
**Uma proposta para gerenciamento de fases de
jogos educacionais desenvolvidos com Phaser.js
para não desenvolvedores no contexto de
Química**

João Gabriel de Matos Dairel



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

João Gabriel de Matos Dairel

**Uma proposta para gerenciamento de fases de
jogos educacionais desenvolvidos com Phaser.js
para não desenvolvedores no contexto de
Química**

Trabalho de Conclusão de Curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, Minas Gerais, como
requisito exigido parcial à obtenção do grau de
Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Professor Dr. Rafael Dias Araújo

Monte Carmelo - MG

2021

Este trabalho é dedicado aos meus pais, por nunca terem medido esforços para me proporcionar um ensino de qualidade durante todo o meu período escolar, e ao meu orientador, que conduziu o trabalho com paciência e dedicação, sempre disponível a compartilhar todo o seu vasto conhecimento.

Agradecimentos

Aos meus pais e irmã, que sempre estiveram ao meu lado, e pelo apoio demonstrado ao longo de todo o período de tempo em que me dediquei a este trabalho.

Ao professor Rafael, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

Aos professores, por todos os conselhos, pela ajuda e pela paciência com a qual guiaram o meu aprendizado.

Aos meus colegas e amigos, por compartilharem comigo tantos momentos de descobertas e aprendizado e por todo o companheirismo ao longo deste percurso.

A todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho de pesquisa, enriquecendo o meu processo de aprendizado.

*“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda
pensou sobre aquilo que todo mundo vê.”
(Arthur Schopenhauer)*

Resumo

Professores da Educação Básica têm buscado estratégias que vão além das metodologias tradicionais de ensino. Uma possibilidade é a utilização de jogos digitais educacionais. No entanto, a criação de jogos educacionais multiplataforma é uma tarefa difícil e que demanda equipes técnicas especializadas. Assim, há uma lacuna temporal entre a demanda dos especialistas de domínio, como professores da Educação Básica, e o desenvolvimento e disponibilização desses artefatos.

Dessa forma, este trabalho propõe um projeto de arquitetura de software que permite a criação dinâmica de fases de jogo utilizando o *framework* Phaser.js no contexto da disciplina de Química. A proposta armazena dados em arquivos JSON referentes a usuários, módulos, níveis, *quizzes* e turmas. Além disso, a proposta inclui a coleta de interações por meio de registro de *logs* para envio a uma *Application Programming Interface* (API).

Foi implementado um protótipo funcional como prova de conceito para a validação por meio de entrevista com um possível usuário de uma ferramenta como esta. Os resultados obtidos foram relevantes e mostraram a viabilidade da arquitetura proposta e o potencial impacto para professores.

Palavras-chave: Jogos Digitais Educacionais, Autoria, Arquitetura de Software, Química, Phaser.js.

Abstract

Basic Education teachers have been looking for strategies that go beyond traditional teaching methodologies. One possibility is the use of educational digital games. However, creating cross-platform educational games is a difficult task and requires specialized technical teams. Thus, there is a temporal gap between the demand of domain specialists, such as Basic Education teachers, and the development and deployment of such artifacts.

Thus, this work proposes a software architecture that allows the dynamic creation of game phases using the framework Phaser.js in the context of the Chemistry discipline. The proposal stores data in JSON files related to users, modules, levels, quizzes and classes. In addition, the proposal includes the collection of interactions through the registration of *logs* to be sent to an Application Programming Interface (API).

A functional prototype was implemented as a proof of concept for validation through an interview with a possible user of a tool like this. The results obtained were relevant and showed the feasibility of the proposed architecture and the potential impact for teachers.

Keywords: Educational Digital Games, Authoring, Software Architecture, Chemistry, Phaser.js.

Lista de ilustrações

Figura 1 – Diagrama UML representando os arquivos JSON.	22
Figura 2 – Exemplo de uma classe que representa um módulo de fase de jogo. . .	23
Figura 3 – Diagrama de Sequência de criação de fase.	24
Figura 4 – Diagrama de Sequência de abertura de fase.	25
Figura 5 – Diagrama de Casos de Uso.	26
Figura 6 – Interface de criação de fases.	27
Figura 7 – Interface de criação de fases, com destaque para a parte onde os módulos são apresentados.	27
Figura 8 – JSON representando uma fase criada.	28
Figura 9 – Interface de listagem de fases.	28
Figura 10 – Exemplo de módulo de menu.	29
Figura 11 – Exemplo de um módulo inicial sem obstáculos.	30
Figura 12 – Exemplo de um módulo com balões representando elementos químicos.	30
Figura 13 – Exemplo de um módulo com plataforma deslizável e obstáculos.	31
Figura 14 – Exemplo de um módulo final com um item para abertura de um <i>quiz</i>	31
Figura 15 – Exemplo de módulo de quiz.	32
Figura 16 – Alterações propostas após entrevista com usuário.	35

Sumário

1	INTRODUÇÃO	9
1.1	Problema	10
1.2	Justificativa	10
1.3	Objetivos	11
1.4	Organização do Texto	11
2	FUNDAMENTAÇÃO TEÓRICA	12
2.1	Conceitos e Definições	12
2.1.1	Jogos Digitais Educacionais	12
2.1.2	Frameworks para desenvolvimento de jogos multiplataforma	13
2.1.3	Bancos de Dados Não Relacionais	15
2.2	Trabalhos Relacionados	16
2.2.1	Criação de Fases	16
2.2.2	Jogos no Contexto de Química	18
3	DESENVOLVIMENTO	21
3.1	Metodologia	21
3.2	Proposta	22
3.3	Prova de Conceito	25
4	RESULTADOS E DISCUSSÕES	33
4.1	Entrevista	33
4.2	Resultados	34
5	CONSIDERAÇÕES FINAIS	36
5.1	Trabalhos Futuros	37
5.2	Contribuições em Produção Bibliográfica	37
	REFERÊNCIAS	38

Introdução

Atualmente, professores da Educação Básica têm buscado estratégias que vão além das metodologias tradicionais de ensino, aquelas em que o conteúdo é apresentado apenas de forma expositiva e o professor atua como figura central do processo de ensino e aprendizagem. Especialmente as estratégias que incluem Tecnologias Digitais de Informação e Comunicação (TDIC) proporcionam contribuições positivas e desafios ao processo de ensino e aprendizagem (NOVAES et al., 2021). Jogos digitais educacionais figuram uma das várias possíveis alternativas e têm se mostrado promissores e engajadores no processo de ensino e aprendizagem de diversas áreas do conhecimento em diversas idades (OLIVEIRA; HILDEBRAND, 2018; PAIVA; TORI, 2017). Nesse contexto, a tecnologia tem atuado como uma aliada no contexto escolar e tem o potencial de mudar a realidade da falta de recursos didáticos das escolas e mudar o paradigma educacional (VALENTE, 1993).

Por mais que se tenha o desejo da utilização de jogos no meio educacional, há uma dificuldade em sua criação. É um processo que demanda recursos humanos, como especialistas de domínio durante a fase de projeto, especialistas técnicos (programadores) durante a fase de desenvolvimento, bem como recursos computacionais, como *hardware* capaz de desenvolver e renderizar as fases. Além disso, há uma demanda de tempo, muitas vezes grande, para que as fases sejam desenvolvidas, testadas e sejam colocadas em produção para uso efetivo.

No contexto da Química, o uso de jogos digitais tem atraído bastante a atenção a fim de proporcionar métodos que proporcionem ensino de melhor qualidade e facilitem seu aprendizado, dada sua complexidade e importância (RASTEGARPOUR; MARASHI, 2012). Alguns professores já utilizam de jogos digitais para tornar o aprendizado de Química mais interessante e atrativo, com o intuito de fazer com que os estudantes tirem melhor proveito de seu aprendizado e desenvolver a inteligência cinestésica. O aprendizado por meio de ações e reações faz com que o cognitivo seja mais eficiente em seu trabalho de adquirir conhecimentos (MAYER, 2019)

O currículo de Química é bem extenso e contempla diversos conteúdos. A Tabela Pe-

riódica (TP) é um dos mais importantes e serve como base de muitos outros na sequência. A Tabela Periódica consiste em uma tabela contendo os elementos químicos agrupados em grupos e famílias, sendo definidos por semelhanças quanto às suas propriedades químicas. Sua importância é dada pela necessidade do conhecimento da quantidade de elétrons na camada de valência (última camada da eletrosfera dos átomos), que posteriormente são utilizados em cálculos estequiométricos, e balanceamento de equações. Ocorre que, muitas vezes, estudar os elementos químicos sem uma contextualização com o cotidiano dos estudantes faz com que esse tópico se torne abstrato e, muitas vezes, de difícil absorção de conhecimento.

Assim, é importante introduzir, durante o processo de aprendizagem, uma contextualização de onde e como esses elementos existem e/ou são utilizados. Por exemplo, um possível cenário onde se trata de características de gases nobres pode ser utilizado para contextualizar a densidade e como se comportam diferentes elementos em relação ao ar atmosférico. Entretanto, a implementação de jogos digitais que atendam os diferentes requisitos de diferentes turmas e conteúdos não é uma tarefa fácil. Além disso, os jogos não devem ser muito complexos, a fim de que os alunos atinjam os objetivos e compreendam o conteúdo (RASTEGARPOUR; MARASHI, 2012).

1.1 Problema

Este trabalho surgiu a partir de um projeto de iniciação científica realizada, cujo objetivo foi a implementação de fases de jogos multiplataforma como suporte ao ensino da Tabela Periódica. Durante o projeto mencionado, foram criadas duas fases de um jogo e observou-se a dificuldade e tempo gasto com a implementação. Assim, ficou claro que um dos desafios do processo de desenvolvimento de jogos digitais educacionais é a autoria, dada a infinidade de domínios de aplicação existentes e o consumo de tempo para programar e disponibilizar cada novo jogo, o que pode fazer com que a demanda se torne inviável até que seja disponibilizada a solução.

Além disso, existe o desafio computacional relacionado com a extensa variedade de dispositivos e plataformas. O desenvolvimento de softwares compatíveis com todos esses dispositivos se torna complexo, pois existem muitas diferenças nos tamanhos de tela, sistemas operacionais e *hardware*. Assim, o uso de *frameworks* Web que dão suporte ao desenvolvimento multiplataforma torna o processo menos árduo.

1.2 Justificativa

O ensino da Tabela Periódica (TP) é um dos mais importantes e fundamentais para o entendimento da disciplina de Química. A TP contempla mais de cem elementos diferentes, cada um tendo diversas propriedades específicas individuais.

