UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Tecnologia

Bruno Rossetto Pereira

Software para criação de jogos educacionais

Limeira

2021

Bruno Rossetto Pereira

Software para criação de jogos educacionais

Trabalho apresentado à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Marcos Augusto Francisco Borges

Este exemplar corresponde à versão final do trabalho desenvolvido por Bruno Rossetto Pereira e orientado pelo Prof. Dr. Marcos Augusto Francisco Borges.

Limeira

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Luiz Felipe Galeffi - CRB 8/10385

P414s

Pereira, Bruno Rossetto, 1997-
Software para criação de jogos educacionais / Bruno Rossetto Pereira. – Limeira, SP : [s.n.], 2021.

Orientador: Marcos Augusto Francisco Borges.
Trabalho de Conclusão de Curso (graduação) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Jogos educacionais. 2. Software - Desenvolvimento. I. Borges, Marcos Augusto Francisco,1971-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações adicionais, complementares

Título em outro idioma: Educational game creation software
Palavras-chave em inglês:
Educational games
Computer software - Development
Titulação: Bacharel
Banca examinadora:
Marcos Augusto Francisco Borges [Orientador]
Gabriel Coutinho Natucci
Ulisses Martins Dias
Data de entrega do trabalho definitivo: 20-12-2021

Folha de Aprovação

Abaixo se apresentam os membros da comissão julgadora desta monografia para o Título de Bacharel em Sistemas de Informação, a que se submeteu o aluno Bruno Rossetto Pereira, na Faculdade de Tecnologia – FT/UNICAMP, em Limeira/SP.

Prof. Dr. Marcos Augusto Francisco Borges
Presidente da Comissão Julgadora

Gabriel Coutinho Natucci
Universidade Estadual de Campinas – UNICAMP

Prof. Dr. Ulisses Martins Dias
Universidade Estadual de Campinas – UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Trabalhos de Conclusão de Curso e na Secretaria de Graduação da FT.

Resumo

Os jogos têm sido cada vez mais utilizados para outras iniciativas além do entretenimento, como aprendizado, ganhos em criatividade, autonomia e concentração. Eles também potencializam o raciocínio lógico e o pensamento crítico. Hoje, a criação de jogos não é simples. Por esse motivo, grande parte dos educadores ainda não fazem uso desta abordagem em suas práticas didáticas. Este trabalho tem como objetivo a construção de um protótipo de um sistema para a criação de jogos educacionais, a "*Papiro Engine*". O objetivo do sistema é facilitar a criação de jogos através de interfaces simples, intuitivas e com a possibilidade de expansão do produto em um momento futuro, a partir da inclusão de novas funcionalidades.

Abstract

Games have been increasingly used for initiatives other than entertainment, such as learning, gains in creativity, autonomy, and concentration. They also enhance logical reasoning and critical thinking. The creation of games is not a simple task, for this reason, it is not usual that educators use this approach in their teaching practices. This work aims to build a prototype of a system for the creation of educational games, the “Papiro Engine”. It is expected that “Papiro Engine” will facilitate the creation of games through simple, intuitive interfaces and with the possibility of expanding the product in the future, based on the inclusion of new features.

Lista de Figuras

Sumário

Capítulo 1

Introdução

A evolução da tecnologia fez com que a demanda por agilidade aumentasse significativamente. Com o avanço da tecnologia ao longo do tempo, os esforços para realizar as tarefas básicas e até as mais complexas do dia a dia foram diminuídos, fazendo com que o senso de urgência dos indivíduos aumentasse.

Observando o modelo de educação atual, nota-se uma discordância entre a vida fora das salas de aula e dentro delas. Dentro da sala de aula, dificilmente se observa o uso de tecnologias comuns ao cotidiano, como o uso de podcasts, jogos e mesmo outras formas tradicionais de aprendizado, como cursos online massivos (MOOC) e vídeos-tutoriais disponíveis em plataformas como Youtube[1], sendo mais comum os procedimentos educacionais serem aplicados por meio de papéis, cadernos ou qualquer outro tipo de anotação manual.

O surgimento de sistemas informatizados trouxe grandes possibilidades para a educação. Os jogos digitais, por exemplo, "são parte da vida de 3,1 bilhões de pessoas ao redor do mundo, ou cerca de 40% da população mundial" (DFC Intelligence, 2020). Comumente vistos como forma de diversão e lazer, jogos também podem representar um papel importante na educação. Jogos apresentam um grande potencial de ensino-aprendizado, tendo obtidos resultados positivos em diversas áreas, como, por exemplo, nas áreas de exatas e programação, como pode ser observado nas revisões sistemáticas publicadas por Santos e Bíscaro (2019) e Souza e Coutinho (2016). Neste contexto, Kishimoto (2009) afirma que "Os jogos de construção são considerados de grande importância por enriquecer a experiência sensorial, estimular a criatividade e desenvolver habilidades".

Jogos cujos objetivos vão além do entretenimento, sendo usados também para fins de aprendizado, marketing e simulação de ambientes reais são conhecidos por jogos sérios. Herpich (2014) afirma que os jogos sérios surgem como tecnologia educacional por meio de softwares interativos que tem como objetivo simular os ambientes reais. É possível utilizar esses jogos em sala de aula em uma prática conhecida como aprendizado baseado em jogos (ou *game-based learning*) como definido por Trybus (2013), porém essa prática não é ainda muito usual. Um dos possíveis motivos para o pouco uso dessa prática no ensino pode ser a dificuldade da criação desses jogos por parte dos professores. Segundo Theodosiou e Karasavvidis (2015), alguns professores têm domínio do conteúdo de suas matérias, porém não tem as habilidades necessárias para traduzir o conteúdo acadêmico em tarefas de jogo. Além disso, outra

[1] https://www.youtube.com/

dificuldade é a necessidade de domínio em diferentes áreas de conhecimento, como design gráfico, programação, animação, design de áudio, escrita, design interativo e o conteúdo do jogo.

Existem soluções desenvolvidas para facilitar a criação de jogos, porém, elas não atendem totalmente as necessidades de usuários que não estão familiarizados com a área de desenvolvimento, pois necessitam de conhecimento básico em programação ou não possuem interfaces muito intuitivas.

1.1 Objetivo

Este trabalho busca criar um protótipo que possibilite facilitar a criação de jogos sérios chamado de *Papiro Engine*, possibilitando o uso deste tipo de software com mais frequência na área educacional. Para tal objetivo, foi proposto um protótipo com interfaces simples, de fácil entendimento, para facilitar a criação de jogos com foco na educação por parte de pessoas sem muito conhecimento sobre a área de desenvolvimento de jogos. Para facilitar ainda mais o uso e criação de jogos por parte de educadores, o protótipo também oferece integração com um módulo de *analytics*, disponibilizando informações mais detalhadas para uma melhor avaliação dos alunos.

1.2 Organização do Texto

Este trabalho está organizado como se segue. No Capítulo 2, são apresentadas as plataformas já existentes que possuem o objetivo de facilitar o desenvolvimento de jogos e um jogo chamado “Reigns”, que juntamente com as plataformas inspirou a criação do protótipo. No Capítulo 3 são descritas as tecnologias e métodos utilizados no desenvolvimento deste trabalho. Em seguida, no Capítulo 4 são apresentados os diagramas criados para o desenvolvimento e cada componente do protótipo, assim como suas funcionalidades, também é apresentado um exemplo de uso do protótipo. Por fim, no Capítulo 5, são apresentados a conclusão e os trabalhos futuros.

Capítulo 2

Plataformas Associadas

Hoje, existem algumas soluções para facilitar o desenvolvimento de jogos digitais, as quais foram inspirações para a criação do protótipo, como o Twine que será explicado em mais detalhes na Seção 2.1, Construct 3 que será explicado em mais detalhes na Seção 2.2 e o Scratch que será explicado em mais detalhes na Seção 2.3. Este capítulo conta também com a Seção 2.4, que apresenta o jogo "Reigns", uma das inspirações para o estilo de jogo criado pela plataforma *Papiro Engine*. Por fim, na Seção 2.5, são feitas algumas considerações finais sobre as plataformas apresentadas.

Os jogos desenvolvidos através da *Papiro Engine* são baseados em narrativas interativas não lineares, porque assim, consegue-se facilitar o desenvolvimento de um jogo por parte dos professores, pois embora nem todos os professores tenham experiencia com jogos, muitos deles tem experiencia com filmes, livros e outras formas de narrativa não-interativa, o que facilita a construção de jogos baseados nessa modalidade de mídia.

Este conceito de narrativa, como definido por Fraga e Pedroso (2006), são narrativas que mudam com base nas escolhas do jogador. Ou seja, em cada jogo existe uma história que tem vários caminhos diferentes, a partir disso, o usuário é exposto a uma pequena parte da história e é ele quem decide, a partir de alternativas escolhidas por ele, como a história irá prosseguir.

2.1 Twine

O Twine[2] é uma plataforma para criação de narrativas interativas não lineares. Como ilustrado na Figura 1 e Figura 2, a plataforma possui nós e ligação entre eles, sendo que os quadrados brancos na Figura 1 representam os nós e a seta entre eles representa a conexão, a Figura 1 ilustra a área de criação de jogos da plataforma. Na Figura 2 está representado o jogo em execução, desenvolvido na Figura 1.

[2] https://twinery.org/

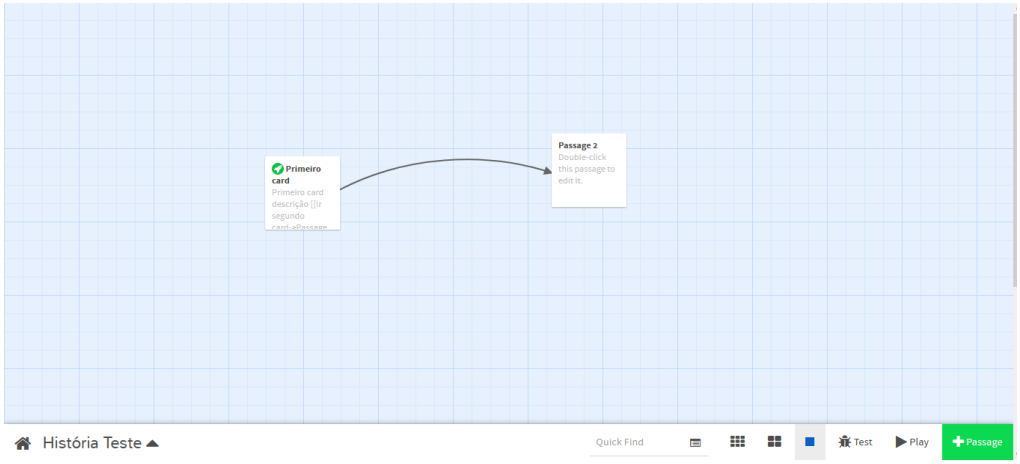

Figura 1: Desenvolvimento de Histórias do Twine

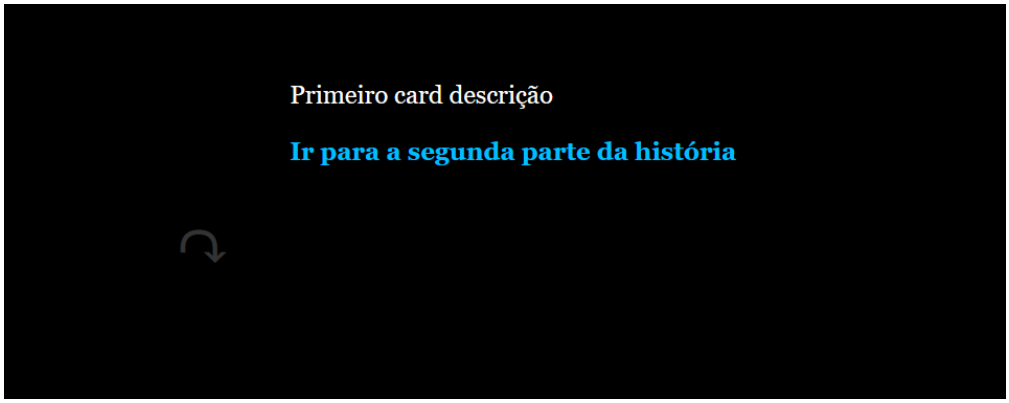

Figura 2: Layout de história criada no Twine

2.2 Construct 3

O Construct 3[3] é uma plataforma para criação de jogos 2D com o estilo *drag-and-drop* através de um editor visual. O *designer* deve criar jogos conectando blocos de programação e alterar sua lógica de acordo com a ocorrência de eventos, como ilustra a Figura 3, a qual está representada a área de edição e criação de um jogo. Ao lado direito existe um menu com as possibilidades de objetos que podem ser utilizados na criação do jogo, como sons, efeitos de luz, movimento, personagens, entre outros. O menu de cima contém opções para jogar o jogo em desenvolvimento, desfazer alterações, refazer alterações que foram desfeitas e alternar entre abas do editor. O menu à esquerda possui opções de alteração do editor, *layout* e efeitos. A Figura 4 ilustra o jogo em andamento.

