UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO ALVES VIANA

SIMUCAR VIRTUAL - LABORATÓRIO VIRTUAL DE UM CARRINHO PARA O ENSINO DE PROGRAMAÇÃO

VITÓRIA DA CONQUISTA - BA

2025

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THIAGO ALVES VIANA

SIMUCAR VIRTUAL - LABORATÓRIO VIRTUAL DE UM CARRINHO PARA O ENSINO DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Sudoeste da Bahia-UESB, para a disciplina de Trabalho Supervisionado II, como requisito para aprovação na disciplina, ministrada pela Prof. Dr. Maísa Soares dos Santos Lopes, do curso de Ciências da Computação.

Prof.ª Orientadora: Maísa Soares dos Santos Lopes

VITÓRIA DA CONQUISTA – BA
2025

RESUMO

O objetivo deste trabalho é apresentar o desenvolvimento do Simucar Virtual, um laboratório virtual projetado para auxiliar estudantes no aprendizado de programação por meio da simulação de um robô móvel. A plataforma tem como principal finalidade proporcionar um ambiente interativo e acessível que permita aos alunos praticar conceitos fundamentais de lógica de programação e controle de robôs sem a necessidade de equipamentos físicos.O desenvolvimento do Simucar Virtual envolveu a criação de uma interface intuitiva e minimalista, a implementação de um editor de código baseado no CodeMirror e a elaboração de uma biblioteca personalizada para controle do carrinho. Além disso, foram incorporadas funcionalidades como detecção de cores, inserção de obstáculos e criação de circuitos personalizáveis, permitindo que os usuários experimentem diferentes abordagens na resolução de problemas computacionais. A ferramenta foi projetada para ser acessada via internet, sem a necessidade de instalação de software, garantindo acessibilidade a um público amplo, tendo em vista que a utilização de simulação computacional permite reduzir custos com equipamentos físicos e proporciona um ambiente seguro para experimentação e aprendizado. Como resultado, foi desenvolvido um laboratório virtual, de acordo com os requisitos, voltado para o ensino de programação, utilizando a simulação de um robô móvel como elemento central da aprendizagem, que pode ser uma ferramenta eficaz para complementar o ensino de programação, incentivando a prática e o raciocínio lógico. Dessa forma, o Simucar se apresenta como uma solução inovadora para o ensino de programação e robótica.

Palavras-chave: Laboratório virtual; Programação; Robô móvel; Ensino prático; Ambiente virtual.

ABSTRACT

This work presents the development of Simucar Virtual, a virtual laboratory designed to assist students in learning programming through the simulation of a mobile robot. The platform's main objective is to provide an interactive and accessible environment that allows students to practice fundamental concepts of programming logic and robot control without the need for physical equipment. The development of Simucar Virtual involved creating an intuitive and minimalist interface, implementing a CodeMirror-based code editor, and developing a custom library for cart control. Additionally, features such as color detection, obstacle insertion, and customizable circuits were incorporated, allowing users to experiment different approaches to solving computational problems. The tool was designed to be accessed via internet without requiring software download, which ensures accessibility for a broad audience. The use of computational simulation reduces costs associated with physical equipment and provides a safe environment for experimentation and learning. As a result, a virtual laboratory was developed according to the specified requirements, focusing on programming education by using mobile robot simulation as the central learning element. This tool can effectively complement programming education, encouraging hands-on practice and logical reasoning. Thus, Simucar presents itself as an innovative solution for programming and robotics education.

Keywords: Virtual laboratory; Programming; Mobile robot; Practical learning; Virtual environment.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, Adeir e Isabel, que não mediram esforços para que eu concluísse a minha graduação, sempre me dando todo suporte. Ao meu irmão, Gabriel, e aos meus amigos, tanto de Almenara, quanto de Vitória da Conquista, por me proporcionarem momentos de descontração e apoio. A minha namorada, Ana Claudia, pelo carinho e incentivo dado para a conclusão deste trabalho. Por fim, aos meus professores e a minha orientadora, Maísa Santos, pela paciência e pelos ensinamentos transmitidos.

