus inimigos.
Inicia o jogo com os valores de atributos:

- Força: 30
- Inteligência: 10
- Destreza: 10
- Constituição: 10

Para realizar suas atribuições, conta com as seguintes habilidades especiais:

- Mestre de Anatomia: confere ao próximo golpe desferido pelo personagem acerto crítico, o que incrementa o dano causado ao inimigo golpeado em 100%. Consome 80 pontos de mana.
- Arte da Guerra: brinda ao personagem um golpe extra numa mesma rodada. Consome 80 pontos de mana.
- Proteção Divina: protege o personagem de quaisquer danos por uma rodada, reduzindo o dano causado pelos oponentes a 0%. Consome 100 pontos de mana.

4.1.4.4 Controles

Para movimentar seu personagem, o jogador usará as teclas A (norte), S (sul), D (oeste) e W (leste).
Para usar as habilidades especiais, serão usadas as teclas de 1 a 3.
A tecla M, ao ser pressionada, abrirá um mapa do mundo onde o jogo se desenrola; a tecla C exibirá os valores dos atributos do personagem; a tecla E mostrará o enigma proposto pelo criador do jogo; a tecla P exibirá os pergaminhos previamente adquiridos pelo jogador, que poderá navegar entre os mesmos com as teclas $<$ e $>$. Para consumir uma poção de HP (Health Points, pontos de vida), será usada a tecla 4, e para a poção de MP (Mana Points, pontos de mana), a tecla 5.
Tanto para a ativação de habilidades quanto para o consumo de poções, poderá ser usado o mouse, apertando-se ícones que estarão à disposição do jogador.

4.1.4.5 Recompensas

A cada oponente vencido, o personagem ganhará 50 pontos de experiência (XP), além de um item que poderá ser, dependendo da espécie vencida:

- Cobras verde e dourada: Presas
- Aranhas: o corpo
- Coelhos e sapos: a pata
- Escorpiões: o ferrão

4.1.4.6 Níveis

A cada 1000 pontos de experiência obtidos, o personagem ganhará um nível. A cada nível galgado, o jogador poderá distribuir 10 pontos entre os atributos de seu personagem.

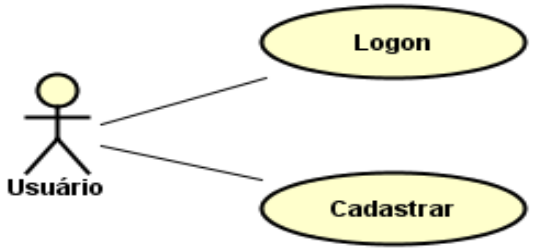
4.2 Projeto

Nesta sessão, será discutido o projeto da aplicação, ilustrado com diagramas UML. Nas sub-sessões seguintes, os requisitos funcionais definidos na sessão 4.1 são subdivididos em funcionalidades, cada uma em um documento correspondente, os quais serão descritos e ilustrados com diagramas de caso de uso.

4.2.1 Autenticação de usuário

Nesta página, é apresentada a escolha entre autenticação, realizada com dados pessoais (e-mail e senha) do membro da comunidade LearnCraft; ou cadastro, caso este ainda não esteja matriculado no sistema.
A figura 16 contém o diagrama de caso de uso correspondente.

Figura 16 – Diagrama de Caso de Uso - Logon



Fonte: Elaborada pelo autor

Figura 17 – Diagrama de Caso de Uso - Cadastro

Fonte: Elaborada pelo autor

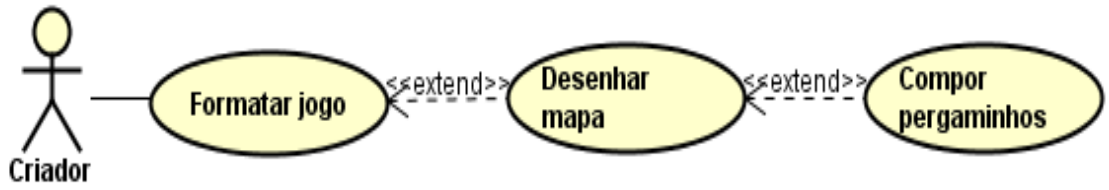
4.2.2 Cadastro

O cadastro de usuário é feito em dois passos: primeiramente, o candidato a membro entra com seus dados cadastrais (nome, e-mail e senha), e uma vez verificado se seu e-mail não foi previamente cadastrado, entra com os dados de seu personagem (nome e classe), como ilustra o diagrama de caso de uso da figura 17.

4.2.3 Criação de jogo

A criação de novos jogos inclui sua formatação, com nome, enigma e dimensões do mapa; o desenho do mapa, com terrenos, cidades, e ponto inicial; e, finalmente, a criação de pergaminhos, com texto e cidade em que cada um deles será vendido, como é ilustrado na figura 18.

Figura 18 – Diagrama de Caso de Uso - Criação de jogo

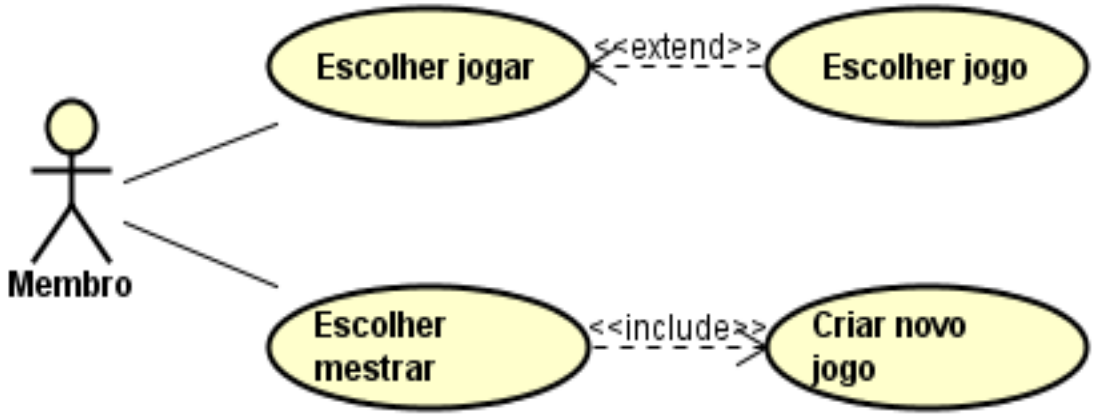


Fonte: Elaborada pelo autor

4.2.4 Escolha de jogo

Como ilustrado na figura 19, o membro terá nesta página as opções de jogar ou mestrar; caso escolha a primeira, será encaminhada para o jogo escolhido; caso decida mestrar, poderá criar um novo jogo.

Figura 19 – Diagrama de Caso de Uso - Escolha de jogo

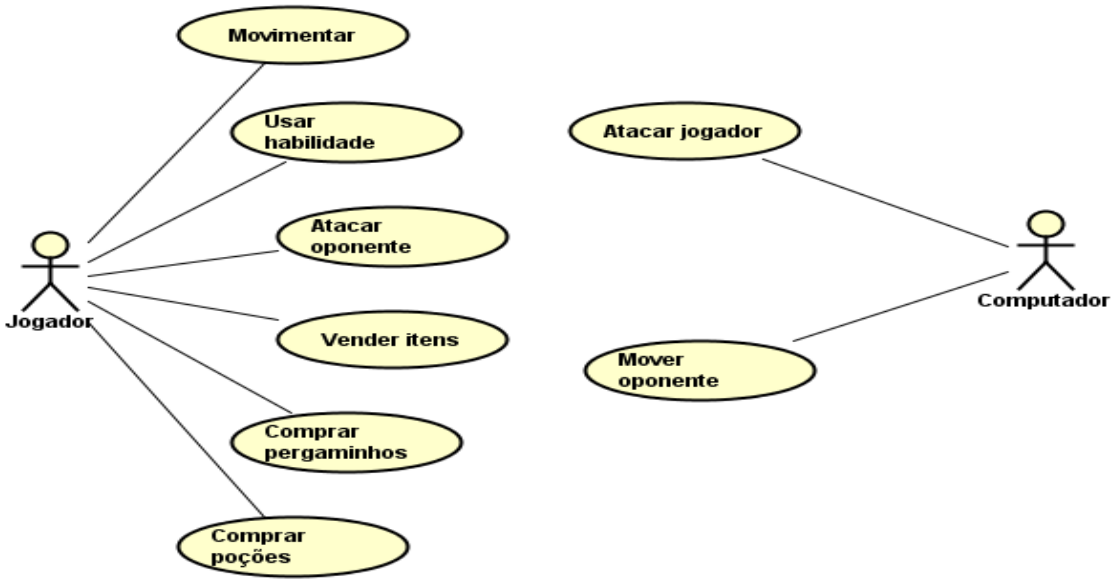


Fonte: Elaborada pelo autor

4.2.5 O jogo

Uma vez iniciado o jogo, a cada turno, como ilustrado no diagrama constante da figura 20, o jogador poderá escolher entre as ações de se movimentar, usar habilidades contra seus inimigos, atacá-los fisicamente, ou interagir com NPCs, quando possível, incluindo as ações de vender itens que obterá ao vencer cada inimigo; comprar pergaminhos; e adquirir poções de vida e mana. Já o computador poderá, a cada turno, mover os oponentes ou atacar o jogador.

Figura 20 – Diagrama de Caso de Uso - O jogo

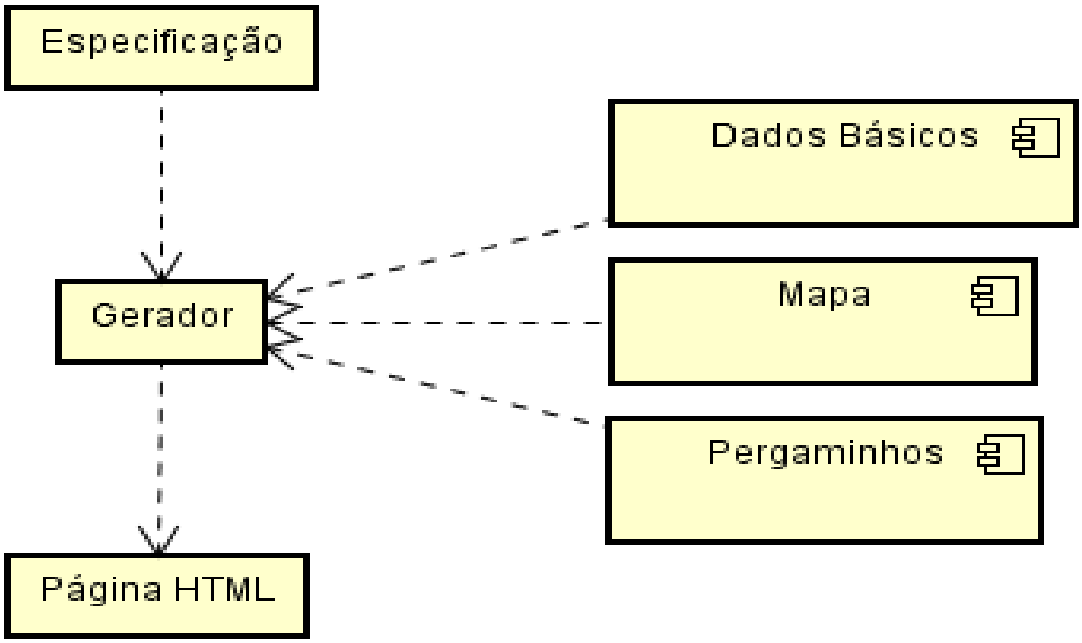


Fonte: Elaborada pelo autor

4.2.6 A Arquitetura Lógica

A modelagem do gerador de aplicações, como ilustrada na figura 21, começa com uma especificação do usuário, feita num documento HTML, enviada ao gerador de aplicação, que se serve de três componentes, para entrada de dados básicos do jogo a ser criado, a saber: título, enigma e dimensões; mapa, onde é desenhada a matriz do mapa a ser usado para o jogo, com terrenos e cidades com respectivos nomes; e finalmente, o componente de pergaminhos, onde são entrados os textos dos pergaminhos de cada cidade do mapa.

