UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DARLEY LEONARDO KREFTA

PLATAFORMA PARA AUXÍLIO NO DESENVOLVIMENTO DE RACIOCÍNIO LÓGICO DE CRIANÇAS

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2019

DARLEY LEONARDO KREFTA

PLATAFORMA PARA AUXÍLIO NO DESENVOLVIMENTO DE RACIOCÍNIO LÓGICO DE CRIANÇAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Departamento Acadêmico de Informática
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

PLATAFORMA PARA AUXÍLIO NO DESENVOLVIMNTO DE RACIOCÍNIO LÓGICO DE CRIANÇAS

POR

DARLEY LEONARDO KREFTA

Este trabalho de conclusão de curso foi apresentado no dia 09 de julho de 2019, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Profª Drª
Beatriz Terezinha Borsoi

Profa. MSc
Andréia Scariot Beulke

Prof. MSc
Vinicius Pegorini

Prof. Dr. Edilson Pontarolo
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Profª Drª Beatriz Terezinha Borsoi
Responsável pela Atividade de Trabalho de Conclusão de Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

O desenvolvimento de atividades lúdicas pelas crianças visando despertar-lhes o interesse e auxiliar no aprendizado de conceitos e para o aperfeiçoamento do raciocínio tem sido visto como forte auxiliar nas primeiras fases da educação escolar. Além disso, com o amplo uso de recursos de tecnologias de informação e com a necessidade cada vez maior de soluções mais complexas de software cresceu exponencialmente o interesse e/ou necessidade de desenvolvimento do raciocínio lógico e abstrato. Esse tipo de raciocínio tem sido cada vez mais necessário, em maior ou menor grau, em todas as atividades humanas, mas para o desenvolvimento de programas e aplicativos computacionais, assim como atuação profissional em carreiras de exatas, ele é indispensável e quanto mais apurado, melhor. Programar computadores é uma atividade essencialmente abstrata, embora o resultado da execução do código possa ser apresentado de forma gráfica (componentes e elementos em telas de computador). Os próprios recursos computacionais (hardware e software) podem ser utilizados como ferramentas didáticas para o ensino de lógica e de conceitos envolvidos em programação e para o desenvolvimento do raciocínio lógico. Existem diversas iniciativas nesse sentido, a exemplo do code.org, que visam fornecer ferramentas tecnológicas para o aprendizado de conceitos de programação e desenvolvimento de raciocínio. Por meio da realização deste trabalho foi implementada uma plataforma web que permite a composição de atividades que visam promovem o desenvolvimento de raciocínio lógico em crianças. Essas atividades são no formato de seguir um caminho pré-determinado utilizando comandos de mover para cima, para baixo, para a esquerda e para a direita. Esses caminhos podem ser encapsulados em estruturas de repetição (mover direita 5 vezes). Para o desenvolvimento foram utilizadas tecnologias como: Material-UI, Axios, Konva.js, ReactJS, Redux, Node.js, Express, MongoDB, Mongoose, JavaScript e Robo 3T.

Palavras-chave: Lógica de programação. Programação para crianças. Aprendizado de conceitos básicos de programação.

ABSTRACT

The development of playful activities by children aiming to arouse their interest and help in the learning of concepts and for the improvement of reasoning has been seen as a strong auxiliary in the early stages of school education. In addition, with the widespread use of information technology resources and the ever-increasing need for more complex software solutions, the interest and/or need to develop logical and abstract thinking has grown exponentially. This type of thinking has been increasingly needed in all human activities, but for the development of computer programs and applications it is indispensable and how much more accurate, better. Programming computers is an essentially abstract activity, although the result of code execution can be presented graphically (components and elements on computer screens). However, the computational resources themselves (hardware and software) can be used as teaching tools for teaching logic and concepts involved in programming and for the development of logical reasoning. There are several initiatives in this direction, such as code.org, which aims to provide technological tools for the learning of programming concepts and reasoning development. Through the accomplishment of this work a web platform was implemented that allows the composition of activities that aim to promote the development of logical reasoning in children. These activities are in the form of following a predetermined path using commands to move up, down, left, and right. These paths can be encapsulated in repeat structures (move right 5 times). For the development were these technologies as main: Material-UI, Axios, Konva.js, ReactJS, Redux, Node.js, Express, MongoDB, Mongoose, JavaScript e Robo 3T.

Keywords: Programming logic. Programming for children. Learning basic programming concepts.

LISTA DE FIGURAS

Figura 1 – Diagrama de casos de uso..... 23
Figura 2 – Diagrama do banco de dados..... 27
Figura 3 – Estrutura do documento atividaderealizada..... 30
Figura 3 – Protótipo da tela para criar atividades..... 31
Figura 4 – Tela de login..... 32
Figura 5 – Tela de cadastro de usuário..... 32
Figura 6 – Tela de cadastro de atividade..... 33
Figura 7 – Tela de lista de atividade tutores..... 34
Figura 8 – Tela de lista de atividade alunos..... 34
Figura 9 – Tela de realização da atividade..... 35
Figura 10 – Tela de realização da atividade concluída..... 36
Figura 11 – Tela de lista de atividades realizadas pelo aluno..... 36

LISTA DE QUADROS

Quadro 1 - Ferramentas e tecnologias 17
Quadro 2 – Requisitos funcionais 22
Quadro 3 – Requisitos não funcionais 23
Quadro 4 – Operação de inclusão de cadastro de usuários 24
Quadro 5 – Operação para altera dados de cadastros de usuários 24
Quadro 6 – Operação de exclusão de cadastro de usuários 25
Quadro 7 – Operação de consulta de dados de cadastro de usuário 26
Quadro 8 – Caso de uso compor atividade 26
Quadro 9 – Caso de uso realizar atividade 26

LISTAGEM DE CÓDIGO

Listagem 1 – Documento entidade usuario do banco de dados..... 28
Listagem 2 – Documento entidade atividade do banco de dados..... 28
Listagem 3 – Documento entidade atividade realizada do banco de dados..... 29
Listagem 4 – server.js 37
Listagem 5 – db.js..... 38
Listagem 6 – auth.js 39
Listagem 7 – models/Atividade.js 40
Listagem 8 – routes/api/atividade.js 40
Listagem 9 – CadastroAtividade.js..... 42
Listagem 10 – CadastroAtividade.js..... 42
Listagem 11 – CadastroAtividade.js: state..... 44
Listagem 12 – CadastroAtividade.js: handleSubmit..... 44
Listagem 13 – CadastroAtividade.js: async()..... 45
Listagem 14 – actions/toast.js..... 46
Listagem 15 – reducers/toast.js 46

LISTA DE SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheet*
DOM	*Document Object Model*
ERP	*Enterprise Resource Planning*
HTML	*HyperText Markup Language*
MIT	Massachusetts Institute of Technology
ODM	*Object Data Mapping*
ORM	*Object Relational Mapping*
RF	Requisito Funcional
RNF	Requisito Não Funcional
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO **10**
1.1 CONSIDERAÇÕES INICIAIS 10
1.2 OBJETIVOS 12
1.2.1 Objetivo Geral 12
1.2.2 Objetivos Específicos 12
1.3 JUSTIFICATIVA 12
1.4 ESTRUTURA DO TRABALHO 14
2 ENSINO DE PROGRAMAÇÃO PARA CRIANÇAS **15**
3 MATERIAIS E MÉTODO **17**
3.1 MATERIAIS 17
3.2 MÉTODO 19
4 RESULTADO **21**
4 RESULTADO **21**
4.1 ESCOPO 21
4.2 MODELAGEM DO SISTEMA 22
4.3 APRESENTAÇÃO DO SISTEMA 31
4.4 IMPLEMENTAÇÃO DO SISTEMA 37
5 CONSIDERAÇÕES FINAIS **47**
5.1 CONTEXTO 47
5.2 TECNOLOGIAS 48
5.3 TRABALHOS FUTUROS 49
REFERÊNCIAS **50**

1 INTRODUÇÃO

Este capítulo apresenta a introdução do trabalho que abrange as considerações iniciais, os seus objetivos e a justificativa. O capítulo é finalizado com a apresentação do texto por meio da descrição sumária dos próximos capítulos.

1.1 CONSIDERAÇÕES INICIAIS

As tecnologias de informação e comunicação estão cada vez mais presentes nos processos de produção agrícola e industrial, no comércio e no terceiro setor, automatizando e auxiliando na realização de atividades e na tomada de decisão nos diversos níveis (estratégico, gerencial e operacional) das corporações e instituições. Essas tecnologias também estão presentes no cotidiano das pessoas seja para entretenimento, acesso à informação ou comunicação, entre outros. Além desses usos, essas tecnologias trazem novas possibilidades aos processos de aprendizagem. Isso porque elas permitem mais facilmente a criação por meio da experimentação, estimulando descobertas.

As tecnologias na educação têm sido vistas como ferramentas didáticas de apoio ao ensino, isso é fundamentado na amplitude e diversidade de uso nas mais diversas atividades humanas, pelo amplo acesso que crianças e adolescentes possuem a essas tecnologias e pelas possibilidades que elas podem oferecer ao processo de ensino e aprendizagem. Wing (2006) ressalta que o pensamento computacional é um método que tem como objetivo solucionar problemas, conceber sistemas e compreender o comportamento humano inspirado em conceitos da Ciência da Computação.

Na educação infantil a utilização desse tipo de pensamento pode ajudar as crianças a aprenderem técnicas de resolução de problemas, que podem ser úteis para o desenvolvimento de habilidades e no aprendizado de conceitos necessários para lidar com problemas e desafios enfrentados ao longo da vida (ZANETTI et al. 2017). O aprendizado de lógica de programação ajuda a resolver problemas, incentiva o trabalho em equipe e aumenta a capacidade de pensar de forma sistematizada e criativa (NASCIMENTO, 2015).

Outro viés mais específico para a necessidade do ensino de lógica voltada para a programação de computadores encontra amparo na visível evolução da informática, no sentido de quantidade e diversidade de uso de aplicativos (software). Para atender a demanda crescente de soluções computacionais é necessário haver profissionais qualificados que

possam desempenhar um bom trabalho (BEZERRA; DIAS, 2014) seja no uso de sistemas informatizados, no desenvolvimento desses sistemas ou na exploração das capacidades das tecnologias de informação e comunicação para a resolução de problemas e no auxílio na realização de atividades.

A necessidade de profissionais para as áreas de computação e tecnologias de informação e comunicação é amparada por pesquisa realizada pela consultoria IDC que indicou que em 2013 havia 39,9 mil vagas disponíveis no mercado nacional de tecnologia (ALMEIDA, 2013). Dados do estudo The Network Skills in Latin America, encomendado pela Cisco à Consultoria IDC, indicavam que faltaria 449 mil profissionais na América Latina até 2019, sendo mais de um terço desse montante no Brasil (COMPUTERWORLD, 2016).

Contudo, há que se ressaltar que essas áreas exigem bastante esforço de aprendizado pelo seu grau de dificuldade, principalmente no que se diz respeito à lógica de programação e à abstração de raciocínio, que são requisitos fundamentais nos cursos na área de computação (tecnologias, bacharelados e engenharias) (PEREIRA JÚNIOR; RAPKIEWICZ, 2004). Essas dificuldades podem ser indicadas como estando entre os fatores que contribuem para as altas taxas de evasão e retenção nesses cursos.

A utilização de novas estratégias de ensino de lógica de programação na educação básica pode ser uma forma de diminuir os índices de evasão nos cursos superiores que envolvem áreas exatas como Matemática, Física e Computação. Além disso, essas estratégias quando utilizadas a partir do ensino fundamental podem auxiliar a despertar o interesse pelas áreas exatas como computação e fazer com que os alunos desenvolvam com mais efetividade a capacidade cognitiva necessária ao aprendizado das demais disciplinas do ensino fundamental, básico e médio.

Contudo, é necessário não acreditar que somente o uso da tecnologia vai desenvolver habilidades e agregar conhecimento que sejam úteis ao desenvolvimento do raciocínio lógico, do pensamento abstrato e da análise crítica nas diversas áreas de conhecimento humano. Stella (2016) mostra que só o uso da ferramenta pode não ser efetivamente motivador, sendo necessário o uso de métodos mais estruturados para despertar a atenção e envolver os alunos. E a busca por esses métodos é um tema de pesquisa na computação que vincula educadores (pedagogos) e profissionais (programadores, analistas e outros).

Considerando esse contexto de importância das tecnologias de informação e comunicação como ferramentas auxiliares na realização de atividades nas mais diversas áreas de conhecimento humano e, ainda, que a simples disponibilização de equipamentos computacionais não é suficiente para que o desenvolvimento do raciocínio lógico e da

aprendizagem de conceitos de programação, neste trabalho é proposto o desenvolvimento de uma plataforma *web*. Essa plataforma permitirá compor (criar) e disponibilizar atividades que visam o desenvolvimento de raciocínio lógico e do conhecimento que possa contribuir e fundamentar o aprendizado de programação.