O processo de criação de fases é complexo e demanda muito tempo para ser desenvolvido por programadores. Para pessoas sem conhecimento em computação se torna ainda mais difícil e desafiador pela necessidade de conhecimentos técnicos. Por mais que haja interesse em utilizar jogos digitais na educação, é visto como uma limitação a necessidade de desenvolvedores para a implementação de fases.

Um sistema de gerenciamento dinâmico de fases contaria com diversos módulos que podem ser montados dinamicamente pelo professor. Além disso, também é importante que dados de utilização do jogo sejam coletados e analisados de forma a prover informações acerca do processo de aprendizagem dos estudantes e facilitar a identificação de tópicos com mais dificuldades.

1.3 Objetivos

O objetivo geral deste trabalho é projetar uma arquitetura de software que permita o gerenciamento dinâmico de fases de um jogo utilizando o *framework* Phaser.js.

Os objetivos específicos incluem:

- ❑ Projetar uma arquitetura de software que permita a criação de jogos educacionais por meio de módulos pré-programados;
- ❑ Modelar um banco de dados considerando regras no contexto da disciplina de Química;
- ❑ Coletar registros de utilização (*logs*) de jogo via API;
- ❑ Implementar uma prova de conceito em um cenário real para avaliação da proposta com professores.

1.4 Organização do Texto

O Capítulo 2 apresenta os principais conceitos necessários para o entendimento da proposta, bem como os trabalhos relacionados; o Capítulo 3 apresenta a metodologia de trabalho e a abordagem proposta; o Capítulo 4 traz os as discussões acerca dos resultados obtidos; e, por fim, o Capítulo 5 apresenta as considerações finais e propostas de trabalhos futuros.

Fundamentação Teórica

2.1 Conceitos e Definições

2.1.1 Jogos Digitais Educacionais

Atualmente, os jogos digitais vêm se tornando parte da vida de crianças, adolescentes e adultos como meios para se distrair e se divertir. A cada dia, surgem novos jogos com propostas totalmente novas e que conseguem atrair a atenção dos usuários e fazer com que joguem por dias, meses e até anos. Os jogos digitais também têm outras funções além de diversão, como a educação. É possível encontrar jogos digitais para as mais diversas áreas do conhecimento e objetivos educacionais, como aprendizado de novos idiomas, ciências exatas, anatomia, entre outras áreas.

Neste contexto, quando jogos são inseridos no meio educacional, ajudam os estudantes a pensar de diversas maneiras diferentes para resolver problemas e quando captam a atenção dos estudantes faz com que lembrem de conceitos utilizados (MAYER, 2019). Os experimentos realizados em Lay e Osman (2018) relacionados com a utilização de jogos digitais com grupos de alunos apresentaram resultados significativamente melhores do que a utilização de métodos tradicionais de ensino.

Além da utilização dos jogos como meios que proporcionam um efeito motivador nos estudantes, eles também possuem papel de facilitadores da educação e fazem com que os usuários sempre busquem entender o que se passa em determinada fase ou nível. A procura de informações visando o aprendizado, quando aplicado às disciplinas da grade curricular do ensino médio, causam um grande impacto na aprendizagem e influenciam nas habilidades cognitivas dos estudantes, tais como reflexos rápidos, raciocínio lógico, reconhecimento de padrões, dentre outros (SAVI; ULBRICHT, 2008).

A disciplina de Química é uma das disciplinas que os jogos digitais educacionais podem auxiliar. Química é a área da ciência que estuda as relações entre a matéria, as reações que ocorrem, a transformação de um material em outro e contribui substancialmente para o avanço tecnológico e científico da humanidade (GUIMARÃES, 2021). A

utilização de jogos pode quebrar o paradigma de que Química é uma matéria muito difícil e complexa, pois à primeira vista, realmente parece ser bem complexa, entretanto ao começar a aprender os conceitos gera interesse por parte dos estudantes.

2.1.2 Frameworks para desenvolvimento de jogos multiplataforma

Existem diversos softwares e *frameworks* para o desenvolvimento de jogos, alguns específicos para uma plataforma computacional específica (como smartphones ou computadores *desktop*) e outros com suporte multiplataforma, ou seja, permite que o jogo seja acessado por diferentes plataformas e sistemas operacionais. De acordo com (AMATYA; KURTI, 2014), uma solução que trataria o problema de acesso por meio de dispositivos multiplataforma é a utilização de browsers para a execução dos jogos. A escolha do *framework* é algo essencial para facilitar o processo de desenvolvimento e abstrair questões e dificuldades específicas de plataformas.

A Engine Unreal¹ possui um sistema de *blueprints*, que consiste na criação dos scripts do jogo sem a necessidade de programação. No quesito linguagens de programação, a *engine* utiliza C++ como linguagem principal. A interface é intimidadora à primeira vista, cheia de informações e opções, porém oferece muita flexibilidade com as opções em sua interface. Os jogos podem ser desenvolvidos para *mobile*, *desktop*, *consoles*, dispositivos de Realidade Virtual (VR) e Web. Quanto à qualidade dos gráficos, possui uma qualidade muito alta, compete diretamente com os gráficos do Unity, sendo até melhores.

A Engine Unity² possui um sistema semelhante ao *blueprints* da Unreal, chamado visual scripting que tem o mesmo intuito. Diferente da Engine Unreal, ela utiliza as linguagens C# e javascript, possui uma interface relativamente complexa à primeira vista, mas ao se acostumar se torna bem simples e intuitiva. Possui suporte multiplataforma e os jogos podem ser desenvolvidos para *mobile*, *desktop*, *consoles*, VR e Web. A qualidade gráfica dos jogos compete fortemente com seu rival, o Unreal, entretanto, fica um pouco atrás de seu rival.

O MIT App Inventor³ é uma *engine* com interface bem simples, conta com o esquema de arrastar trechos de códigos e conectá-los apenas alterando os valores dentro dos códigos já criados. Possui suporte apenas para a plataforma *mobile* e não oferece uma liberdade total, pois não se tem a possibilidade de escrever seus próprios códigos.

Existem, ainda, *engines* focadas para o desenvolvimento de jogos para a Web que, por natureza, permite a execução em diferentes plataformas por utilizar apenas linguagens entendidas por navegadores Web, como HTML5, CSS e Javascript. As *engines* Unity e Unreal são capazes de exportar jogos para Web, entretanto, não geram um código nativo

¹ <<https://www.unrealengine.com/>>

² <<https://unity.com/>>

³ <<https://appinventor.mit.edu/>>

de HTML, CSS e Javascript que consegue ser lido e interpretado pelos navegadores. Para executá-los, os navegadores utilizam *plugins* específicos.

A Babylon.js é uma *engine* específica para desenvolvimento de jogos para a Web. É de código aberto⁴ e é capaz de produzir jogos robustos em 3D. Possui capacidades gráficas incríveis, tem a possibilidade de utilizar um editor online para testar códigos, possui diversos exemplos de códigos e uma documentação bem ampla.

O PlayCanvas também é uma *engine* de código aberto⁵ que permite a utilização de códigos escritos em Javascript para a criação de jogos 2D e 3D, sendo compatível com quase todos os navegadores. Não necessita de um editor para se analisar e depurar os códigos, podendo se utilizar das ferramentas de desenvolvedor do navegador para realizar tais tarefas.

O Phaser.js é uma *engine* robusta para Web, focada em jogos 2D. Tem a possibilidade de escolher entre duas *engines* para a física de jogo (matter.js ou arcade physics, nativa do phaser) e possui uma ótima documentação, com fóruns bastante ativos e uma comunidade que se ajuda bastante. Como as outras *engines* para Web, por serem *frameworks* Javascript, facilita sua integração com outros sistemas, sendo necessária apenas conexão com a Internet.

A Tabela 1 apresenta uma comparação das principais funcionalidades dos *frameworks* apresentados, analisados do ponto de vista dos recursos que as *engines* de jogos Javascript disponibilizam e a dificuldade da criação dinâmica de fases. Para contextualizar a tabela, os itens comparativos foram divididos em quatro categorias, sendo elas Foco em 2D, que diz respeito ao objetivo central do *framework* ser o desenvolvimento de jogos 2D; Foco em 3D, que representa o objetivo central do *framework* no desenvolvimento de jogos 3D; Multiplataforma, na qual é verificada a compatibilidade do *framework* para desenvolvimento de jogos multiplataforma; e, por último, Fácil criação dinâmica de fases, na qual é verificada a dificuldade em se implementar um jogo que contenha vários módulos.

Tabela 1 – Tabela comparativa de *frameworks* para desenvolvimento de jogos para Web.

	Phaser.js	Babylon.js	PlayCanvas
Foco em 2D	X		X
Foco em 3D		X	X
Multiplataforma	X	X	X
Fácil criação dinâmica de fases	X		

A escolha do Phaser.js se deu principalmente pela facilidade em se projetar a criação de fases de jogo por meio de módulos pré-programados, além da familiaridade já adquirida durante o projeto de iniciação científica mencionado.

⁴ <<https://github.com/BabylonJS/Babylon.js>>

⁵ <<https://github.com/playcanvas/engine>>

2.1.2.1 Phaser.js

Como já mencionado, o Phaser.js é um framework de criação de jogos que utiliza a linguagem de programação JavaScript, e por meio do JavaScript é introduzido um canvas no HTML. Um canvas é uma área retangular na qual é possível renderizar gráficos, esses gráficos devem ser renderizados por meio do JavaScript, e o mesmo gerencia as *Scenes*.

Em jogos digitais, *Scenes* são cenários onde cada cenário contém diferentes informações, essas informações podem ser desde simples pressionar um botão à uma fase completa de um jogo. Durante o processo de criação dos módulos, é necessário que a classe herde as características do Phaser.Scene para ser tornar um módulo de jogo.

A partir da herança feita, uma *Scene* precisa de uma “key”, que é o atributo utilizado para inicializar uma Scene por meio de uma instância do Phaser.Game, na qual são passadas ao Phaser as configurações de inicialização, altura e largura do canvas gerado no HTML, uma escala de renderização, as configurações da *engine* de física, e, por fim, as *Scenes* que serão renderizadas. Os parâmetros de altura e largura são utilizados como referência para as Scenes na qual renderizam as unidades de largura e altura da *Scene*, já o parâmetro de escala de renderização se refere ao modo em que o canvas é renderizado no HTML. Os demais parâmetros de uma *Scene* são utilizados para controlar como a troca e inicialização de módulos irá ocorrer, e são definidos como *false* em *active* e *visible*.