[3] https://www.construct.net/en

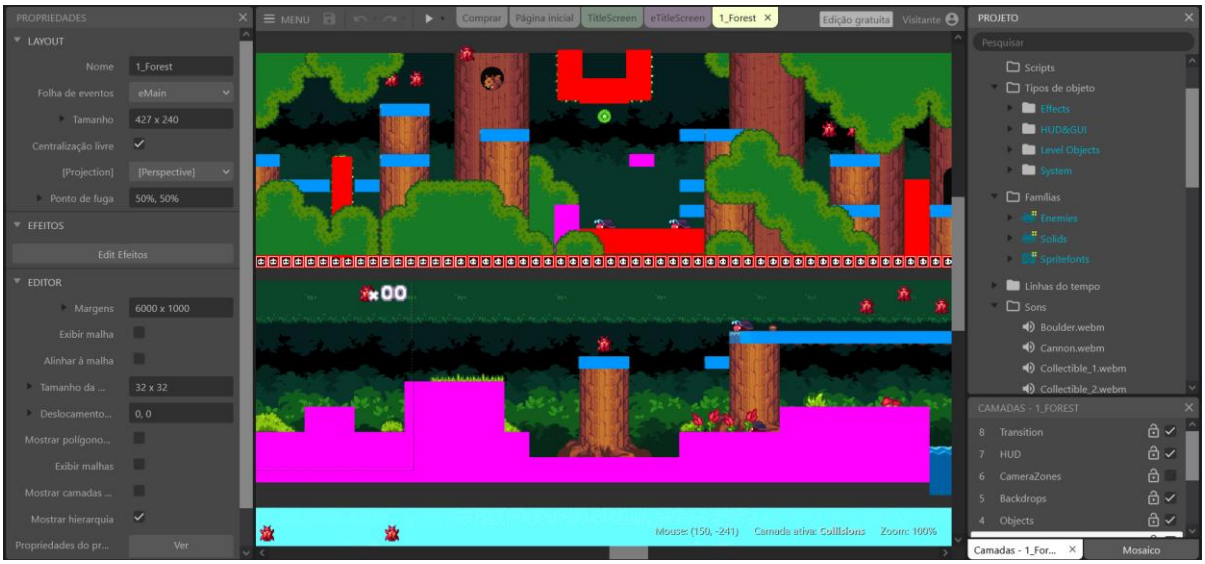

Figura 3: Área de edição do Construct 3

Figura 4: Jogo em execução no Construct 3

2.3 Scratch

O Scratch[4] é uma linguagem de programação baseada em blocos que foi desenvolvida pelo grupo Lifelong Kindergarten no *Media Lab* do *Massachussetts Institute of Technology* (MIT). O Scratch foi criado para ensinar lógica de programação para iniciantes na área de desenvolvimento de software. Uma linguagem de programação baseada em blocos funciona com o encaixe de peças, assim como um quebra-cabeça: cada comando é uma peça e consegue-se desenvolver um programa através do encaixe das peças representando os comandos. A Figura 5 contém a área de edição e criação de um jogo no Scratch, sendo que do lado esquerdo existem alguns menus com opções para a construção do jogo, como movimento, som, eventos, entre outros. No centro da tela está a área reservada para a criação do jogo: os objetos do menu à esquerda são arrastados para esta área, soltos e combinados, construindo, desta forma, o jogo.

[4] https://scratch.mit.edu/

O menu à direita, contém os elementos contidos no jogo e opções de testar o jogo e parar o teste.

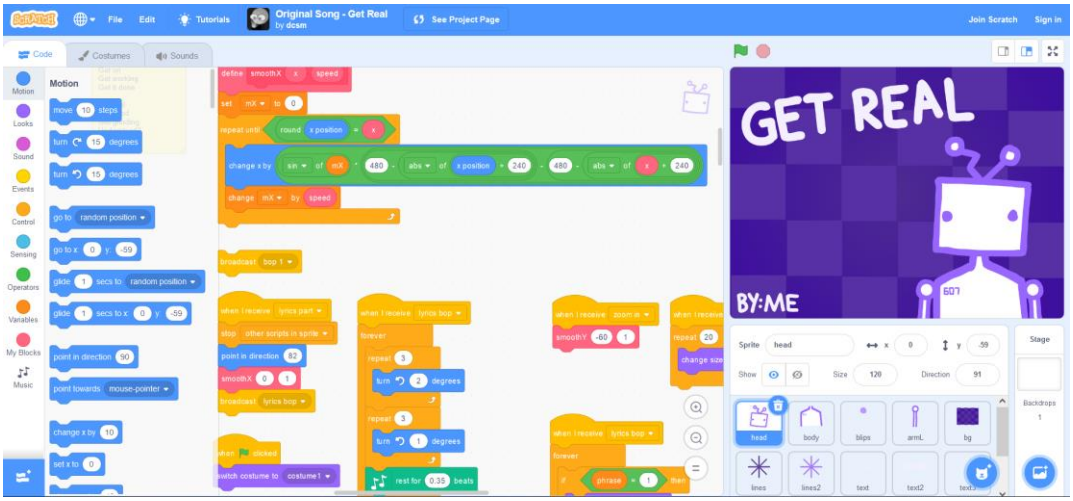

Figura 5: Área de criação e edição de jogos no Scratch

2.4 Reigns

O jogo Reigns[5] atua de forma similar às narrativas do Twine[6]. Entretanto, ao invés de apresentar uma grande quantidade de texto e mostrar as ações do personagem como botões na parte inferior, Reigns baseia-se na apresentação e troca de cartões (Figura 6): cada cartão contém uma breve história, os atributos do personagem e um ponto de decisão que deve ser tomada pelo usuário, movendo o cartão para a esquerda ou para a direita para prosseguir com a história. A decisão tomada pelo jogador afeta seus atributos e a história do jogo.

Figura 6: Cartão do jogo "Reigns"

[5] https://reignsgame.com/reigns/
[6] https://twinery.org/

2.5 Considerações Finais

As plataformas apresentadas neste capítulo não atendem integralmente a necessidade dos educadores: as plataformas Scratch e Construct 3 por exemplo exigem conhecimentos básicos de lógica de programação, restringindo o número de educadores que podem usar essa ferramenta como material didático; a plataforma Twine, embora apresente funcionalidades simples, não possui uma interface intuitiva, com botões e textos apresentando pouco contraste, como ilustra a Figura 7.

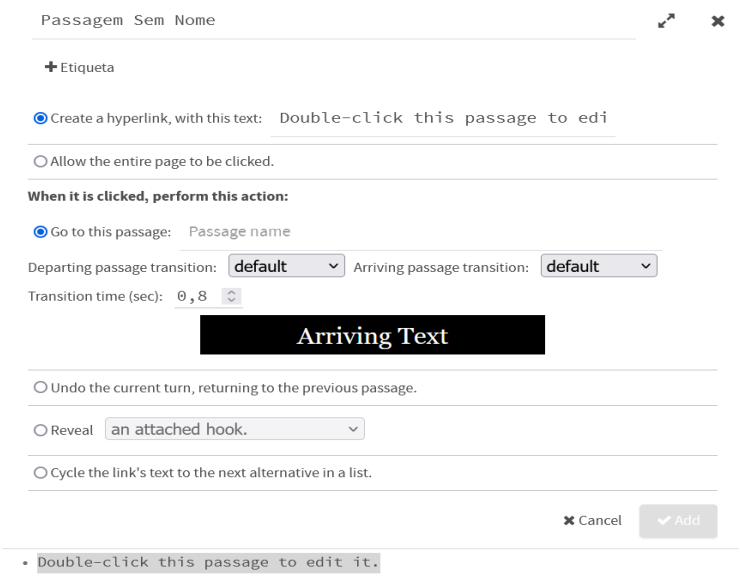

Figura 7: Edição de um nó na plataforma Twine

Para resolver esses desafios e limitações das plataformas atuais, foi desenvolvida *a Papiro Engine* (PE), uma plataforma focada em aspectos educacionais, desenvolvida para facilitar a criação de jogos por meio dos professores e baseada em jogos comerciais de sucesso, buscando aumentar a frequência de jogos na educação e aumentar o engajamento dos alunos com conteúdo diverso. Ela tem como foco a construção de jogos do tipo narrativa interativa, nos quais, o jogador assume o papel de um personagem em uma história fictícia e deve escolher suas ações dentre algumas opções pré-determinadas, de forma similar aos livros-jogo popularizados nos anos 80-90, segundo Pereira (2019).

O foco em jogos narrativos se deve pelo fato de ser uma modalidade que faz o uso de menos elementos estéticos, como arte, som e movimento, exigindo uma variabilidade menor de conhecimentos para realizar a construção de um jogo. Essa plataforma busca ser a ideal para pessoas inexperientes na área de desenvolvimento de jogos, como é uma grande proporção dos professores. Este tipo de jogo faz com que o jogador reflita sobre quais decisões podem ser tomadas em determinado momento do jogo, o que também o torna ideal para práticas didáticas.

Capítulo 3

Tecnologias e Métodos Utilizados

Para o desenvolvimento da aplicação, foram utilizadas tecnologias divididas entre *back-end* e *front-end*. O *back-end* é a parte da aplicação que fica oculta ao usuário, sendo uma "ponte" entre o banco de dados e o *front-end*, que é a parte da aplicação que fica visível ao usuário, como interfaces gráficas. As tecnologias utilizadas, conforme a Tabela 1, serão detalhadas nas seções posteriores.

Tabela 1: Tecnologias utilizadas no projeto

FRONT-END	*BACK-END*	FERRAMENTAS
React (react-markdown, react-modal, react-router)	NodeJS(Cors, Mongoose, Express, Multer, Path)	Visual Studio Code
Typescript	MongoDB	Github
	Firebase	Git
	Redis	Trello
		Heroku

3.1 React[7]

O React é uma biblioteca desenvolvida pela equipe do Facebook em 2011. Em 2013, a biblioteca se tornou, fazendo com que sua popularidade aumentasse significativamente. A linguagem de programação utilizada com React é o Javascript, o que também contribui com a sua popularidade.