1.2 OBJETIVOS

A seguir estão o objetivo geral e os objetivos específicos definidos para este trabalho.

1.2.1 Objetivo Geral

Desenvolver uma plataforma para composição de atividades que visam promovem o desenvolvimento de raciocínio lógico.

1.2.2 Objetivos Específicos

- Possibilitar que tutores possam criar atividades para o desenvolvimento de raciocínio lógico e conceitos de programação para crianças;
- Promover o aprendizado de conceitos básicos de programação por meio da realização de atividades lúdicas que desenvolvam o raciocínio lógico.
- Disponibilizar atividades que possam desenvolver a ideia de conceitos básicos de programação.

1.3 JUSTIFICATIVA

Alunos de graduação geralmente apresentam dificuldades de aprendizado e de abstração de raciocínio. Isso fica evidente em disciplinas de lógica e de programação. Essas disciplinas são complexas, tornando-se desafiadoras e causando receio e/ou insegurança nos estudantes (NASCIMENTO, 2015). É possível que se o desenvolvimento do raciocínio lógico

inicie na infância, fornecendo as bases para o pensamento argumentativo, os adolescentes e os adultos tenham menos dificuldades e receio em relação ao aprendizado de disciplinas que exigem maior formalização de raciocínio e abstração. Essa exigência ocorre nas diversas disciplinas das Ciências Exatas, como Matemática, Física e Computação.

Todo mundo deveria aprender a programar computadores porque isso ensina como pensar (tradução livre da frase de Steve Jobs que consta na página *web* br.code.org (BR.CODE.ORG, 2018). Essa frase sustenta a importância do raciocínio lógico computacional não apenas para o desenvolvimento de aplicativos computacionais, mas para as diversas áreas de atividade humana.

O uso de tecnologia de jogos eletrônicos também é visto como uma forma de desenvolver rapidez de memorização e de raciocínio. Exemplos são os jogos que propõem problemas para serem resolvidos por meio da digitação de valores ou da escolha entre opções para atingir determinados objetos (como naves) que apresentem a solução do problema envolvido no jogo. Na Química, por exemplo, recursos tecnológicos podem ser usados para demonstrar reações e na Biologia para visualizar bactérias; esses dois exemplos mostram apenas algumas das muitas possibilidades do uso da tecnologia como auxiliar no ensino e na aprendizagem das ciências (VALENTIM, 2000).

A ideia de realização de atividades lúdicas com crianças para o desenvolvimento de raciocínio lógico não está fundamentada em o processo ser iniciado pelo ensino de uma linguagem de programação específica, como Java, C, C# ou PHP, mas sim ensinar a desenvolver a fundamentação do raciocínio lógico de programação que é semelhante para as diversas linguagens. Parte-se da premissa que se o aluno possuir raciocínio da lógica computacional desenvolvido ele terá facilidade para aprender a sintaxe de diferentes linguagens, pelo menos as que atendem a um mesmo paradigma de programação. O raciocínio da lógica computacional é visto como a definição e a organização das premissas e ações que fundamentam o desenvolvimento de problemas: o algoritmo que representa a solução do problema. Independentemente de implementação computacional, as entradas para resolver o problema e as saídas que representam a solução, considerando o contexto que é necessário armazenar dados (variáveis) e operar com eles (operações aritméticas, lógicas e relacionais).

Garlet, Bigolin e Silveira (2018) ressaltam que nem todos os alunos se tornarão programadores, é certo que muitos optarão por outras áreas de conhecimento. Até porque são variadas as necessidades de profissionais nas diversas áreas de conhecimento, interesse e necessidades humanas. Contudo, como destacam esses autores, os profissionais com raciocínio lógico e computacional desenvolvido que atuarão nas diversas áreas do

conhecimento terão como diferencial a capacidade de pensar de forma mais argumentada e com mais criatividade. Esses autores atribuem essas habilidades à aprendizagem da lógica de programação que pode auxiliar a desenvolver habilidades muitas vezes ocultas (GARLET; BIGOLIN; SILVEIRA, 2018).

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. Este é o primeiro capítulo é apresenta as considerações iniciais com o contexto do sistema a ser desenvolvido, os seus objetivos e a justificativa. O Capítulo 2 apresenta o referencial teórico centrado em comércio internacional. No Capítulo 3 estão as ferramentas e as tecnologias utilizadas na modelagem do sistema e que serão utilizadas na implementação subsequente do sistema. No Capítulo 4 é apresentado o resultado da realização do trabalho, ou seja, a modelagem e a implementação do sistema. Por fim estão as considerações finais seguidas das referências utilizadas na composição do texto.

2 ENSINO DE PROGRAMAÇÃO PARA CRIANÇAS

O ensino de programação para crianças tem início na década de 60, com a implementação de uma linguagem computacional denominada Logo (PAPERT; RESNICK, 1995). A linguagem de programação Logo foi desenvolvida com finalidades educacionais por um grupo de pesquisadores do Massachusetts Institute of Technology (MIT), liderados pelo Professor Seymour Papert (PRADO, 2000).

A partir da linguagem Logo surgiram outras ferramentas, tecnologias e ideias para que o aprendizado fosse centrado na pessoa que está aprendendo, no caso na criança sendo a protagonista do seu próprio aprendizado. E nesse sentido a tecnologia passou a ser vista como uma ferramenta que auxilia na construção do próprio conhecimento (ALMEIDA, 2013). Scratch e Hora do Código podem ser citadas como iniciativas sucessora da linguagem Lego, mas objetivos semelhantes.

O Scratch, como sucessor recente do Logo, é uma ferramenta desenvolvida no Media Laboratory do MIT. Embora seja inspirada na linguagem Logo, Scratch visa ser mais simples e intuitiva, uma vez que utiliza a metodologia de "clicar e arrastar" por meio de blocos. A linguagem de programação Scratch utiliza diversos tipos de mídia, possibilitando a criação de histórias interativas, animações, jogos e músicas e é possível compartilhar essas criações na Internet. Ainda que se trate de uma linguagem de programação, o processo de iniciação é rápido e o usuário pode imediatamente conceber projetos ajustados a sua faixa etária, com maior ou menor grau de mediação de um professor (OLIVEIRA; SILVA; ANDRADE, 2013).

Hora do Código (https://hourofcode.com/br) é uma iniciativa global voltada para a desmistificação do ensino da programação de computadores. A Hora do Código no Brasil está disponível no endereço https://br.code.org/. O movimento foi originalmente pensado para ser um conjunto de tutoriais, com duração de uma hora e voltado para o ensino de programação de computadores. O material não exige do usuário conhecimento prévio de programação. Esse projeto já foi traduzido para mais de 40 idiomas. Em relação à faixa etária, recomenda-se teoricamente a participação de crianças a partir de 4 anos de idade, mas a prática tem demonstrado que o ideal é a partir do momento em que as crianças estão começando a ler, ou seja, geralmente por volta dos 5 ou 6 anos (MARINHEIRO et al., 2016).

Novas metodologias e uso de tecnologias no processo de aprendizagem são amparadas na necessidade que as pessoas aprendam a programar e pelo amplo e diversificado uso de computadores na realização das mais diversas atividades humanas. O uso de recursos

computacionais tem sido visto como uma forma de melhorar, aparelhar e modernizar o processo de ensino. Isso porque são evidentes as dificuldades encontradas pelos estudantes nas disciplinas de ciências exatas e que envolvem raciocínio lógico (OCDE, 2013). A compreensão da matemática e demais ciências exatas é essencial para o desenvolvimento tecnológico. Visando minimizar as dificuldades de aprendizado apresentadas pelos estudantes e a desmotivação com a educação formal, é necessário redimensionar as abordagens metodológicas de ensino com estratégias didático-pedagógicas que permitam a construção do próprio conhecimento (TAJRA, 2012).

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado deste trabalho.

3.1 MATERIAIS

O Quadro 1 apresenta as ferramentas e as tecnologias utilizadas na modelagem e no desenvolvimento do sistema.

Quadro 1 - Ferramentas e tecnologias

Nome	Versão	Aplicação no projeto
Astah Community	v7.0	Ferramenta de modelagem.
Balsamiq Mockups	3	Ferramenta para modelagem de interface de aplicações
Material-UI	v 4.0.2	*Framework front-end* para criação de conteúdos HTML, CSS e JavaScript.
Axios	V0.18.0	Biblioteca que permite realizar requisições HTTP.
Konva.js		Biblioteca JavaScript para criação de *canvas* interativos.
ReactJS	v16.2.0	Biblioteca de criação de componentes *front-end*.
Redux		Biblioteca para controlar o estado da aplicação.
Node.js	v8.10.0	Plataforma de desenvolvimento *server-side*.
Express	v4.16.3	*Framework* Node.js.
MongoDB	v10.2	Banco de dados orientado a documentos.
Mongoose	V5.4.2	Biblioteca Node.js que permite a conexão com o banco de dados MongoDB utilizando esquemas.
Visual Studio Code	V1.35.1	Ambiente de desenvolvimento.
BitBucket		Ferramenta de controle de versão.
JavaScript	ES6	Linguagem de programação.
Robo 3T	1.3	Para desenvolvimento de interface gráfica para MongoDB.

Fonte: autoria própria.

A seguir está uma breve descrição das ferramentas apresentadas no Quadro 1:

a) Astah Community – ferramenta para modelagem utilizando a *Unified Modeling Language* (UML). A versão *community* é gratuita. A versão paga possui funcionalidades adicionais como a de gerar o código Java a partir das classes de modelagem (ASTAH, 2018).

b) Balsamiq Mockups – ferramenta para a modelagem da interface do aplicativo utilizando a ideia de desenho a mão. Os componentes de interface utilizados para compor e definir o leiaute dos aplicativos fornecem a impressão de desenho em papel. Contudo, essa

ferramenta oferece vantagens de arrastar-e-soltar componentes para organizá-los na tela do formulário, relatório ou outra interface da aplicação (BALSAMIQ MOCKUPS, 2018).

c) Material-UI - é uma biblioteca *open source* de componentes *front-end* para ReactJS baseada em JavaScript licenciada pelo MIT (MATERIAL-UI, 2019).

d) Axios - é uma biblioteca *Application Programming Interface* (API) baseada em *promises* que permitem a interação com XMLHttpRequest, funcionando em navegadores ou em ambientes de servidor Node.js (BERNARDES, 2015).

e) Konva.js - é uma biblioteca JavaScript para criação de *canvas* 2D interativos (KONVAS.JS, 2019).

f) ReactJS - é uma ferramenta para criar componentes de interfaces interativas (REACT, 2018)**.** Componentes *web* podem ser vistos como *tags HyperText Markup Language* (HTML) customizadas que encapsulam estrutura (HTML), estilo *Cascading Style Sheet* (CSS) e comportamento (JavaScript) com o objetivo de serem reaproveitáveis.

g) Redux - é uma biblioteca JavaScript *open source*, baseada na arquitetura Flux utilizada pelo Facebook. O Redux é uma alternativa para controle de estado global de uma aplicação (REDUX, 2019).

h) Node.js – é uma plataforma construída sobre o motor JavaScript do Google Chrome. Node.js usa um modelo de entrada e saída direcionada a evento não bloqueante, tornando-se ideal para aplicações em tempo real com grande volume de dados trocados por meio de dispositivos distribuídos (NODE.JS, 2018).

i) Express – é um *framework* para Node.js que fornece um conjunto de recursos para desenvolvimento de aplicativos *web* e móveis (EXPRESS, 2018).

j) MongoDB - é um banco de dados distribuído, baseado em documentos de uso geral e para a computação em nuvem (MONGODB, 2019).

k) Mongoose - é uma biblioteca do Nodejs que oferece uma solução baseada em esquemas para modelar os dados da aplicação. Mongoose oferece um sistema de conversão de tipos, validação, criação de consultas e outros. Mongoose fornece um mapeamento de objetos do MongoDB similar ao *Object Relational Mapping* (ORM) que é o *Object Data Mapping* (ODM), traduzindo os dados do banco de dados para objetos JavaScript para que possam ser utilizados pela aplicação (NODEBR, 2016).

l) Visual Studio Code – é um editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS (MICROSOFT, 2019).

m) BitBucket – é um sistema de controle de versão distribuído que visa facilitar a colaboração entre os membros de uma equipe de desenvolvimento ou de projeto

(BITBUCKET, 2018). BitBucket é uma solução Git colaborativa similar ao GitHub. O Git é um sistema de controle de versão distribuído e de gerenciamento de código fonte. Cada diretório de trabalho do Git é um repositório com um histórico completo e permite o acompanhamento das revisões, sem dependência de acesso a uma rede ou a um servidor central.

n) Robo 3T – formalmente denominada Robomongo é uma ferramenta de interface gratuita para o MongoDB (ROBO 3T, 2019).