A *Scene* também possui 3 funções herdadas de Phaser.Scene, a função *preload*, na qual o carregamento dos *assets* é executado, e após o carregamento, é executada a função *create* na qual são instanciados os *gameObjects*. Após a criação dos *gameObjects*, é executada a função *update*, na qual ocorre de fato o *loop* do jogo até que entre em uma condição de parada definida pelo desenvolvedor.

2.1.3 Bancos de Dados Não Relacionais

Banco de dados não relacionais, conhecidos como NoSQL pelo fato de não utilizar somente a Linguagem de Consulta Estruturada (SQL, do inglês *Structured Query Language*), são construídos para serem flexíveis, escaláveis e capazes de responder rapidamente às demandas de gerenciamento de dados (SADALAGE; FOWLER, 2013). Bancos de dados não relacionais não necessitam de tantas modificações para seu crescimento em escalabilidade, entretanto, como o nome já diz, eles não consideram o modelo relacional, não considerando questões relacionadas com integridade referencial e transações conforme o ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

Uma das formas de armazenamento nesse tipo de banco de dados é por meio de documentos, que podem ter diferentes estruturas. Armazenamento em Chave-Valor, que como o nome sugere, consiste em pares de chave-valor onde pode armazenar diferentes tipos de arquivos. Armazenamento de Grafos, que como o nome sugere também, os dados são representados como nós e as relações entre os dados são representadas por

arestas. Armazenamento em famílias de colunas, no qual os dados são armazenados de forma semelhante aos bancos de dados relacionais, sua diferença se dá na performance de entradas e saídas, o qual é superior, e como mencionado anteriormente, focados em escalabilidade.

Uma estrutura amplamente utilizada é a Notação de Objetos Javascript (JSON, do inglês *JavaScript Object Notation*), que é um tipo de arquivo de texto bastante utilizado como arquivo de intercâmbio de dados entre sistemas. A notação consiste na utilização de pares chave-valor para o armazenamento de dados. A principal vantagem de se utilizar arquivos JSON é a alta portabilidade, pois a maioria das linguagens de alto nível tem suporte para tratar arquivos JSON, além de serem facilmente lidos e entendidos por desenvolvedores. Assim, este trabalho utiliza arquivos JSON para armazenamento dos dados, cuja estrutura será apresentada no próximo capítulo.

2.2 Trabalhos Relacionados

Foi realizada uma busca exploratória na literatura sobre trabalhos a este, que foram divididos em duas categorias: (i) criação/gerenciamento dinâmica de fases e (ii) jogos digitais no contexto de Química. Para a categoria de criação de fases, foram encontrados métodos propostos que englobam áreas da inteligência artificial para a criação de níveis de modo procedural, por exemplo, para a categoria de jogos digitais, existem jogos que englobam diversos tópicos, desde o nome de cada elemento químico até ao balanceamento de equações químicas.

2.2.1 Criação de Fases

2.2.1.1 Adaptive Game Level Creation through Rank-based Interactive Evolution

O modelo proposto nesse trabalho consiste na evolução interativa baseada em classificação, utilizando o método *Rank-based Interactive Evolution* (RIE) para personalizar o conteúdo do jogo com base em um processo de adaptação de duas etapas: enquanto o conteúdo é evoluído para maximizar um conjunto de condicionamentos, o impacto de cada condicionamento é ajustado para se aproximar da preferência do usuário. A estrutura do RIE aprimora as capacidades de escolha baseada em evolução interativa, pois oferece mais controle aos usuários e permite um processo mais rápido se aproximando de seu gosto pessoal. O procedimento baseado em pesquisa para a geração de mapas de jogos de estratégia é usada como uma plataforma de teste para o RIE, com mapas sendo avaliados por heurísticas inspiradas na jogabilidade propriedades como navegação espacial e distribuição de recursos bem como propriedades visuais, como simetria (LIAPIS et al., 2013).

2.2.1.2 Computer-Assisted Game Level Authoring

O trabalho consiste no desenvolvimento de uma ferramenta chamada Sentient Sketchbook⁶ que permite a criação de jogos por meio de uma interface de projeto assistida computacionalmente. A ferramenta dá suporte por meio de automatização de avaliações de mapas, visualizando-os na tela e propondo alternativas de *design*. A criação de sugestões de mapas combina o paradigma Viável-Inviável de duas populações com a pesquisa de originalidade e se mostra mais robusto do que a pesquisa restrita ao estado da arte nos casos em que a descoberta de indivíduos viáveis é difícil.

2.2.1.3 A genetic approach in procedural content generation for platformer games level creation

O trabalho propõe um algoritmo genético para auxiliar o *level designer* a criar jogos com diferentes níveis de dificuldade. Esse algoritmo é capaz de criar ritmos-alvo com uma função de aptidão que é a soma de quatro equações. O sistema pode produzir pelo menos 50.000 ritmos diferentes com diferentes dificuldades. O gerador de geometria gera, então, geometrias correspondentes de ritmos produzidos e pode ter geometria ilimitada para cada ação (MOGHADAM; RAFSANJANI, 2017).

2.2.1.4 BuildBox

O software BuildBox⁷ é uma *engine* de criação de jogos sem a necessidade de programação. Entretanto, é limitado aos tipos de jogos que podem ser criados (mecânicas), dada sua simplicidade de interface e a não utilização de codificação (e, sim por meio de blocos de arrastar e soltar), porém há a possibilidade de se criar jogos 2D e 3D. Como não é necessário programar os jogos, é também limitado as possibilidades de criação que envolvam lógicas mais complexas.

A proposta do BuildBox consiste na criação da fase completa, e não em módulos. Dessa forma, apesar de permitir criar a fase sem precisar saber alguma linguagem de programação, não é possível gerenciar trechos da fase a fim de se criar novas fases.

2.2.1.5 FazGame

O FazGame⁸ é um software que permite a criação de jogos do tipo “apontar e clicar”, nos quais o usuário utiliza especificamente este meio para jogar. A limitação deste software é o desenvolvimento de um estilo único de jogo. No entanto, o desenvolvimento também não é necessário conhecimento de programação, pois utiliza da mecânica de arrastar e soltar blocos de código.

⁶ <<https://www.um.edu.mt/library/oar/handle/123456789/29607>>

⁷ <<https://buildbox.com/>>

⁸ <<https://www.fazgame.com.br/>>

Como mencionado anteriormente, o software é capaz de desenvolver jogos de apenas um tipo. Diferentemente da proposta deste trabalho, uma fase pode ser composta por módulos que exercitam diferentes mecânicas de jogo.

2.2.2 Jogos no Contexto de Química

2.2.2.1 Element Hangman

O jogo Element Hangman⁹ consiste em um jogo da forca, no qual o objetivo é descobrir o nome de elementos químicos dada uma dica. Há a possibilidade de se abrir uma tabela periódica e ter acesso às informações de todos os elementos, tais como nome, número atômico, massa atômica, aplicações, distribuição eletrônica, entre outros. Conta com uma interface bem simples e intuitiva, é interessante em alguns pontos como curiosidades sobre os elementos. O jogo foi desenvolvido para Web, sendo possível utilizar tanto em dispositivos *mobile* quanto desktop, porém, no caso *mobile*, não é tão atrativo pois não apresenta responsividade.

2.2.2.2 Balancing Chemical Equations

O jogo Balancing Chemical Equations¹⁰ trabalha o conceito de balanceamento de equações, no qual o jogador seleciona o número de reagentes para resultar em um produto. O jogador tem duas chances de acertar, caso erre ambas as tentativas, é dada a opção de o jogador ver a resposta e passar para a próxima fase. Também é de interface bem simples e intuitiva, por ser apenas de balanceamento de equações acaba sendo bem limitado a um único contexto. O jogo foi desenvolvido para Web, podendo ser utilizado em qualquer navegador, entretanto, quando acessado por dispositivos móveis, apresenta problemas de jogabilidade, como dificuldade de cliques nos botões devido ao seu tamanho.

2.2.2.3 Outer Space Molecule Chase

O jogo Outer Space Molecule Chase¹¹ consiste em um jogo de labirinto no qual o objetivo é coletar átomos e criar compostos para ajudar o jogador a chegar ao final do labirinto. Alguns níveis são bem grandes e repetitivos, tornando-os pouco atrativos. O jogo possui suporte multiplataforma dando a oportunidade de o jogador utilizar toques para se mover.

⁹ <<https://education.jlab.org/elementhangman/>>

¹⁰ <http://phet.colorado.edu/sims/html/balancing-chemical-equations/latest/balancing-chemical-equations_en.html>

¹¹ <<https://www.acs.org/content/acs/en/education/whatischemistry/adventures-in-chemistry/games/outer-space-molecule-chase/chase-game.html>>

2.2.2.4 Quiz Molecular

O jogo Quiz Molecular (SILVA; LOJA; PIRES, 2020) é um jogo de perguntas e respostas clássico, disponibilizado apenas para a plataforma *mobile*. No jogo, o usuário seleciona uma fase e dentro de cada fase existe uma série de perguntas, cada uma contendo quatro respostas. Ao responder todas as perguntas, o jogador conclui uma fase e ao concluir uma fase, um painel é aberto mostrando quantas perguntas o jogador acertou com a opção de ir para o próximo nível ou voltar para o menu.

2.2.2.5 IFQuimical

O jogo IFQuimical é um jogo de perguntas e respostas, no qual o jogador escolhe um ou mais modos (simulado, balanceamento de equações químicas ou materiais de laboratório). O jogo é montado aleatoriamente de acordo com os tópicos escolhidos pelo jogador. No caso do modo de balanceamento de equações, ao final, o jogador recebe as informações do elemento correto. No caso do simulado, ao final, o jogo mostra uma tela de gabarito com cada alternativa, mostrando se acertou ou errou e dá a opção de abrir as questões. A seção dos materiais de laboratório é mais como um informativo, disponibilizando os materiais e a opção de abrir as informações de cada material, como utilizar, quando utilizar, características etc. O jogo não é multiplataforma e foi desenvolvido para dispositivos desktop (LEITE, 2018).

2.2.2.6 Comparativo

A Tabela 2 apresenta um comparativo entre os trabalhos relacionados apresentados nesta seção. Para comparação, considera-se “Multiplataforma” a capacidade de ser utilizado em dispositivos de diferentes sistemas operacionais e diferentes características de *hardware*, como tamanhos de tela. O item “Exercitar a TP” está relacionado com a utilização de elementos químicos em um contexto de jogos onde possam ser observadas suas características. Por fim, o item “Criação de fases” se refere à capacidade de gerenciar a criação de fases de um jogo a partir de módulos.