Uma de suas características é ser orientada a componentes, ou seja, as interfaces são divididas em componentes. No Capítulo 4, Seção 4.2.2, por exemplo, é descrito o ambiente de desenvolvimento de jogos. Nessa tela, existem alguns componentes como por exemplo os nós, que estão localizados no menu de nós, os dois nós são um único componente que é customizável, fazendo com que os componentes se tornem reutilizáveis para as demais telas, ou seja, os nós do

[7] https://reactjs.org/

menu de nós podem ser reutilizados em qualquer outra tela com qualquer conteúdo neles, pois são um componente customizável.

O React foi selecionado para o desenvolvimento *front-end* por ser uma biblioteca estável, por conter uma ótima documentação, com código reutilizável e utiliza Javascript que é uma linguagem muito utilizada no desenvolvimento de aplicações web. A versão utilizada foi a 17.0.2.

Foram utilizados alguns componentes adicionais junto ao React no desenvolvimento porque a biblioteca não possui todos os componentes necessários na criação da plataforma, por conta disso, existem componentes externos já desenvolvidos para dar apoio ao React. Os componentes adicionais utilizados estão descritos na Tabela 2.

Tabela 2: Componentes do ReactJS

BIBLIOTECA	DESCRIÇÃO	VERSÃO
react-markdown[8]	Componente de processamento de texto no formato *markdown* para React. Foi utilizado para processar o texto em markdown da descrição dos nós criados.	5.0.3
react-modal[9]	Componente que permite a criação de uma janela flutuante customizável. Foi utilizado na criação dos formulários do protótipo, como o de criação de jogos e edição de nós.	3.12.0
react-router[10]	Coleção de componentes de navegação que se compõem declarativamente com o aplicativo. Foi utilizado para a criação de rotas internas da aplicação.	5.1.7

3.2 Typescript[11]

O Typescript é um super conjunto de Javascript, sendo convertido em Javascript antes de ser executado. Algumas vantagens do Typescript são, por exemplo, a adição de tipagem estática e orientação a objetos.

Essa tecnologia foi selecionada para o desenvolvimento pois, através dela, pode-se ter um código mais bem estruturado, utilizando a tipagem estática, por exemplo. A versão utilizada foi a 4.1.2.

[8] https://www.npmjs.com/package/react-markdown
[9] https://www.npmjs.com/package/react-modal
[10] https://reactrouter.com/
[11] https://www.typescriptlang.org/

3.3 NodeJS[12]

O NodeJS é um ambiente de execução Javascript assíncrono e orientado a eventos. Essa tecnologia foi selecionada para o desenvolvimento do *back-end*, pois foi decidido utilizar o Javascript em todo o projeto. Além disso, ela possui uma integração muito simples com o MongoDB, que foi a plataforma de banco de dados selecionada para a realização do protótipo. A versão utilizada foi a 15.6.0.

Os componentes utilizados no desenvolvimento junto ao NodeJS estão descritos na Tabela 3.

Tabela 3: Componentes NodeJS

BIBLIOTECA	DESCRIÇÃO	VERSÃO
Cors[13]	É um pacote node.js para fornecer um *middleware* Connect/Express que pode ser usado para habilitar CORS com várias opções.	2.8.5
Mongoose[14]	Ferramenta de modelagem de objetos MongoDB projetada para funcionar em um ambiente assíncrono. O Mongoose oferece suporte a promessas e retornos de chamada. Utilizado para realizar as interações com o banco de dados do protótipo.	5.12.3
Express[15]	Framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.	4.17.1
Multer[16]	Middleware node.js para lidar com *multipart/form-data*, que é usado principalmente para fazer upload de arquivos. Utilizado para realizar o cadastro de imagens.	1.4.2
Path[17]	Fornece utilitários para trabalhar com caminhos de arquivo e diretórios.	0.12.7

[12] https://nodejs.org/en/

[13] https://www.npmjs.com/package/cors

[14] https://mongoosejs.com/

[15] https://expressjs.com/pt-br/

[16] https://www.npmjs.com/package/multer

[17] https://www.npmjs.com/package/path

3.4 MongoDB[18]

O MongoDB é um sistema gerenciador de banco de dados NoSQL baseado em documentos, que usa como formato do armazenamento o JSON.

Embora a modelagem do banco de dados da *Papiro Engine* siga um modelo relacional, muitos dos atributos salvos no banco são dinâmicos e estruturados em formato JSON. Por conta disso, foi decidido utilizar o NoSQL, mais especificamente o MongoDB. Além disso, como o projeto está em seu início, o banco pode ser alterado significativamente ao longo do tempo, e um banco NoSQL possui mais liberdade para receber essas alterações sem uma migração completa dos dados. De acordo com Sareen e Kumar (2015), essas são algumas das principais vantagens do NoSQL. A versão utilizada foi a 3.6.6.

3.5 Firebase[19]

Firebase é uma plataforma desenvolvida pelo Google, essa plataforma possui funcionalidades que facilitam o desenvolvimento web e mobile. Para o desenvolvimento deste trabalho, foi utilizada a funcionalidade do Firebase chamada “Firebase Authentication”.

O Firebase Authentication tem como objetivo realizar o login, logout do sistema e armazenar os dados do usuário, como e-mail, último login, data de criação do usuário e provedor do cadastro/login. Foi utilizado o Google e e-mail alternativo como forma de cadastro/login.

3.6 Visual Studio Code[20]

O Visual Studio Code é um editor de código, com suporte para a grande maioria das linguagens de programação. Conta também com milhares de extensões, fazendo com que o usuário tenha a possibilidade de “moldar” o editor da forma que desejar. A versão utilizada foi a 1.57.

3.7 Redis[21]

O Redis é um armazenamento de estrutura de dados de chave-valor na memória. Essa tecnologia foi utilizada no formato pub/sub para realizar o envio de informações relevantes para o módulo de *analytics* através de um tópico. Existem alguns pontos da plataforma em que são enviadas mensagens através do tópico citado anteriormente, são eles: no momento de cadastro de

[18] https://www.mongodb.com/
[19] https://firebase.google.com/
[20] https://code.visualstudio.com/
[21] https://redis.io/

um novo usuário, ao criar um jogo, ao modificar um jogo já existente e ao passar de cartões durante o teste do jogo. A versão utilizada foi a 3.1.2.

3.8 Git[22]

O Git é um sistema de controle de versão com código aberto. Essa tecnologia foi selecionada para realizar o controle de versão do código, por ser de fácil utilização, desempenho extremamente rápido, pelo possuir uma extensão no Visual Studio Code, facilitando ainda mais o uso. A versão utilizada foi a 2.29.2.windows.3.

3.9 Github[23]

Github é uma plataforma totalmente online, podendo-se criar repositórios para hospedar projetos.

Essa tecnologia foi selecionada para realizar o armazenamento do código fonte e por possibilitar que o projeto seja de código aberto, uma vez que no Github, a *Papiro Engine* está aberta para novos contribuidores.

3.10 Heroku

Heroku[24] é uma plataforma em nuvem que permite a hospedagem de aplicações em um ambiente totalmente escalável e provê suporte para muitas tecnologias.

Essa tecnologia foi selecionada para a realização do deploy e hospedagem da aplicação e foi selecionada, também, pois, conta com uma integração ao Github, facilitando o deploy e é uma plataforma que fornece um plano grátis.

3.11 Fluxo de Produção/Deployment

Durante o desenvolvimento deste trabalho, foi utilizada uma metodologia de desenvolvimento, na qual foram feitas reuniões semanais com média de uma hora e meia por semana para relatar os avanços e impedimentos relacionados ao desenvolvimento. Para controle de atividades a serem feitas, foi utilizado o Trello[25], que é uma ferramenta de gerenciamento de

[22] https://git-scm.com/
[23] https://github.com/
[24] https://www.heroku.com
[25] https://trello.com/

projetos baseado em quadros, listas e cartões. Foram criadas 5 listas: “references”, “backlog”, “todo”, “doing” e “done”. Na lista “references” foram criados cartões com referências para desenvolvimento das atividades, “backlog” é a lista na qual foram criados cartões com todas as atividades a serem desenvolvidas, “todo” são as tarefas que foram movidas do backlog para serem as próximas a serem desenvolvidas, “doing” são as tarefas que estão em andamento no momento e “done” são as tarefas que já foram concluídas.

Para o deployment foi utilizado o Git, Github e o Heroku. O Git foi utilizado para enviar as atualizações de código para o Github. Através do Heroku, foi realizado o fluxo de continuous deployment. O Continuous Deployment, como citado por Niu, Brinkkemper, Franch, Partanen e Savolainen (2018), é o processo em que as novas funcionalidades do sistema são atualizadas de forma automatizada, no caso da PE, sempre que o código fonte do Github receber uma atualização, é feito o deploy para o Heroku automaticamente.

O código-fonte do projeto está disponível no Github com os nomes de “*designerModule*” (branch development) e “*backendDesignerModule*”, pode ser acessado através do endereço: https://bityli.com/KcAjPv. O protótipo está disponível através do Heroku e pode ser acessado através do endereço: https://bityli.com/kyctMQ.

Capítulo 4

O Produto

Este capítulo apresenta a descrição da *Papiro Engine* com uma breve passagem por todos os seus módulos e maior ênfase no módulo do *designer*, foco deste trabalho. Em seguida, são apresentados os diagramas criados durante o projeto: Diagrama de Caso de Uso e Diagrama Entidade-Relacionamento. Esses diagramas foram criados para que se tivesse uma visão mais clara e simples do caminho a ser seguido no desenvolvimento. Por fim, é apresentado o protótipo do sistema, com suas telas e descrição do funcionamento, para melhor entendimento da *Papiro Engine*.

A *Papiro Engine* é composta por três módulos: o módulo do *designer*, o módulo do jogador, e o módulo de *analytics*. O módulo do jogador é a área na qual o jogador irá jogar o jogo desenvolvido na plataforma. O módulo de *analytics* concentra as informações e estatísticas geradas durante as execuções e criação do jogo. Por fim, o módulo do *designer*, que é foco deste trabalho. Neste módulo é feito o envio de dados dos jogos e jogadores para o módulo de *analytics* e é feito também, a criação, edição e teste dos jogos. A *Engine* foi inspirada na plataforma Twine e no jogo Reigns.

Os jogos criados através da *Papiro Engine* são baseados em histórias e decisões tomadas pelo jogador, ou seja, o rumo da história é definido pelas decisões. Cada jogo tem uma história e, como em Reigns, é dividido em cartões. Cada cartão apresenta uma parte da história e uma decisão que deve ser tomada pelo jogador. Dependendo da decisão, a história segue caminhos diferentes em termos de narrativa, conforme criado pelo *game designer*.

Buscando uma interface simples e intuitiva, o sistema de criação de jogos foi desenvolvido em formato de nós e ligações entre eles, com uma mecânica de "*drag and drop*", como nas plataformas Scratch e Twine; assim, para a criação de um nó, basta arrastá-lo até a tela e soltar. Cada nó representa um cartão no jogo e as ligações entre os nós estão atreladas às alternativas, que são as decisões que devem ser tomadas pelo jogador.

4.1 Diagramas

Para guiar o desenvolvimento técnico do módulo de *designer*, um Diagrama de Casos de Uso geral do sistema com todos os módulos foi desenvolvido, descrevendo os cenários de utilização da aplicação. O Diagrama de Casos de Uso elaborado na modelagem da "*Papiro Engine*" é ilustrado nas Figuras 8 e 9.

A Papiro foi pensada para três diferentes atores: desenvolvedores de jogos que são pessoas já habituadas a criação de jogos digitais, professores que são pessoas que geralmente não possuem familiaridade com o desenvolvimento de jogos digitais e jogadores/alunos que serão responsáveis por jogar os jogos já desenvolvidos. Os desenvolvedores e professores estarão aptos para realizar as mesmas tarefas dentro da aplicação: gerenciar e criar jogos e realizar as ações dentro do módulo de *analytics* para geração de dados sobre a criação e edição de jogos e ações dos jogadores enquanto estiverem jogando. Os jogadores/alunos têm acesso somente ao módulo do jogador, podendo acessar todos os jogos criados na plataforma.