3.2 MÉTODO

As definições iniciais e a ideia básica para o desenvolvimento da plataforma foram inspiradas no code.org.br. A definição de como o tutor faria a composição das atividades foi inspirada na composição de interface de aplicativos baseada em componentes que são arrastados para a tela. A forma de composição das atividades utiliza conceitos de orientação a objetos.

Os requisitos foram definidos a partir da análise do code.org.br e de material pesquisado para a composição do referencial teórico. Foram, ainda, analisadas atividades como jogos e quebra-cabeças disponibilizados na Internet. Dois atores, tutor e aluno, com as suas permissões de acesso foram estabelecidos. O primeiro compõe as atividades e as disponibiliza. O segundo realiza atividades disponibilizadas. Tanto a composição quanto a realização das atividades são desenvolvidas da forma semelhante, ou seja, não há necessidade de comandos ou instruções especiais para compor uma atividade.

Os requisitos foram organizados tendo como base as funcionalidades planejadas para o sistema, considerando o seu escopo e contexto, e as permissões desses atores considerando essas funcionalidades.

Um protótipo da tela para composição de atividades foi desenvolvido com a ferramenta Balsamiq para entendimento de como o tutor poderia compor as atividades e o aluno realizar essas atividades. A distribuição dos elementos na tela e a forma de interação também foram avaliadas a partir desse protótipo.

A partir do protótipo da tela e do banco de dados definido foi realizada a implementação. O protótipo forneceu a ideia básica de como seria a interface principal de interação com o sistema. O leiaute final da interface da aplicação ficou relativamente

diferente do protótipo, até porque alguns requisitos inicialmente planejados foram revistos e redimensionados. Havia sido inicialmente planejado existir elementos que o usuário arrastasse na tela, mas o contexto da aplicação ficou centrado na indicação de instruções para que as ações fossem realizadas, fazendo, assim, maior necessidade de pensar na sequência lógica das instruções necessárias para resolver o problema.

Os testes foram informais e realizados à medida que o código era desenvolvido visando identificar erros de codificação e verificar o atendimento aos requisitos propostos.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho, incluindo a modelagem e a implementação do código do sistema.

4.1 ESCOPO

O sistema é definido como uma plataforma *web* para "quebra-cabeças" voltados para o desenvolvimento do raciocínio lógico e aprendizado de conceitos básicos de programação. O escopo da plataforma é definido para crianças como forma de delimitar o seu contexto e, de certa forma, pela ênfase na possibilidade de aprendizado por meio do uso de jogos e *puzzles*. Porém, ressalta-se que jogos e *puzzles* podem ser utilizados para o desenvolvimento de raciocínio lógico e o aprendizado de conceitos fundamentais de programação por pessoas de qualquer faixa etária.

A criação das atividades na plataforma desenvolvida é realizada pelos tutores e as atividades podem ser elaboradas com níveis de dificuldade diferentes e envolver conceitos de programação, básicos ou mais avançados, adaptando-se, assim, ao seu público alvo.

O acesso à plataforma é realizado por dois perfis de usuário distintos: alunos e tutores. Os tutores fazem a inclusão de atividades que ficarão disponíveis para que os alunos possam realizá-las. Os alunos terão acesso à plataforma e aos conteúdos criados pelos tutores e realizarão as tarefas cadastradas e disponibilizadas.

Essas atividades serão criadas por meio de uma ferramenta da plataforma desenvolvida com auxílio de uma biblioteca JavaScript chamada Konva.js. A ferramenta é específica para a criação de atividades envolvendo instruções e estruturas de programação. A ferramenta disponibilizará elementos com atributos que serão utilizados para montar fluxogramas e modelar processos que o usuário realizará utilizando-os como método de aprendizado.

As atividades cadastradas devem ser de fácil compreensão para quem irá resolvê-las. Um exemplo de atividade seria de um caminho que uma abelha tem que fazer até chegar até determinada flor para colher seu pólen. Desafios como esses são simples, mas o raciocínio para resolver o problema terá que partir do usuário (aluno). O deslocamento do "cursor" para realizar o caminho necessário ou desenhos pré-determinados será realizado por meio de comandos para frente, para trás, esquerda e direita.

As atividades disponibilizadas pela plataforma terão como forma de ensino atividades que envolvam conceitos de variáveis e repetições e outros relacionados ao desenvolvimento de raciocínio lógico e de programação.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta os requisitos funcionais definidos para o sistema. Nesse quadro a sigla RF significa Requisito Funcional.

Quadro 2 – Requisitos funcionais

Identificação	Requisito	Descrição
RF01	Manter usuários	O sistema terá controle de acesso por tipo de usuário. O próprio usuário fará o seu cadastro e poderá alterar dados do cadastro ou desativar o seu usuário. Cada usuário terá permissões conforme as selecionar durante o seu cadastro. Os usuários que realização o seu próprio cadastro poderão escolher entre os perfis de tutores ou alunos. Os usuários do tipo tutor terão acesso aos requisitos relacionados ao cadastro e alterações de atividades. Já usuários do tipo alunos poderão realizar as atividades cadastradas na plataforma. De forma geral, os tutores criam as tarefas e os alunos as realizam.
RF02	Compor Atividades	Cada atividade será tratada como um projeto, tendo um nome, o tutor que está realizando o cadastro da referida atividade e os elementos que o tutor inserirá na composição da atividade. Ao iniciar uma nova atividade o usuário poderá vincular elementos pré-cadastrados com o objetivo de formar um fluxograma ou processo, sendo possível representar raciocínio lógico e decisões necessárias para resolver a atividade. Somente após essa alteração o aluno terá acesso à atividade previamente elaborada pelo tutor.
RF03	Realizar Atividades	A realização das atividades será conduzida por usuários do tipo aluno. Os usuários realizarão os fluxogramas para que a atividade seja concluída. As atividades terão *status* por aluno de "concluída" ou "não concluída". Para concluir uma atividade, o aluno terá que finalizá-la. Os alunos terão acesso somente às atividades ativas, ou seja, disponibilizadas para uso. As atividades realizadas pelo aluno serão listadas e terão *status* de concluídas ou não concluídas, conforme o aluno realizar o seu desenvolvimento. Um aluno pode refazer uma atividade tenha ela sido concluída com êxito ou não.

Fonte: autoria própria.

No Quadro 3 estão os requisitos não funcionais definidos para o sistema. Nesse quadro RNF significa Requisito Não Funcional.

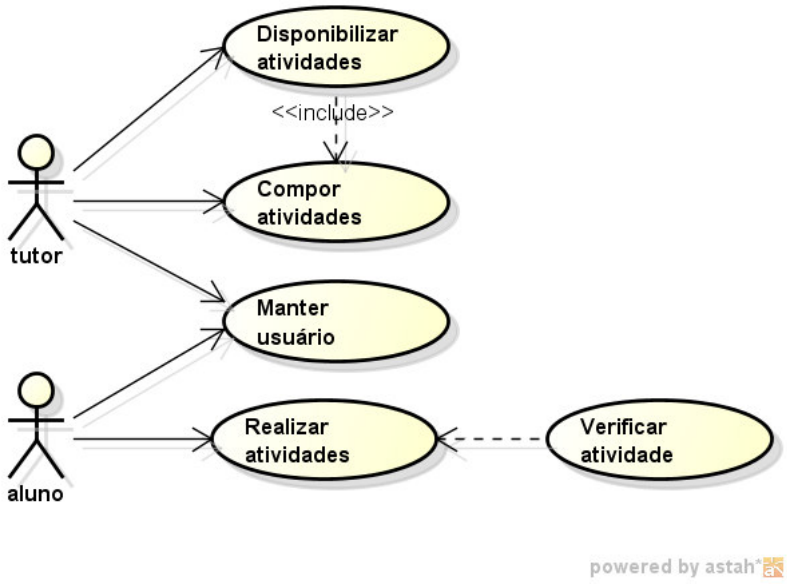
Quadro 3 – Requisitos não funcionais

Identificação	Requisito	Descrição
RNF01	Acesso ao sistema	Para ter acesso é necessário ter *login* e senha previamente cadastrados. O cadastro é realizado pelo próprio usuário que escolhe entre um dos dois perfis de acesso disponíveis: tutor e aluno. Após o cadastro o tipo do usuário não poderá ser alterado, pelos vínculos que cada perfil de usuário gera no sistema.
RNF02	Validação de atividades	As atividades são criadas pelos usuários com o perfil de tutor, mas elas somente são disponibilizadas para os alunos.
RNF03	Definir permissões de acesso	As permissões serão atribuídas ao usuário na realização do cadastro. As permissões de usuário serão separadas em dois níveis: o usuário tutor que terá acesso ao cadastro e à edição de projetos de atividades; e o usuário com perfil de aluno que terá acesso as atividades cadastradas e disponibilizadas para realização. As permissões serão vinculadas aos usuários assim que eles forem cadastrados, não haverá validação de usuários cadastrados.
RNF04	Recuperar usuário	Se um usuário que estiver inativo realizar autenticação no sistema, com *login* e senha válidos para os registros constantes no banco de dados, o sistema informará que o cadastro já existe e que o histórico do referido usuário será recuperado.

Fonte: autoria própria.

O diagrama da Figura 1 apresenta os casos de uso do sistema. Esses casos de uso estão organizados por ator que são: a) tutor – ator responsável pela criação ou composição de atividades que posteriormente são disponibilizadas para realização pelos alunos; b) aluno – acessa a plataforma para realizar atividades propostas.

Figura 1 – Diagrama de casos de uso



Fonte: autoria própria.

Nos Quadros 4 a 7 estão as descrições das ações de inclusão, alteração, exclusão e consulta do caso de uso manter usuários, representado na Figura 1.

Quadro 4 – Operação de inclusão de cadastro de usuários

Casos de uso: Manter usuários: operação de inclusão de um novo usuário para acesso ao sistema. **Descrição:** Inclusão dos dados cadastrais de usuário no sistema, com a definição do perfil de acesso. **Evento Iniciador:** Ator solicita inclusão de um usuário no sistema. **Atores:** Usuário que ainda não possui acesso ao sistema e está fazendo o próprio cadastro. **Pré-condição:** Não há. **Sequência de Eventos:** **1.** Ator acessa a tela para realizar o cadastro inserindo as informações necessárias. **2.** Ator seleciona o tipo de usuário. O ator está realizando o próprio cadastro, ele poderá escolher aluno ou tutor como perfil de acesso. **3.** O sistema insere as informações no banco de dados e informa o *status* do procedimento. **Pós-Condição:** Usuário inserido no banco de dados.	
Nome do fluxo alternativo (extensão)	Descrição
1.Campos obrigatórios não informados	1.1. O ator deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2.Campos informados em formato incorreto	2.1. O ator informa dados em formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.

Fonte: autoria própria.

No Quadro 5 está a descrição da operação alterar dos casos de uso Manter usuário.

Quadro 5 – Operação para altera dados de cadastros de usuários

Caso de uso: Manter usuários: operação de alteração de dados cadastrais de um usuário do sistema. **Descrição:** Alteração dos dados cadastrais no sistema. **Evento Iniciador:** Usuário aluno ou tutor acessa o seu cadastro para alterar dados de cadastro no sistema. **Atores:** Tutor ou aluno. **Pré-condição:** O cadastro do usuário deve estar no sistema e usuário com *login* ativo. **Sequência de Eventos:**

1. Ator acessa a tela para visualização dos dados do registro e seleciona o registro que deseja alterar. 2. O sistema apresenta o registro selecionado para alteração. 3. Usuário altera os dados do registro. 4. O sistema altera as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Dados de cadastro alterados no banco de dados.

Nome do fluxo alternativo (extensão)	Descrição
1.Campos obrigatórios não informados	1.1. O ator deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2.Campos informados em formato incorreto	2.1. O ator informa dados em formato incorreto e clica em salvar. 2.2. O sistema valida que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.

Fonte: autoria própria.

A operação de exclusão de cadastros de usuários no sistema é descrita no Quadro 6.

Quadro 6 – Operação de exclusão de cadastro de usuários

Caso de uso: Manter usuários: operação de exclusão de cadastros de um usuário do sistema. **Descrição:** Exclusão dos dados cadastrais no sistema. A operação de exclusão tornará o usuário inativo, mas não excluirá os registros vinculados ao referido usuário do banco de dados. O referido cadastro apenas será marcado como inativo. **Evento Iniciador:** Aluno ou tutor solicita exclusão do seu cadastro. **Atores:** Tutor, aluno. **Pré-condição:** Cadastro estar incluso no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para exclusão do registro. 2. O sistema exclui as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Cadastro de usuário marcado como inativo no banco de dados.