Tabela 2 – Tabela comparativa de jogos educativos no contexto de Química.

	Multiplataforma	Exercita a TP	Criação de fases
Element Hangman	X	X	
Balancing Chemical Equations		X	
Outer Space Molecule Chase		X	
Quiz Molecular		X	
IFQuimical		X	
Proposta deste trabalho	X	X	X

Os jogos, no geral, apresentam semelhanças nos métodos utilizados para aprendizagem. A abordagem das questões como perguntas faz com que o usuário preste atenção em

certos elementos do nível para que consiga prosseguir. Como mencionado anteriormente, existem diversos tipos de jogos, cada um focado para uma ou mais plataformas. No entanto, poucos são considerados completamente multiplataforma, mesmo sendo Web apresentam problemas quando executados em dispositivos móveis, e nenhum permite a criação dinâmica de fases pelo professor, sendo sempre estáticas. Além disso, todos os trabalhos apresentados nesta seção exercitam conceitos da Tabela Periódica.

Desenvolvimento

Este trabalho propõe uma arquitetura de software que permite a criação dinâmica de fases de jogos digitais, a partir de módulos previamente definidos, com o intuito de que usuários não técnicos (nesse caso, professores da educação básica) possam apenas escolher quais módulos farão parte da fase de jogo criada.

3.1 Metodologia

Neste trabalho, a metodologia se baseou na abordagem *Design Science Research* (DSR) e foi guiada pelo *framework* conceitual no contexto de Sistemas de Informação (HEVNER et al., 2004), que consiste em um processo de iteração onde é verificada a necessidade de um negócio, denominado ambiente, e por meio de um processo iterativo de construção, avaliação e refinamento de artefatos tecnológicos, a pesquisa gera uma base de conhecimento. No contexto deste trabalho, o ambiente se define por pessoas (professores), organizações (instituições de ensino) e tecnologias (HTML, CSS, JavaScript); a base de conhecimento é definida como: fundamentação (Phaser.js, Node.js; Tabela Periódica) e metodologias (modelo iterativo incremental); e a pesquisa que se divide em Desenvolvimento (Prova de conceito) e Avaliação (experimental). Ressalta-se que foi utilizada apenas uma iteração do modelo neste trabalho.

A implementação de um protótipo funcional como prova de conceito mostrou, por si, a viabilidade do modelo proposto, indicando a facilidade em se criar novas fases de jogo por meio de módulos pré-programados. Apesar disso, é relevante incluir potenciais usuários finais durante o processo de design de soluções tecnológicas com vistas a identificar problemas específicos de domínio. Assim, decidiu-se por realizar uma entrevista com um professor de Química da Educação Básica.

3.2 Proposta

Esta seção apresenta detalhes arquiteturais da proposta criada. A ideia é permitir que não desenvolvedores consigam criar novas fases de jogo por meio do gerenciamento de módulos pré-programados, que são colocados em uma ordem definida pelo usuário, respeitando algumas regras impostas pela definição de alguns módulos, como o menu, que deve ser o módulo inicial, e o módulo de quiz, que deve ser o módulo final.

Os dados são persistidos por meio de arquivos JSON. Existem cinco arquivos em que são armazenados dados referentes à Usuários, Módulos, Níveis, *Quizzes* e Turmas, estruturados conforme o Diagrama de Classes mostrado na Figura 1.

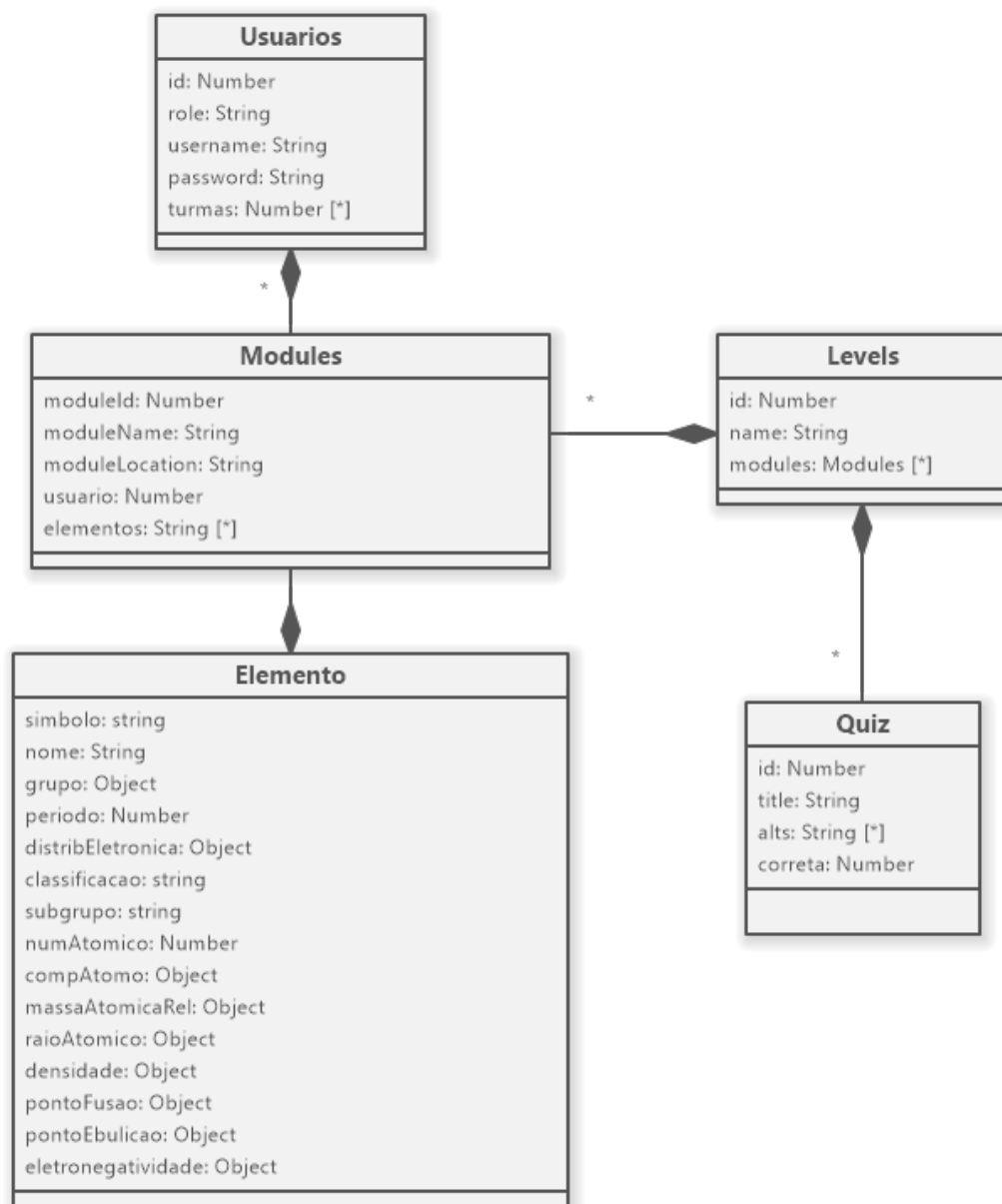


Figura 1 – Diagrama UML representando os arquivos JSON.

De acordo com o diagrama apresentado, um nível (que representa uma fase de jogo) é um agregado de módulos, onde cada módulo tem suas características próprias e é adicionado/criado por algum usuário. Os usuários possuem as turmas nas quais se relacionam as fases, permitindo que alunos cadastrados em uma determinada fase joguem apenas as fases disponíveis para aquela turma. Além disso, um quiz, que consiste em um título, suas alternativas e a alternativa correta, também é vinculado a uma fase, sendo que um mesmo nível pode ter mais de um quiz relacionado a ele.

Como a proposta tem foco principal na disciplina de Química, um módulo da fase possui um *array* de elementos químicos que são vinculados a ele no momento de sua criação. O modelo desenvolvido foi pensado para que possa atender genericamente outras disciplinas sem que sejam necessárias muitas alterações. Uma possibilidade para outras disciplinas é a alteração do *array* de elementos químicos para um *array* que contenha dados do contexto desejado.

Como mencionado nos objetivos do projeto, foi proposto que o envio de registros de interação (*logs*) por meio de uma API (*Application Programming Interface*) Web. O *log* é enviado dentro de cada módulo de jogo por meio de chamada assíncronas Ajax. Apesar da API fazer parte do projeto como um todo, sua proposta não é objeto deste trabalho.

Cada módulo de jogo é criado e disponibilizado no sistema apenas por administradores, os quais terão de selecionar o arquivo .js do módulo para cadastrá-lo, como mostrado na Figura 2. Por padrão, cada módulo é construído por meio de uma classe cujo nome deve ser igual ao nome utilizado para chamar o arquivo (nome do módulo). A Figura 2 apresenta um exemplo de um módulo com nome 'Exemplo'.

```
class Exemplo extends Phaser.Scene {
  constructor() {
    super({
      key: 'Exemplo',
      active: false,
      visible: false
    })
  }
  preload() {
    //Carregamento dos assets
  }
  create() {
    //Estágio de criação dos gameObjects
  }
  update() {
    //Loop do game
  }
}
```

Figura 2 – Exemplo de uma classe que representa um módulo de fase de jogo.

O Diagrama de Sequência apresentado na Figura 3 mostra o processo de criação de uma fase. O usuário clica no botão relacionado com a criação de uma fase de jogo na interface, que faz com que o *browser* dispare uma chamada para a rota ‘/menu’, que, por sua vez, faz uma requisição para a API para buscar os módulos de fase disponíveis para que o usuário possa adicioná-los na fase. A API retorna os módulos ao MenuController que manipula esses dados e os envia tratados ao *browser* para que sejam exibidos ao usuário. A partir deste momento, as interações de adicionar e remover módulos são efetuadas em interações entre o usuário e o *browser* apenas. Após o fim dessas interações, o usuário digita o nome que deseja dar à fase e, em seguida, clica no botão criar fase. Nesse momento, o *browser* dispara uma chamada para a rota ‘/addFase’ que, por sua vez, faz uma nova requisição para a API para cadastrar a nova fase no arquivo ‘levels.json’ e devolve uma resposta com o status da operação.