LISTA DE FIGURAS

Figura 1 - Ilustração das etapas do DSR
Figura 2 - Simulação de um circuito no Open Roberta
Figura 3 - Prototipação no Tinkercard
Figura 4 - Tela principal do Lara Domótica
Figura 5 - Tela principal do projeto Narrativas Interativas
Figura 6 - Tela principal do projeto Narrativas Interativas
Figura 7 – Tela inicial de Login
Figura 8 - Tela de criação de conta
Figura 9 – Tela inicial de visualização do laboratório
Figura 10 – Tela de Planejamento
Figura 11 – Ambiente de programação
Figura 12 – Console de erros da aplicação
Figura 13 – Carrinho criado no Simucar Virtual
Figura 14 – Carrinho ao término do comando cart.frente (100)
Figura 15 – Carrinho após andar para trás
Figura 16 – Carrinho rotacionado em 45 graus
Figura 17 – Carrinho após andar para frente e com angulo de 45 graus
Figura 18 – linha inserida na horizontal
Figura 19 – Segunda linha, com uma espessura maior, no circuito.
Figura 20 – Carrinho com o detector na cor branca, localizado na parte frontal
Figura 21 – Detector na cor azul, pois o carrinho está sobre a linha de mesma cor
Figura 22 – Cone inserido no circuito
Figura 23 - Vários cones inseridos no circuito
Figura 24 – Inserido linha para laço de repetição
Figura 25 – Carrinho detectando uma linha, com seu detector na mesma cor preta
Figura 26 – Carrinho ao término do laço de repetição

SUMÁRIO

1 INTRODUÇÃO....8
1.1. Justificativa....11
1.2 Objetivos....12
1.2.1 Objetivos gerais....12
1.2.2 Objetivos específicos....12
1.3 Metodologia....12
2 REVISÃO BIBLIOGRÁFICA....15
2.1 Laboratório virtual....15
2.2 Robótica....16
2.3 Trabalhos relacionados....17
2.3.1 Open Roberta Lab....17
2.3.3 Trabalhos de conclusão de curso....21
2.3.3.1 Lara Domótica - Protótipo de Laboratório virtual de Domótica....21
2.3.3.2 Desenvolvimento de um laboratório virtual para criação de histórias interativas....22
3.3.3 Lara Core Server – API: Implementação do LARA (Laboratório em Redes de aprendizagem) como um serviço....24
3 SIMUCAR VIRTUAL....24
3.1 Proposta....24
3.2 Requisitos....25
3.2.1 Requisitos Funcionais....25
3.2.2 Requisitos não funcionais....25
3.3 Ferramentas e tecnologias utilizadas....26
3.3.1 Linguagens....26
3.3.2 Frameworks....27
3.3.3 Vue.JS....27
3.3.4 CodeMirror....28
3.3.5 Supabase....28
3.4 Arquitetura....29
3.5 Laboratório....31
3.5.1 Descrição dos componentes da interface....31
3.5.2 Iniciando a programação....37
3.5.3 Rotação....39
3.5.4 Inserir linha....41
3.5.5 Detector de cores....43
3.5.6 Inserir Obstáculos....44
3.5.7 Laço de repetição....46
3.6 Considerações Finais....49

4. CONCLUSÃO..... 49
REFERÊNCIAS..... 51
APÊNDICE B – EXEMPLO PARA DEMONSTRAÇÃO COMPLETA..... 55
APÊNDICE C – LINK PARA REPOSITÓRIO NO GITHUB.....57

1 INTRODUÇÃO

O avanço tecnológico tem transformado a forma como a educação é concebida, trazendo consigo novas abordagens e metodologias inovadoras para o processo de aprendizagem de programação. Nesse contexto, os laboratórios virtuais despontam como ferramentas importantes, proporcionando ambientes de experimentação e prática em um mundo cada vez mais digital. Um dos grandes problemas das disciplinas de ensino de programação, tanto em escolas técnicas, quanto em universidades, seja qual for o curso, é a necessidade de realização de muitas atividades práticas para a internalização dos conceitos. Sendo assim, é preciso encontrar maneiras que facilitem o entendimento e a compreensão do estudante, que, muitas vezes, não está familiarizado com esses novos conceitos que envolvem o ambiente de desenvolvimento de *software*.