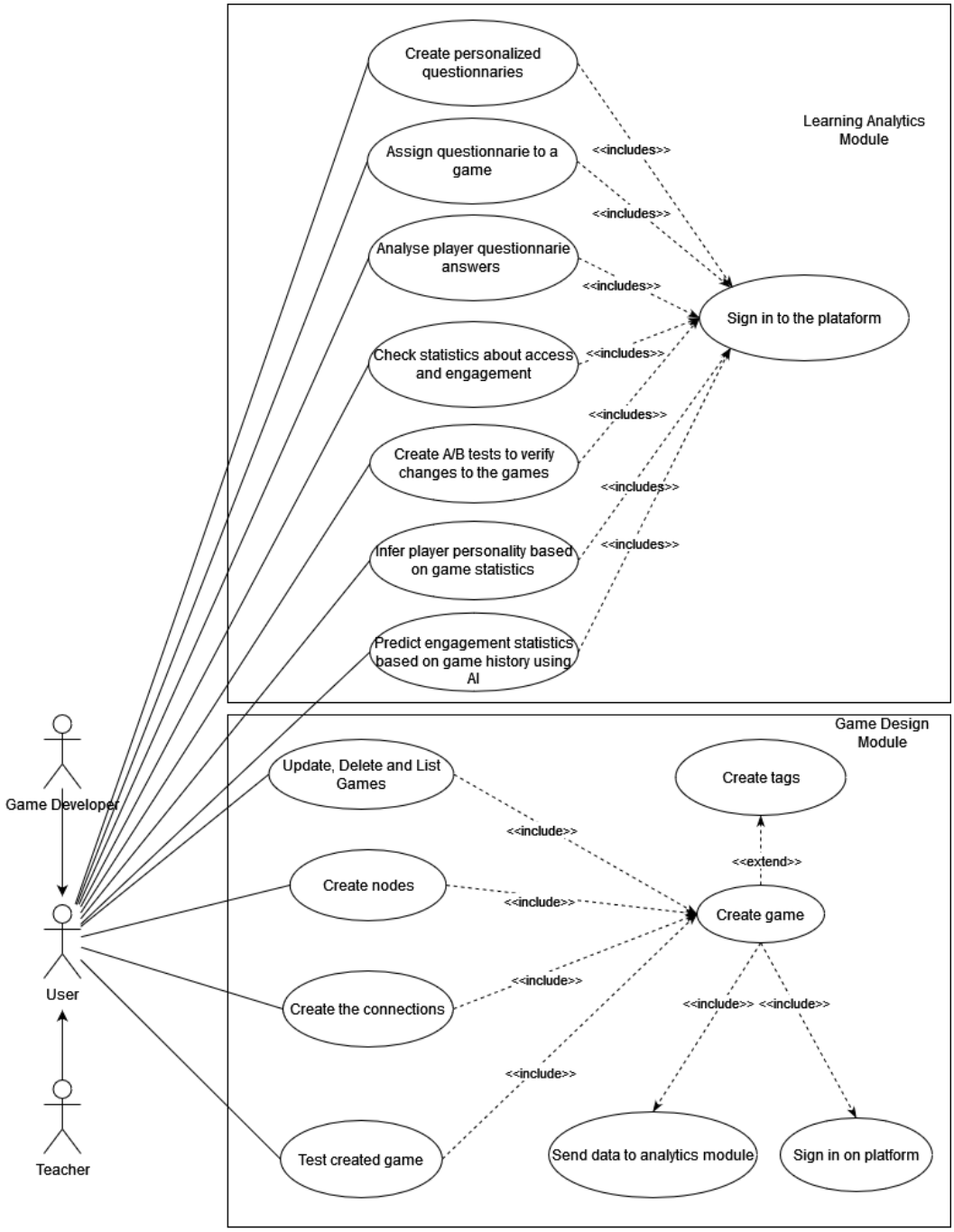

Figura 8: Diagrama de Caso de Uso I

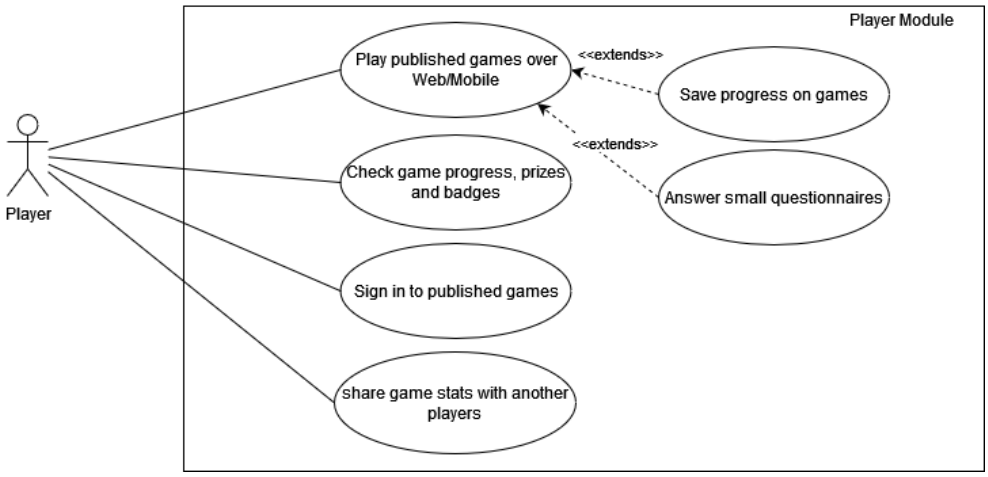

Figura 9: Diagrama de Caso de Uso ll

A partir do Diagrama de Casos de Uso, foi possível ter uma visão mais clara do que precisaria ser desenvolvido neste trabalho. Com isso, partiu-se para a modelagem do banco de dados da aplicação, com o desenvolvimento de um Diagrama Entidade-Relacionamento, como ilustrado na Figura 10, na qual os "Nodes" são os nós presentes na criação de um jogo, cada nó, como descrito anteriormente, representa um cartão no jogo, os nós do jogo possuem alternativas, cada alternativa está ligada a um outro nó, essa ligação entre os nós é representada pela tabela "Node_connection". A tabela "Games" representa os jogos que são criados dentro da PE, que, por sua vez, possuem um ou mais nós. A tabela "Labels" representa as tags dos nós no ambiente de desenvolvimento de jogos, descrito com mais detalhes na Subseção 4.2.2.

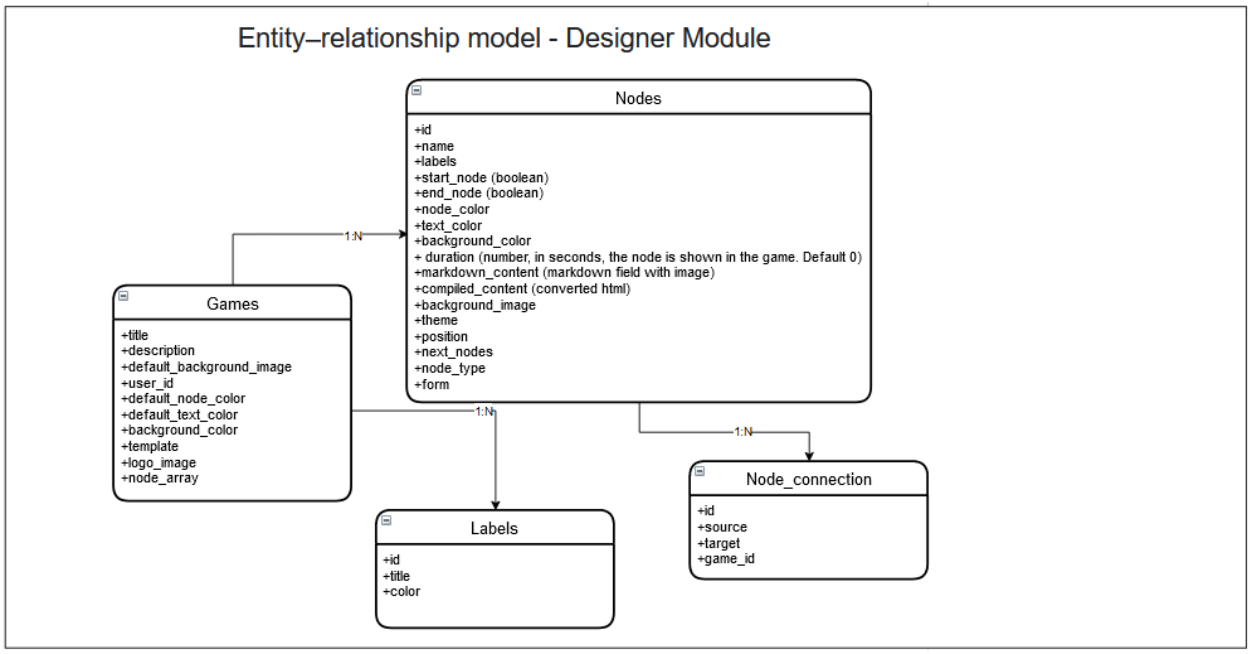

Figura 10: Diagrama Entidade-Relacionamento

Por fim, para facilitar a visualização do protótipo como um todo e como irá funcionar o fluxo de interação entre o usuário e a plataforma, foi desenvolvido um Diagrama de Atividades, facilitando também a visualização das ações que podem ser tomadas pelos atores que interagem diretamente com o protótipo desenvolvido, que são os *game developers* e os educadores, como ilustrado na Figura 11.