Figura 21 – O Gerador de Aplicação



Fonte: Elaborada pelo autor

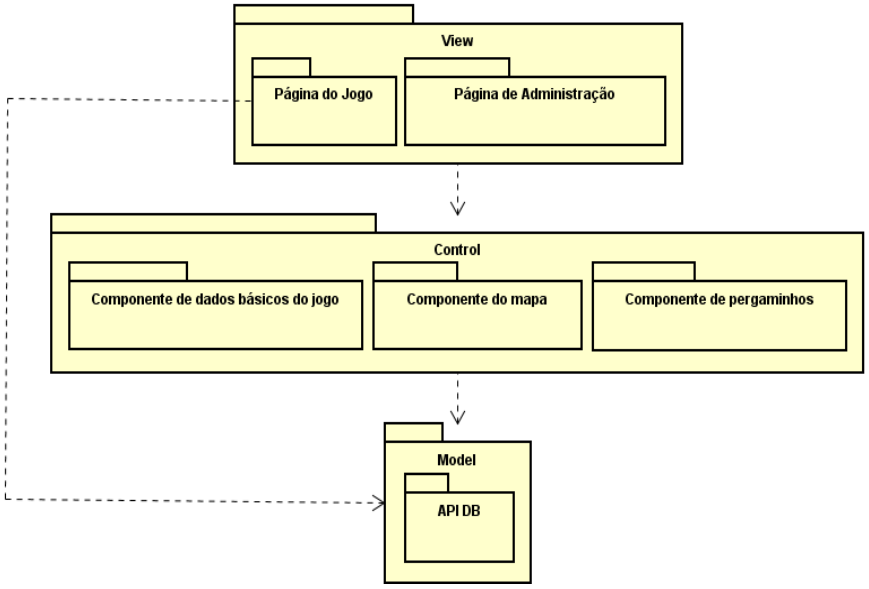
A arquitetura em camadas do gerador foi construída dentro do padrão MVC (*Model View Controller*), onde, de acordo com (CURRY; GRACE, 2008), as partes constantes do projeto são subdivididas em três camadas:

- *Model* Camada de modelo, responsável pela gerência de dados; neste projeto é representada pela API de comunicação com o banco de dados.
- *View* Camada de visão, responsável pela *interface* com o usuário; neste projeto é representada pelas páginas de jogo e de administração.
- *Controller* Camada de controle, se comunica com as outras duas camadas, para o recebimento e envio de comandos de alteração de dados; neste projeto é representada pelos componentes de autenticação de usuário, cadastro de usuário e criação de jogo.

A arquitetura em camadas é ilustrada pelo diagrama de pacotes exibido na figura 22, onde é possível notar que foi adotado o modelo de arquitetura relaxada, onde, de acordo

com (DONOHOE, 2012), a camada de visão se comunica diretamente com a camada de modelo, ao contrário da arquitetura estrita, onde apenas camadas vizinhas se comunicam entre si.

Figura 22 – Arquitetura em Camadas



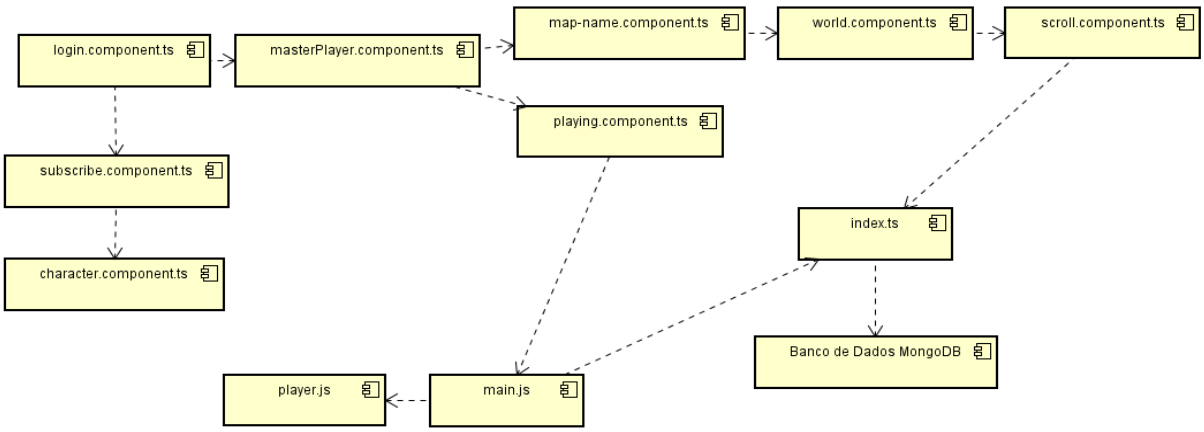
Fonte: Elaborada pelo autor

A arquitetura lógica do sistema é aqui ilustrada no diagrama de componentes constante da figura 23.
A porta de entrada para a ferramenta é o componente login.component.ts.
Nesta página, o usuário tem as opções de se autenticar no sistema ou fazer seu cadastro, posteriormente explanado na sessão 4.3.2.2.
Caso já tenha feito previamente seu cadastro, uma vez autenticado o usuário, a aplicação é redirecionada para o componente masterPlayer.component.ts, onde o membro da comunidade LearnCraft terá a escolha entre jogar e gerenciar jogos:

- Caso escolha jogar, será redirecionado ao arquivo main.js, que contém a implementação do jogo em si.
- Caso escolha gerenciar jogos, será direcionado ao componente map-world.component.ts, onde poderá começar a criar um novo jogo, com nome, enigma e dimensões do mapa; posteriormente será redirecionado ao componente world.component.ts, onde prosseguirá seu trabalho com o desenho do mapa, sendo, por fim, redirecionado ao componente scroll.component.ts, onde completará sua tarefa de criação com a entrada de textos de pergaminhos e respectivas cidades de vendas.

Tanto o jogo quanto a gerência de jogos se servirão de uma API de acesso a banco de dados MongoDB, localizada no arquivo index.ts.

Figura 23 – Diagrama de Componentes



Fonte: Elaborada pelo autor

4.3 Implementação

Uma vez definidas as fases de modelagem e projeto do aplicativo proposto, fica viabilizado o início do processo de implementação. Para tal, faz-se necessária uma reunião do ferramental necessário, e a definição do seu uso, para codificação e criação de elementos gráficos e sonoros a ser usados nos jogos criados com a ferramenta LearnCraft.

4.3.1 Ferramental Utilizado

Nesta sessão, será descrito o ferramental usado para a implementação da LearnCraft, incluindo-se as linguagens, *frameworks*, solução para armazenamento de dados, servidores e ferramentas utilizados.

4.3.1.1 Linguagens

- HTML *HyperText Markup Language* (Linguagem de Marcação de Hipertexto), segundo (W3C, 2014), em sua versão HTML5, oferece funcionalidades para a programação do lado cliente de uma vasta gama de aplicações web. Esta versão, portanto, foi escolhida para a exibição de elementos gráficos. A escolha por esta linguagem se deu também pela sua popularidade e consequente ubiquidade nos ambientes de programação web e *mobile*, o que lhe confere uma substancial oferta de material didático de apoio disponível para consulta na www (*World Wide Web*, a rede mundial de computadores). A obrigatoriedade de interpretação de seu código por quaisquer browsers certificados pelo World Wide Web Consortium (W3C), entidade responsável pela padronização da www, de acordo novamente com (W3C, 2014), também influenciou para a escolha.

- SASS *Syntactically Awesome Stylesheets* [1] Linguagem destinada a adicionar estilo a páginas HTML; de acordo com (SASS, 2017), é derivada da CSS, entretanto adicionando significativas melhores nesta, foi eleita para adicionar estilo (cores, fontes, formatação), na parte servidora da aplicação. Adicionando em relação à sua originadora características da programação orientada a objetos, tais como herança, variáveis e iteração, sua utilização se mostrou conveniente para separar a estilização das páginas, restrita a arquivos SCSS, da exibição dos componentes, localizada em arquivos HTML.

- JavaScript Linguagem de programação, segundo (W3SCHOOLS, 2018), voltada ao desenvolvimento conjunto com HTML, interpretada do lado do cliente, por qualquer navegador certificado pelo W3C, foi escolhida para a codificação da lógica do jogo, por sua nativa integração com o *framework* eleito para sua implementação, o Phaser.

- TypeScript Segundo (RASTOGI et al., 2015), linguagem tipada, desenvolvida e mantida pela Microsoft a partir do JavaScript, sendo caracterizada como um superconjunto desta. Foi utilizada para codificação do módulo de implementação dos jogos e da parte servidora da aplicação, em conjunto com o NodeJS, para o controle da criação de mapas e disponibilização da API de acesso aos dados, respectivamente.

4.3.1.2 *Frameworks*

- Angular[2] De acordo com (RUEBBELKE; FORD, 2015), é um *framework* JavaScript open-source, desenvolvido e mantido pelo Google, orientado ao desenvolvimento de SPA (*Single-Page Applications*), ou seja, aplicações que funcionam numa única página, ao invés de usar redirecionamento entre documentos, com o objetivo de facilitar a escalabilidade de aplicativos que podem ser acessados por um navegador web. Foi escolhido pelas características de sua biblioteca, que, de acordo com a mesma fonte, lê o documento em formato HTML, que contém tags específicos do *framework*; logo após executando a diretiva à qual esta tag pertence, e faz a ligação entre a apresentação e seu modelo, representado por variáveis JavaScript.

- Node.js[3] Foi utilizado como servidor da aplicação. Segundo (CANTELON et al., 2017) foi desenvolvida pelo engenheiro de software norte-americano Ryan Dahl em 2009 e atualmente mantido pela Node.js Foundation, para a execução de código JavaScript do lado servidor, permitindo que conteúdo dinâmico de páginas web seja produzido, antes de ser enviado ao cliente. Código escrito em TypeScript é traduzido em JavaScript para posterior execução, num processo denominado transpilação.

[1] Disponível em https://sass-lang.com/
[2] Disponível em https://angularjs.org/
[3] Disponível em https://nodejs.org/en/

- Phaser[4] Framework, segundo (PHOTONSTORM, 2018), *open source* para desenvolvimento de jogos em HTML5, na linguagem JavaScript, com foco na criação de jogos que rodam em navegadores tanto para mobile quanto desktop. Foi eleita tendo a vista a facilitação de aspectos como renderização de imagens, sprites (contidos em spritesheets) e sons, em formatos como mp3.

4.3.1.3 Armazenamento de dados

MongoDB[5] É uma plataforma de base de dados, segundo (GYŐRÖDI et al., 2015), classificada como NoSQL (dispensa o uso da linguagem de manipulação de bancos de dados relacionais SQL), o que lhe garante uma interação direta, em termos de codificação, com linguagens orientadas a objetos, ao persistir diretamente as variáveis utilizadas em formato JSON (JavaScript Object Notation).

4.3.1.4 Aplicações

- Atom[6] é um ambiente de desenvolvimento, segundo (GITHUB, 2018), desenvolvido pela GitHub Inc, para MacOS, Linux e Windows, dando suporte a um espectro de linguagens que abrange C/C++, C#, CSS, Go, Git, HTML, JavaScript, Java, JSON, Objective-C, PHP, Perl, Python, Ruby on Rails, Ruby, Sass, Shell Script, Scala, SQL, TOML, XML e YAML.

- Paint.net[7] é um aplicativo, de acordo com (DOTPDN, 2018), voltado à edição de imagens, desenvolvido pelo engenheiro de software norte-americano Rick Brewster, e originalmente concebido como uma alternativa gratuita ao Microsoft Paint. Nos dias atuais, é considerado como um concorrente a populares programas de edição de imagens, como Adobe Photoshop, Corel Paint Shop Pro, Microsoft Photo Editor e GIMP.

- Character Generator[8] ferramenta online, que fornece templates para personagens, visualizados em várias perspectivas e realizando diferentes ações, tais como caminhar e atacar, que podem ser personalizados pelo usuário, com adereços como roupas e armas, também disponíveis no site, para criar a aparência das classes e NPCs de seus próprios jogos.