Segundo Santos e Costa (2006), a falta de compreensão do raciocínio lógico pode ser a principal razão para o alto índice de reprovação nas disciplinas de Algoritmos e Programação e, em alguns casos, para a desistência de um curso. Sendo assim, quando alunos se deparam com algo totalmente novo, podem surgir dificuldades importantes, pois, como apontam Gomes, Henriques e Mendes (2008), eles estão frequentemente habituados a disciplinas nas quais é possível ser bem-sucedido através de abordagens de estudo baseadas em leituras sucessivas, memorização de fórmulas e certa mecanização de procedimentos. Desse modo, é necessário que os estudantes mudem as estratégias de estudo, para que possam alcançar bons resultados nas disciplinas de programação, buscando aquilo que melhor se adapta às suas características e utilizando ferramentas que agregam o seu entendimento sobre o assunto. Com isso, o entendimento e a abstração dos conhecimentos, ainda com novas metodologias, acaba ficando mais suave e mais fácil de se compreender.

Outro ponto crítico é a ausência de uma aplicação prática imediata dos conceitos teóricos, o que pode tornar o aprendizado abstrato e desconectado da realidade percebida pelos estudantes. Isso não é um problema exclusivo da programação. Segundo Batista, Fusinato e Blini (2009, p. 44),

> o modelo de ensino tradicional é ainda amplamente utilizado por muitos educadores nas nossas escolas de Ensino Fundamental e Médio. Tal modelo

> de educação trata o conhecimento como um conjunto de informações que são simplesmente transmitidas pelos professores para os alunos, não resultando em um aprendizado efetivo. Os alunos têm o papel de ouvintes e, na maioria das vezes, os conhecimentos transmitidos pelos professores não são realmente absorvidos por eles, são apenas memorizados por um curto período de tempo e, em geral, esquecidos posteriormente, comprovando a não-ocorrência de um aprendizado significativo.

No ensino de disciplinas de ciências da natureza, por exemplo, em que também são abordados muitos conceitos teóricos, a dificuldade dos alunos em compreender esses conceitos é bastante conhecida pelo sistema educacional. Por isso, muitas escolas e professores recorrem ao ensino prático, ou seja, o conceito estudado é, primeiramente, apresentado em sua forma totalmente teórica em sala de aula, com explicações orais, e, posteriormente, abordado de forma prática, geralmente por meio de um experimento em que se simula e se demonstra aquele conceito teórico abordado em sala de aula. Segundo Moraes (1998), as aulas práticas desempenham um papel complementar às aulas teóricas, atuando como um elemento essencial no processo de construção de novos conhecimentos. A experiência direta vivenciada durante essas atividades contribui significativamente para a assimilação dos conteúdos relacionados. Com isso, conclui-se que aquele conceito demonstrando com um experimento é muito mais fixado e compreendido pelos estudantes, pois a abordagem prática adotada nas aulas.

Outro ponto que podemos abordar é a falta de *feedback* e a pressão por resultados rápidos. Um acompanhamento de aprendizagem que não é feito de forma contínua e personalizada, com rapidez e falta de atenção do professor, pode impedir que os alunos identifiquem e corrijam suas dificuldades em tempo hábil e ocasionar problemas comuns como ansiedade e desmotivação. Com isso, um aluno pode achar que está entendendo o assunto e que está tudo certo com seu aprendizado ou ainda estar ciente de que não aprendeu o assunto e não conseguir mais acompanhar o andamento da disciplina. Sendo assim, em um momento de avaliação, fica explícito que o seu conhecimento não foi o suficiente. Entretanto, quando o aluno entende o assunto, coloca em prática aquilo que aprendeu e recebe um *feedback*, identificar suas dificuldades emais fácil, uma vez que houve tempo para analisar e estudar melhor as suas dificuldades.