Fonte: autoria própria.

No Quadro 7 está a descrição da operação consultar referente aos casos de uso manter usuários.

Quadro 7 – Operação de consulta de dados de cadastro de usuário

Caso de uso: Manter usuários: operação para consulta cadastros de um usuário do sistema. **Descrição:** Consulta dos dados de usuário do sistema. **Evento Iniciador:** Ator solicita consulta de um registro no sistema. **Atores:** Administrador. **Pré-condição:** Registro estar incluso no sistema. **Sequência de Eventos:** **1.** O autor solicita a visualização os seus dados cadastrais. **2.** O sistema apresenta os dados da consulta ao usuário. **Pós-Condição:** Dados do cadastro apresentados para o usuário.

Fonte: autoria própria.

A expansão do caso de uso compor atividade é apresentada no Quadro 8.

Quadro 8 – Caso de uso compor atividade

Casos de uso: Compor atividade **Descrição:** Tutores compõem atividades a partir do Konva.js que é uma biblioteca JavaScript para criação de *canvas* animados disponibilizado na plataforma. **Evento Iniciador:** Tutor cria uma atividade para disponibilizar na plataforma **Atores:** Tutor. **Pré-condição:** Não há. **Sequência de Eventos:** 1. Ator acessa a tela do *canvas* na plataforma e utilizando os recursos disponibilizados cria a atividade. Ator salva atividade. 2. O sistema insere a atividade no banco de dados e informa ao usuário o *status* do procedimento. Atividade será disponibilizada para usuários do tipo aluno para que possam ser realizadas. **Pós-Condição:** Registro inserido no banco de dados.

Fonte: autoria própria.

A descrição do caso de uso realizar atividade é apresentada no Quadro 9.

Quadro 9 – Caso de uso realizar atividade

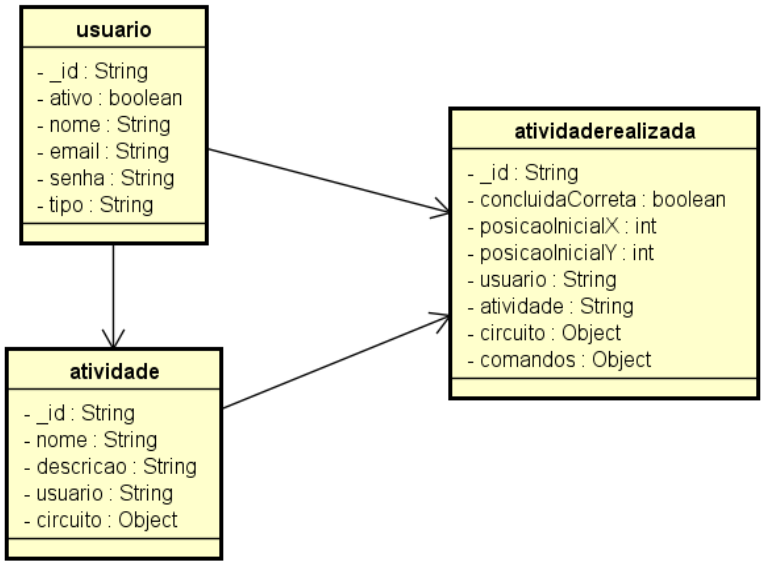
Casos de uso: Realizar atividade. **Descrição:** Alunos realizam atividades disponibilizadas na plataforma. **Evento Iniciador:** Aluno seleciona a atividade para ser realizada.

Atores:
Aluno.
Pré-condição:
Não há.
Sequência de Eventos:
1. Ator acessa a tela para realizar atividades e seleciona a atividade que quer realizar.
2. Sistema disponibiliza a atividade.
3. Aluno realiza a atividade e salva mantendo a solução vinculada ao seu usuário.
4. Sistema faz a verificação da atividade e armazena o *status* e se a mesma foi realizada corretamente conforme o padrão.
Pós-Condição:
Atividade realizada pelo aluno com o respectivo *status* armazenado no banco e vinculado ao usuário.

Fonte: autoria própria.

A Figura 2 apresenta o diagrama com as entidades do banco de dados.

Figura 2 – Diagrama do banco de dados



Fonte: autoria própria.

Na Figura 2, a entidade “usuario” mantém os dados para autenticação no sistema, por meio de usuário e senha, cada usuário terá um tipo, conforme escolhido no momento do cadastro. A entidade “atividade” contém dados das atividades inseridas (cadastradas) na plataforma. A entidade “atividaderealizada” terá associações com o usuário criador da atividade e o usuário que realizou a atividade.

Na Listagem 1 está a descrição do documento que representa a entidade “usuário” no banco de dados.

Listagem 1 – Documento entidade usuário do banco de dados

```
{
    "_id" : "",
    "ativo" : true,
    "nome" : "",
    "email" : "",
    "senha" : "",
    "tipo" : ""
}
```

Fonte: autoria própria.

Na Listagem 2 está o documento que é a entidade "atividade" do banco de dados que contém dados das atividades cadastradas na plataforma. Esse documento contém as informações de nome da atividade, descrição, o usuário que a criou e um *array* de objetos circuito que possuí todas as informações de posição e *checked* referente à atividade cadastrada. Uma atividade contém todas as posições do *grid* e a indicação se alguma foi alterada ou não.

Listagem 2 – Documento entidade atividade do banco de dados

```
{
    "_id" : "",
    "circuito" : [
        [
            {
                "_id" : "",
                "pos" : "",
                "checked" : true
            }
        ]

    ],
    "nome" : "",
    "descricao" : "",
    "usuario" : ""
}

```

Fonte: autoria própria.

O documento "atividaderealizada", apresentado na Listagem 2, terá associações com o usuário que realizou a atividade, as posições iniciais e finais do circuito, o id da atividade referência, um *array* de objetos, que espelha o *array* da própria atividade como sendo o

circuito realizado pelo aluno. E terá, ainda, um *array* de objetos de comandos que contém o nome do comando, o *index* da lista se é um comando de repetição e, também, um *array* de objetos de comandos de repetição e um atributo para verificar se a atividade foi concluída corretamente ou não.

Listagem 3 – Documento entidade atividade realizada do banco de dados

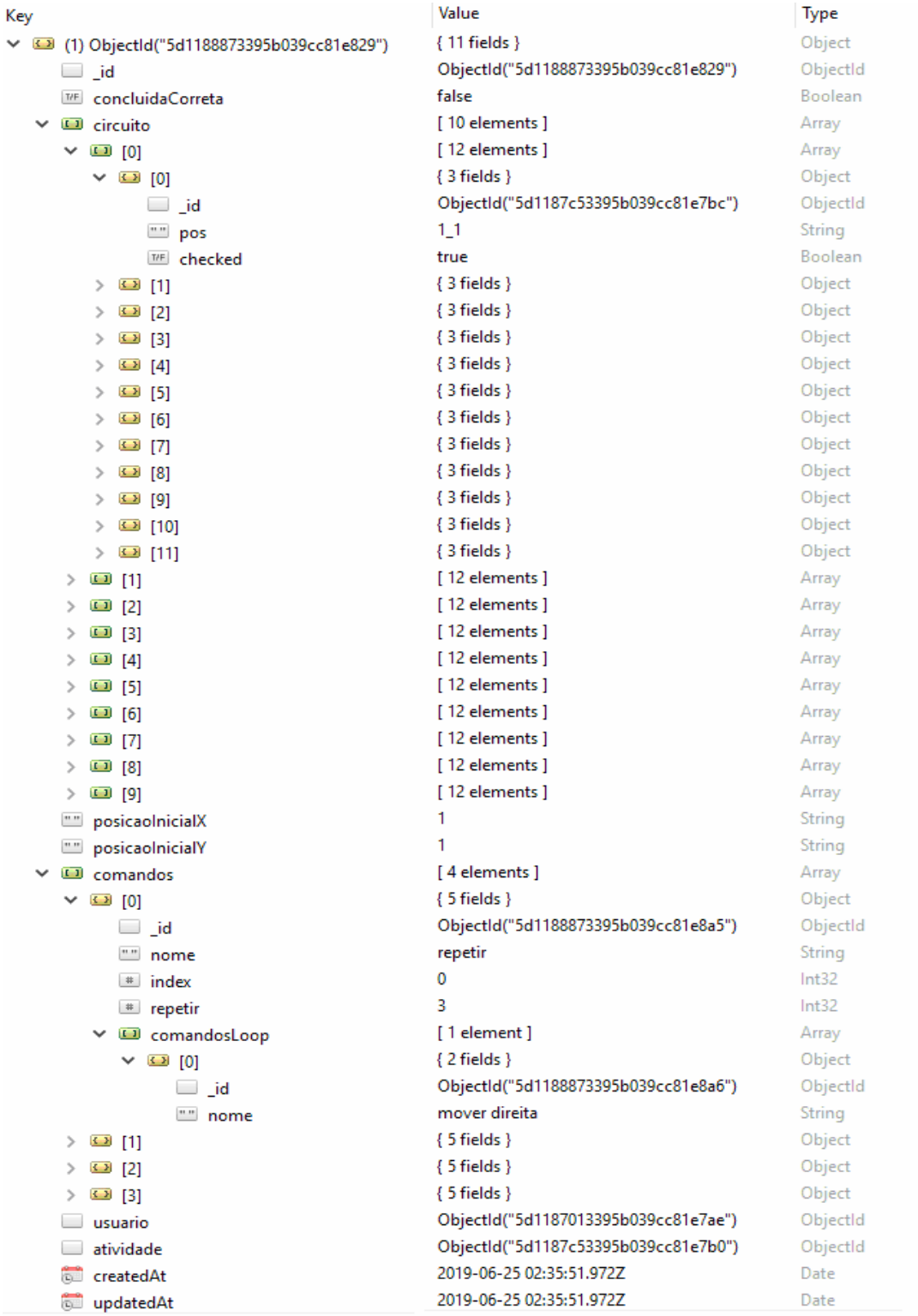
```
{
    "_id" : "",
    "concluidaCorreta" : false,
    "circuito" : [
        [
            {
                "_id" : "",
                "pos" : "",
                "checked" : true
            }
        ]

    ],
    "posicaoInicialX" : 0,
    "posicaoInicialY" : 0,
    "comandos" : [
        {
            "_id" : "",
            "nome" : "",
            "index" : 0,
            "repetir" : 0,
            "comandosLoop" : [
                {
                    "_id" : "",
                    "nome" : ""
                }
            ]
        }
    ],
    "usuario" : "",
    "atividade" : ""
}
```

Fonte: autoria própria.

O Robo 3T é utilizado como ferramenta de gerenciamento para acesso ao banco de dados MongoDB. A Figura 3 apresenta a estrutura de documento de atividaderealizada, como exemplo. Nesse documento são utilizados dois tipos de referência: uma com o ObjectId do documento de atividade e usuário; e outra no formato de objeto que é o caso dos comandos e dos circuitos (o caminho que deve ser desenhado na tela para resolver a atividade proposta).

Figura 3 – Estrutura do documento atividaderealizada

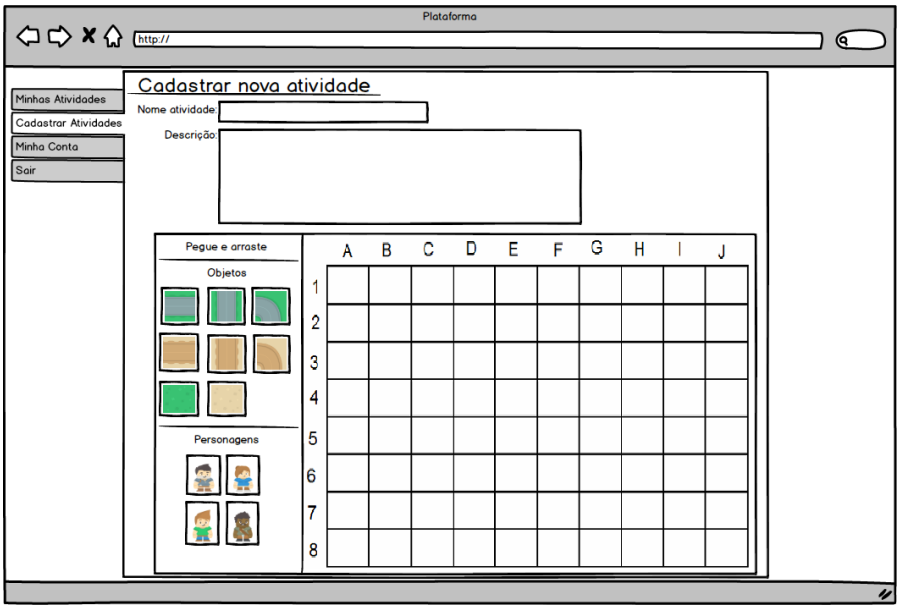


Fonte: autoria própria.