[4] Disponível em https://phaser.io/
[5] Disponível em https://www.mongodb.com/
[6] Disponível em https://atom.io/
[7] Disponível em https://www.getpaint.net/
[8] Disponível em http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator/

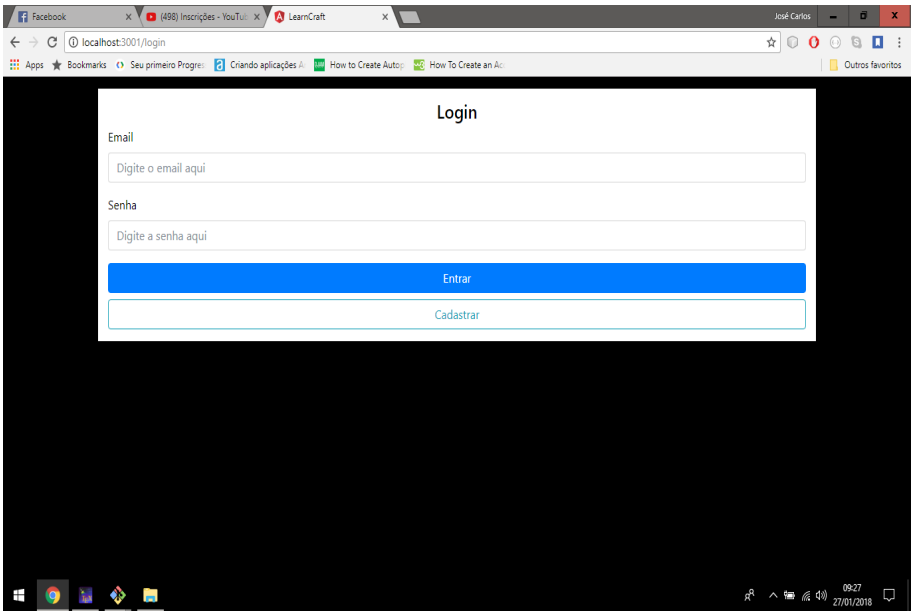
4.3.2 O aplicativo

Nas sessões subsequentes é descrita a implementação das diversas partes do aplicativo LearnCraft, explicitada com trechos de código e ilustrada com o uso de *screenshots* e diagramas de caso de uso.
O código fonte da aplicação, em sua versão completa, encontra-se disponibilizado na URL https://github.com/JCMilito/LearnCraft

4.3.2.1 Autenticação de Usuário

O logon (figura 24), conforme projeto exposto na sessão 4.2.1, é realizado de forma tradicional, com endereço de e-mail e senha de um usuário previamente cadastrado. Na mesma tela, o usuário poderá fazer seu cadastro, que o redirecionará para a página correspondente.

Figura 24 – Logon



Fonte: Elaborada pelo autor

Os dados usados na autenticação são enviados para busca no banco de dados, através da API index.ts. Caso o usuário seja corretamente autenticado, seus dados serão armazenados na área de *LocalStorage* do navegador, e a aplicação será redirecionada para a página de criação/escolha de jogo; caso contrário, um alerta de logon inválido será exibido.
Tal procedimento é responsabilidade da função a seguir, localizada no componente login.component.ts:

```
logon() {
   this.loginService.logon(this.user).subscribe(
   data => {
      if (data) {
```

```
            localStorage.setItem('user', JSON.stringify(data));
            this.route.navigate(['/playermaster']);
        } else {
            alert('Logon inválido!');
        }
    },
    error => console.log(error));
}
```

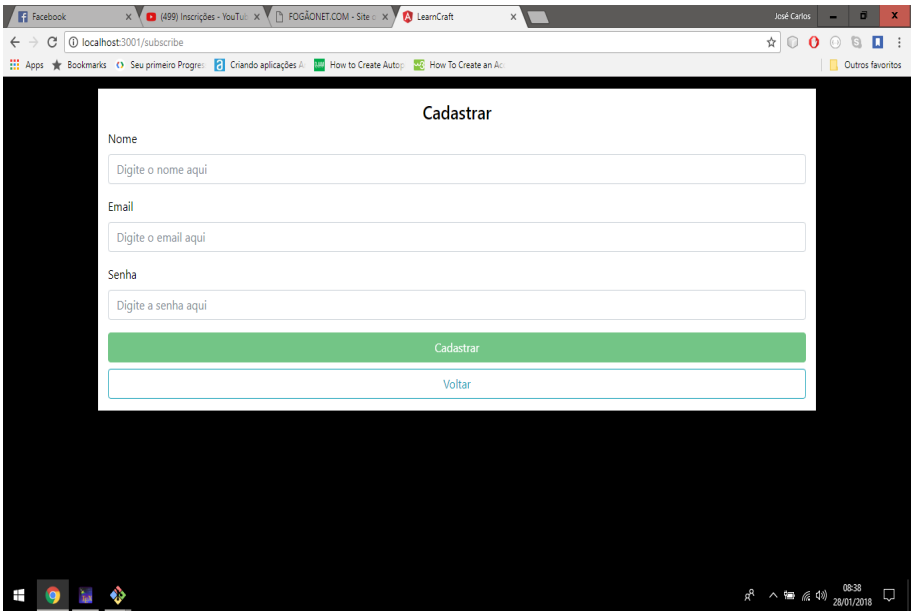
4.3.2.2 Cadastro

Como explanado na sessão 4.2.2, o cadastro é subdividido em entrada de dados do usuário e criação do personagem.

4.3.2.2.1 Dados do Usuário

Para realizar seu cadastro, o usuário deverá digitar seu nome, endereço de e-mail e senha (figura 25).

Figura 25 – Dados do Usuário



Fonte: Elaborada pelo autor

Os dados digitados são enviados para a API responsável, que, após verificar que o e-mail digitado não tenha sido registrado em outro usuário anteriormente, os incluirá como um novo documento no banco de dados. A seguir, seus dados serão armazenados na área de *LocalStorage* do navegador, na forma do item UserIncomplete, e a aplicação será redirecionada para a tela de criação do personagem.

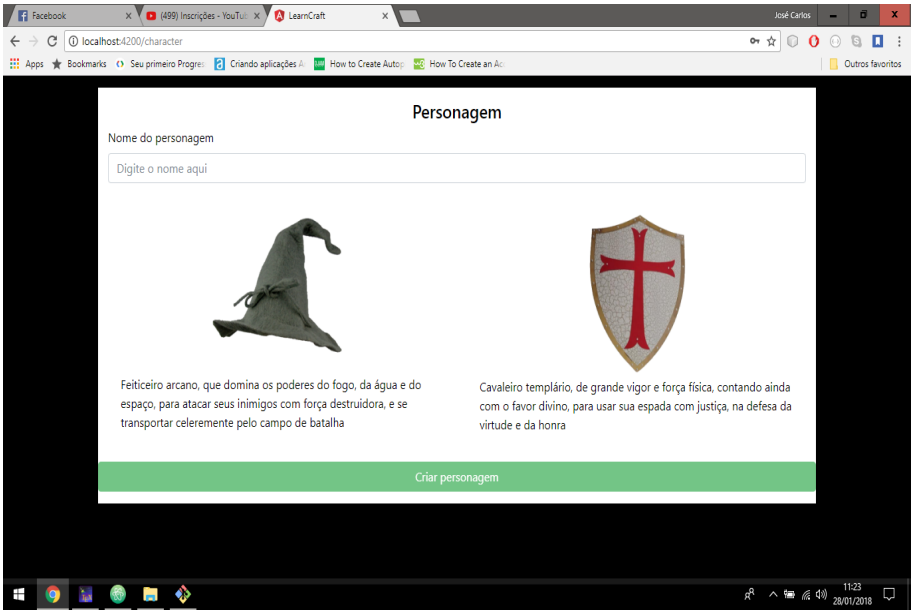
A função create(), localizada no componente subscribe.component.ts, é ilustrada a seguir:

```
create() {
   localStorage.setItem('userIncomplete', JSON.stringify(this.user));
   this.route.navigate(['/character']);
}
```

4.3.2.2.2 Criação de Personagem

Os dados do personagem consistem de seu nome e classe (mago ou cavaleiro, conforme delineado na sessão 4.1.4.3) (figura 26).

Figura 26 – Criação de Personagem



Fonte: Elaborada pelo autor

Os nome e a classe são enviados para a API, que os incluirá no banco de dados, concluindo, assim, o documento correpondente ao novo usuário no banco de dados. A seguir, a aplicação será redirecionada para a rota /login. A função a realizar este processo é create(), localizada no componente character.component.ts:

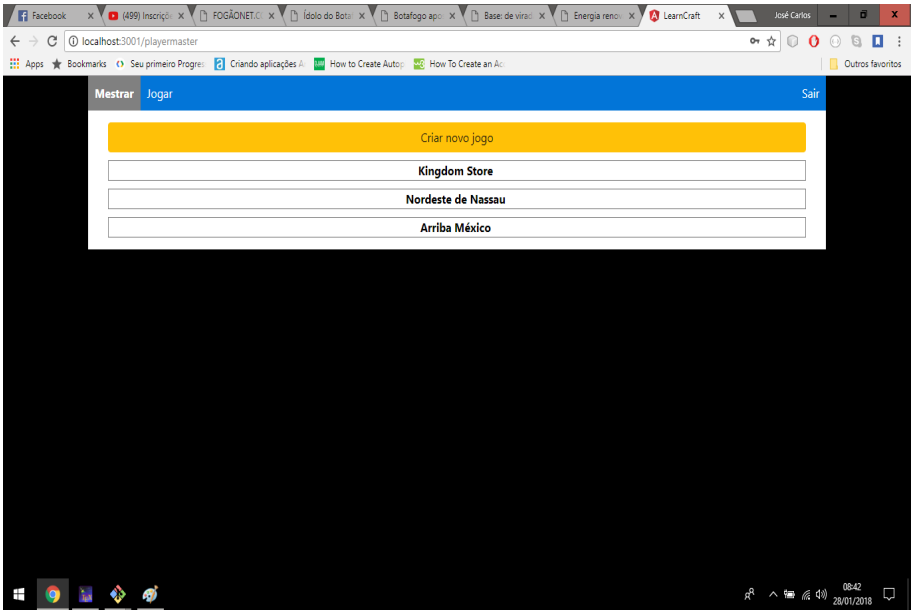
```
create() {
   let user = JSON.parse(localStorage.getItem('userIncomplete'));
   user.character = this.character;
   this.loginService.create(user).subscribe (
   data => {
```

```
        console.log(data)
        alert('Criado com sucesso!')
        this.route.navigate(['/login'])
    },
    error => console.log(error));
}
```

4.3.2.3 Criação de jogo

A criação de um jogo novo (figura 27) começa com um botão presente na aba Mestrar, na tela oferecida ao usuário uma vez que este se autentique no sistema.

Figura 27 – Criação de Jogo



Fonte: Elaborada pelo autor

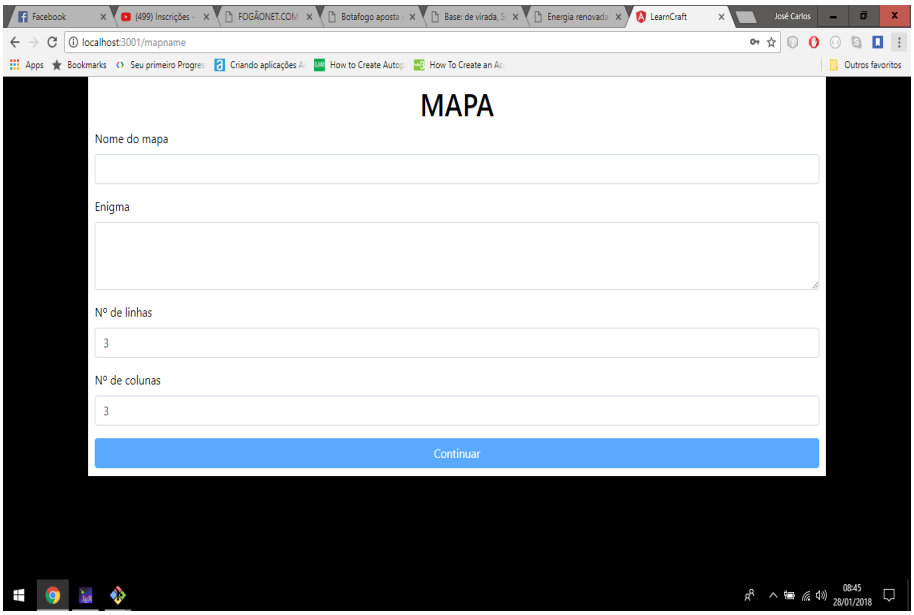
Na mesma tela, será exibida uma listagem dos jogos previamente cadastrados, realizada pela seguinte função, contida no componente masterPlayer.component.ts:

```
list() {
    this.masterService.list(this.user._id)
    .subscribe(data => {
        console.log(data.data)
        this.games = data.data;
    },
    error => console.log(error));

}
```

4.3.2.3.1 Formatação de Jogo

Na formatação do jogo (figura 28), são digitados seu nome, seu enigma e as medidas do mapa onde este se desenrolará.