Voltando para o campo tecnológico, a integração de novas tecnologias e ferramentas de ensino, embora potencialmente benéfica, muitas vezes é feita de

maneira inadequada, sem a devida preparação e capacitação dos professores, o que pode resultar em uma experiência de aprendizado fragmentada e ineficaz. Além disso, o uso inadequado da tecnologia pode levar a um sentimento de isolamento, tanto para alunos quanto para educadores, tornando o processo de ensino menos envolvente e colaborativo. Para evitar esse distanciamento, é essencial que as ferramentas adotadas incentivem a interação e a cooperação, promovendo um ambiente de aprendizado mais fluido e inclusivo. Sistemas colaborativos podem transformar a tecnologia em um elo de conexão, garantindo que seu uso realmente enriqueça a experiência educacional, ao invés de criar barreiras. Por isso, é necessário que a colaboração com essa tecnologia seja de fácil aprendizagem, tanto por parte dos professores como dos alunos, a fim de proporcionar uma maior fluidez e um melhor aproveitamento do ensino. De nada adianta agregar a tecnologia ao processo de ensino-aprendizagem se essa agregação acontecer de uma forma que pode ser lenta e difícil de lidar, pois o processo será desgastante e não atingirá o seu objetivo.

Por fim, a falta de um ambiente colaborativo e de apoio dentro da sala de aula pode isolar os alunos, fazendo com que percam o interesse e a confiança em suas habilidades de programação. Todos esses fatores combinados representam desafios significativos que precisam ser abordados para melhorar a eficácia do ensino de programação e garantir que os alunos possam desenvolver plenamente suas habilidades e seus interesses na área.

Diante do exposto, surge a seguinte questão: como desenvolver um laboratório virtual que auxilie no ensino de programação? Visando responder essa pergunta, a proposta central deste trabalho é criar um espaço de aprendizado online, no qual os estudantes de ciência da computação poderão desenvolver suas habilidades de programação de maneira prática e dinâmica. A singularidade desse laboratório virtual reside na interação do aluno com um robô móvel, cujos comandos são efetuados diretamente na interface online. Esse conceito busca proporcionar uma experiência de aprendizado imersiva, na qual os alunos podem aplicar seus conhecimentos de programação para controlar virtualmente o movimento do robô na tela.

A escolha por um robô móvel como objeto de estudo no laboratório virtual visa oferecer aos estudantes um ambiente prático e estimulante, alinhado com os

objetivos do LARA (Laboratório de Acadêmico em Rede de Aprendizagem). O LARA é um projeto que tem como objetivo projetar e implementar uma arquitetura pedagógica que integra recursos tecnológicos e metodologia de ensino para melhorar o processo de aprendizagem de disciplinas do curso de ciência da computação. O LARA envolve aspectos relacionados a laboratórios remotos, engenharia de *software*, robótica e metodologia pedagógica (Lopes, 2017).

Ao criar um espaço online onde os comandos programáticos se traduzem em movimentos perceptíveis do robô, busca-se proporcionar uma abordagem única para o ensino de programação, aproximando os conceitos teóricos da prática aplicada.

Ao longo desta monografia, serão abordados os elementos fundamentais para o desenvolvimento e implementação desse laboratório virtual, explorando suas potencialidades, desafios e relevância para a formação profissional dos alunos de ciência da computação. Esta monografia está dividida em 4 seções: introdução, onde são apresentados a justificativa, os objetivos do trabalho e a metodologia utilizada, revisão bibliográfica, na qual é feita uma revisão sobre laboratório virtual e robótica e apresentado os trabalhos relacionados do LARA com este projeto, Sinuca Virtual, mostrando todo o trabalho feito e as tecnologias utilizadas e por fim a conclusão, apresentando os trabalhos futuros.

1.1. Justificativa

A robótica é uma área que desperta a curiosidade e o desenvolvimento de muitos estudantes, e o conhecimento em programação é fundamental para o sucesso em várias áreas em que está envolvido o ensino da ciência da computação. Buscando agregar as duas áreas de conhecimento para que seja mais fácil o processo de ensino-aprendizado do estudante, nasce esse trabalho.