A Figura 3 apresenta um protótipo desenvolvido com a ferramenta Balsamiq Mockups. Esse é o protótipo planejado para a tela de cadastro e composição de atividades. O

usuário tutor tem acesso a essa tela e a utiliza para criar atividades que serão disponibilizadas para que os usuários alunos possam realizá-las.

Figura 3 – Protótipo da tela para criar atividades



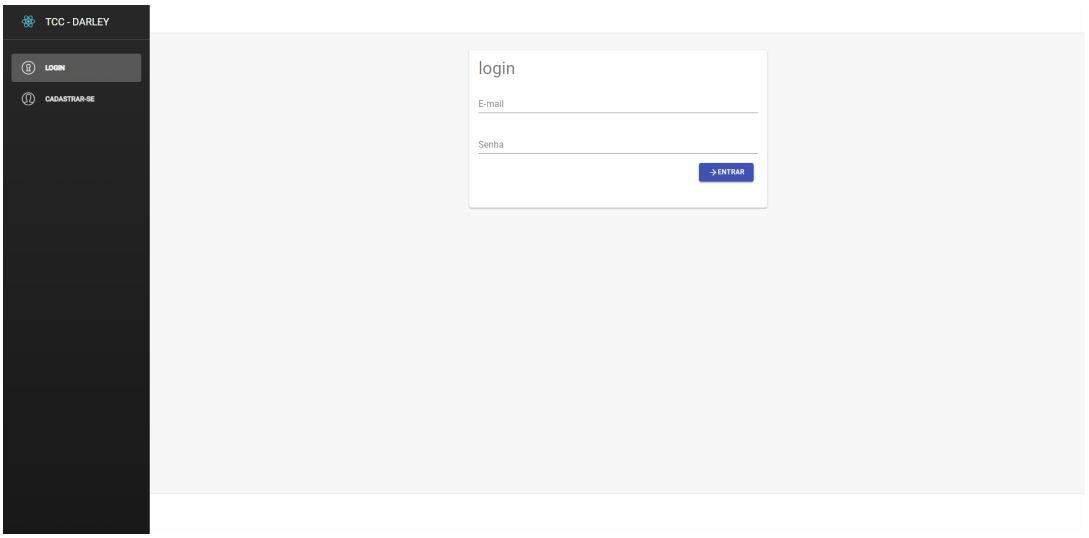
Fonte: autoria própria.

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto por duas partes: a sidebar esquerda que inclui os módulos disponíveis e o contêiner central que contém os formulários e as listas, conforme seleção do módulo na *sidebar*.

A Figura 4 apresenta o processo de autenticação (*login*) do sistema e mostra a separação da *sidebar* na esquerda e o contêiner central de informação.

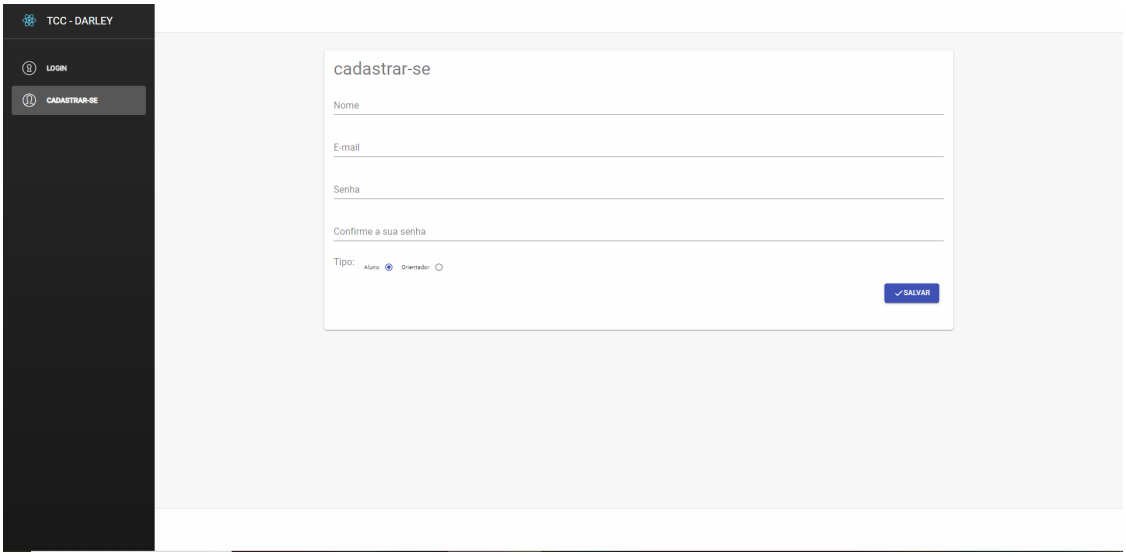
Figura 4 – Tela de login



Fonte: autoria própria.

A Figura 5 mostra o formulário de cadastro de um novo usuário, que pode ser tutor ou aluno, sendo que todos os dados do formulário são de preenchimento obrigatório. Caso algum campo não seja preenchido, o sistema avisará por meio de um *toast* (notificação) na tela sinalizando os campos que devem ser preenchidos. Ao efetuar o cadastro, o usuário estará autenticado no sistema, não sendo necessário inserir suas credenciais novamente na tela de *login*.

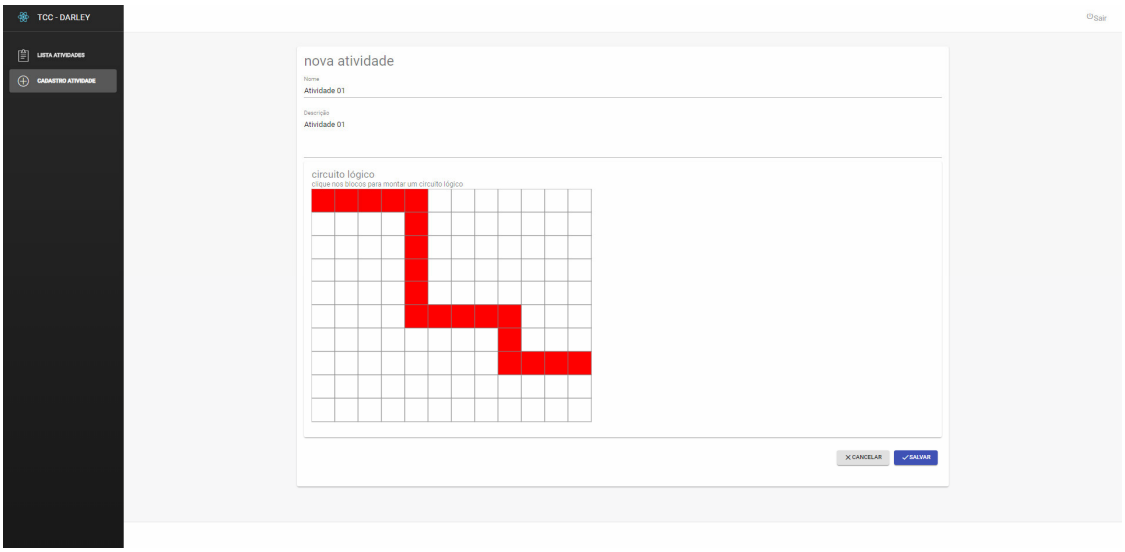
Figura 5 – Tela de cadastro de usuário



Fonte: autoria própria.

A Figura 6 apresenta as informações de um usuário do tipo “Tutor”. Na tela de cadastro de uma atividade, o tutor inserirá as informações de nome e descrição, ambas obrigatórias e criará um caminho lógico no *canvas*, para que o aluno possa realizar/resolver a atividade.

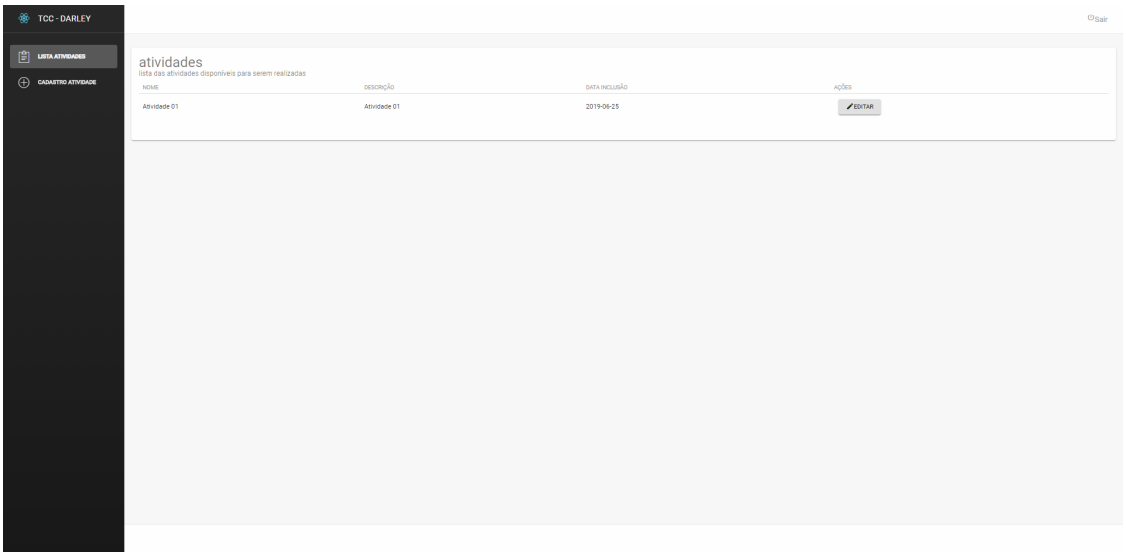
Figura 6 – Tela de cadastro de atividade



Fonte: autoria própria.

A Figura 7 contém as atividades cadastradas pelos tutores, podendo ser editadas caso ainda não tenham vínculos com as atividades realizadas pelos alunos. Nesse caso uma notificação será mostrada em tela informando que não poderá ocorrer a alteração da atividade, pois está vinculado às atividades realizadas.

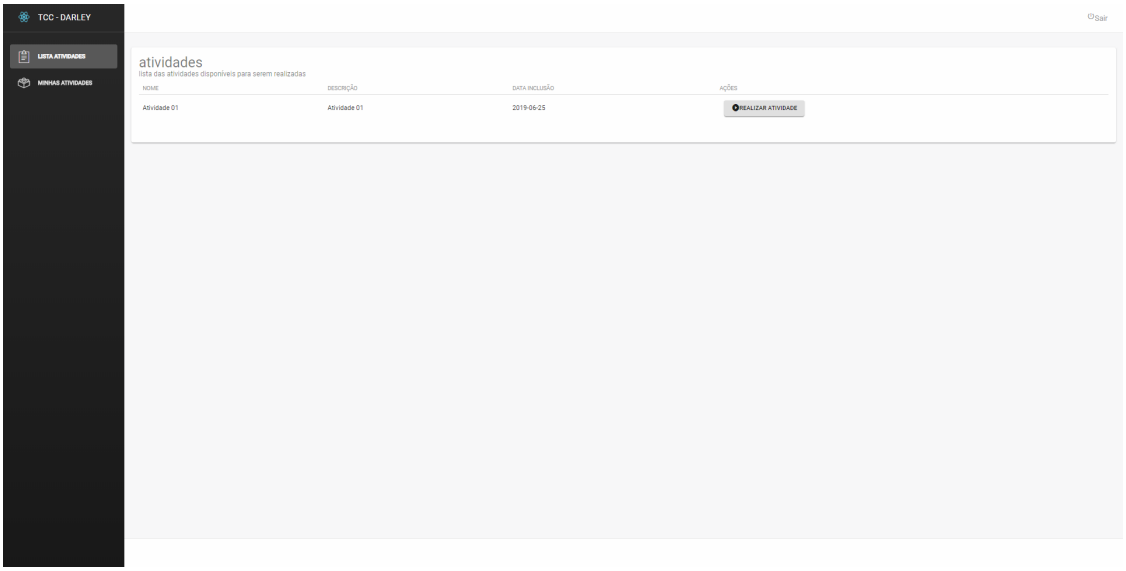
Figura 7 – Tela de lista de atividade tutores



Fonte: autoria própria.

A Figura 8 apresenta a tela de listagem das atividades no perfil de aluno. O botão de editar será alterado para “realizar atividade”, se clicado redirecionará para a tela para realizar a atividade.