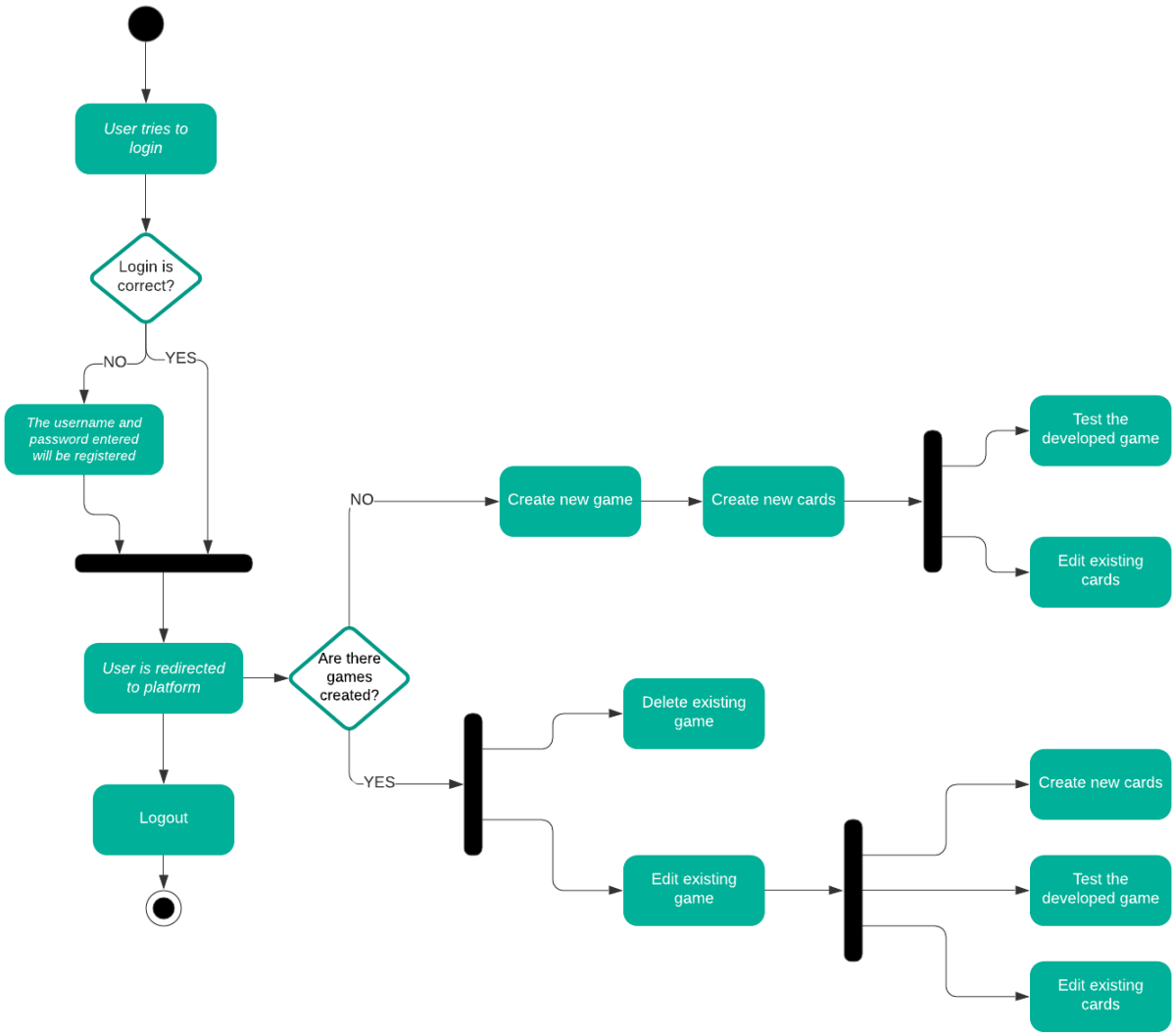

Figura 11: Diagrama de Atividades

4.2 Componentes do Sistema

Para atender os atores e ações dos diagramas de caso de uso, o módulo do *designer* será dividido em três partes para facilitar a compreensão: login, dashboard e o ambiente de desenvolvimento. O login é a área dedicada a cadastro e entrada na plataforma através de autenticação do usuário por e-mail e senha ou através da conta do google; o dashboard é responsável pelo gerenciamento de jogos; a área de desenvolvimento é responsável por fornecer todos os atributos necessários para que o usuário consiga desenvolver e testar o jogo criado.

4.2.1 Dashboard

O dashboard, como ilustrado na Figura 12, é a página inicial da aplicação, sendo possível visualizar os jogos criados, criar, editar e excluir jogos e se desconectar da plataforma. Esta área da aplicação conta com os seguintes componentes:

- Biblioteca
- Menu lateral
- Criação de Jogos

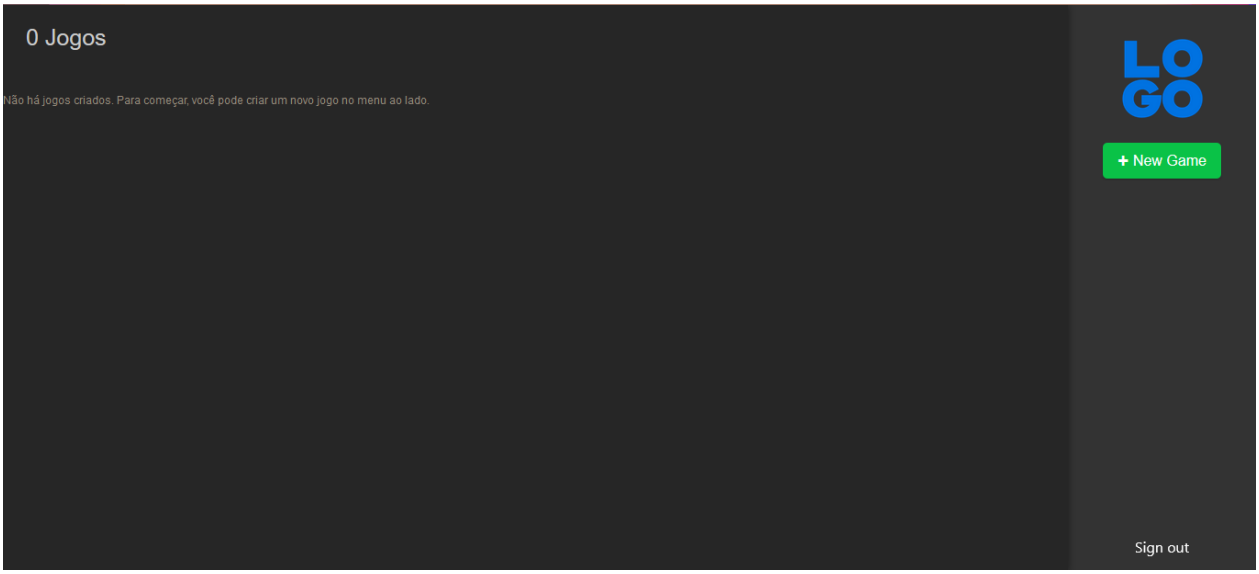

Figura 12: Dashboard

A Biblioteca, como ilustrado na Figura 13 e Figura 14, é um componente que está presente no Dashboard. Tem o objetivo de mostrar ao usuário quantos jogos já foram criados por ele. É possível acessar os jogos já existentes clicando nos mesmos, sendo redirecionado para o ambiente de desenvolvimento (Seção 4.2.2).

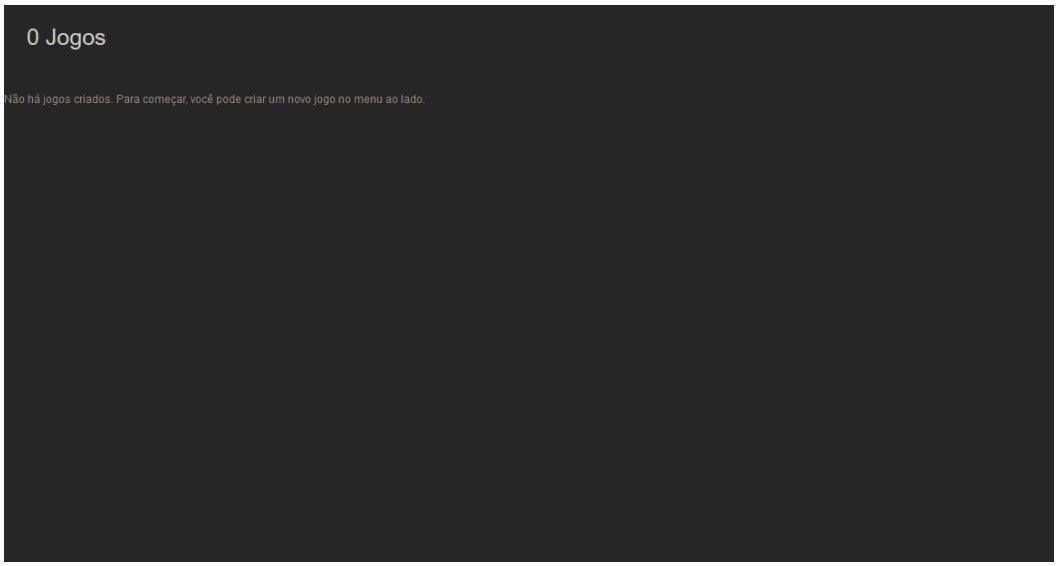

Figura 13: Biblioteca de jogos vazia

Figura 14: Biblioteca de jogos

O menu lateral, como ilustrado na Figura 15, é um componente que está presente no Dashboard. Pode ser feito o logout e iniciar a criação de um novo jogo através do botão “New Game”, abrindo um formulário, como ilustrado na Figura 16.

Figura 15: Menu lateral localizado na tela “Dashboard”

A área de criação de jogos, como ilustrado na Figura 16, é um formulário com as informações essenciais para a criação de um jogo. Essas informações são: Título, descrição do jogo, padrão de cores, imagem de fundo padrão, se este jogo vai ser um modelo para ser usado futuramente e o logo do jogo.

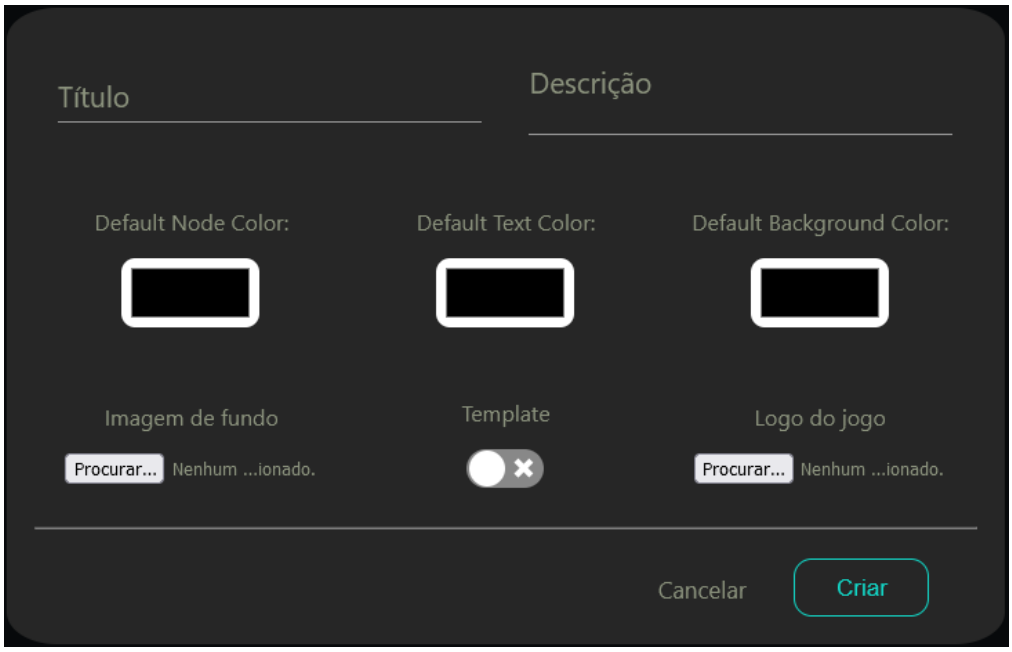

Figura 16: Criação de jogos

4.2.2 Ambiente de Desenvolvimento de Jogos

O ambiente de desenvolvimento de jogos, ilustrado na Figura 17, é a área responsável pela criação, edição e teste de jogos. Esta área conta com os seguintes componentes:

- Tela de desenvolvimento
- Menu superior
- Menu de nós
- Edição de nós

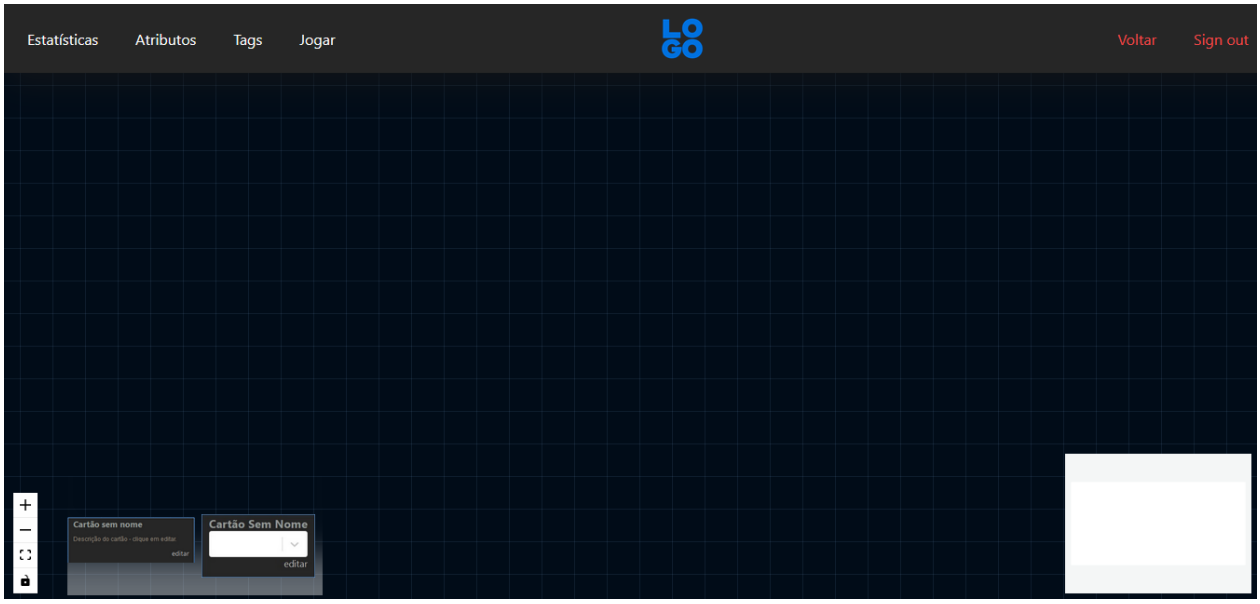

Figura 17: Ambiente de desenvolvimento de jogos

A Tela de Desenvolvimento é o principal componente. Nele, os jogos são criados ou editados. Como mostra a Figura 18, pode-se realizar a criação de nós (cada nó representa um cartão no jogo, como ilustrado na Figura 22) e as ligações entre eles (caminhos em que o jogador tem a possibilidade de seguir a partir de suas decisões no jogo).

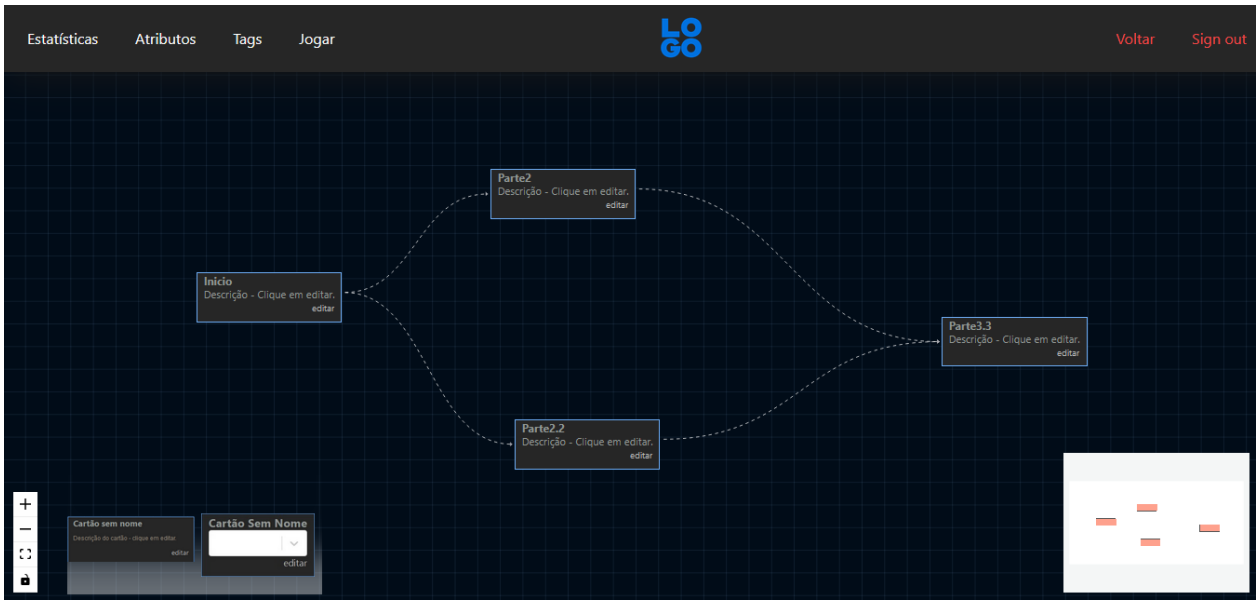

Figura 18: Grid de desenvolvimento

O Menu superior é o menu da área de edição e criação de jogos, sendo possível realizar a criação de tags e testar o jogo.

Figura 19: Menu superior na tela “Ambiente de desenvolvimento”

Ao selecionar a opção “Tags” no menu superior (Figura 19), abre-se a possibilidade de criar as chamadas “Tags” que tem a função de “etiquetas”. *Tags* tem o objetivo de facilitar a identificação de cada nó mais facilmente na hora do desenvolvimento. Na Figura 20, pode-se ver um exemplo de como criar as *tags*. Na Figura 21, há um exemplo de como ficam as *tags* quando adicionadas aos nós.

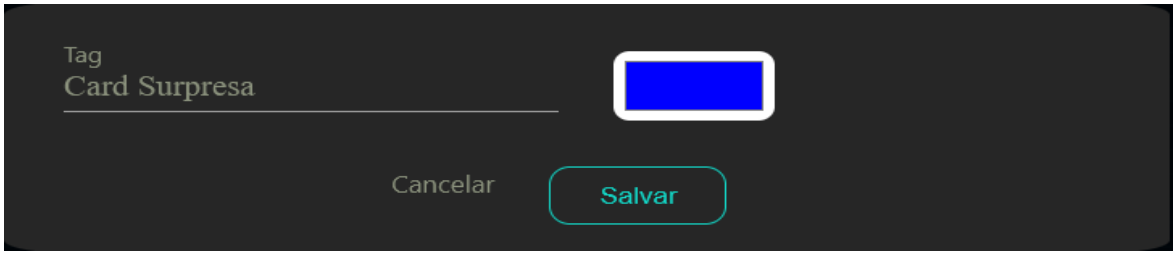

Figura 20: Área para criação de tags

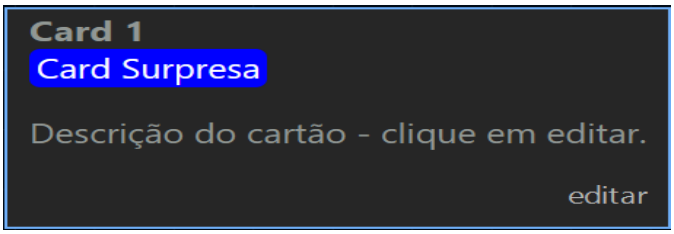

Figura 21: Nó com uma tag atrelada

Pode-se também realizar o teste do jogo criado, como na Figura 22. Esta função encontra-se no botão “jogar” ao lado de “tags”. Cada nó presente na área de desenvolvimento, corresponde a um cartão ao executar o jogo.

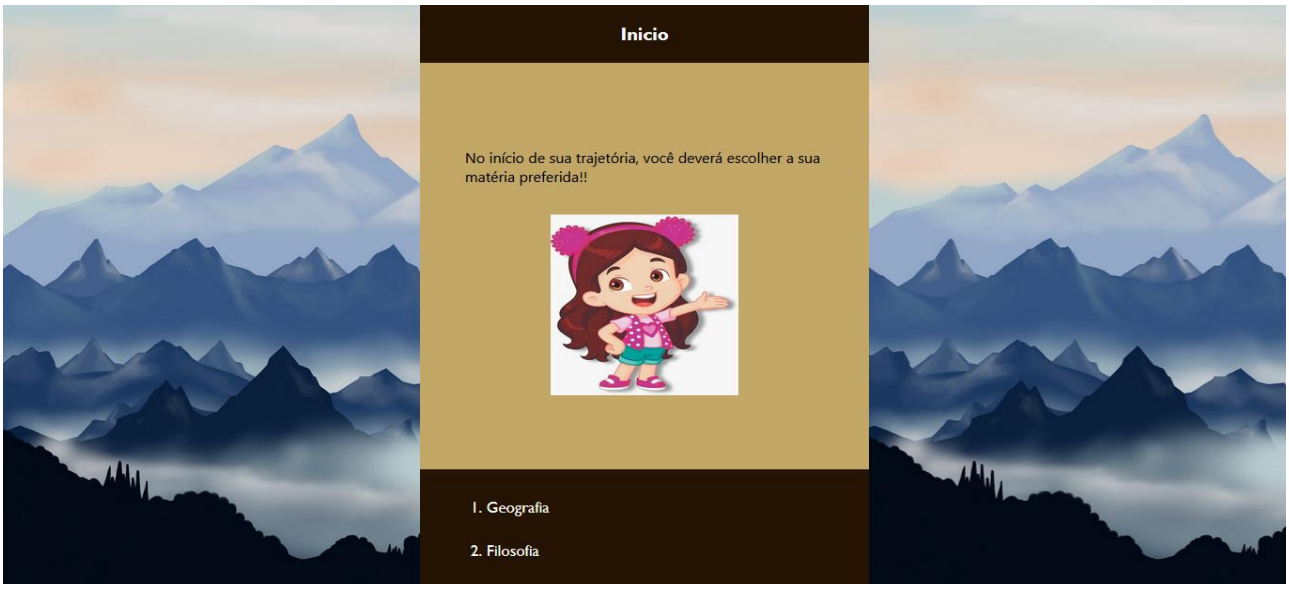

Figura 22: Jogo em execução

A seção “Atributos” será desenvolvida futuramente e a seção “Estatísticas” está sendo desenvolvida em outro trabalho.

No menu de nós (Figura 23) estão os nós que podem ser criados no jogo. Neste menu, o usuário arrasta o nó para a posição desejada na tela de desenvolvimento: ao soltar, o nó é criado.

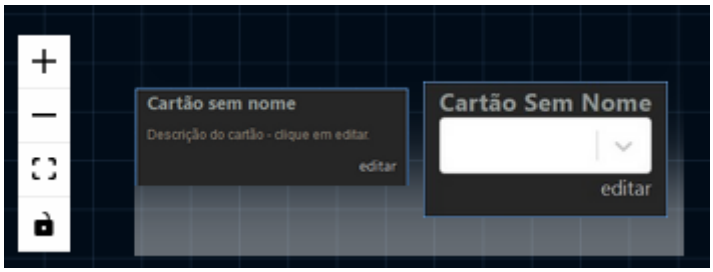

Figura 23: Menu de nós

Existem duas opções de nós a serem criados: o nó tradicional, que equivale a um cartão no jogo, assim como ilustrado na Figura 22; E um nó formulário, responsável por listar e integrar questionários de avaliação didática, criados no módulo *analytics*. Com um nó formulário, o usuário

tem acesso aos formulários criados no módulo de *analytics*, que serão incluídos como transições de cartões no módulo do jogador para que este responda o questionário dentro do jogo. Suas respostas são enviadas então para o módulo *analytics* através do sistema de mensageria do Redis, como ilustrado na Figura 24.