Figura 28 – Formatação de Jogo



Fonte: Elaborada pelo autor

De forma análoga aos passos anteriores, os dados entrados são enviados para a API correspondente, exceto caso alguma das dimensões do mapa seja inferior a 2 ou superior a 15, quando o usuário será notificado. A função responsável por este passo na criação do jogo está localizada no componente map-name.component.ts:

```
save() {
   if ((this.map.cols < 16 && this.map.cols > 2) && (this.map.rows < 16
   && this.map.rows > 2)) {
      localStorage.setItem('map', JSON.stringify(this.map));
      this.router.navigate(['world']);
   } else {
      alert('Minimo de linhas e colunas 3x3 e máximo 30x30');
   }
}
```

4.3.2.3.2 Desenho de Mapa

Para realizar o desenho do mapa (figura 29), o usuário terá à sua disposição um matriz com as dimensões eleitas no passo anterior, a qual deverá preencher com terrenos (água,

deserto, campo e floresta) e cidades, que deverão ser nomeadas. Uma vez completo o mapa, deverá ser escolhida a casa da matriz onde o jogador iniciará seu jogo.

Figura 29 – Desenho de Mapa

Fonte: Elaborada pelo autor

O método confirm(), presente no componente world.component.ts, realizará o envio das características finais do mapa para o banco de dados, após verificar que a localização inicial do jogador é válida, não estando numa casa de água dentro da matriz:

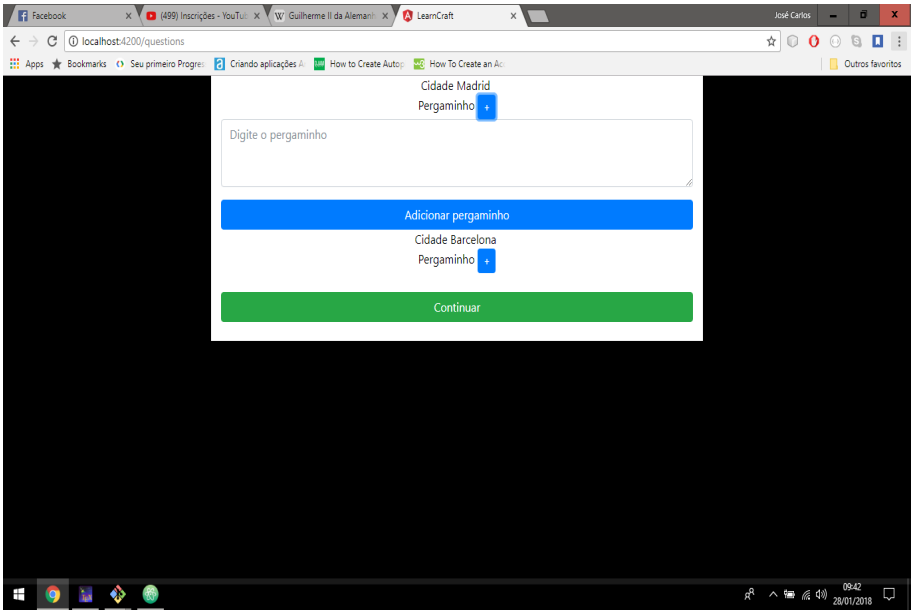
```
confirm() {
   console.log(this.mark);
   if (this.mark.col != null) {
      if (this.finalMatrix[this.mark.row][this.mark.col] != 'water') {

         let world = {
            numberOfColumns: this.map.cols,
            numberOfRows: this.map.rows,
            contents: this.finalMatrix,
            towns: this.towns,
            startPoint:{
               column: this.mark.col,
               row: this.mark.row,
            }
         }
         localStorage.setItem('world', JSON.stringify(world));
         this.route.navigate(['/questions']);
```

```
        } else {
            alert('O jogo não pode ser iniciado na água');
        }
    } else {
        alert('Escolha um local para iniciar');
    }
}
```

4.3.2.3.3 Criação de Pergaminhos

A criação de pergaminhos (figura 30) é realizada ao se escolher uma cidade, apertar o botão de + correspondente a esta, e digitar o conteúdo do novo pergaminho, a ser adquirido pelo jogador junto ao npc correspondente, habitante daquela cidade. Ao apertar o botão Adicionar Pergaminho, o usuário confirmará a entrada deste.

Figura 30 – Criação de Pergaminhos



Fonte: Elaborada pelo autor

O método finish(), localizado no componente scroll.component.ts, conclui a autoria do novo jogo, enviando as características do mapa para o banco de dados e retornando o usuário para a rota /playermaster, onde este poderá criar outros mapas ou jogar algum mod previamente constante do banco de dados da aplicação:

```
finish() {
    this.world.map = JSON.parse(localStorage.map);
    let finish = {
        name: this.world.map.name,
        subject: this.world.map.subject,
```

```
      numberOfColumns: this.world.numberOfColumns,
      numberOfRows: this.world.numberOfRows,
      contents: this.world.contents,
      towns: this.world.towns,
      idMaster: JSON.parse(localStorage.getItem('user'))._id,
      startPoint: {
         column: this.world.startPoint.column,
         row: this.world.startPoint.row
      },
      startGame: true
   }
   localStorage.setItem('finish', JSON.stringify(finish));
   this.masterService.create(finish)
   .subscribe(
   data => {
      alert('Jogo criado com sucesso!');
      this.route.navigate(['/playermaster'])
   },
   error => console.log('data' + error));
}
```

4.3.2.4 Escolha de Jogo

A escolha do mod o qual o usuário deseja jogar (figura 31) é feita com um botão com a designação do jogo desejado, presente na aba Jogar, na tela oferecida ao usuário uma vez que este se autentique no sistema. De forma análoga, caso o membro da comunidade deseje criar ou editar um mod, este terá à sua disposição na aba correspondente.

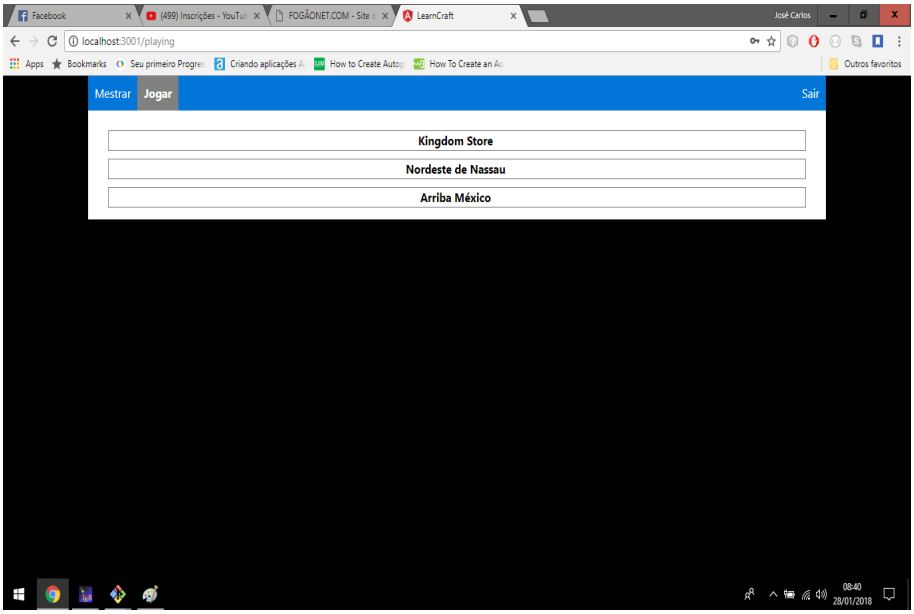
O método getData(), localizado no componente playing.component.ts, recebe os dados do jogo eleito dentre a lista de mods, os envia à área de Local Storage do navegador, e redireciona o membro à página de jogo, na rota "/gametime":

```
getData(data){
   window.location.replace('http://localhost:3001/gametime');
   localStorage.setItem('gameData', JSON.stringify(data));
   localStorage.setItem('startPlaying', 'true');
}
```

4.3.2.5 O jogo

Nesta sessão, será descrita a implementação do jogo, com detalhes concernentes à interface gráfica, os terrenos e cidades por onde o personagem do jogador estará apto a

Figura 31 – Escolha de Jogo



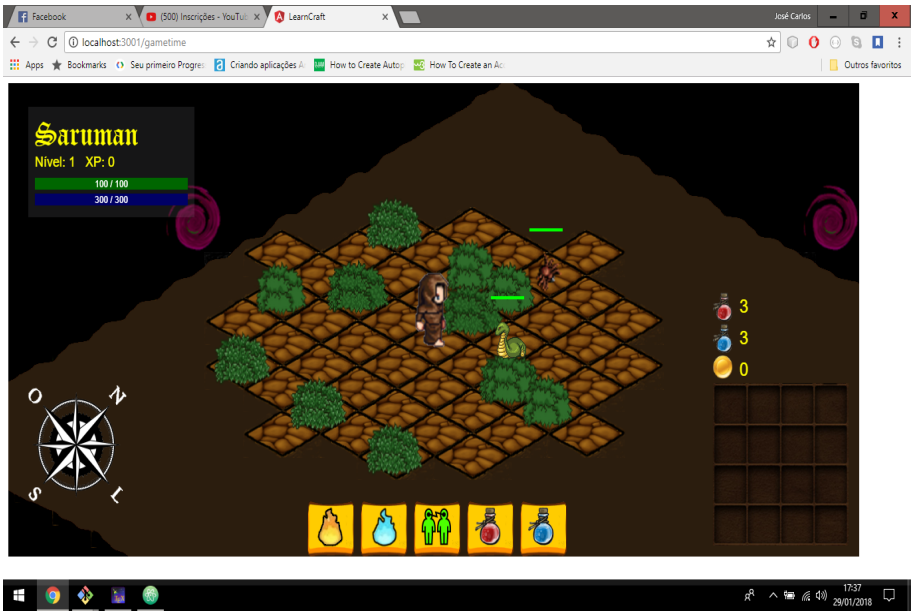
Fonte: Elaborada pelo autor

percorrer durante uma partida.

4.3.2.5.1 Terrenos

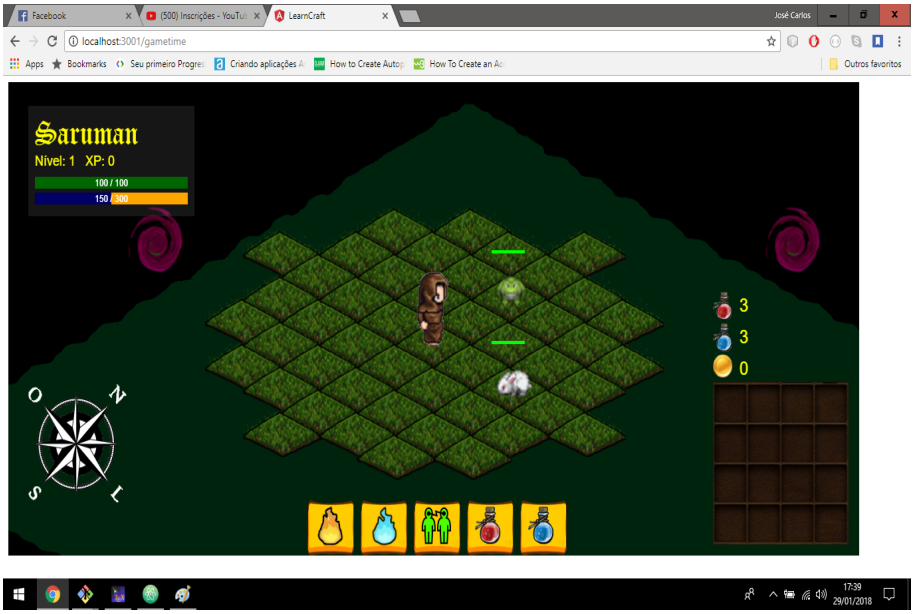
Neste parágrafo, conforme descritos na sessão 4.1.2, são ilustrados por *screenshots* os terrenos por onde os personagens poderão trafegar. A floresta é ilustrada na figura 32. O campo, na figura 33. E finalmente, o deserto, na figura 34.

Figura 32 – Floresta



Fonte: Elaborada pelo autor

Figura 33 – Campo



Fonte: Elaborada pelo autor

Figura 34 – Deserto

Fonte: Elaborada pelo autor

4.3.2.5.2 Cidade

As cidades (figura 35) são terrenos livres de inimigos, onde o jogador poderá recuperar seus pontos de vida e mana. Terá ainda à sua disposição três NPCs, cada um prestando um serviço, como descrito na sessão 4.1.4.1.

Figura 35 – Cidade

Fonte: Elaborada pelo autor

4.3.2.5.3 Interface

Neste parágrafo, será descrita a GUI que informará ao jogador o estado atual de seu personagem dentro do jogo.