Nesse contexto, o desenvolvimento de um laboratório virtual de um robô móvel para o ensino da programação representa uma iniciativa fundamental, pois permite que os alunos acessem um ambiente simulado de qualquer lugar do mundo, a qualquer momento, utilizando apenas a internet. Essa abordagem elimina barreiras geográficas e financeiras, possibilitando a realização de experimentos e práticas em programação de robôs móveis sem a necessidade de equipamentos

físicos, que podem ser caros e de difícil acesso. Com essa flexibilidade, estudantes e pesquisadores podem explorar conceitos de robótica e programação de forma contínua e acessível, potencializando o aprendizado por meio de uma plataforma interativa e disponível 24/7. Além disso, o laboratório virtual pode ser utilizado por alunos de diferentes níveis de conhecimento em programação, desde iniciantes até alunos mais avançados. Com isso, é possível proporcionar aos estudantes uma formação mais ampla e adequada às suas necessidades.

Assim, o trabalho de conclusão de curso sobre o desenvolvimento de um laboratório virtual de um robô móvel para o ensino da programação busca contribuir para a melhoria da qualidade do ensino de programação, oferecendo aos alunos um ambiente de aprendizado mais acessível e prático, que possa ser utilizado em diferentes instituições de ensino, como escolas e universidades, além de promover um melhor entendimento dos conceitos de programação.

1.2 Objetivos

1.2.1 Objetivos gerais

O principal objetivo deste trabalho é desenvolver um laboratório virtual que auxilie no aprendizado de programação de computadores de maneira prática através do movimento de um robô móvel.

1.2.2 Objetivos específicos

- Analisar os requisitos fundamentais de laboratório virtual, suas tecnologias e ferramentas de desenvolvimento;
- Desenvolver o laboratório de acordo com os requisitos funcionais e não funcionais;
- Testar para garantir um bom funcionamento do sistema.

1.3 Metodologia

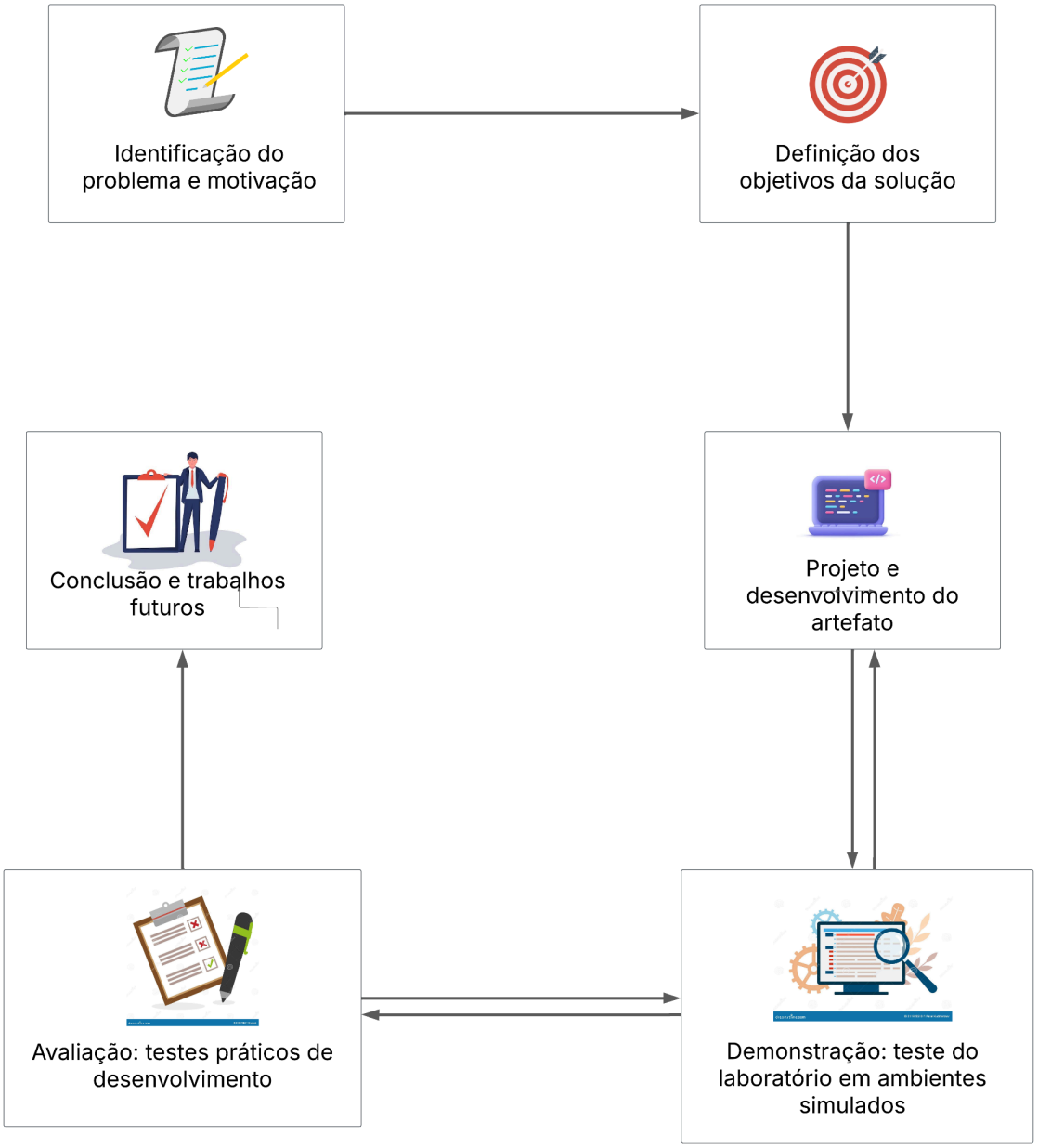
A metodologia utilizada nesse trabalho foi a DSR (*Design Science Research*). Essa metodologia é bastante utilizada em monografias relacionadas à tecnologia, uma vez que esse tipo de trabalho possui características e desenvolvimento diferentes dos demais.