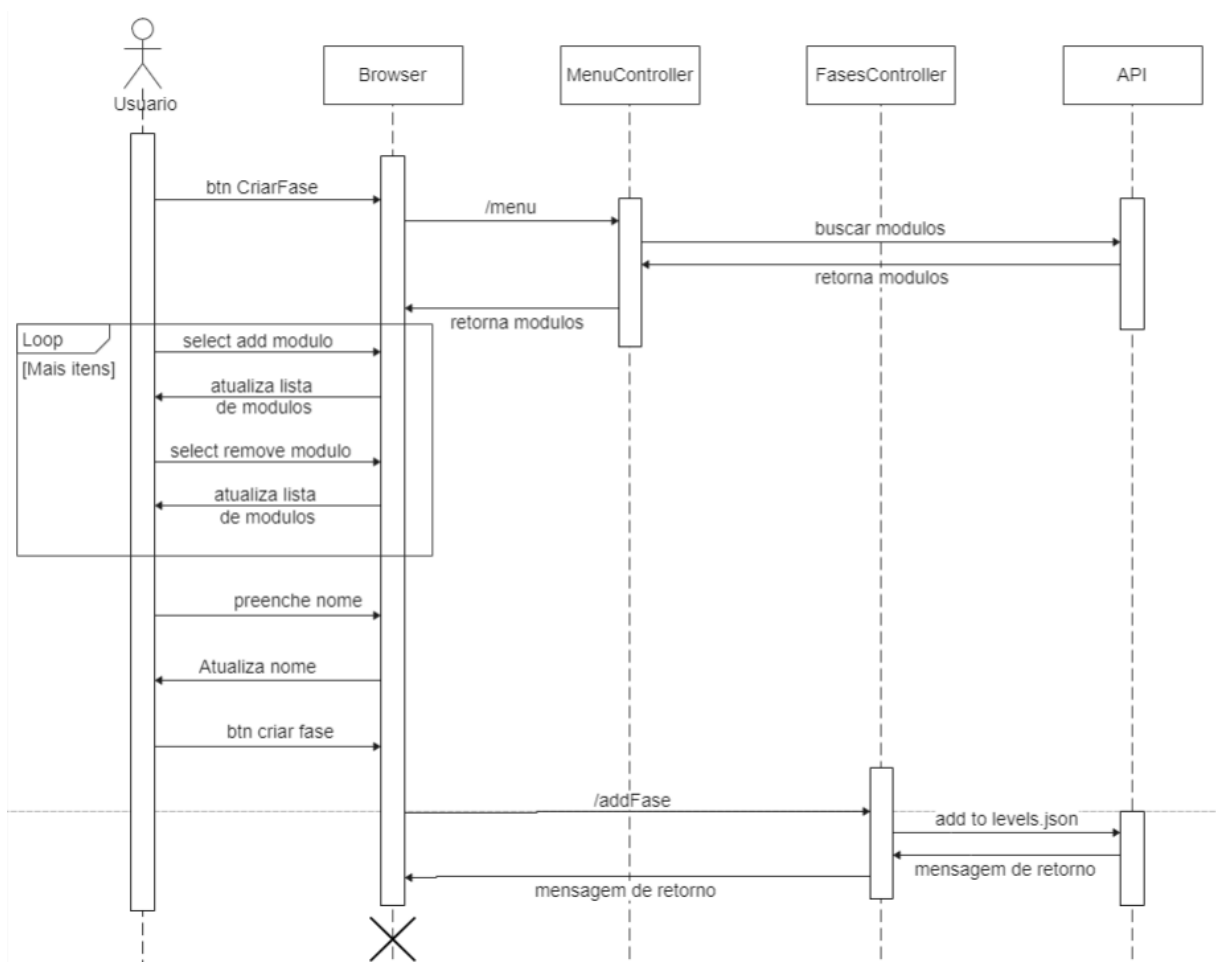


Figura 3 – Diagrama de Sequência de criação de fase.

O Diagrama de Sequência apresentado na Figura 4 mostra o processo de abertura de uma fase de jogo com os módulos já organizados. Ao clicar no botão “jogar”, o navegador acessa a rota ‘/fase/:id’, na qual ele faz o envio do ID da fase para o *fasesController* que dispara um evento de buscar fase pelo ID para a API, e o retorno volta para o

fasesController, que por sua vez, repassa a fase ao navegador. Após a busca da fase, se inicia um *loop* no qual consiste no processo de carregar e executar uma fase. Na etapa de carregar a fase, o navegador dispara uma chamada para carregar módulo que, na sequência, chama a função `preload()` nativa do Phaser.js, cuja função é carregar os *assets* e retorná-los carregados ao módulo. Ao fim do carregamento dos *assets*, o módulo executa a função `create()`, também nativa do Phaser.js, cujo objetivo é criar os *gameObjects* e os retorná-los ao módulo. Após o processo de criação dos *gameObjects*, o módulo entra no *loop* de execução do jogo, o qual é executado pela função `update()`, também nativa do Phaser.js, e que atualiza o estado do jogo e devolve a atualização ao navegador, até que uma condição de finalização seja encontrada. Assim que uma condição de finalização, o navegador faz uma chamada ao módulo requisitando o próximo módulo na fila e repete o processo até o encerramento do jogo no último módulo.

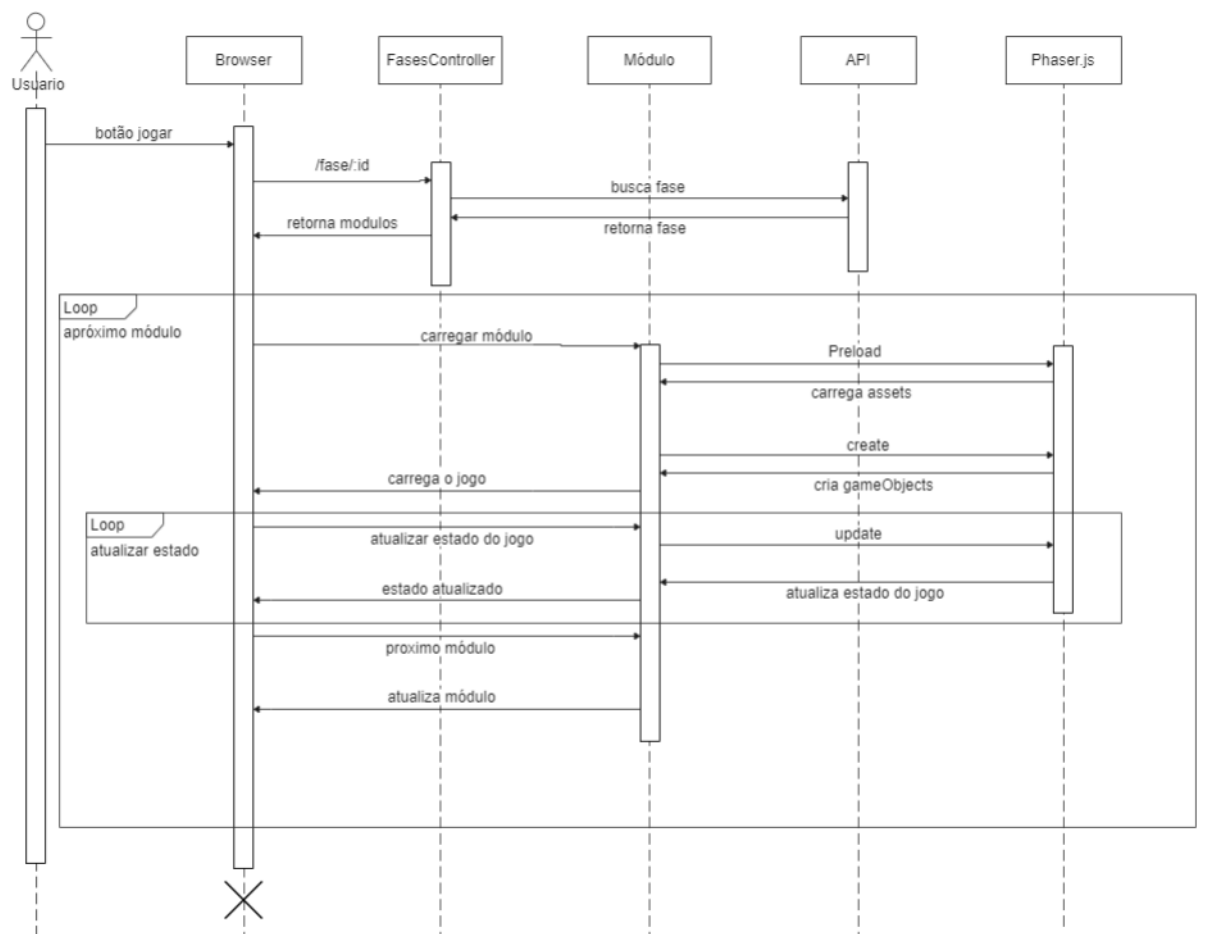


Figura 4 – Diagrama de Sequência de abertura de fase.

3.3 Prova de Conceito

Com o intuito de verificar a viabilidade da proposta e permitir a sua avaliação, foi construído um sistema Web que conta com três perfis de usuários (administrador, professor

e aluno) e algumas funcionalidades, como mostrado pelo Diagrama de Casos de Uso da Figura 5.



Figura 5 – Diagrama de Casos de Uso.

O software foi desenvolvido utilizando as linguagens de programação HTML, CSS e JavaScript. Com a utilização do servidor em Node.js e o framework phaser.js para o desenvolvimento dos módulos, seguindo a arquitetura cliente-servidor.

A Figura 6 mostra a interface de criação de fase utilizada na prova de conceito, onde cada número representa uma funcionalidade. O botão representado pelo número 1, ao ser pressionado, abre os campos para cadastro de uma nova fase. Já o botão 2, ao ser pressionado, leva a uma interface que exibe a lista das fases cadastradas pelo usuário que estiver autenticado. O item representado pelo número 3 corresponde ao elemento de interface do tipo *select* que é utilizado para adicionar um módulo à lista de módulos da fase de jogo e o item representado pelo número 4 corresponde ao elemento *select* utilizado para remover um módulo da lista de módulos da fase que está sendo criada. O item 5 corresponde ao campo de texto onde o usuário insere o nome que gostaria de dar para a fase. Ao pressionar o botão que corresponde ao item 6, o sistema faz as verificações necessárias e, caso as verificações forem bem-sucedidas, uma nova fase de jogo é criada. A Figura 6 mostra, na seção destacada em vermelho, onde os módulos são apresentados na interface.