Status O *status* atual do jogador é representado por seu nome, seus pontos de vida e mana, podendo ser visualizado no canto superior esquerdo das figuras 32 a 35 deste trabalho. A função que mostra na tela o status do jogador está localizada no arquivo main.js da aplicação:

```
function createStatusPanel() {
    let bmd = game.add.bitmapData(250, 140);
    bmd.ctx.fillStyle = "#e0e0eb";
    bmd.ctx.beginPath();
    bmd.ctx.rect(0, 0, 250, 140);
    bmd.ctx.fill();
    panelPlayer = game.add.sprite(30, 30, bmd);
    panelPlayer.alpha = 0.1;
    panelPlayer.fixedToCamera = true;

    playerNameText = game.add.text(40, 40, player.name, {
        font: "40px Old English Text MT",
        fill: "#FFFF00"
    });
```

```
    playerNameText.fixedToCamera = true;

    playerHealthBar = new PlayerBar(40, 120, "#660000", "#006600");
    playerManaBar = new PlayerBar(40, 140, "#FFA500", "#000066");
    playerHPText = game.add.text(155, 120, player.hp + " / " + player.maxHp, {
        font: "12px Arial",
        fill: "#FFF"
    });
    playerHPText.anchor.set(0.5, 0);
    playerHPText.fixedToCamera = true;
    playerMPText = game.add.text(155, 140, player.mp + " / " + player.maxMp, {
        font: "12px Arial",
        fill: "#FFF"
    });
    playerMPText.anchor.set(0.5, 0);
    playerMPText.fixedToCamera = true;
}
```

Inventário No inventário, são representados a quantidade de poções de vida (vermelha) e mana (azul), bem como a quantidade de moedas de ouro, à disposição do jogador. Numa matriz de 4x4 são exibidos, também, os itens obtidos de oponentes derrotados. Pode ser visualizado no canto inferior direito das figuras 32 a 35 deste trabalho. A função que realiza a construção do inventário está localizada no arquivo player.js:

```
createBackpack() {
    this.backpack = [];
    let backpack = game.add.sprite(1060, 380, "backpack");
    backpack.fixedToCamera = true;

    let gold = game.add.sprite(1060, 345, "coin");
    gold.fixedToCamera = true;
    this.goldText = game.add.text(1100, 350, this.gold.toString(), {
        font: "24px Arial",
        fill: "#FFFF00"
    });
    this.goldText.fixedToCamera = true;
    let mp = game.add.sprite(1060, 305, "mp");
    mp.fixedToCamera = true;
    this.mpText = game.add.text(1100, 310, this.mpPotion.toString(), {
        font: "24px Arial",
```

```
      fill: "#FFFF00"
   });
   this.mpText.fixedToCamera = true;
   let hp = game.add.sprite(1060, 265, "hp");
   hp.fixedToCamera = true;
   this.hpText = game.add.text(1100, 270, this.hpPotion.toString(), {
      font: "24px Arial",
      fill: "#FFFF00"
   });
   this.hpText.fixedToCamera = true;
}
```

Habilidades e Poções Habilidades e poções podem ser acessadas pelos botões, que podem ser visualizados no canto inferior central das figuras 32 a 35 deste trabalho, ou pelas teclas de 1 a 5. O código responsável por adicionar à interface os botões está no corpo principal do arquivo main.js:

```
button1 = game.add.button(450, 530, "buttonFireball", fireball, this, 2, 1, 0);
button2 = game.add.button(530, 530, "buttonIceball", iceball, this, 2, 1, 0);
button3 = game.add.button(610, 530, "buttonTeleport", teleport, this, 2, 1, 0);
button4 = game.add.button(690, 530, "buttonHp", drinkHpPotion, this, 2, 1, 0);
button5 = game.add.button(770, 530, "buttonMp", drinkMpPotion, this, 2, 1, 0);
button1.fixedToCamera = true;
button2.fixedToCamera = true;
button3.fixedToCamera = true;
button4.fixedToCamera = true;
button5.fixedToCamera = true;
```

Mapa Ao pressionar a tecla M, o jogador terá à sua disposição o mapa (figura 36), que traz uma representação gráfica da matriz do jogo, com localizações de terrenos, cidades com respectivos nomes e a localização atual do jogador. A função que mostra na tela o mapa está localizada no arquivo main.js:

```
function showMap() {
   player.turn = "mapCheck";
   mapScroll = game.add.sprite(50, -100, "mapScroll");
   mapScroll.fixedToCamera = true;
   mapContents = [];
   for (let i = 0; i < world.numberOfRows; i++) {
      for (let j = 0; j < world.numberOfColumns; j++) {
```

Figura 36 – Mapa

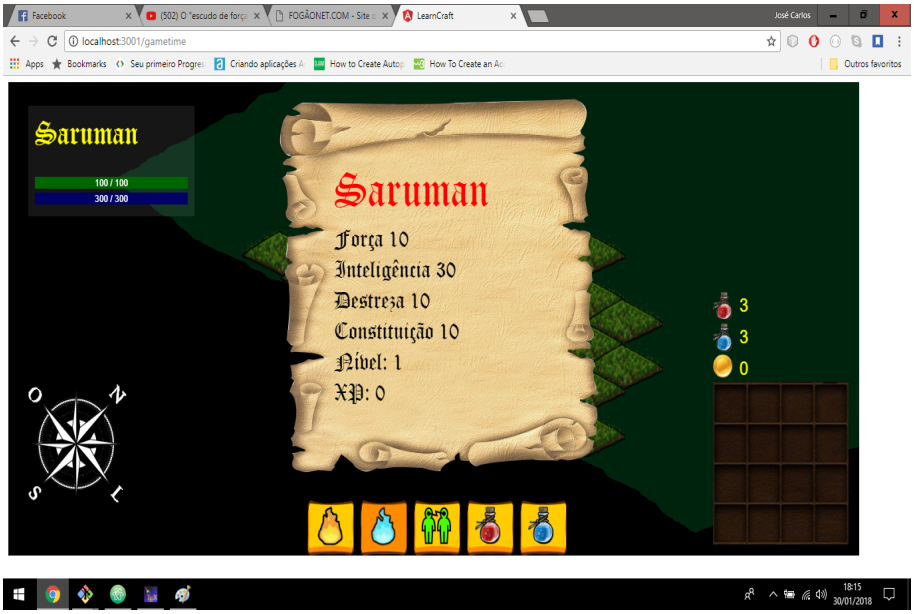
Fonte: Elaborada pelo autor

```
        mapContents.push(game.add.sprite(200 + j * 36, 20 + i * 36, world.squareCo
        mapContents[i * world.numberOfColumns + j].fixedToCamera = true;
    }
}
let listOfTowns = "";
cityNumbers = [];
for (var i = 0; i < world.towns.length; i++) {
    let town = world.towns[i];
    listOfTowns += (i + 1) + ". " + town.name + "\n";
    cityNumbers.push(game.add.text(217 + town.col * 36, 20 + town.row * 36, (i +
        font: "30px Old English Text MT",
        fill: "#FF0"
    }));
    cityNumbers[i].fixedToCamera = true;
    cityNumbers[i].anchor.set(0.5,0);
}
textCities = game.add.text(760, 20, listOfTowns, {
    font: "40px Old English Text MT",
    fill: "#000"
});
textCities.fixedToCamera = true;
marker = game.add.sprite(202 + player.col * 36, 15 + player.row * 36, "x");
marker.fixedToCamera = true;
```

```
    marker.alpha = 0;
    game.add.tween(marker).to( { alpha: 1 }, 1000, Phaser.Easing.Linear.None, true,
}
```

Atributos Ao pressionar a tecla C, são exibidos os atributos atuais do jogador (figura 37), representados por força, inteligência, constituição e destreza, com seus respectivos valores, além de seu nível e pontos de experiência. A função showPlayerScroll(), que

Figura 37 – Atributos



Fonte: Elaborada pelo autor

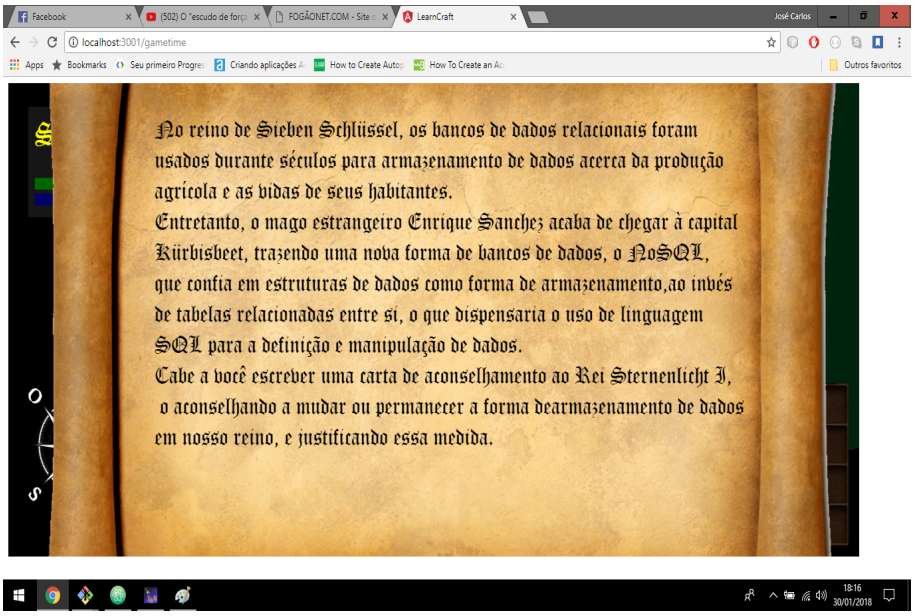
mostra na tela tais informações, está localizada no arquivo main.js:

```
function showPlayerScroll() {
    player.turn = "characterCheck";
    playerScroll = game.add.sprite(400, 10, "playerScroll");
    playerScroll.fixedToCamera = true;
    playerDataText.name = game.add.text(490, 100, player.name, {
        font: "60px Old English Text MT",
        fill: "#F00"
    });
    playerDataText.name.fixedToCamera = true;
    playerDataText.attributes = game.add.text(490, 180,
    "Força " + player.strength +
    "\nInteligência " + player.intelligence +
    "\nDestreza " + player.dexterity +
    "\nConstituição " + player.constitution +
        "\nNível: " + player.level +
```

```
        "\nXP: " + player.xp, {
            font: "30px Old English Text MT",
            fill: "#000"
        }
    );
    playerDataText.attributes.fixedToCamera = true;
}
```

Enigma Pressionando a tecla E, é exibido o enigma proposto para o jogo atual (figura 38). A função showEnigma(), que mostra na tela o enigma proposto ao jogador,

Figura 38 – Enigma



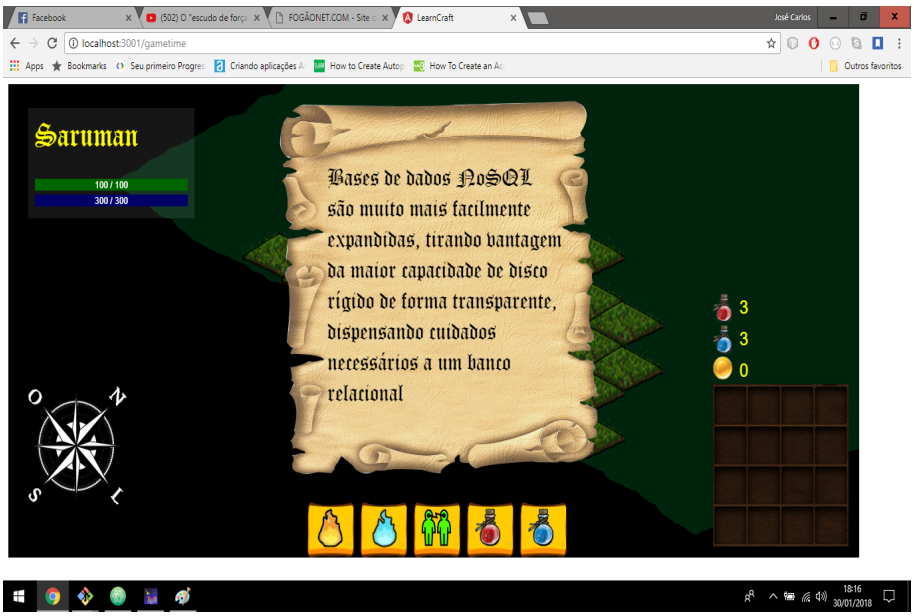
Fonte: Elaborada pelo autor

está localizada no arquivo main.js:

```
function showEnigma() {
    player.turn = "enigmaCheck";
    enigmaScroll = game.add.sprite(50, -100, "mapScroll");
    enigmaScroll.fixedToCamera = true;
    enigmaText = game.add.text(220, 40, enigma
    , {
        font: "30px Old English Text MT",
        fill: "#000"
    });
    enigmaText.fixedToCamera = true;
}
```

Pergaminhos Ao pressionar a tecla P, o jogador terá à sua disposição os pergaminhos previamente adquiridos (figura 39). Pressionando as teclas < e >, este poderá navegar entre eles. A função showHint(), que mostra na tela os pergaminhos pertencentes ao

Figura 39 – Pergaminhos



Fonte: Elaborada pelo autor

jogador, está localizada no arquivo main.js:

```
function showHint() {
   player.turn = "hintCheck";
   hintScroll = game.add.sprite(400, 10, "playerScroll");
   hintScroll.fixedToCamera = true;
   hintText = game.add.text(480, 100, hints[0],
      fill: "#000"
   });
   hintText.fixedToCamera = true;
}
```

5 ESTUDO DE CASO

Neste capítulo, será descrito um estudo de caso, que consistiu de um desafio em formato de enigma, proposto a um grupo de alunos de nível superior.