Figura 8 – Tela de lista de atividade alunos

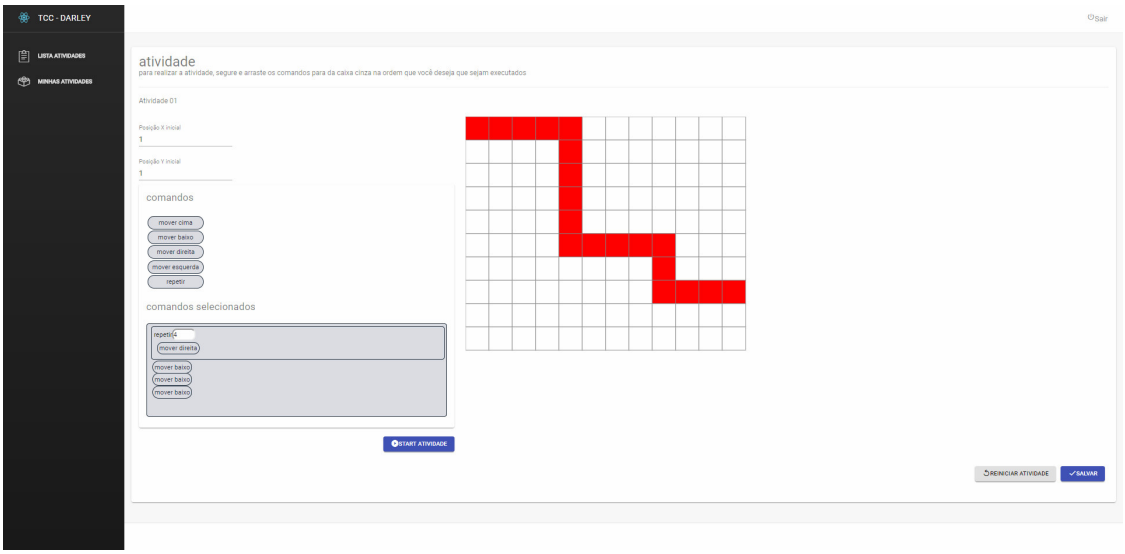


Fonte: autoria própria.

A Figura 9 apresenta o processo para realizar uma atividade. Após o aluno selecionar a atividade que deseja resolver, ao clicar no botão realizar atividade, será redirecionado para a

tela de realização da atividade. Nessa tela, o aluno inserirá as informações conforme o circuito lógico desenhado pelo tutor. Primeiramente é necessário informar as posições iniciais do circuito, após isso, devem ser arrastados os comandos mover cima, mover baixo, mover esquerda, mover direita e repetir para a caixa de comandos. Esses comandos serão executados na sequência inserida, respeitando os *loops* e comandos de mover nas direções, após isso o usuário poderá iniciar a realização da atividade. Essa ação faz com que as instruções inseridas sejam transformadas na representação gráfica que é apresentada na parte central da tela (traço de cor vermelha no *grid* apresentado na Figura 9).

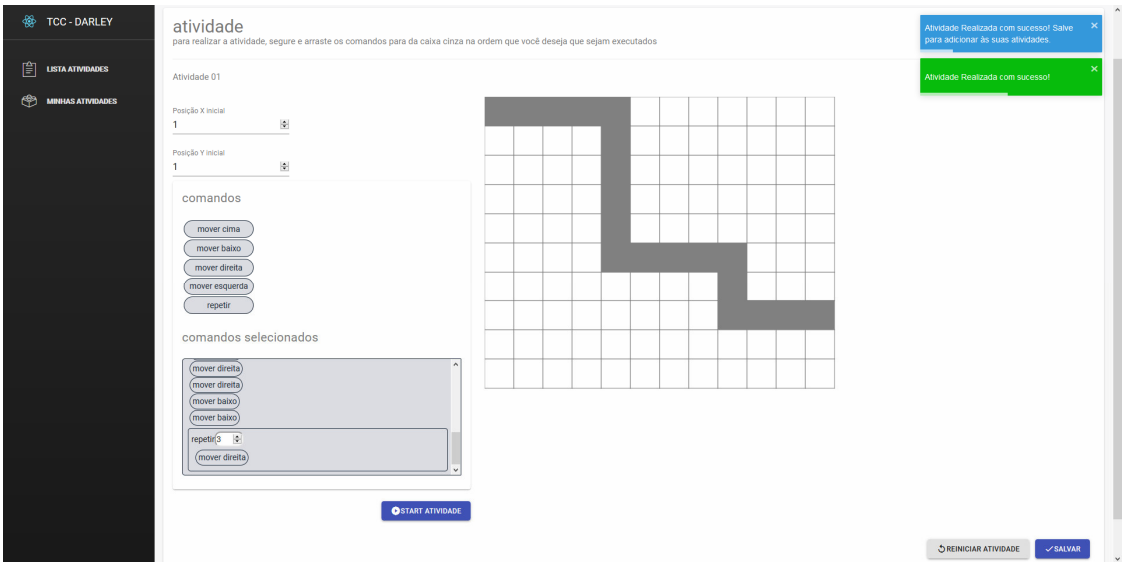
Figura 9 – Tela de realização da atividade



Fonte: autoria própria.

Na Figura 10 um usuário aluno completou corretamente o percurso, utilizando os comandos solicitados. Ao realizar a atividade, o sistema mostrará uma notificação. Caso o usuário queira armazenar o histórico da atividade é necessário salvá-la.

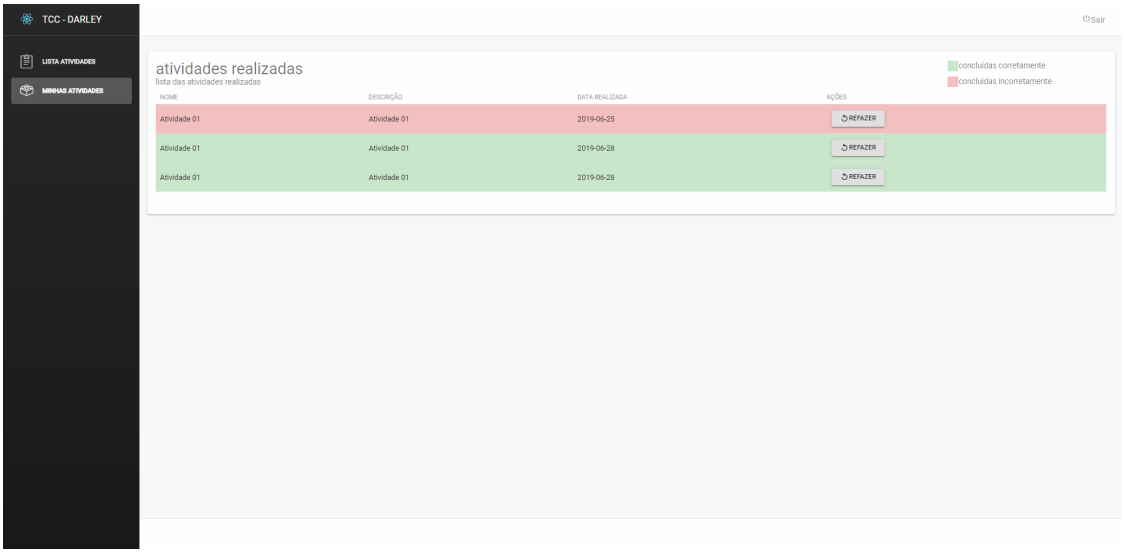
Figura 10 – Tela de realização da atividade concluída



Fonte: autoria própria.

A Figura 11 apresenta a lista das atividades realizadas pelo usuário aluno que está autenticado no sistema. São apresentadas em cores diferentes, se a atividade foi concluída corretamente ou não. Nessa tela, o usuário poderá refazer a atividade, sendo possível carregar a atividade como foi realizada anteriormente, com os mesmos comandos e posições iniciais.

Figura 11 – Tela de lista de atividades realizadas pelo aluno



Fonte: autoria própria.

4.4 IMPLEMENTAÇÃO DO SISTEMA

Para o desenvolvimento do *back-end* foi utilizada a linguagem JavaScript, juntamente com o Node.js que é uma plataforma de desenvolvimento *server-side*, que utiliza a Engine V8 do Google que é um interpretador JavaScript.

A Listagem 4 mostra o início do desenvolvimento da aplicação *server-side*. Para que seja possível iniciar um servidor *node* foi utilizada a biblioteca *Express*, que facilita a criação, utilizando o método *listen*, para que o servidor permaneça ativo executando em uma porta definida. A sequência de código dessa listagem mostra o arquivo server.js, que contém as configurações de conexão, adições de *middlewares* e definições de rotas. Inicialmente foi definida uma constante *app* que recebe o escopo do *express*. A partir dela é possível adicionar *middlewares* para que sejam definidos no escopo da aplicação *express*. O método connectDB é responsável pela conexão com o banco de dados MongoDB. A biblioteca *cors* é utilizada para que seja possível a troca de mensagens entre navegadores (*browsers*) e servidores de domínios distintos. Em seguida, foi alterada a configuração do *express* para tratar o objeto *body* como JSON e adicionadas as rotas de acesso no escopo da aplicação. Cada rota requer um arquivo de rota. Por fim, foi adicionado um *listen* para que a aplicação possa ficar disponível na porta que for definida. O método *listen* tem dois parâmetros: o primeiro é a porta que será liberada para acesso e o segundo uma função de *callback* que retorna a conexão do servidor.

Listagem 4 – server.js

```
const express = require('express');
const connectDB = require('./config/db');
const cors = require('cors');
const app = express();

// Connect Database
connectDB();

// Cors
app.use(cors());

// Init middleware
app.use(express.json({ extended: false }));

// Define Routes
app.use('/api/auth', require('./routes/api/auth'));
app.use('/api/usuario', require('./routes/api/usuarios'));
```

```
app.use('/api/atividade', require('./routes/api/atividade'));
app.use('/api/atividade-realizada',
require('./routes/api/atividadeRealizada'));
app.use('/api/rendimento', require('./routes/api/rendimento'));
const PORT = process.env.PORT || 3001;
app.listen(PORT, () => {
console.log(`Server started on port ${PORT}`);
});
```

Fonte: autoria própria.

A Listagem 5 mostra como foi realizada a conexão com o banco de dados MongoDB. Primeiramente é requerido o módulo do *mongoose* e o módulo Config que contêm as configurações padrões de conexão e do servidor. Em seguida, é criada uma função na qual será iniciada a conexão com o MongoDB. A função utiliza *async*, ou seja, é uma função assíncrona. Quando utilizado o comando *await* em alguma requisição que tenha como resposta um *callback*, o processo será pausado até que a função *callback* retorne o valor esperado, nesse caso a resposta é a conexão bem sucedida com o banco de dados.

Listagem 5 – db.js

```
const mongoose = require('mongoose');
const config = require('config');
const db = config.get('mongoconnection');

const connectDB = async () => {
try {
await mongoose.connect(db.URI, {
useNewUrlParser: true,
useCreateIndex: true,
useFindAndModify: false
});
console.log('MongoDB conectado...');
} catch (e) {
console.error('Error:' + e.message);
// código 1 = falha
process.exit(1);
}
}
module.exports = connectDB;
```

Fonte: autoria própria.

A Listagem 6 apresenta a implementação de um *middleware* de autenticação, que utiliza a biblioteca JSONWEBTOKEN (JWT) para validar a autenticidade do usuário que

requisitou qualquer que seja o *end-point* do servidor. Primeiramente, ele requer o módulo do *jsonwebtoken* e o módulo com os dados padrões de configuração. Após isso, ele cria uma função com os parâmetros *request*, *response* e *next*. No *request* estão contidas todas as informações referentes àquela requisição, o *response* é o parâmetro de resposta e o *next* faz com que a próxima requisição seja chamada em caso de fila de requisições. Ao ter acesso ao *token* que está no *header* da requisição, é verificado na função *verify* se o *token* ainda é válido conforme a *secret* que está salva nas configurações padrão. Se validado, retorna o objeto usuário do *payload* do *token* para o *request*, caso contrário retorna um erro 401 com a mensagem de *token* inválido.

Listagem 6 – auth.js

```
const jwt = require('jsonwebtoken');
const config = require('config');

module.exports = function (req, res, next) {

    // Pega o token da tag Authorization do header
    const token = req.header('Authorization');

    // Verifica se existe um token
    if (!token) {
        return res.status(401).json({ msg: 'Token de autorização
inválido!' });
    }
    // Decodifica o token e insere o usuário no request
    try {
        const decoded = jwt.verify(token, config.get('jwtSecret'));
        req.usuario = decoded.usuario;
        next();
    } catch (err) {
        res.status(401).json({ msg: 'Token de autorização inválido!' });
    }
}
```

Fonte: autoria própria.

A Listagem 7 representa o esquema da *collection* atividade do banco de dados MongoDB. Primeiramente é requerido o módulo do *mongoose*, a biblioteca de conexão com o MongoDB. Em seguida é criado um novo objeto *schema*. Os atributos de um *schema* podem ser objetos, *arrays* ou variáveis, todas com um tipo, conforme a utilização. Por fim, o *schema* é exportado com a função *model* do *model mongoose*.