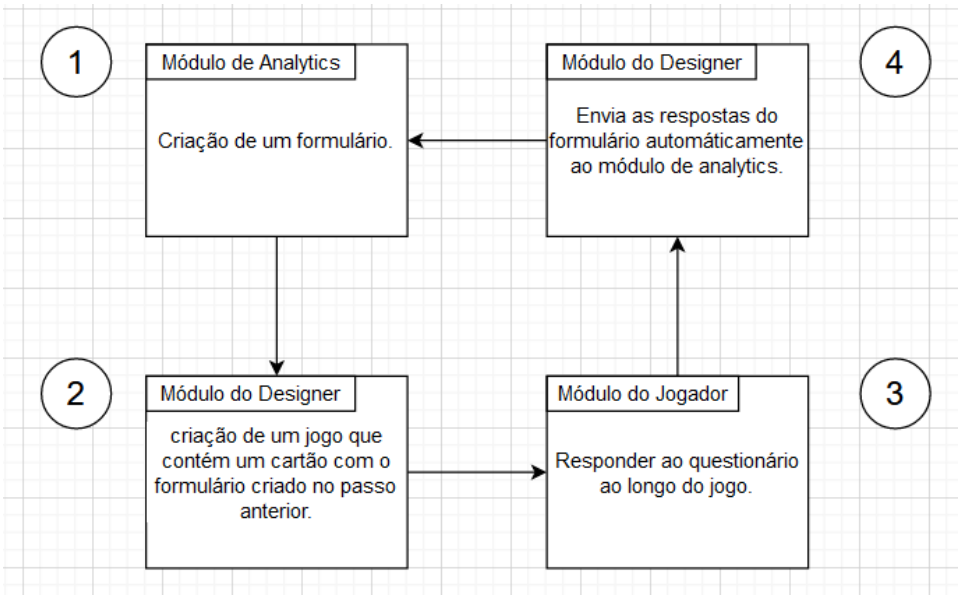

Figura 24: Fluxograma de interação entre os módulos da plataforma

Ou seja, ao responder o formulário, é enviada uma mensagem com todas as respostas para um tópico do Redis, que nada mais é do que um canal de comunicação entre o módulo do *designer* e o módulo de *analytics*.

A área de edição de nós é aberta ao clicar em “editar”, localizado na parte inferior direita de todos os nós, como ilustrado anteriormente na Figura 21. Esta área corresponde a um formulário, no qual pode-se editar todas as informações do nó em questão, como a descrição no formato *markdown* (que será convertida em formato html para que seja renderizada ao executar o jogo), o título do cartão, a duração em que ele ficará disponível na tela, caso o cartão tenha duração, ao testar o jogo aparecerá o tempo em que o jogador tem para responder aquele cartão, caso não responda a tempo, ele irá para o próximo cartão aleatoriamente; As tags atreladas, as alternativas e para qual cartão cada alternativa leva, as cores dos nós (se nenhuma cor for selecionada, prevalecerá as cores padrão selecionadas ao criar um jogo), se este cartão é um cartão de final de jogo e uma imagem de fundo, como ilustrado na Figura 25.

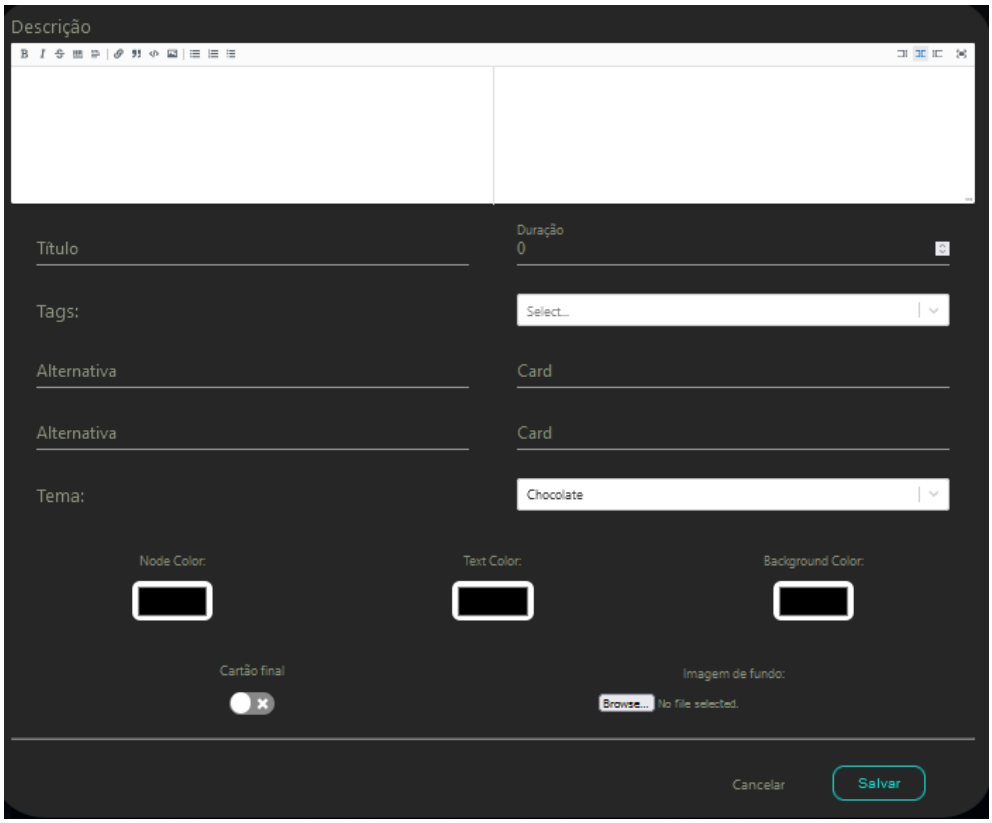

Figura 25: Edição de nós

4.3 Produto Resultante

Nesta seção será apresentado um exemplo de uso do protótipo com a sequência de telas para ilustrar melhor o funcionamento do sistema.

O primeiro passo é fazer o login na plataforma através das opções ilustradas na Figura 26. Caso o usuário não tenha uma conta cadastrada, ambas as opções farão o cadastro e em seguida o login. Caso o usuário já tenha realizado o cadastro, ambas as opções só farão o login.

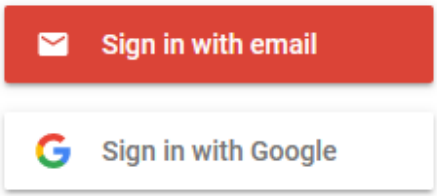

Figura 26: Passo 1 - Realizar o cadastro/login

Após esse passo, o usuário terá acesso ao Dashboard da plataforma. Para iniciar a criação de um jogo, basta clicar no botão “New Game” no menu lateral a direita. Assim como ilustrado na figura 27.

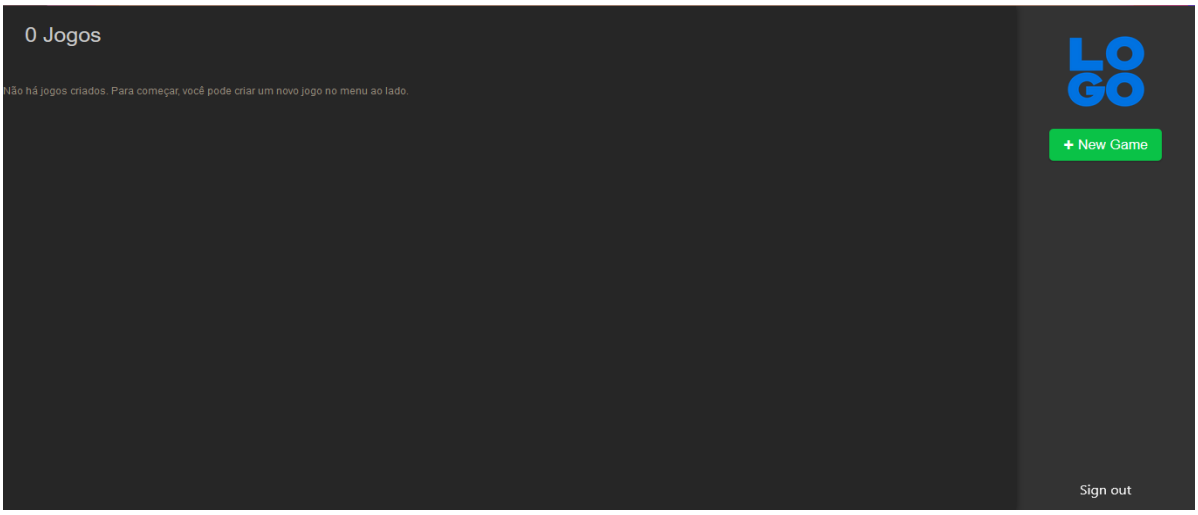

Figura 27: Passo 2 - Iniciar a criação de um jogo

Em seguida, um formulário será aberto, nele, é necessário que seja preenchido ao menos o Título e clique em “Criar”. Assim como ilustrado na Figura 28, nela foi preenchido o Título, imagem de fundo e o logo do jogo.

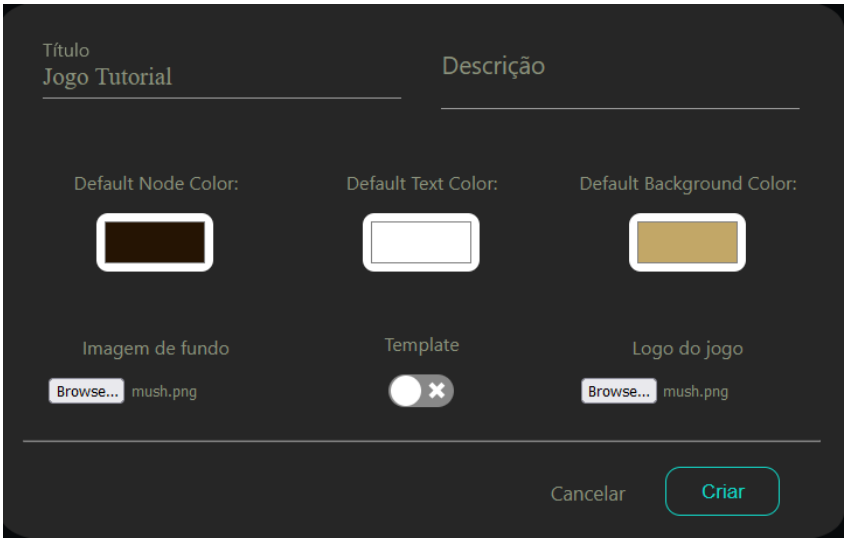

Figura 28: Passo 3 - Inserir informações do jogo

Após ter um jogo criado, o usuário é redirecionado para o ambiente de desenvolvimento de jogos. Para dar início ao desenvolvimento, é necessário clicar em um nó que está localizado no menu de nós no canto inferior esquerdo e arrastá-lo para o local da tela que desejar. Para melhor entendimento, na Figura 29, o nó da esquerda do menu de nós foi arrastado e posicionado no grid de desenvolvimento.