5.1 O Jogo

O mod submetido ao estudo é denominado Kingdom Store, tem como área específica Bancos de Dados, e seu mapa está ilustrado na figura 36 deste trabalho.
Este tem como enigma o texto a seguir: "No reino de Sieben Schlüssel, os bancos de dados relacionais foram usados durante séculos, para armazenamento de dados acerca da produção agrícola e as vidas de seus habitantes. Entretanto, o mago estrangeiro Enrique Sanchez acaba de chegar à capital Kürbisbeet, trazendo uma nova forma de bancos de dados, o NoSQL, que confia em estruturas de dados como forma de armazenamento, ao invés de tabelas relacionadas entre si, o que dispensaria o uso de linguagem SQL para a definição e manipulação de dados. Cabe a você escrever uma carta de aconselhamento ao Rei Sternenlicht I, o aconselhando a mudar ou permanecer a forma de armazenamento de dados em nosso reino, e justificando essa medida".
As cidades do mapa são a progressista capital portuária Kürbisbeet, entusiasta das novas tecnologias; e a conservadora AlteLeute, situada em meio ao deserto sul do reino de Sieben Schlüssel.
Os pergaminhos representantes da cultura de Kürbisbeet são assim escritos:

- Bases de dados NoSQL são muito mais facilmente expandidas, tirando vantagem da maior capacidade de disco rígido de forma transparente, dispensando cuidados necessários a um banco relacional (HAN et al., 2011).

- Administradores de bancos (DBAs) não se fazem mais necessários para o projeto e instalação de bases de dados, podendo ser essas funções executadas pelos programadores (HAN et al., 2011).

- Restrições antes inevitáveis, no caso do uso de bancos de dados relacionais, para a relação entre tabelas, não existem mais em bancos NoSQL, com os dados podendo ser armazenados de forma mais flexível(HAN et al., 2011).

Em contraste, os pergaminhos à venda em AlteLeute assim advogam:

- Bancos relacionais estão num estado muito mais maduro, proporcionado pelas décadas de seu uso em grandes e pequenas corporações; bancos NoSQL ainda estão em seu início (HAN et al., 2011).

- Existem literalmente milhões de desenvolvedores familiares com bancos relacionais, enquanto para bancos NoSQL, a comunidade apenas começa a florescer (HAN et al., 2011).

5.2 A Atividade

A atividade em questão é classificada como quantitativa-qualitativa; quantitativa, pois envolve a quantificação das relações entre variáveis, neste caso, as respostas dos alunos e respectivas avaliações de um especialista (SOUSA; DRIESSNACK; MENDES, 2007); e qualitativa, pois envolve questões abertas, que permitem aos respondentes expressar seus pensamentos de forma livre, sem as amarras de estruturas confinadas, como múltipla escolha (BOGDAN; BIKLEN, 1992).
Um número de dez alunos de uma turma, discente de um curso de ciência da computação, foi dividido em dois grupos de cinco, onde exclusivamente o primeiro grupo jogou o Kingdom Store. Todos os participantes da atividade compartilharam o mesmo laboratório, com acesso à Internet. Cabe também destacar que o grupo a não jogar teve à sua disposição um arquivo texto, com os conteúdos dos pergaminhos constantes do jogo. A atividade teve a duração de duas horas e foi estritamente individual, sem compartilhamento de texto entre os alunos.
Finalmente, os textos obtidos da atividade, disponibilizados no apêndice A do presente trabalho, foram submetidos a um especialista da área, que ficaria responsável por tecer comentários acerca de cada um destes, e atribuir-lhes uma nota final, entre 0 e 10.
Os critérios utilizados para a obtenção de cada nota foram:

- Clareza: uso denotativo da linguagem, evitando ambiguidades que venham a dificultar a compreensão do texto;
- Conhecimento: o domínio do assunto abordado na atividade;
- Concisão: manutenção do foco no assunto abordado, evitando termos e frases desnecessários;
- Coerência: relacionamento adequado entre as frases, sem contradição entre as partes do texto
- Argumentação: capacidade do texto de persuadir o leitor a reconhecer sua validade.

5.3 Resultados

Nesta sessão, os resultados dos dois grupos serão expostos e comparados entre si. Foi estabelecida a nota 5,0 como mínima para aprovação do texto produzido, de acordo com o regimento estabelecido na instituição onde foi realizada a experiência.

5.3.1 Jogantes

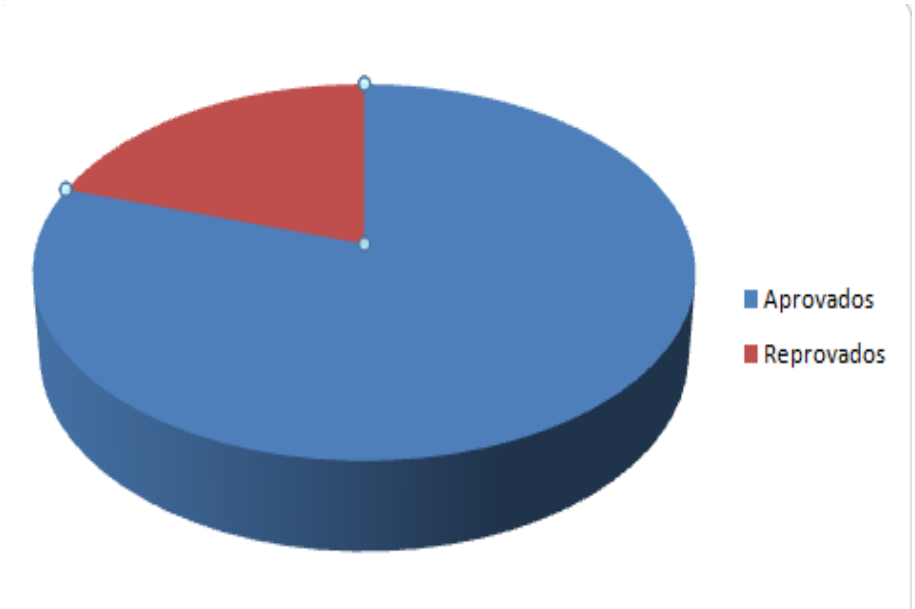
Os textos dos alunos jogantes obtiveram, junto ao especialista designado, uma nota 4,0; três notas 7,0; e uma nota 10,0; a média dentre os participantes que jogaram o mod proposto, portanto, foi de 7,0.
A tabela seguinte mostra as pontuações de cada trabalho dos participantes jogantes, em cada um dos atributos, bem como suas notas finais, obtidas pelas somas das notas em cada critério:

Tabela 2 – Trabalhos e respectivas notas dos jogantes

Trabalho	Clareza	Conhecimento	Concisão	Coerência	Argumentação	Total
1	2,0	2,0	2,0	2,0	2,0	10,0
2	0,0	2,0	2,0	1,0	2,0	7,0
3	1,0	1,0	2,0	2,0	1,0	7,0
4	0,0	2,0	2,0	2,0	1,0	7,0
5	0,0	1,0	2,0	1,0	0,0	4,0

A proporção entre aprovados, neste caso, 80% e reprovados, aqui de 20%, é ilustrada no gráfico constante da figura 40.
Assim se configuraram as considerações tecidas pelo especialista, acerca de cada texto:

Figura 40 – Resultados dos Participantes Jogantes



Fonte: Elaborada pelo autor

- No tocante ao texto a obter a nota 10,0, o aluno realçou de forma adequada o valor das características do NoSQL;
- Com relação aos textos a obter a nota 7,0, os alunos mostraram, ainda que de forma incompleta, argumentos adequados para a transição ao NoSQL;

- Quanto ao texto do aluno jogante a obter a nota 4,0, o aluno destaca os benefícios do SQL, porém aconselha mudar para uma nova arquitetura, tornando o texto confuso.

5.3.2 Não Jogantes

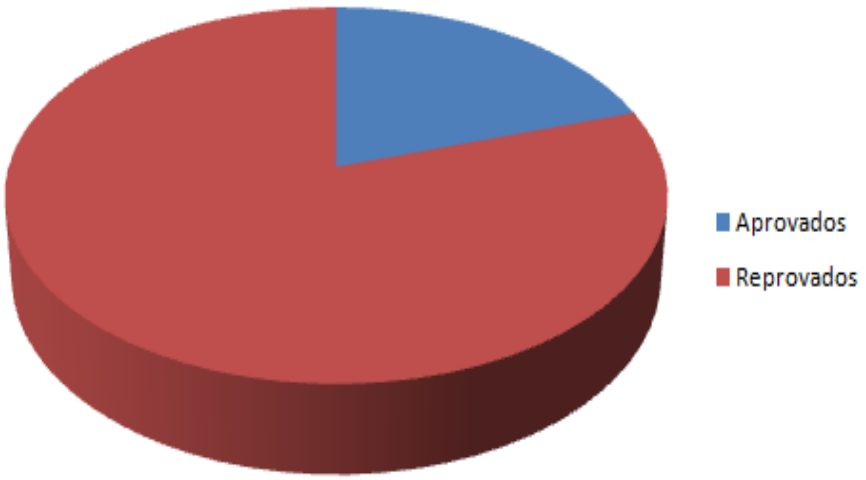
Em contraste, dentre os restantes participantes da atividade, as notas obtidas foram: quatro notas 4,0 e uma nota 7,0; tais desempenhos conferem, de acordo com o mesmo especialista, aos textos destes participantes, uma média de 4,6.
A tabela seguinte mostra as pontuações de cada trabalho dos participantes não jogantes, em cada um dos atributos, bem como suas notas finais, obtidas pelas somas das notas em cada critério:

Tabela 3 – Trabalhos e respectivas notas dos não jogantes

Trabalho	Clareza	Conhecimento	Concisão	Coerência	Argumentação	Total
1	2,0	2,0	1,0	1,0	1,0	7,0
2	2,0	1,0	0,0	1,0	0,0	4,0
3	1,0	1,0	2,0	0,0	0,0	4,0
4	0,0	2,0	1,0	0,0	1,0	4,0
5	1,0	1,0	2,0	0,0	0,0	4,0

É ilustrada no gráfico constante da figura 41, a proporção entre aprovados (20%)e reprovados (80%):

Figura 41 – Resultados dos Participantes Não Jogantes



Fonte: Elaborada pelo autor

Com relação aos textos produzidos por não jogantes, estas foram as considerações do especialista:

- Com relação ao texto a obter a nota 7,0, o aluno mostrou prudência em seus argumentos a permanecer com a forma atual de armazenamento de dados;
- Quanto aos demais textos, todos com nota 4,0, o consenso foi de que aluno não demonstrou argumentos fortes para apoiar a sua decisão.

5.4 Considerações acerca dos resultados

Assim, os resultados obtidos da pesquisa apontam para a comunhão de ideias com (MCGONIGAL, 2011), quando a autora afirma que comparada a jogos, a realidade é entediante; jogos nos desafiam com obstáculos voluntários, nos ajudando a tirar o melhor de nossas forças pessoais.
Tais obstáculos voluntários, aqui representados pelos desafios dos mods criados com LearnCraft, ao ressignificar nas mentes do alunado, a informação adquirida em sala de aula, têm se mostrado um significativo aditivo às ideias construcionistas em particular, e arautos de uma nova era na relação entre aprendiz e conteúdo programático, em geral.

6 CONCLUSÃO

Neste capítulo serão discutidas as considerações finais do trabalho desenvolvido, sua contribuição para a comunidade científica, resultados obtidos com a pesquisa, bem como eventuais trabalhos futuros.