Listagem 7 – models/Atividade.js

```
const mongoose = require('mongoose');

const AtividadeSchema = new mongoose.Schema({
    nome: {
        type: String,
        required: true
    },
    descricao: {
        type: String,
        required: true
    },
    circuito: [[
        {
            pos: String,
            checked: Boolean
        }
    ]],
    usuario: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'usuario'
    }
}, { timestamps: true });

module.exports = mongoose.model('atividade', AtividadeSchema);
```

Fonte: autoria própria.

O código da Listagem 8 é do *end-point post* para inserir uma atividade. O *router* é uma função do Express para realizar a tratativa de rotas da aplicação. No método *post* são passados três parâmetros:

a) o primeiro é a rota que será respondida pelo *end-point*;

b) o segundo é um *array* de *middlewares*, que é o *middleware* de autenticação já mostrado anteriormente e os *middlewares* de tratativa de valores *check* importados do módulo *express-validator*;

c) o terceiro é uma função de *call-back*, que recebe um *request* e um *response*.

Inicialmente são validados possíveis erros retornados pelo *check*, posteriormente é realizada a desconstrução do objeto *request*, apanhando somente as informações que são necessárias. Após isso, um novo objeto Atividade é instanciado e é realizado o salvamento da entidade.

Listagem 8 – routes/api/atividade.js

```
router.post('/',
    [
        auth,
        check('nome', 'O nome é obrigatório!').not().isEmpty(),
        check('descricao', 'A descrição é obrigatória!').not().isEmpty(),
```

```
        check('circuito', 'O circuito é obrigatório!').not().isEmpty()
    ],
    async (req, res) => {

        const errors = validationResult(req);

        if (!errors.isEmpty()) {
            return res.status(400).json({ errors: errors.array() });
        }

        const {
            nome,
            descricao,
            circuito,
        } = req.body;

        const {
            usuario: { id: usuario}
        } = req;

        try {

            const atividade = new Atividade({
                nome,
                descricao,
                circuito,
                usuario
            });

            atividade.save();

            res.send(atividade);

        } catch (err) {
            console.error(err.message);
            res.status(500).send('Erro de servidor!');
        }
    });
```

Fonte: autoria própria.

O sistema foi desenvolvido com a linguagem JavaScript utilizando a Biblioteca de interface ReactJS, com ela foi possível desenvolver todos os módulos, itens, listas, botões separando em componentes. Esses componentes utilizam a extensão da linguagem JavaScript denominada JSX. JSX possui uma sintaxe semelhante às *tags* HTML. Essas *tags* auxiliam na criação de *components* e separação de lógica, pois ainda que utilizado JSX, ele permite a utilização de JavaScript junto com as declarações de *components*. Para facilitar o desenvolvimento da plataforma, foi utilizado um *template* Bootstrap-React, com a estrutura básica do sistema, incluindo estruturação de pastas, responsividade do sistema e configurações do Bootstrap.

A Listagem 9 é referente ao componente CadastroAtividade.js. Um componente ReactJS é uma classe JavaScript que estende o módulo *component* do React, com isso é

possível declarar JSX no método *render()* que é responsável por manipular o JSX e interpretar para JavaScript. No primeiro trecho de código é apresentado um construtor da classe que recebe por parâmetro *props*. Os *props* são parâmetros que podem ser recebidos por atributos do componente e são imutáveis no escopo do *component* atual. Com os *props* a passagem de parâmetros entre os componentes é realizada por meio de atributos definidos na chamada de um *component*. O *state* é uma propriedade de *component* do React. O *state* é o estado de um componente. Nele é possível armazenar informações referentes a *inputs* nos quais haverá manipulação do usuário. O *state* é uma propriedade de *component* do React.

Listagem 9 – CadastroAtividade.js

```
  constructor(props) {
    super(props);
    this.state = {
      atividade: {
        _id: '',
        nome: '',
        descricao: '',
        usuarioCriacao: '',
        circuito: circuitoDefault
      }
    }
  }
```

Fonte: autoria própria.

A Listagem 10 apresenta o método *render(),* também do componente de atividade. Ele é responsável por renderizar os elementos, no caso do formulário, foram utilizados componentes visuais do Material-UI. No método render é realizada a desconstrução do *state*, para que o código não fique verboso e de difícil entendimento.

Listagem 10 – CadastroAtividade.js

```
 render() {
   const { atividade: { nome, descricao, circuito } } = this.state;
    return (
      <div className="content">
        <Grid container spacing={3}>
          <Grid item xs={2}></Grid>
          <Grid item xs={8}>
            <Card>
              <CardContent>
                <form onSubmit={e => this.handleSubmit(e)}
autoComplete={'off'}>
                  <Typography variant="h3" color="textSecondary">
                    nova atividade
                  </Typography>
                  <TextField
                    InputLabelProps={{ style: { fontSize: 16 } }}
                    InputProps={{ style: { fontSize: 16 } }}
```

```
                    label="Nome"
                    margin="normal"
                    fullWidth
                    name={'nome'}
                    value={nome}
                    onChange={(e) => this.handleForm(e)}
                  />
                  <br />
                  <TextField
                    InputLabelProps={{ style: { fontSize: 16 } }}
                    InputProps={{ style: { fontSize: 16 } }}
                    label="Descrição"
                    margin="normal"
                    type="password"
                    multiline={true}
                    rows="4"
                    fullWidth
                    name={'descricao'}
                    value={descricao}
                    onChange={(e) => this.handleForm(e)}
                  />
                  <br />
                  <Grid item xs={12}>
                    <Card>
                      <CardContent>
                        <Typography variant="h4" color="textSecondary">
                          circuito lógico
                        </Typography>
                        <Typography variant="h5" color="textSecondary">
                          clique nos blocos para montar um circuito
lógico
                        </Typography>
                          <Atividade grid={circuito}
clicked={this.handleCheckBlock} />
                      </CardContent>
                    </Card>
                  </Grid>
                  <br />
                  <CardActions >
                    <Button style={{ 'marginLeft': 'auto', 'fontSize':
'12' }} variant="contained" color="default" type="reset">
                    <Icon style={{'fontSize': 17}}>close</Icon>cancelar
                  </Button>
                  <Button style={{ 'marginLeft': '10', 'fontSize': '12'
}} variant="contained" color="primary" type="submit">
                    <Icon style={{'fontSize': 17}}>check</Icon>salvar
                  </Button>
                  </CardActions>
                </form>
              </CardContent>
            </Card>
          </Grid>
        </Grid>
      </div>
    );
  }
```

Fonte: autoria própria.

Em cada *input* do formulário, há um onChange(), que chama uma função que é responsável por atualizar o *state* como mostra a Listagem 11 para o cadastro de atividade. A função recebe o evento do onChange() como parâmetro, a partir do evento é possível recuperar o *name* e o *value* do *input*. É nessa função que o *state* tem a sua alteração, com o método this.setState(). O setState() espera como parâmetro o novo valor de atributos do *state*. Neste caso, está sendo realizado um *spread* do objeto atividade para manter os dados diferentes de *target.name*.

Listagem 11 – CadastroAtividade.js: state

```
  handleForm = (event) => {
    const atividade = this.state.atividade;
    this.setState({ atividade: { ...atividade, [event.target.name]:
event.target.value } });
  }
```

Fonte: autoria própria.

O método handleSubmit, apresentado na Listagem 12, é responsável por realizar o *post/put* dos dados do *state* para API. Esse método é assíncrono, pois faz requisições para uma API que pode ou não responder rapidamente. Para realizar uma requisição para a API é utilizada a biblioteca Axios, que permite fazer requisições HTTP. Após realizar o *submit* das informações, se for uma inserção é atualizado no *state* o ID da atividade. Em caso de erro é realizado um *map* no *array* de erros respondido pelo servidor, adicionando notificações que são tratadas pelo gerenciador de estado global Redux.

Listagem 12 – CadastroAtividade.js: handleSubmit

```
handleSubmit = async (event) => {
    event.preventDefault();
    const atividade = {
      _id: this.state.atividade._id,
      nome: this.state.atividade.nome,
      descricao: this.state.atividade.descricao,
      circuito: this.state.atividade.circuito
    }
    if (atividade._id) {
      try {
        await axios.put(`/atividade/${atividade._id}`, atividade);
        this.props.addToast('Atividade alterada com sucesso!',
TOAST_SUCCESS);
      } catch (err) {
        const errors = err.response.data.errors;
        if (errors) {
          errors.map(error => this.props.addToast(error.msg,
TOAST_ERROR))
        }
      }
```

```
    } else {
      try {
        const res = await axios.post('/atividade', atividade);
        this.props.addToast('Atividade cadastrada com sucesso!',
TOAST_SUCCESS);
        this.setState({ atividade: { ...this.state.atividade, _id:
res.data._id } })
      } catch (err) {
        const errors = err.response.data.errors;
        if (errors) {
          errors.map(error => this.props.addToast(error.msg,
TOAST_ERROR))
        }
      }
    }
  }
```

Fonte: autoria própria.

O ReactJS possuí métodos que controlam o ciclo de vida de um componente. Nesse componente é utilizado o método componentDidMount, apresentado na Listagem 13 que é disparado quando o componente é montado. Se o componente Atividade.js for chamado corretamente, esse método é disparado somente uma vez. É nesse método que devem ser realizadas as requisições para a APIs e tratativas de dados iniciais buscados de outros *resources* para o *state*. Neste caso, quando o componente é montado e for uma edição, o *props* será alimentado com um valor no campo ID. Após realizar o *get* com o Axios, o objeto atividade é atribuído para o *state* por meio do this.setState(). Caso contrário, é chamado o método do Redux addToast() para notificação.

Listagem 13 – CadastroAtividade.js: async()

```
componentDidMount = async () => {
    if (this.props.location.state) {
      const { id } = this.props.location.state;
      if (id) {
        try {
          const res = await axios.get(`/atividade/${id}`);
          this.setState({ atividade: res.data });
        } catch (err) {
          this.props.addToast(err.message, TOAST_ERROR);
        }
      }
    }
  }
```

Fonte: autoria própria.

Como o Redux é uma biblioteca para ReactJS que gerencia o estado global da aplicação, com ela é possível definir um *state* que ficará disponível em toda a aplicação conectando o componente com o Redux. O código apresentado na Listagem 14 é a

configuração do Redux para manipular as notificações da aplicação. Nesse código, está a função addToast que recebe por parâmetro a mensagem, o tipo da mensagem e um *middleware dispatch*. O *dispatch* é o *middleware* responsável por requisitar a alteração de alguma informação no *state* global gerenciado pelo Redux, ao realizar o *dispach* de ADD_TOAST.

Listagem 14 – actions/toast.js

```
import uuid from 'uuid';
import { ADD_TOAST, REMOVE_TOAST } from './types';

export const addToast = (mensagem, toastTipo) => dispatch => {
    const id = uuid.v4();
    dispatch({
        type: ADD_TOAST,
        payload: { mensagem, toastTipo, id }
    });
    dispatch({
        type: REMOVE_TOAST,
        payload: id
    })
};
```

Fonte: autoria própria.

No bloco de código da Listagem 15, o *reduce* é encarregado de retornar a informação para o *state* global, conforme o parâmetro do tipo.

Listagem 15 – reducers/toast.js

```
import { ADD_TOAST, REMOVE_TOAST } from '../actions/types';

const initialState = [];

export default function (state = initialState, action) {

    const { type, payload } = action;

    switch (type) {
        case ADD_TOAST:
            return [...state, payload];
        case REMOVE_TOAST:
            return state.filter(msg => msg.id !== payload)
        default:
            return state;
    }
}
```

Fonte: autoria própria.

5 CONSIDERAÇÕES FINAIS

As considerações finais estão organizadas em contexto do trabalho realizado, com a avaliação do alcance dos objetivos; as tecnologias utilizadas e trabalhos futuros.

5.1 CONTEXTO

O objetivo de desenvolver uma plataforma para composição de atividades que promovem o desenvolvimento de raciocínio lógico foi alcançado, ainda que o escopo tenha sido reduzido do que havia sido inicialmente planejado, que era a possibilidade de o usuário arrastar objetos para a composição da atividade.

O sistema desenvolvido, denominado de plataforma *web*, possibilita que tutores possam criar atividades para o desenvolvimento de raciocínio lógico e conceitos básicos de programação, especialmente a ideia de instruções e de repetição dessas instruções. O público alvo inicialmente pensado é de crianças, mas a plataforma de aplica para a realização de atividades para desenvolvimento de raciocínio lógico e aprendizado da ideia básica de programação para qualquer idade.