A aplicação da DSR neste projeto seguiu um processo iterativo dividido nas seguintes etapas:

1. Identificação do Problema e Motivação: a necessidade de um ambiente de ensino mais prático e acessível para estudantes de programação foi o ponto de partida deste trabalho. A dificuldade em compreender conceitos abstratos e a falta de ferramentas interativas motivaram a criação de um laboratório virtual baseado em simulação de um robô móvel.
2. Definição dos Objetivos da Solução: com base na identificação do problema, estabeleceu-se que o laboratório virtual deveria proporcionar uma experiência interativa e acessível, permitindo que alunos programassem e visualizassem o comportamento do robô em tempo real. Além disso, o sistema deveria ser intuitivo, colaborativo e adequado a diferentes níveis de conhecimento em programação.
3. Projeto e Desenvolvimento do Artefato: a construção do Simucar Virtual seguiu um processo iterativo, no qual a interface, os componentes e as funcionalidades foram planejadas com base em requisitos funcionais e não funcionais. O ambiente foi implementado utilizando Vue.js, garantindo modularidade e escalabilidade, e a simulação do robô móvel foi desenvolvida com bibliotecas como CodeMirror para a edição de código e WBO para colaboração e planejamento de algoritmos.
4. Demonstração: o protótipo do laboratório foi testado em cenários simulados, permitindo que os usuários interagissem com o ambiente, executassem comandos e observassem os resultados de sua programação na movimentação do robô.
5. Avaliação: a avaliação do sistema foi realizada por meio de testes práticos, verificando a usabilidade, a eficiência e a experiência dos usuários. Os *feedbacks* obtidos foram analisados para identificar melhorias e ajustes necessários.

6. Conclusões e Comunicação dos Resultados: com base nos testes e na validação do artefato, foram destacadas as contribuições do Simucar Virtual para o ensino de programação, reforçando a importância de abordagens práticas e interativas no aprendizado de conceitos computacionais.

Figura 1 - Ilustração das etapas do DSR



Fonte: autoria própria, feito no Lucidchart (2025)

A aplicação da DSR garantiu que o desenvolvimento do laboratório virtual fosse fundamentado em um processo científico, permitindo a criação de uma solução inovadora e eficaz para o ensino da programação.