6.1 Considerações Finais

O trabalho apresentado nesta dissertação teve por objetivo a modelagem e implementação de uma *engine* para jogos construcionistas do gênero RPG, com o desenvolvimento de um jogo, junto a sua aplicação em sala de aula, com o objetivo de validar seu uso e testar sua aplicabilidade junto ao público alvo. A *engine* LearnCraft objetiva, por meio de uma *interface* gráfica, viabilizar o desenvolvimento de mods por membros de corpos docentes e discentes, ainda que estes não tenham conhecimento de linguagens de programação. Por meio dos resultados obtidos no estudo de caso ficou patente uma sensível diferença positiva dentre aqueles que usaram a ferramenta, em oposição aos que não lançaram mão do uso desta, no desempenho de atividades propostas ao alunado, segundo avaliação de especialista na área.

6.2 Contribuição

As contribuições advindas da elaboração deste trabalho são elencadas a seguir:

- A partir da proposta de uso do software implementado neste trabalho, se abrem novas perspectivas para o compartilhamento de informações em sala de aula.
- Se tratando de uma aplicação web escrita em JavaScript, sua portabilidade para dispositivos móveis está viabilizada, oportunizando, assim, a capitalização do uso das referidas plataformas em sala de aula; sendo estas particularmente populares entre as últimas gerações de alunos, é fornecido, assim, um significativo subsídio para a construção de conhecimento.
- A democratização no tocante à elaboração de atividades, no formato de mods, fica viabilizada, tanto pela inclusão de membros sem conhecimentos de linguagem de programação como potenciais criadores de jogos, como pela possibilidade de tanto professores quanto alunos atuarem como criadores e jogadores dentro da comunidade LearnCraft.

6.3 Resultados Obtidos

O estudo realizado viabilizou obter os seguintes resultados:

- Um estudo sobre os conceitos de jogos construcionistas
- Um estudo sobre os conceitos de *engines*
- Um estudo sobre os conceitos de jogos RPG
- Construção do mod RPG Kingdom Store
- Aplicação de Kingdom Store em sala de aula
- Estudo sobre os resultados obtidos da aplicação do mod

6.4 Trabalhos Futuros

Para trabalhos futuros, é vislumbrado um estudo envolvendo a criação de uma *engine* com maior potencial de variabilidade no tocante às características básicas de jogos criados, que por sua vez, ampliariam as possibilidades de usabilidade em sala de aula, com potencialidades que poderiam incluir a criação de missões, ambientes, classes e inimigos.
O desenvolvimento de outros gêneros de jogos, tais como RTS e FPS poderia ser contemplado em futuras *engines*, bem como a possibilidade de criar partidas *multiplayer*, em que vários jogadores poderiam interagir entre si, em modo competitivo ou colaborativo.
Jogos educacionais que unam as características de construcionismo e PBL (Problem Based Learning) também podem ser contemplados, a fim de fomentar a interdisciplinaridade e a redução da distância entre o aprendizado da teoria e a prática.
Ferramentas geradoras de jogos RPG baseados em *Storytelling* podem também ser concebidas, objetivando, assim, a contextualização do conhecimento por meio de experiências emocionais positivas, ajudando ainda na memorização dos fatos.
Pode também ser concebida Uma SPL destinada à implementação de jogos educacionais, com *features* que enfoquem em seu escopo componentes construcionistas, como uma série de tarefas que levem à construção de um artefato palpável.
Finalmente, é vislumbrada uma *engine* que faça uso das funcionalidades (tais como física, gráficos e sons) de seus pares hoje no mercado, como os mencionados na sessão 2.2 do presente trabalho, voltada ao desenvolvimento de jogos construcionistas.

Referências

ARRUDA, E. P. O papel dos videogames na aprendizagem de conceitos e analogias históricas pelos jovens. *Ensino em Re-Vista*, 2011.

BARTON, M. *Dungeons and desktops: The history of computer role-playing games*. [S.l.]: CRC Press, 2008.

BARZILAI, S.; BLAU, I. Scaffolding game-based learning: Impact on learning achievements, perceived learning, and game experiences. *Computers & Education*, Elsevier, v. 70, p. 65–79, 2014.

BATES, B. Game design second edtion. *Boston: Thomson Course Technology*, 2004.

BOAVENTURA, F.; SARINHO, V. T. Mendiga: A minimal engine for digital games. *International Journal of Computer Games Technology*, Hindawi, v. 2017, 2017.

BODNER, G. M. Constructivism: A theory of knowledge. *J. Chem. Educ*, ACS Publications, v. 63, n. 10, p. 873, 1986.

BOGDAN, R.; BIKLEN, S. K. Qualitative research for education. Allyn & Bacon Boston, 1992.

BOUZID, Y.; KHENISSI, M. A.; JEMNI, M. Designing a game generator as an educational technology for the deaf learners. In: IEEE. *Information & Communication Technology and Accessibility (ICTA), 2015 5th International Conference on*. [S.l.], 2015. p. 1–6.

CANTELON, M. et al. *Node. js in Action*. [S.l.]: Manning Publications, 2017.

CARVALHO, M. B. et al. Towards a service-oriented architecture framework for educational serious games. In: IEEE. *Advanced Learning Technologies (ICALT), 2015 IEEE 15th International Conference on*. [S.l.], 2015. p. 147–151.

CLEMENS, S.; DOMINIK, G.; STEPHAN, M. Component software: Beyond object-oriented programming. *Addison-Wesley*, 2002.

CRPGADDICT. *The Earliest CRPGs*. 2011. Disponível em <http://2.bp.blogspot.com/-0JRb7mcrHEs/TvVbk0vrzII/AAAAAAAAGVo/JedZA5qF6ho/s1600/dungeon2.GIF>. Acesso em: 09/09/2017.

CRYENGINE. *Cry Engine*. 2017. Disponível em <https://www.cryengine.com/>. Acesso em: 07/09/2017.

CRYTEK. *CryEngine*. 2013. Disponível em <http://docs.cryengine.com/download/attachments/1048844/area_shape.jpg?version=1&modificationDate=1320153546000&api=v2>. Acesso em: 07/09/2017.

CURRY, E.; GRACE, P. Flexible self-management using the model-view-controller pattern. *IEEE software*, IEEE, v. 25, n. 3, 2008.

DALMAU, D. S.-C. *Core techniques and algorithms in game programming*. [S.l.]: New Riders, 2004.

DONOHOE, P. *Software product lines: Experience and research directions*. [S.l.]: Springer Science & Business Media, 2012. v. 576.

DOTPDN. *paint.net*. 2018. Disponível em <https://www.getpaint.net/>. Acesso em: 12/03/2018.

EL-NASR, M. S.; SMITH, B. K. Learning through game modding. *Computers in Entertainment (CIE)*, ACM, v. 4, n. 1, p. 7, 2006.

ELECTRONICARTS. *SimCity 4*. 2012. Disponível em <https://web-vassets.ea.com/Assets/Richmedia/Image/Screenshots/SimCity%204%20Deluxe%20Screenshot%202.jpg?cb=1412974766>. Acesso em: 05/09/2017.

ENSEMBLE. *Age of Empires*. 2013. Disponível em <http://www.ensemblestudios.com/>. Acesso em: 05/09/2017.

ERTMER, P. A.; NEWBY, T. J. Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance improvement quarterly*, Wiley Online Library, v. 6, n. 4, p. 50–72, 1993.

ERTMER, P. A.; NEWBY, T. J. Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly*, Wiley Online Library, v. 26, n. 2, p. 43–71, 2013.

FERNÁNDEZ-PANADERO, C. et al. A framework to design educational mobile-based games across multiple spaces. In: *Design for Teaching and Learning in a Networked World*. [S.l.]: Springer, 2015. p. 407–413.

FILHO, C. A. C. L.; HERNÁNDEZ-DOMÍNGUEZ, A. Jindie: Uma linha de produto de software para jogos educativos com foco no construcionismo. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 350.

FINALFANTASY. 2017. Disponível em <http://finalfantasy.wikia.com>. Acesso em: 11/09/2017.

FIRAXIS. *Civilization*. 2016. Disponível em <http://www.firaxis.com/>. Acesso em: 05/09/2017.

GEORGIEV, A. et al. The rage game software components repository for supporting applied game development. Serious Games Society, 2017.

GITHUB. *Atom A Hackable Text Editor for the 21st Century*. 2018. Disponível em <https://atom.io/>. Acesso em: 12/03/2018.

GUIDE, D. M. *Advanced Dungeons and Dragons*. [S.l.]: Renton, Wash.: TSR, 1995.

GYŐRÖDI, C. et al. A comparative study: Mongodb vs. mysql. In: IEEE. *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on*. [S.l.], 2015. p. 1–6.

HAMARI, J. et al. Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behavior*, Elsevier, v. 54, p. 170–179, 2016.

HAN, J. et al. Survey on nosql database. In: IEEE. *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. [S.l.], 2011. p. 363–366.

HARDCOREGAMING101. *Temple of Apshai C64*. 2010. Disponível em <http://www.hardcoregaming101.net/dunjonquest/templeofapshai-c64-1.png>. Acesso em: 10/09/2017.

HAY, K. E.; BARAB, S. A. Constructivism in practice: A comparison and contrast of apprenticeship and constructionist learning environments. *The Journal of the Learning Sciences*, Taylor & Francis, v. 10, n. 3, p. 281–322, 2001.

HEIN, G. Constructivist learning theory. *Institute for Inquiry. Available at:/http://www. exploratorium. edu/ifi/resources/constructivistlearning. htmlS*, 1991.

HOLBERT, N. et al. Combining video games and constructionist design to support deep learning in play. International Society of the Learning Sciences, 2014.

HWANG, G.-J.; CHIU, L.-Y.; CHEN, C.-H. A contextual game-based learning approach to improving students' inquiry-based learning performance in social studies courses. *Computers & Education*, Elsevier, v. 81, p. 13–25, 2015.

INDRAPRASTHA, A.; SHINOZAKI, M. The investigation on using unity3d game engine in urban design study. *Journal of ICT Research and Applications*, v. 3, n. 1, p. 1–18, 2009.

IROWIKI. 2017. Disponível em <http://irowiki.org>. Acesso em: 12/09/2017.

JOHNSON, C. Learning to program with game maker. *Online Submission*, ERIC, v. 1, n. 2, 2017.

JOHNSON, R. E. Frameworks=(components+ patterns). *Communications of the ACM*, ACM, v. 40, n. 10, p. 39–42, 1997.

JOHNSON, W. L.; WANG, N.; WU, S. Experience with serious games for learning foreign languages and cultures. In: SIMULATION INDUSTRY OF AUSTRALIA. *Proceedings of the SimTecT Conference*. [S.l.], 2007.

JONES, M. G.; BRADER-ARAJE, L. The impact of constructivism on education: Language, discourse, and meaning. *American Communication Journal*, v. 5, n. 3, p. 1–10, 2002.

KAFAI, Y. B. Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and culture*, Sage Publications Sage CA: Thousand Oaks, CA, v. 1, n. 1, p. 36–40, 2006.

KAFAI, Y. B.; BURKE, Q. Constructionist gaming: Understanding the benefits of making games for learning. *Educational psychologist*, Taylor & Francis, v. 50, n. 4, p. 313–334, 2015.

KRUEGER, C. W. Software reuse. *ACM Computing Surveys (CSUR)*, ACM, v. 24, n. 2, p. 131–183, 1992.

LEWIS MICHAEL E JACOBSON, J. Game engines. *Communications of the ACM*, v. 45, 2002.

MACHADO, A.; SANTOS, P.; DIAS, J. On the structure of role playing game quests. *Revista de Ciências da Computação*, v. 12, 2017.

MAXIS. *SimCity*. 2015. Disponível em <http://www.maxis.com/>. Acesso em: 05/09/2017.

MCGONIGAL, J. *Reality is broken: Why games make us better and how they can change the world*. [S.l.]: Penguin, 2011.

MCGREGOR, J. D. Testing a software product line. In: SPRINGER. *Pernambuco Summer School on Software Engineering*. [S.l.], 2007. p. 104–140.

MEC. *Relatório educação para todos no brasil 2000-2015*. 2016. <http://portal.mec.gov.br/docman/junho-2014-pdf/15774-ept-relatorio-06062014/file>. Acessado em 25/08/2017.

MENDONCA, M.; BRANCO, M.; COWAN, D. Splot: software product lines online tools. In: ACM. *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. [S.l.], 2009. p. 761–762.