A ideia inicial de implementar a possibilidade de arrastar e soltar objetos no *grid* não foi possível desenvolver pelas dificuldades encontradas com as tecnologias escolhidas. Isso não significa que não seja possível implementar, mas no tempo utilizado para o desenvolvimento do trabalho, não foi encontrada solução para os problemas apresentados pelas tecnologias utilizadas para que essa funcionalidade pudesse ter sido desenvolvida. Outras tecnologias poderiam ser utilizadas em conjunto, resolvendo, assim, os problemas encontrados.

A ideia de o próprio usuário cadastrar-se, sem necessidade de validação de usuário, e de escolher o tipo de usuário, é decorrente de que o aplicativo não envolve questões de segurança, como dados que devam ser protegidos. Apenas o usuário que criou uma atividade poderá acessá-la em modo edição. A plataforma pode ser utilizada para que as próprias crianças (ou outro que no momento tem o principal papel de aluno, ou seja, realizador das atividades) componham atividades. Para isso ele fará um cadastro de acesso ao sistema como tutor. O objetivo da plataforma é estimular e incentivar a realização de atividades que promovam o desenvolvimento de raciocínio lógico, abstração e de conceitos (ideias) fundamentais de programação, como é o caso de repetição de comandos.

5.2 TECNOLOGIAS

Em termos de uso de tecnologias, o desenvolvimento da aplicação foi bem desafiador, iniciando pela escolha das tecnologias a serem utilizadas. Essas tecnologias não foram vistas no ambiente acadêmico, somente uma breve visão de como funcionavam e como se comportavam. Para realizar o desenvolvimento foi preciso aprender a manipular informações de um banco de dados orientado a documentos no caso do MongoDB, realizar o desenvolvimento de uma aplicação *front-end* somente com JavaScript utilizando o ReactJS e conectar essas duas tecnologias com JavaScript utilizando o Node.js.

Pelo desenvolvimento realizado e para o contexto da aplicação, o ReactJS foi a escolha mais acertada para o *front-end*, a manipulação de componentes interface do usuário é facilitado com a sua forma de trabalhar baseada em componentes que se comunicam por meio de parâmetros. O desempenho é outro ponto forte, conseguindo renderizar utilizando o Virtual *Document Object Model* (DOM) para que somente o necessário seja alterado fazendo com que alterações na tela sejam extremamente performáticas e quase que irrelevantes em questão de impacto negativo em desempenho.

O Node.js talvez não tenha sido a melhor escolha para o servidor, ele não chega a ser um servidor robusto, podendo ocasionar lentidão ao carregar os documentos das atividades, que por sua vez não são pequenos. Uma alternativa a essa tecnologia seria, talvez, utilizar ASP .NET CORE, que em sua versão estável 2.2 possui desempenho considerável e é multiplataforma, não executando somente no Windows como as versões do ASP .NET fazem.

O banco de dados escolhido foi o MongoDB, a escolha deu-se por ser um banco que é mais performático, mas não foi bem pensada na hora da recuperação dos dados. Houve muita dificuldade de manipular os dados para apresentação. Ao invés do MongoDB uma alternativa seria a utilização do PostgreSQL.

Com o desenvolvimento desta aplicação, foi possível entender o porquê essas tecnologias estão sendo tão utilizadas pelo mercado de trabalho, seja para aplicações menores, sites ou *Enterprise Resource Planning* (ERP). Esse projeto motivou o aprendizado de novas tecnologias e técnicas diferentes de desenvolvimento. E pode-se dizer que esse foi um aspecto muito relevante, juntamente, é claro, com o fato e disponibilizar um software, denominado plataforma, que pode ser utilizado para promover o desenvolvimento de raciocínio lógico e conceitos fundamentais de programação para crianças.

5.3 TRABALHOS FUTUROS

Entre os principais complementos indicados como trabalhos futuros destacam-se:

a) A possibilidade de arrastar objetos para o *grid* que compõe o cenário. Possibilitando, assim, indicar o ponto de origem e o ponto de destino para o caminho a ser percorrido e, inclusive, com paradas intermediárias. Essas paradas indicam locais pelos quais o caminho a ser criado deveria obrigatoriamente passar. Exemplo: indicar os pontos de coleta de pólen de uma abelha. Conceitos matemáticos poderiam ser acrescentados com quantidades associadas a cada ponto de coleta e no final do caminho o usuário indicar a soma dos números existentes em cada um dos pontos de coleta.

b) A possibilidade de o caminho poder ser representado por uma estória. O aluno criaria o caminho a partir de uma estória que foi desenvolvida pelo tutor, por exemplo: o gato deu 5 passos para a direita, fez um giro de 90 graus e mais 2 passos. Nessa descrição seriam indicadas as direções dos comandos que podem ser utilizados e a ideia de ângulos. As variáveis da estória poderiam ser randômicas (aleatórias) a cada vez que a atividade fosse executada. A estória permanece, mas os procedimentos são distintos para realizá-la.

c) A existência de um tutor inteligente que pudesse orientar, no sentido de otimizar o conjunto de instruções a partir da solução proposta pelo aluno. A solução proposta pelo aluno é comparada com uma solução padrão. Exemplo: o aluno fez 5 vezes o comando mover para a direita. O tutor sugeriria que ele utilizasse o comando repetir e o comando mover para a direita apenas uma vez.

d) Considerando que a aplicação desenvolvida poderia ser utilizada (haveria necessidade de um estudo por especialista para afirmar-se que pode) para auxílio no desenvolvimento de habilidades básicas em pessoas com dificuldade de aprendizagem. Nesse caso, um tutor inteligente que indicasse a evolução do usuário nas repetidas tentativas de resolver o problema poderia ser utilizado como auxiliar na indicação de progresso, necessidade de reforço de conceitos, dificuldades acentuadas e outros.

REFERÊNCIAS

ALMEIDA, Amanda de. **O que a maioria das escolas não ensina, mas ainda assim podemos e devemos aprender (Parte 1)**. 2013. Disponível em: <http://www.b9.com.br/37238/opiniao/o-que-a-maioria-das-escolas-nao-ensina-mas-ainda-assim-podemos-e-devemos-aprender-parte-1/>. Acesso em: 22 mar. 2018.

ASTAH. **Astah community**. 2018. Disponível em: <http://astah.net/editions/community>. Acesso em: 12 abr. 2018.

BALSAMIQ MOCKUPS. **Balsamiq products**. 2018. Disponível em: <https://balsamiq.com/products/>. Acesso em: 28 jun. 2018.

BERNARDES, Marlon. **How to use Axios as your HTTP client**. 2015. Disponível em: <https://bitbucket.org/>. Acesso em: 2 jun. 2019.

BEZERRA, Fabio; DIAS, Klissiomara. **Programação de computadores no ensino fundamental: experiências com Logo e Scratch em escola pública**. In XXII Workshop sobre Educação em Informática, Brasília, DF: SBC. 2014, p. 1515-1524.

BITBUCKET. **Bitbucket.** 2018. Disponível em: <https://bitbucket.org/>. Acesso em: 12 abr. 2018.

BR.CODE.ORG. **br.code.org**. 2014. Disponível em: <https://br.code.org/>. Acesso em: 24 mar. 2018.

COMPUTERWORLD. **Faltarão 449 mil profissionais de TI na América Latina até 2019**. 2016. Disponível em: <http://computerworld.com.br/falta-de-profissionais-de-ti-na-america-latina-chegara-32-ate-2019>. Acesso em: 26 mar. 2018.

EXPRESS. **Express**. 2018. Disponível em: <http://expressjs.com/pt-br/>. Acesso em: 11 abr. 2018.

GARLET, Daniela, BIGOLIN, Nara Martini, SILVEIRA, Sidnei Renato. **Uma proposta para o ensino de programação de computadores na educação básica.** Dispnível em: <http://w3.ufsm.br/frederico/images/DanielaGarlet.pdf>. Acesso em: 22 mar. 2018.

KONVA.JS. **Konva.js: HTML5 2d canvas library for desktop and mobile applications**. 2019. Dispnível em: <https://konvajs.org/>. Acesso em: 16 jun. 2019.

MARINHEIRO, Fabiana; SILVA, Ivanovitch; MADEIRA, Charles; CORDEIRO, Sandro; SOUZA, Danielle; COSTA, Patrícia; FERNANDES, Gildene. Ensinando crianças do ensino fundamental a programar computadores com o auxílio de jogos digitais. **Revista Tecnologias na Educação**, ano 8, n./v.16. 2016. Edição Temática. Congresso Regional sobre Tecnologias na Educação (CTRL+E). Disponível em: <tecnologiasnaeducacao.pro.br>. Acesso em: 22 mar. 2018.

MATERIAL-IU. 2019. Dispnível em: <https://material-ui.com/>. Acesso em: 16 jun. 2019.

MICROSOFT. **Visual Studio Code**. 2019. Disponível em: <https://code.visualstudio.com/>. Acesso em: 10 jun. 2019.

MONGODB. **The database for modern applications**. Disponível em: <https://www.mongodb.com/>. Acesso em: 10 jun. 2019.

NASCIMENTO, Cledison da S. **Introdução ao Ensino de Lógica de Programação para Crianças do Ensino Fundamental com a ferramenta Scratch**. Monografia de Graduação apresentada ao Núcleo de Educação à Distância da Universidade Federal de Roraima, 2015.

NODE.JS. **Node.js**. 2018. Disponível em: <https://nodejs.org/en/>. Acesso em: 11 abr. 2018.

NODEBR. **Nodejs e MongoDB – introdução ao Mongoose**. 2016. Disponível em: <http://nodebr.com/nodejs-e-mongodb-introducao-ao-mongoose//>. Acesso em: 16 jun. 2019.

OCDE. **Results in Focus**: what 15-year-olds know and what they can do with what they know. PISA 2012. Disponível em: <http://www.oecd.org/pisa/keyfindings/pisa-2012-resultsoverview.pdf>. Acesso em: 5 abr. 2018.

OLIVEIRA, Thiago Henrique Braz de, SILVA, Cherlia, ANDRADE, Mariel José Pimentel de. **Scratch**: utilizando programação para desenvolvimento do raciocínio lógico em crianças. XIII JORNADA DE ENSINO, PESQUISA E EXTENSÃO – JEPEX 2013 – UFRPE: Recife, 2013, p. 1-3.

PAPERT, Seymour; Resnick, Mitchel. **Technological fluency and the representation of knowledge**. Proposal to the National Science Foundation. MIT MediaLab 1995.

PEREIRA JUNIOR, José Carlos R., RAPKIEWICZ, Clevi E. **O processo de ensino-aprendizagem de fundamentos de programação: uma visão crítica da pesquisa no Brasil**, WEI RJES, 2004, p. 1-7.

PRADO, Maria Elisabette B. B. **Logo**: linguagem de programação e as implicações pedagógicas. Nied-Unicamp. Disponível em: <http://www.nied.unicamp.br/oea/mat/LOGO_IMPLICACOES_bette_nied.pdf>. Acesso em: 09 abr. 2018.

REACT. **React**. 2018. Disponível em: <https://reactjs.org/>. Acesso em: 11 abr. 2018.

REDUX. **A predictable state container for JavaScript apps**. 2019. Disponível em: <https://redux.js.org/>. Acesso em: 11 abr. 2018.

ROBO 3T. **Robo 3T**. 2019. Disponível em: < https://robomongo.org/>. Acesso em: 6 jun. 2019.

STELLA, A. L. **Utilizando o pensamento computacional e a computação criativa no ensino da linguagem de programação Scratch para alunos do ensino fundamenta**l. Dissertação, 2016 (Mestrado), Faculdade de Tecnologia – Universidade Estadual de Campinas. Limeira, 2016.

TAJRA, Samya Feitosa. **Informática na educação**: novas ferramentas pedagógicas para o professor na atualidade. Editora Érica, 2012.

VALENTIM, Henryethe. **Um estudo sobre o ensino-aprendizagem de lógica de programação**. Encontro Nacional de Pesquisa em Educação e Ciências. 2000, p. 1-22. Disponível em: <http://posgrad.fae.ufmg.br/posgrad/viienpec/pdfs/137.pdf>. Acesso em: 22 mar. 2018.

WING, Jeannette. M. Computational thinking. **Communications of the ACM**, v. 49, n 3, mar. 2006. Disponível em: <https://cacm.acm.org/magazines/2006/3/5977-computational-thinking/abstract>. Acesso em: 30 ago. 2015.

ZANETTI, Humberto A. P, BORGES, Marcos A. F., LEAL, Valéria C. G., MATSUZAKI, Igor Y. Proposta de ensino de programação para crianças com Scratch e pensamento computacional. **Tecnologias Sociedade e Conhecimento,** Campinas, v. 4, Dez. 2017, p. 34-58. Disponível em: <www.nied.unicamp.br/ojs/>. Acesso em: 22 mar. 2018.