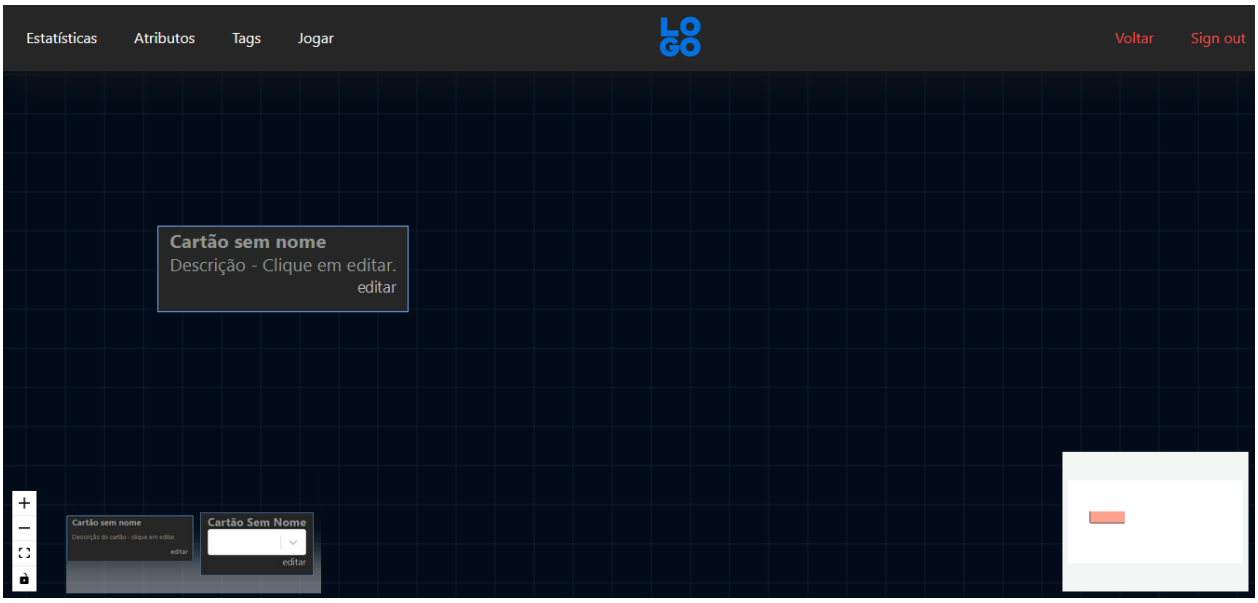

Figura 29: Passo 4 - Criação de um cartão

O próximo passo é editar o nó: para isso, deve-se clicar em “editar” no canto inferior direito do nó que acabou de ser criado no passo anterior e preencher as informações desejadas no formulário.

Para realizar a ligação entre nós, o usuário criar uma “alternativa” e adicionar o nome de um “card” que será o destino caso essa alternativa seja selecionada na hora do jogo. Caso o nome do card destino escolhido ainda não exista, ele será criado automaticamente. Neste caso, a alternativa “Geografia” tem como destino o cartão “Parte2”, o formulário de edição de nó está ilustrado na Figura 30.

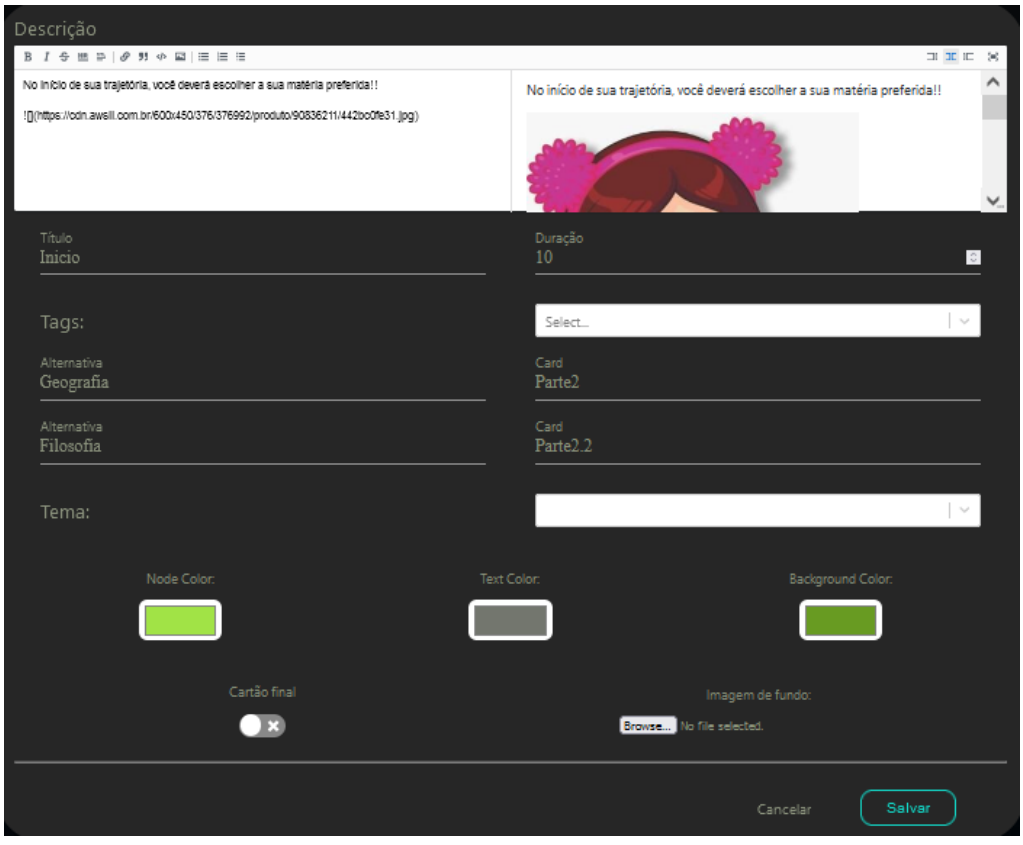

Figura 30: Passo 5 - Inserir informações do cartão

Após desenvolver o jogo no ambiente de desenvolvimento de jogos, para realizar o teste, basta acessar a seção “jogar” no menu superior da tela de desenvolvimento, como ilustrado na Figura 28. Com isso, o jogo desenvolvido anteriormente será aberto, como ilustrado na Figura 31.

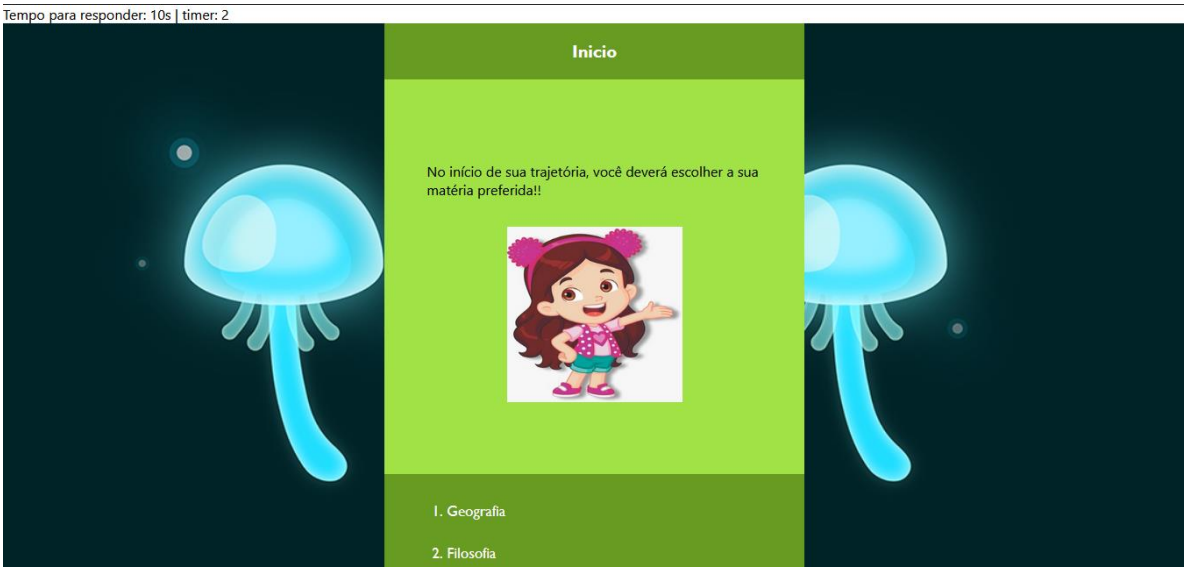

Figura 31: Passo 6 - Teste do jogo desenvolvido

Capítulo 5

Conclusão e Trabalhos Futuros

Este trabalho teve como motivação a facilitação da criação de jogos por parte de usuários que não tenham familiaridade com nenhuma área de desenvolvimento. A *Papiro Engine* foi desenvolvida com esse intuito, facilitar a criação de jogos educativos por professores para uso em contexto educacional.

A *Papiro Engine* é uma plataforma que conta com três principais módulos: *designer*, *analytics* e jogador. O módulo do *designer*, cujo protótipo foi desenvolvido neste trabalho, contém funcionalidades suficientes para realizar a criação, edição e teste dos jogos baseados em narrativas interativas não lineares, sendo também responsável por enviar dados, através de um sistema de mensageria, para alimentar o módulo de *analytics*. O módulo de *analytics* recebe, trata e devolve para o usuário os dados estatísticos gerados a partir do módulo do *designer*, em forma de gráficos. Por fim, o módulo do jogador é a área em que os jogadores têm acesso aos jogos desenvolvidos no módulo do *designer*.

Futuramente, deverão ser implementadas novas funcionalidades na *Engine*, como o sistema de atributos do jogador, exportação do jogo para que possa ser executado em outras plataformas e a inclusão de um personagem para o jogador. Além disso, o módulo do jogador em si ainda se encontra em desenvolvimento. Esse módulo permitirá a exportação do jogo e sua publicação em lojas de aplicativos (como Google Play e App Store) e plataformas de distribuição de jogos tradicionais, como a Itch.io. A *Engine* é de código aberto, possibilitando o desenvolvimento de novas funcionalidades pela comunidade de jogos em um projeto comum, facilitando a colaboração com outros pesquisadores e *designers*. O Protótipo foi testado ao longo do desenvolvimento pela própria equipe que o criou, porém ainda não foi avaliado no contexto de uso, algo planejado para ser conduzido posteriormente.

Referências

DFC Intelligence. (2020). *Global Video Game Consumer Segmentation.* DFC Intelligence.

FRAGA, D., & PEDROSO, F. (2006). *Jogo de RPG no ensino e aprendizagem de narrativas não-lineares.* Rio Grande do Sul: CINTED-UFRGS.

HERPICH, F., JARDIM, R., SILVA, R., VOSS, G., NUNES, F., & MEDINA, R. (2014). Jogo Sério na Educação: Uma Abordagem para Ensino-Aprendizagem de Redes de Computadores (Fase II). *WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI)*, 391-400.

KISHIMOTO, T. (2009). *Jogo, brinqueido, brincadeira e educação.* São Paulo: São Paulo.

M., W. J. (2008). *Computational thinking and thinking about computing.* New York: Phil. Trans. R. Soc. A. Fonte: https://doi.org/10.1098/rsta.2008.0118

NIU, N., BRINKKEMPER, S., FRANCH, X., PARTANEN, J., & SAVOLAINEN, J. (2018). Requirements Engineering and Continuous Deployment. *IEEE Software, vol. 35*, 86-90. doi:10.1109/MS.2018.1661332

PEREIRA, E. B. (2019). *O LIVRO-JOGO COMO RECURSO DIDÁTICO PARA O ENSINO DE BIOLOGIA: UMA PROPOSTA PARA A TEMÁTICA MALÁRIA.* Manaus: Instituto Federal de Educação, Ciência e Tecnologia do Amazonas .

ROBLES, G., MORENO-LEÓN, J., & ROMÁN-GONZÁLEZ, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED-Revista de Educación a Distancia*, 1-23.

SANTOS, S., & BÍSCARO, H. (2019). Revisão sistemática sobre a utilização de jogos . *Revista Eletrônica Paulista de Matemática*, 14.

SAREEN, P., & KUMAR, P. (2015). NOSQL DATABASE AND ITS COMPARISON WITH SQL DATABASE. *International Journal of Computer Science & Communication Networks,Vol 5(5)*, 293-298.

SOUSA, A., & COUTINHO, J. (2016). *Revisão Sistemática sobre Avaliação de Jogos voltados para Aprendizagem de Programação no Brasil.* Rio Tinto: Universidade Federal da Paraíba.

THEODOSIOU, S., & KARASAVVIDIS, I. (2015). Serious games design: A mapping of the problems novice game designers experience in designing games. *Journal of E-Learning and Knowledge Society*, 133-149.

TRYBUS, J. (2013). *Game-Based Learning: What it is, Why it Works, and Where it's Going.* Georgia: New Media Institute .