MINNERY, J.; SEARLE, G. Toying with the city? using the computer game simcity™ 4 in planning education. *Planning Practice and Research*, Taylor & Francis, v. 29, n. 1, p. 41–55, 2014.

MOJANG. *Minecraft*. 2013. Disponível em <https://mojang.com/>. Acesso em: 05/09/2017.

MORA, R. "school is so boring": High-stakes testing and boredom at an urban middle school. *Penn GSE Perspectives on Urban Education*, ERIC, v. 9, n. 1, p. n1, 2011.

MZSTATIC. *Minecraft*. 2016. Disponível em <http://is4.mzstatic.com/image/thumb/Purple122/v4/f9/a8/3b/f9a83b28-e38a-b70b-284f-3d89918f016e/source/720x405bb.jpg>. Acesso em: 05/09/2017.

NAKEVSKA, M. et al. Using game engines in mixed reality installations. *Entertainment Computing–ICEC 2011*, Springer, p. 456–459, 2011.

NERI, M. et al. Motivos da evasão escolar. 2015.

NORTHROP, L. et al. A framework for software product line practice, version 5.0. *SEI.–2007–http://www. sei. cmu. edu/productlines/index. html*, 2007.

NOVARAGNAROK. *Ragnarok Online*. 2015. Disponível em <https://www.novaragnarok.com/themes/nova/img/img/ss1.jpg>. Acesso em: 12/09/2017.

OVERMARS, M. Teaching computer science through game design. *Computer*, IEEE, v. 37, n. 4, p. 81–83, 2004.

PAPERT, S. *Constructionism: A new opportunity for elementary science education*. [S.l.]: Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, 1986.

PAPERT, S.; HAREL, I. Situating constructionism. *Constructionism*, v. 36, n. 2, p. 1–11, 1991.

PEREIRA, C.; ANDRADE, F. M.; FREITAS, L. E. Desafio dos bandeirantes–aventuras na terra de santa cruz. *Rio de Janeiro: GSA*, 1992.

PHONEARENA. *Age of Empires*. 2002. Disponível em <https://i-cdn.phonearena.com/images/articles/87881-image/Age-of-Empires-I.jpg>. Acesso em: 05/09/2017.

PHOTONSTORM. *Phaser Desktop and Mobile MTML5 Game Framework*. 2018. Disponível em <https://phaser.io/>. Acesso em: 12/03/2018.

PLASIL, F.; VISNOVSKY, S. Behavior protocols for software components. *IEEE transactions on Software Engineering*, IEEE, v. 28, n. 11, p. 1056–1076, 2002.

PRESSLEY, D.; GALI, C.; DOGGETT, C. *Software application generator*. [S.l.]: Google Patents, 2014. US Patent App. 14/211,778.

RASTOGI, A. et al. Safe & efficient gradual typing for typescript. *ACM SIGPLAN Notices*, ACM, v. 50, n. 1, p. 167–180, 2015.

RDUSHAY. *Dungeons & Dragons White Box Edition*. 2011. Disponível em <http://rdushay.home.mindspring.com/Museum/Fantasy/DDrevw.html>. Acesso em: 09/09/2017.

ROCKPAPERSHOTGUN. *Ultima Online*. 2015. Disponível em <https://www.rockpapershotgun.com/images/15/jun/ultimaonline03.jpg>. Acesso em: 10/09/2017.

RUEBBELKE, L.; FORD, B. *AngularJS in action*. [S.l.]: Manning, 2015.

RUTHERFORD, A. Radical behaviorism and psychology's public: Bf skinner in the popular press, 1934–1990. *History of Psychology*, Educational Publishing Foundation, v. 3, n. 4, p. 371, 2000.

SASS. *CSS with superpowers*. 2017. Disponível em <https://sass-lang.com/>. Acesso em: 12/03/2018.

SCHICK, L. *Heroic worlds: A history and guide to role-playing games*. [S.l.]: Prometheus Books, 1991.

SOUSA, V. D.; DRIESSNACK, M.; MENDES, I. A. C. An overview of research designs relevant to nursing: Part 1: quantitative research designs. *Revista latino-americana de enfermagem*, SciELO Brasil, v. 15, n. 3, p. 502–507, 2007.

SQIRE, K. Changing the game: What happens when video games enter the classroom? *Innovate: Journal of online education*, v. 1, n. 6, p. 5, 2005.

SQUAREENIX. *Square Enix*. 2017. Disponível em <https://na.square-enix.com/us/home>. Acesso em: 11/09/2017.

THEANTIROOM. *Logo*. 2011. Disponível em <https://theantiroom.files.wordpress.com/2011/04/logo_turtle.jpg>. Acesso em: 03/09/2017.

THEGAMER. *Civilization V*. 2012. Disponível em <http://www.gamer.ru/system/attached_images/images/000/544/593/original/2012-06-21_00011.jpg>. Acesso em: 05/09/2017.

THORSTEINSSON, G.; NICULESCU, A. Pedagogical insights into the use of minecraft within educational settings. *Studies in Informatics and Control*, v. 25, n. 4, p. 508, 2016.

TINSMAN, B. *The Game Inventor's Guidebook*. [S.l.]: Krause Publications, 2003.

ULTIMA. 2012. Disponível em <https://web.archive.org/web/20120902020412/http://www.uo.com/archive/>. Acesso em: 10/09/2017.

ULTIMACODEX. 2017. Disponível em <http://wiki.ultimacodex.com>. Acesso em: 10/09/2017.

UNITY. *Unity Engine*. 2017. Disponível em <https://unity3d.com>. Acesso em: 07/09/2017.

UNREAL. *Unreal Engine*. 2017. Disponível em <https://www.unrealengine.com>. Acesso em: 07/09/2017.

UNREAL-BRASIL. *Unreal*. 2016. Disponível em <http://is4.mzstatic.com/image/thumb/Purple122/v4/f9/a8/3b/f9a83b28-e38a-b70b-284f-3d89918f016e/source/720x405bb.jpg>. Acesso em: 05/09/2017.

VALUECODERS. *10 Top Web Development Frameworks In 2018*. 2018. Disponível em <https://www.valuecoders.com/blog/technology-and-apps/10-top-web-development-frameworks-businesses>. Acesso em: 04/03/2018.

W3C. *HTML5 Differences from HTML4*. 2014. Disponível em <https://www.w3.org/TR/2014/NOTE-html5-diff-20141209/>. Acesso em: 12/03/2018.

W3SCHOOLS. *JavaScript Tutorial*. 2018. Disponível em <https://sass-lang.com/>. Acesso em: 12/03/2018.

WINDOWSCENTRAL. *Unity*. 2012. Disponível em <https://www.windowscentral.com/sites/wpcentral.com/files/postimages/35454/Engine_editor_s.jpg>. Acesso em: 07/09/2017.

WORLDOFWARCRAFT. 2017. Disponível em <https://worldofwarcraft.com>. Acesso em: 12/09/2017.

WOWHEAD. 2017. Disponível em <http://www.wowhead.com/>. Acesso em: 12/09/2017.

YOYOGAMES. *GameMaker Studio*. 2017. Disponível em <https://www.yoyogames.com/gamemaker>. Acesso em: 08/09/2017.

APÊNDICE A – Textos dos Participantes do Estudo de Caso

Os parágrafos seguintes deste apêndice contêm as cartas dos alunos participantes do estudo de caso contido no capítulo 5 deste trabalho.

Majestade, durante anos o uso o banco de dados relacionais supriu e a tendeu a necessidade do reino, mantendo seguro e estável os dados, evoluiu e multiplicou suas possibilidades assim como surgiu uma quantidade satisfatória de profissionais, apesar da segurança que ele nos passa e a comodidade de ser bem mais conhecido está na hora de mudar para algo melhor. O NoSQL é uma nova forma de armazenar dados baseado em estruturas de dados, resolvendo assim uma série de problemas e criando novas possibilidades, como escalabilidade, alta disponibilidade, alto desempenho e confiabilidade. Está na hora de evoluir e aproveitar esta tecnologia desde o começo.

A utilização de bancos de dados NoSQL será uma mudança que, querendo ou não, vai ocorrer, por sua praticidade, facilidade e acesso. Ficará muito mais simples achar quem consiga fazer bancos de dados sem o sistema SQL, dadas suas vantagens de não precisar de um grupo específico para sua montagem e instalação, além de melhor usufruir do espaço do disco rígido. Apesar da consolidação do DBA clássico, é sabido que as mudanças sempre vem pra melhor ou pior, então, mesmo que haja um problema com o NoSQL sempre dá para voltar ao antigo modelo, que não é totalmente desfuncional. Diante das facilidades apresentadas para o NoSQL, acredito que deva ser feita uma atualização para o novo tipo de banco de dados que irá expandir horizontes e melhorar a qualidade de serviços.

Vossa Magestade Rei Sternenlicht I, Quando a linguagem sql virou um padrão, os bancos de dados relacionais tornaram padrão para desenvolvimento de aplicações com bases de dados. SQL relacional e RDBMS tornaram quase sinônimos e instalaram-se na mente de todos os desenvolvedores por décadas. Ninguém imagina outra forma de trabalhar. Várias organizações passaram a desenvolver e utilizar banco de dados não relacionais que suporta a demanda.

À vossa majestade Rei Sternenlicht I, aconselho não substituir o banco de dados relacionais pela novidade NoSQL, levando em consideração o grande valor das informações contidas nos arquivos. Ressalto que migrar seria um risco desnecessário, podendo aguardar a solidificação do novo sistema no mercado, pois é sabido que a não utiliza-

ção de DBA para monitorar e criar restrições especificas, torna a priori vulnerável o controle dos registros. Entretanto, caso seja de vossa vontade Real, sugiro adquirir bakups confiáveis para uma excepcional restauração em caso de possíveis falhas nos processos.

A majestade Rei Sternenlicht I

Vossa majestade venho em meio a essa carta, informa-lo que a troca da nosso banco de dados Sql pelo banco de dados NoSql, seria de fato uma boa mudança, pois com esse novo banco, iriamos poupar tempo no armazenamento das informações e uma maior flexibilidade, sem contar que não precisaremos de administradores para fazer a manutenção, poupando assim nosso gold. Sei que de fato o banco NoSql está em seu inicio, mas creio que suas vantagens são de grande valor para nosso reino majestade.

Vossa majestade, Venho por intermédio desta, venho informar ao senhor que o uso da linguagem SQL que é usada ainda hoje em vosso reino, continua sendo uma das formas mais adequadas de se criar, manter e manipular os dados, uma vez que o próprio nome já o identifica como tal, pois SQL ou Linguagem de Consulta Estruturada traz ferramentas que são amplamente utilizadas para o tratamento dos dados que estarão contidos no banco. Funções como DML, DCL, DDL, DQL entre outros, são funções que estão disponíveis em sistemas de banco de dados que são baseados em SQL. Portanto, Vossa Majestade deve, em minha humilde opinião, manter estas ferramentas que já temos disponíveis e continuar com o mesmo sistema de banco de dados relacional, que usa tabelas relacionadas.

Rei Sternenlicht I , no meu ponto de vista os banco de dados relacionais deveriam continuar no reino de Sieben Schlüssel, com o definição e manipulação de dados, pelo o motivo do bom tempo que vem sendo utilizado e considero seguro , pois o NoSql se trata apenas de uma alternativa e faria com que tivesse alguma mudanças desnecessária. Conclusão não lhe aconselho a fazer essa mudança!

Vossa Majestade, não escute as historias desse enviado de Mordor, ele esta tentando fazer a sua cabeça para enfraquecer o nosso reino pela base e assim tornar mais fácil a sua conquista, precisamos continuar com SQL, ou seja, organizando nossas tropas por meio de tabelas relacionadas, esse modo certamente trará nossa vitória e também a prosperidade de nosso reino. Espero que por meio dessa carta, Vossa Majestade tome a decisão correta e não caia nas ladainhas desse enviado do mal.

Vossa majestade, uma certa pessoa que está causando um alvoroço na sua população, existe um certo mago chamado Enrique Sanchez que quer trazer uma nova forma de banco de dados, o NoSQL, que envolve estrutura de dados de forma de armazenamento, em meu conhecimento, eu não aconselharia o uso desse meio pois causará uma quebra no sistema

de SQL, ao invés disso, aconselho você procurar esse mago e ver uma possibilidade de melhorar ou aprimorar nosso sistema para o bem de todos.

Mude a forma de armazenamento de dados, os bancos de dados relacionais tem limitações de escalabilidade e sempre haverá mais dados para armazenar, já o NoSQL, que é um modelo não-relacional, permite escalabilidade, alta disponibilidade, alto desempenho e confiabilidade.