Figura 6 – Interface de criação de fases.

Figura 7 – Interface de criação de fases, com destaque para a parte onde os módulos são apresentados.

A fase de jogo criada é, então, armazenada em um arquivo JSON, conforme estrutura mostrada na Figura 8. Cada fase tem seu ID, que é controlado pelo arquivo ‘levels.JSON’, na propriedade `lastIndex`. A propriedade ‘name’ representa o nome da fase criada e a propriedade ‘modules’ apresenta uma lista de módulos adicionados na fase, na ordem em que foram cadastrados. Por padrão são inseridos dois módulos, o módulo Menu, que é inserido sempre no início de uma nova fase, e o módulo quiz, que é sempre inserido ao final da fase. No entanto, o quiz deve ser vinculado à fase posteriormente.

A segunda interface do sistema, como mencionado anteriormente, faz uma listagem

```
{
  "id": 0,
  "name": "level teste 1",
  "modules": [
    {
      "moduleId": 0,
      "moduleName": "Menu",
      "moduleLocation": "assets/modules/Menu.js"
    },
    {
      "moduleId": 2,
      "moduleName": "ModuloA",
      "moduleLocation": "assets/modules/ModuloA.js"
    },
    {
      "moduleId": 1,
      "moduleName": "Quiz",
      "moduleLocation": "assets/modules/quiz.js"
    }
  ]
},
```

Figura 8 – JSON representando uma fase criada.

para o usuário das suas fases criadas, como mostrado na Figura 9. O item representado pelo número 1, ao ser clicado, dá a opção de o usuário vincular um quiz. O item 2 dá a opção de o usuário excluir a fase em questão e o botão representado pelo item 3 possibilita que o usuário teste a fase criada. Por fim, o item 4 permite que o usuário atualize a lista de suas fases.

NOME DA FASE	MODULOS	TURMAS	CRIADOR	TESTAR
level teste 1	ModuloA	[]	criador	1 Editar 2 Remover 3 Jogar
level teste 2	ModuloB,ModuloC,ModuloA	[]	criador	Edita Remove Jogar
123	ModuloD,ModuloF,ModuloI,ModuloL	[]	criador	Edita Remove Jogar
TESTE HENRIQUE	ModuloA,ModuloB,ModuloC,ModuloD,ModuloF	[]	criador	Edita Remove Jogar

Figura 9 – Interface de listagem de fases.

A título de exemplificação de como seriam possíveis módulos pré-programados a serem gerenciados e ordenados visando a criação de uma fase completa, as figuras seguintes apresentam trechos de uma fase criada durante o projeto de iniciação científica mencionado anteriormente. A Figura 10 apresenta um exemplo do módulo de início de uma fase com um botão “Jogar”.

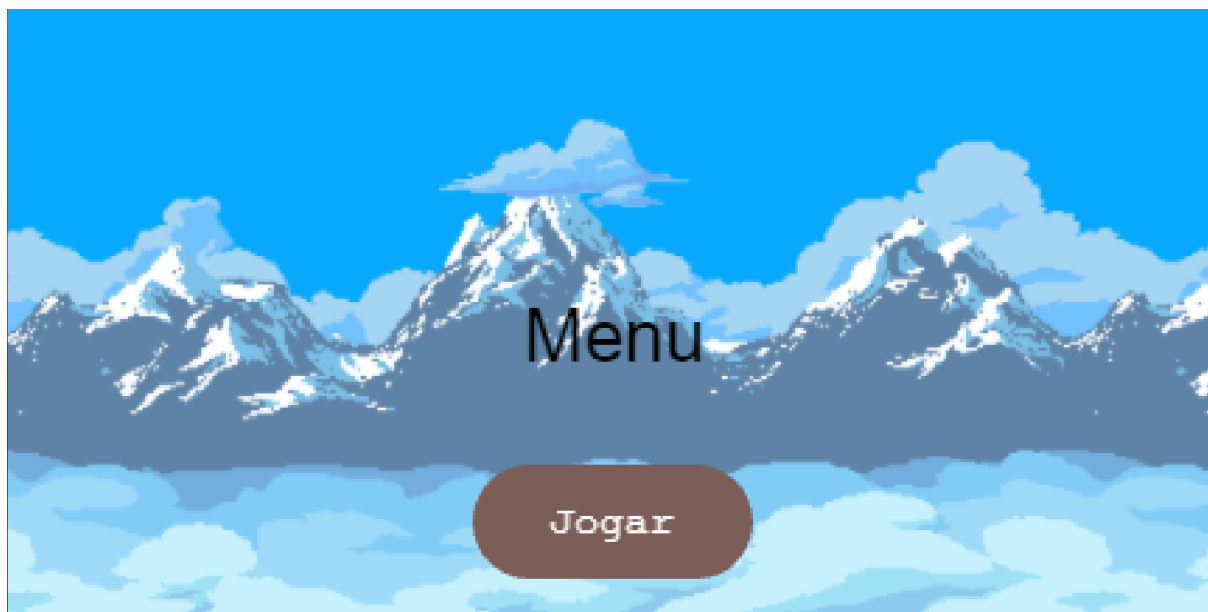


Figura 10 – Exemplo de módulo de menu.

A Figura 11 apresenta o início da fase, onde há algumas moedas a serem coletadas e uma placa indicativa que é utilizada para explicação textual sobre os elementos químicos explorados na fase. A Figura 12 apresenta a segunda parte da fase pode ser entendida como um módulo de balões utilizado para explorar certos elementos químicos (neste exemplo, observa-se os elementos Neônio - Ne, Argônio - Ar e Hélio - He). A Figura 13 apresenta a terceira parte da fase, com obstáculos que representam elementos radioativos (na imagem, observa-se o elemento Radônio - Rd). Por último, na Figura 14, o personagem já venceu todos os obstáculos e chegou ao final da fase que contém uma pergunta de múltipla-escolha (*quiz*) sobre os elementos químicos abordados, mostrada na Figura 15.

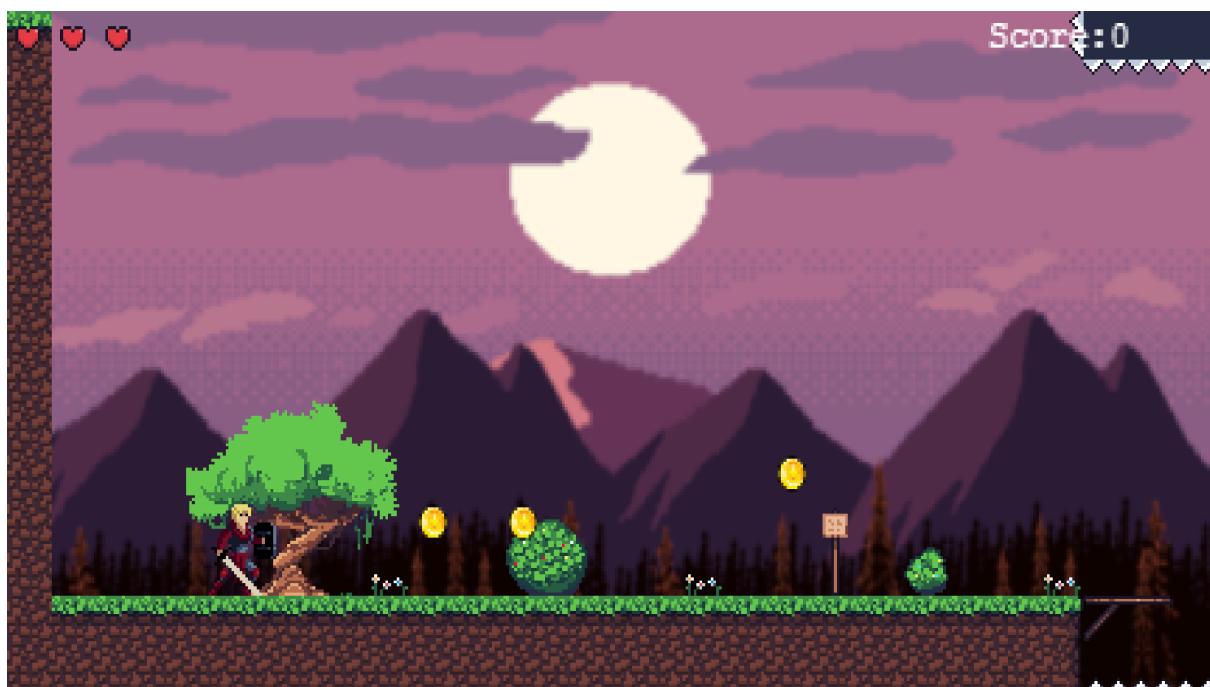


Figura 11 – Exemplo de um módulo inicial sem obstáculos.



Figura 12 – Exemplo de um módulo com balões representando elementos químicos.

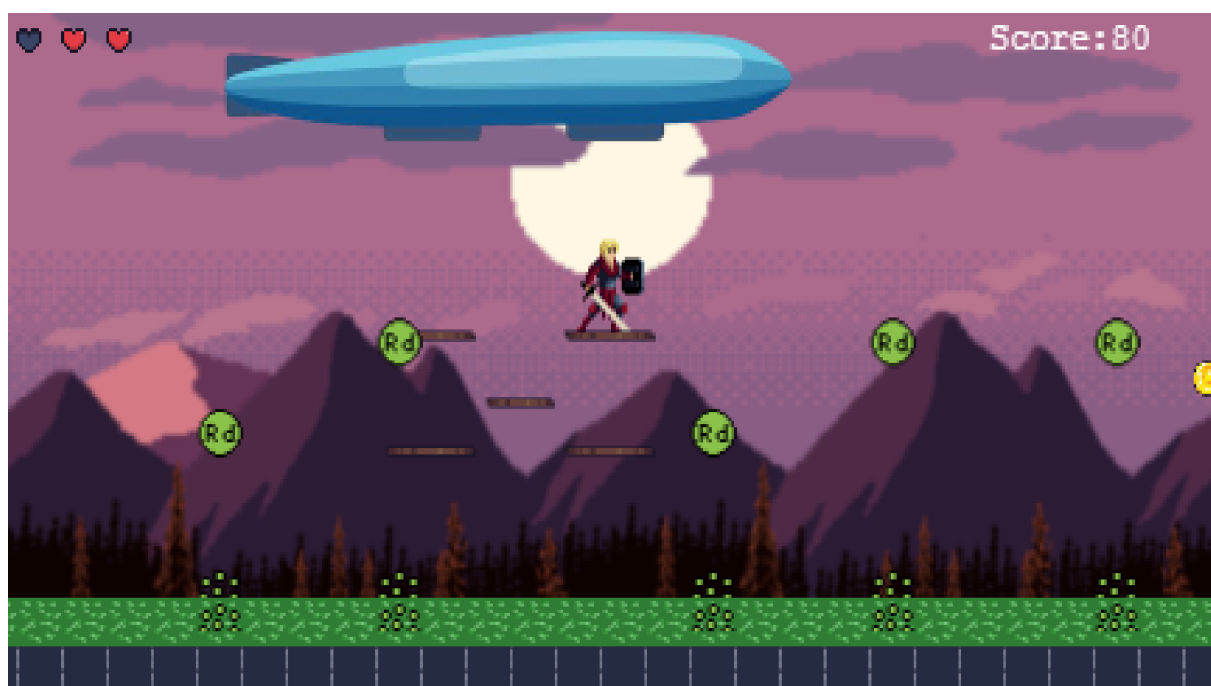


Figura 13 – Exemplo de um módulo com plataforma deslizável e obstáculos.

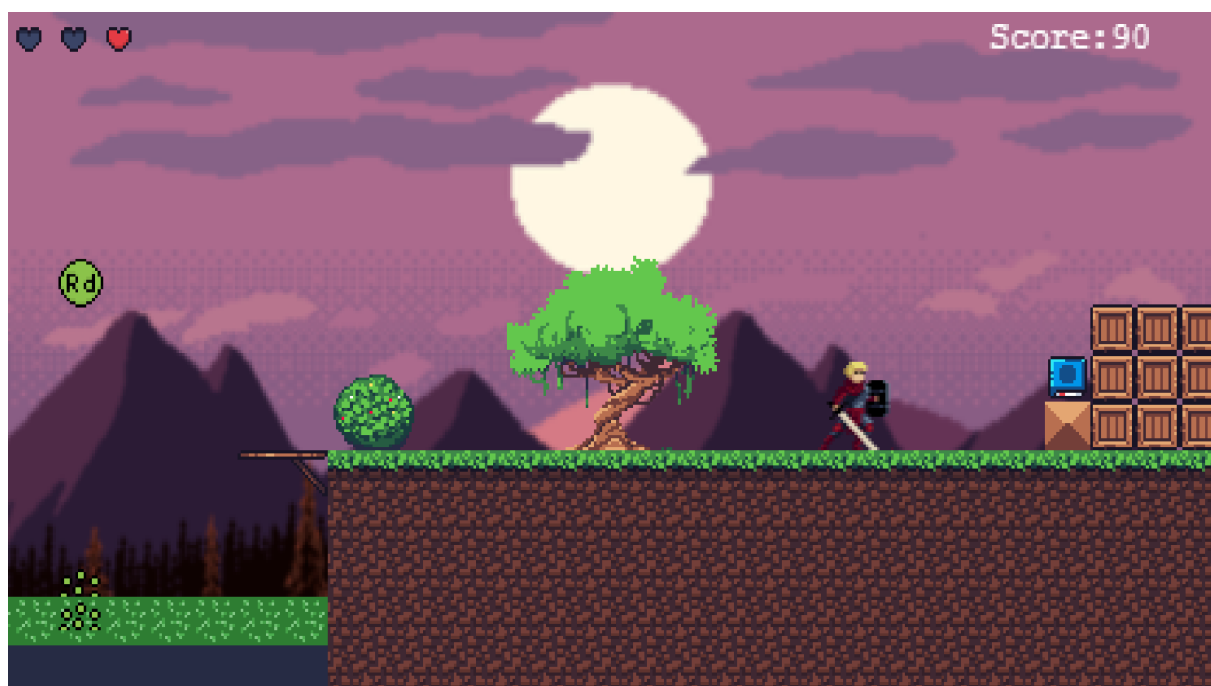
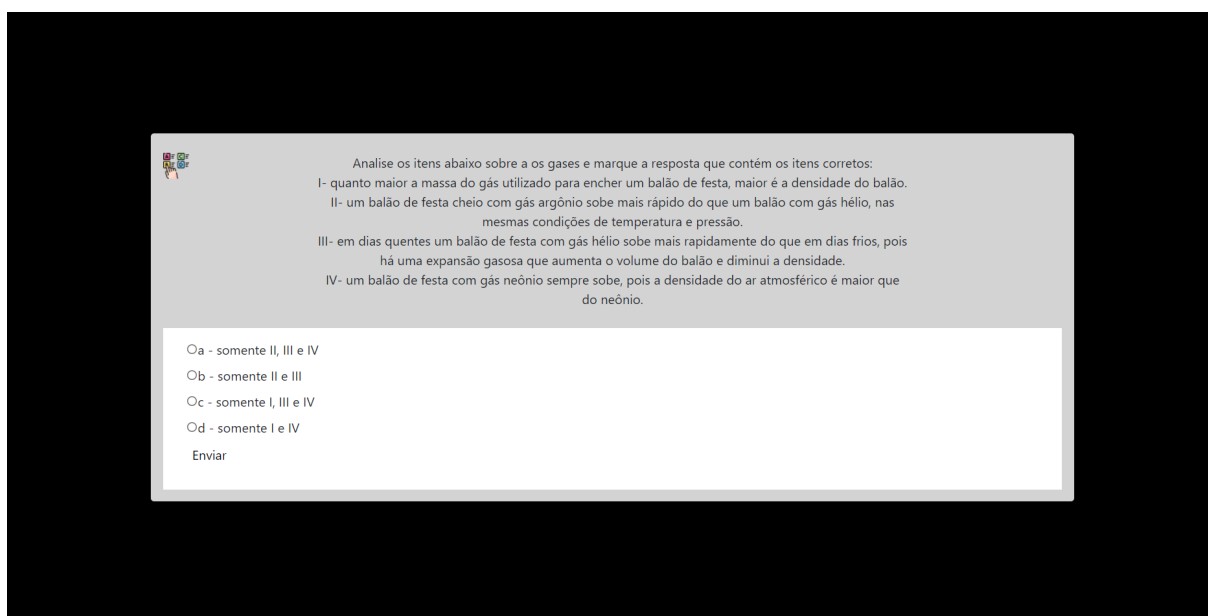


Figura 14 – Exemplo de um módulo final com um item para abertura de um *quiz*.



 Analise os itens abaixo sobre a os gases e marque a resposta que contém os itens corretos:

I- quanto maior a massa do gás utilizado para encher um balão de festa, maior é a densidade do balão.

II- um balão de festa cheio com gás argônio sobe mais rápido do que um balão com gás hélio, nas mesmas condições de temperatura e pressão.

III- em dias quentes um balão de festa com gás hélio sobe mais rapidamente do que em dias frios, pois há uma expansão gasosa que aumenta o volume do balão e diminui a densidade.

IV- um balão de festa com gás neônio sempre sobe, pois a densidade do ar atmosférico é maior que do neônio.

☐ a - somente II, III e IV

☐ b - somente II e III

☐ c - somente I, III e IV

☐ d - somente I e IV

Enviar

Figura 15 – Exemplo de módulo de quiz.

Resultados e Discussões

Existem diversas técnicas de se coletar informações durante o processo de *design* de interação visando compreender aspectos de usabilidade e do domínio do sistema proposto, tais como entrevistas, questionários, observações, grupos focais, dentre outros (ROGERS; SHARP; PREECE, 2013). Neste trabalho, a técnica de entrevista foi utilizada para obter as primeiras impressões de um potencial usuário final da ferramenta, objetivando identificar pontos de melhorias para o ciclo de *design*.

As entrevistas podem ser classificadas como estruturadas, não estruturadas, semiestruturadas, ou em grupo (ROGERS; SHARP; PREECE, 2013). O método escolhido como validação da prova de conceito proposta foi uma entrevista semiestruturada com um professor de Química, que combina características de entrevistas estruturadas e não estruturadas para dar liberdade para o entrevistado expor seus pontos de vista de forma mais espontânea devido ao diálogo que se desenvolve durante a entrevista.

4.1 Entrevista

A entrevista foi dividida em duas partes, sendo que a primeira parte consistiu no usuário realizar duas tarefas no sistema e a segunda parte consistiu em uma arguição para que o usuário pudesse contar como se sentiu em relação ao software. A entrevista foi realizada com um professor de Química do Ensino Médio, do sexo masculino, com título de Mestre e experiência de aproximadamente oito anos em docência na Educação Básica.

Na etapa de planejamento da entrevista, foram definidas as seguintes tarefas a serem executadas pelo entrevistado:

- Tarefa 1 Cadastrar uma fase que contenha cinco módulos quaisquer;
- Tarefa 2: Cadastrar uma fase que contenha cinco módulos quaisquer, efetuar a remoção de dois deles e inseri-los no final.

Na etapa de planejamento da entrevista, foram delineadas as seguintes questões a serem abordadas:

1. “Você gostaria de utilizar o software proposto com frequência?”
2. “Você achou ele mais simples ou mais complexo de mexer?”
3. “Você achou a interface do sistema agradável?”
4. “Quanto a organização da informações na tela, você acha que ficou claro?”

4.2 Resultados

A entrevista se mostrou bastante relevante, pois durante a etapa de arguição foram levantados pontos aos quais não se havia pensado anteriormente, abrindo o leque de possibilidades. O principal aspecto sugerido pelo entrevistado consistiu na dificuldade de reorganizar os módulos após inseridos na fase e foi sugerido para que pudesse criar algum mecanismo para possibilitar arrastar os módulos para reorganizá-los.

A respeito da usabilidade do sistema, o entrevistado relatou que um primeiro contato com o sistema não foi difícil, porém ficou um pouco confuso e sugeriu a incorporação de uma seção contendo um tutorial para novos usuários. O entrevistado relatou também que se sentiu confiante na utilização do software e que acredita que o software é de fácil utilização por novos usuários.

A Figura 16 mostra os pontos da interface com alterações sugeridas pelo entrevistado. A primeira proposta contempla um menu que apareceria com um evento *hover* para mostrar o que é possível criar na parte de “Gerenciar Fases”, ao invés de ter somente um *link* e dois botões dentro da página. A segunda proposta, também marcada em vermelho na imagem, corresponde à lista de módulos e está relacionada com a manipulação da ordem dos módulos selecionados para criação da fase. O entrevistado apontou a dificuldade em entender que era, de fato, a ordem em que os módulos são cadastrados para a fase. Assim, propõe-se uma nova abordagem de interface que possa permitir arrastar os módulos com o mouse para reorganizá-los, sem que fosse necessário remover todos e inseri-los novamente.

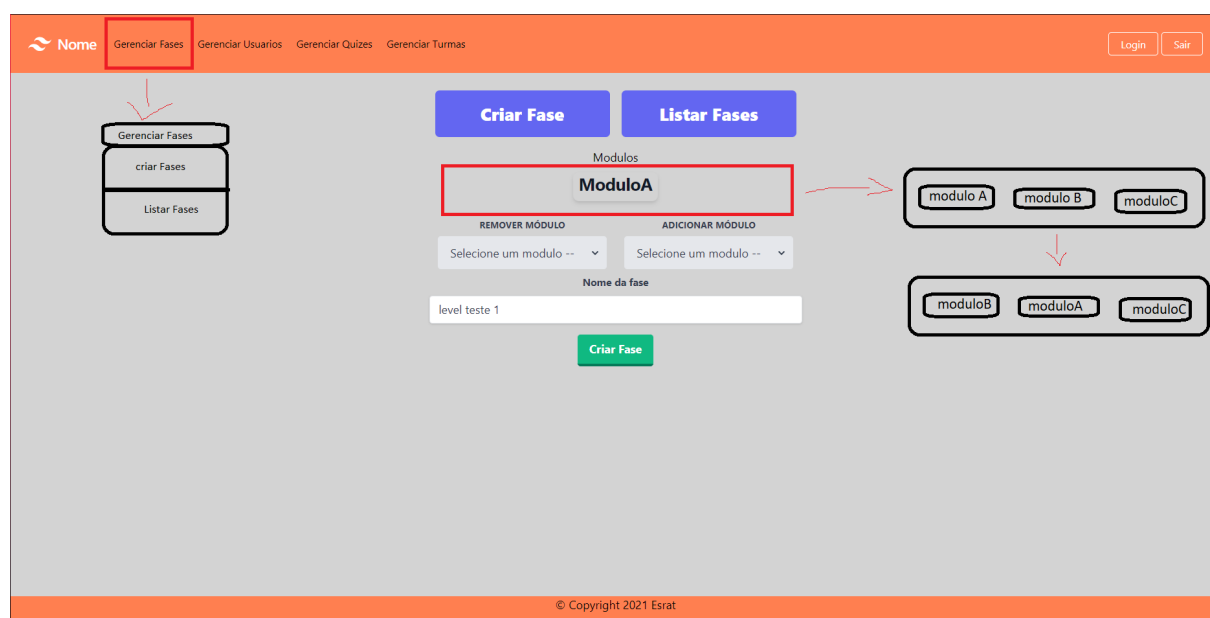


Figura 16 – Alterações propostas após entrevista com usuário.

Considerações Finais

Este trabalho apresentou uma proposta arquitetural para a criação dinâmica de fases de jogo por meio do gerenciamento de módulos pré-programados utilizando o *framework* Phaser.js. As principais contribuições do trabalho se encaixam na utilização de jogos no meio educacional, pois facilita o trabalho de criação de uma fase de acordo com o projeto arquitetura desenvolvido, que permite com que um professor crie uma fase a partir de módulos previamente programados, podendo selecionar quais módulos ele deseja adicionar e suas respectivas ordens.

Dentro da proposta arquitetural, o módulo conta com chamadas Ajax que enviam *logs* de utilização para uma API Web, que é responsável por gerenciar esses dados. Para armazenamento das fases criadas, o modelo de dados foi escolhido seguindo os conceitos NoSQL e pensando em uma posterior reestruturação da API na qual facilite sua integração à API.

Foi desenvolvido um protótipo funcional como prova de conceito, que foi avaliado por meio de uma entrevista com um professor do Ensino Médio. Os resultados indicaram uma boa aceitação e reforçaram a necessidade de prover mecanismos mais fáceis de se criar jogos digitais por pessoas não técnicas. O entrevistado apresentou seus pontos de vista de forma espontânea durante o diálogo estabelecido, apontando os pontos positivos e negativos da proposta, com sugestões válidas para inclusão de novos elementos de interface.

O trabalho se mostrou um desafio muito gratificante, pois agregou uma diversidade de novos conhecimentos, entre eles principalmente novos modos de se pensar devido as dificuldades encontradas durante o processo de desenvolvimento. A ideia de se desenvolver um software que gerencia “mini softwares” independentes e multiplataforma, fazendo com que eles conectem entre si foi um grande desafio. Foram necessárias diversas tentativas até se chegar na arquitetura proposta, além da maneira que se gerencia o processo de criação e armazenamento de fases que foi pensada seguindo o modelo NoSQL.

5.1 Trabalhos Futuros

Como este trabalho está inserido em um projeto maior que visa criar um ecossistema de software para criação de jogos educacionais no contexto da disciplina de Química, a ideia é que este trabalho seja integrado ao sistema já existente¹ como forma de auxiliar na autoria dos jogos.

A importação de jogos externos não é suportada pelo projeto no momento devido às restrições de terem que ser escritos com o *framework* Phaser.js e terem que seguir as regras mencionadas anteriormente para a abertura e execução de uma fase. Entretanto, a criação de um repositório com diversos módulos/jogos pode ser realizada para a utilização do software, gerando, assim, um maior acervo para os usuários finais.

Além disso, a criação dos módulos independentes deve ser trabalhada para facilitar sua criação pelos usuários. A ideia é criar *templates* para permitir que o professor escolha aqueles que mais se adequam à proposta de jogo criada. Ainda, é preciso projetar como cada elemento de *Game Design* (Narrativa, Estética, Mecânicas e Tecnologia) poderia ser alterado/adaptado pelo próprio professor.

5.2 Contribuições em Produção Bibliográfica

Este trabalho é continuação de um trabalho desenvolvido em um projeto de Iniciação Científica e que se integra com outros trabalhos de outros alunos. Apesar de não ser uma publicação específica da proposta deste trabalho, houve uma publicação de um artigo resumido na Trilha Educação do XX Simpósio Brasileiro de Games e Entretenimento Digital (SBGames 2021) que apresenta a integração dos trabalhos mencionados (DAIREL et al., 2021).

¹ <https://quimicotgames.com/>

Referências

- AMATYA, S.; KURTI, A. Cross-platform mobile development: Challenges and opportunities. In: TRAJKOVIK, V.; ANASTAS, M. (Ed.). **ICT Innovations 2013**. Heidelberg: Springer International Publishing, 2014. p. 219–229. ISBN 978-3-319-01466-1. Citado na página 13.
- DAIREL, J. G. de M. et al. Em direção a um ecossistema de software para apoio ao ensino de química por meio de jogos digitais. In: **Proceedings of SBGames 2021**. Gramado/RS: SBC, 2021. Citado na página 37.
- GUIMARÃES, L. M. **Jogos Didáticos como Recurso Educacional para Aprendizagem em Química**. 211 p. Dissertação (Mestrado) — Universidade Federal Fluminense, 2021. Citado na página 12.
- HEVNER, A. et al. Design science in information systems research. **MIS Quarterly**, MISRC, v. 28, n. 1, p. 75–105, 2004. Citado na página 21.
- LAY, A.-N.; OSMAN, K. **Developing 21st Century Chemistry Learning through Designing Digital Games**. [S.l.]: Seyit Ahmet KIRAY, 2018. 81 - 92 p. Citado na página 12.
- LEITE, J. A. e Arthur Almeida e Marcos Vieira e Adriano Carvalho e M. Ifquimical: uma proposta de mediação no processo ensino-aprendizagem de química. **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)**, v. 29, n. 1, p. 61, 2018. ISSN 2316-6533. Disponível em: <<https://br-ie.org/pub/index.php/sbie/article/view/7954>>. Citado na página 19.
- LIAPIS, A. et al. Adaptive game level creation through rank-based interactive evolution. In: **2013 IEEE Conference on Computational Intelligence in Games (CIG)**. [S.l.: s.n.], 2013. p. 1–8. Citado na página 16.
- MAYER, R. E. Computer games in education. **Annual Review of Psychology**, v. 70, n. 1, p. 531–549, 2019. PMID: 30231003. Disponível em: <<https://doi.org/10.1146/annurev-psych-010418-102744>>. Citado 2 vezes nas páginas 9 e 12.
- MOGHADAM, A. B.; RAFSANJANI, M. K. A genetic approach in procedural content generation for platformer games level creation. In: **2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)**. [S.l.: s.n.], 2017. p. 141–146. Citado na página 17.

- NOVAES, M. A. B. d. et al. Active methodologies in the teaching and learning process: Emerging didactic alternatives. **Research, Society and Development**, v. 10, n. 4, p. e37710414091, 2021. Citado na página 9.
- OLIVEIRA, F. M.; HILDEBRAND, H. R. Ludicidade, ensino e aprendizagem nos jogos digitais educacionais. **Informática na Educação: Teoria & Prática**, v. 21, n. 1, p. 106–120, 2018. Citado na página 9.
- PAIVA, C. A.; TORI, R. Jogos digitais no ensino: Processos cognitivos, benefícios e desafios. In: **Proceedings of SBGames 2017**. [S.l.: s.n.], 2017. p. 1052–1055. Citado na página 9.
- RASTEGARPOUR, H.; MARASHI, P. The effect of card games and computer games on learning of chemistry concepts. **Procedia - Social and Behavioral Sciences**, v. 31, p. 597–601, 2012. ISSN 1877-0428. World Conference on Learning, Teaching & Administration - 2011. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877042811030400>>. Citado 2 vezes nas páginas 9 e 10.
- ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação: Além da interação humano-computador**. 3a. ed. Porto Alegre: Bookman, 2013. Citado na página 33.
- SADALAGE, P. J.; FOWLER, M. **NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota**. São Paulo: Novatec, 2013. Citado na página 15.
- SAVI, R.; ULBRICHT, V. R. Jogos digitais educacionais: benefícios e desafios. **RENOTE - Novas Tecnologias na Educação**, v. 6, n. 2, 2008. Citado na página 12.
- SILVA, E. S.; LOJA, L. F. B.; PIRES, D. A. T. Quiz molecular: Aplicativo lúdico didático para o ensino de química orgânica. **Revista Prática Docente**, v. 5, n. 1, p. 172–192, maio 2020. Citado na página 19.
- VALENTE, J. A. Por que o computador na educação? In: VALENTE, J. A. (Ed.). **Computadores e Conhecimento: Repensando a Educação**. Campinas: Unicamp/Nied, 1993. cap. 2. Citado na página 9.