2 REVISÃO BIBLIOGRÁFICA

Nesta seção, será abordada a fundamentação teórica que embasa o desenvolvimento do laboratório virtual proposto neste trabalho. A revisão bibliográfica é essencial para compreender os conceitos e tecnologias envolvidas, bem como para contextualizar a proposta dentro do cenário atual de ensino de programação e robótica. Serão discutidos os principais temas relacionados ao projeto, como laboratórios virtuais e robótica educacional, bem como alguns trabalhos relacionados.

2.1 Laboratório virtual

Os laboratórios virtuais são ambientes de simulação desenvolvidos para facilitar o ensino e a aprendizagem em diversas áreas do conhecimento, especialmente nas disciplinas de Ciência, Tecnologia e Engenharia. Esses ambientes permitem que os estudantes realizem experimentos e adquiram habilidades práticas sem a necessidade de um laboratório físico, sendo uma solução essencial para cursos à distância e para o ensino híbrido (Potkonjak *et al.*, 2010).

Eles surgiram como uma alternativa aos laboratórios físicos, especialmente em contextos em que o acesso a equipamentos e materiais é limitado. Ao permitir que os estudantes pratiquem conceitos teóricos em um ambiente seguro e controlado, os laboratórios virtuais eliminam riscos associados a experimentos reais e garantem maior acessibilidade. Além disso, como podem ser acessados de qualquer lugar com conexão à internet, promovem o aprendizado remoto e flexível. Segundo Potkonjak *et al.* (2013), um dos principais benefícios desses laboratórios é a segurança, pois possibilitam que os alunos cometam erros sem risco de danificar equipamentos ou colocar-se em perigo.

No ensino de programação, os laboratórios virtuais desempenham um papel fundamental ao oferecerem aos alunos uma experiência prática de codificação, depuração e teste de algoritmos. Eles disponibilizam ferramentas como editores de código, compiladores, depuradores e simuladores, que permitem que os estudantes desenvolvam habilidades técnicas de forma interativa e dinâmica.

O desenvolvimento contínuo desses laboratórios busca aprimorar a interação dos estudantes com os experimentos, tornando-os cada vez mais semelhantes à experiência prática encontrada em laboratórios físicos. Como destacado por Huang, Rauch e Liaw (2010), o uso de plataformas de aprendizado colaborativo e sistemas de realidade virtual é uma tendência promissora para aumentar a eficácia dos laboratórios virtuais no ensino de disciplinas técnicas e científicas.

2.2 Robótica

A robótica é uma disciplina multifacetada que envolve o *design*, a construção, a programação e a operação de robôs. Esses dispositivos autônomos ou semi autônomos têm a capacidade de realizar tarefas específicas, muitas vezes de maneira mais eficiente, precisa e segura do que os humanos. Com essa contextualização, é possível agregar e simular vários ambientes de maneira que se desperte o interesse do aluno em interagir e aprender com essa tecnologia.

A robótica educacional é uma área que combina conceitos de robótica e programação para fins educativos, visando estimular o aprendizado de forma prática e lúdica. Segundo Papert (1985), a robótica pode ser uma ferramenta poderosa para o ensino de programação, pois permite que os alunos vejam os resultados de seus códigos de forma tangível, através do movimento e do comportamento de robôs.

A robótica educacional tem sido amplamente utilizada em escolas e universidades para ensinar conceitos de lógica de programação, automação e inteligência artificial. De acordo com Resnick *et al.* (2009), a interação com robôs pode aumentar o engajamento dos alunos e facilitar a compreensão de conceitos abstratos, como *loops*, condicionais e funções.

No contexto deste trabalho, a robótica educacional é aplicada através da simulação de um robô móvel em um ambiente virtual. O objetivo é proporcionar aos

alunos uma experiência prática de programação, na qual eles podem controlar o movimento do robô e observar os resultados de seus códigos em tempo real.

2.3 Trabalhos relacionados

2.3.1 Open Roberta Lab

O *Open Roberta Lab* é uma plataforma de programação online gratuita e de código aberto desenvolvida pelo Fraunhofer IAIS (Instituto Fraunhofer de Análise Inteligente e Sistemas de Informação) como parte da iniciativa “Roberta – Learning with Robots”. Essa plataforma, lançada em 2014, tem como objetivo facilitar a educação em programação e robótica, sendo especialmente voltada para crianças e adolescentes.

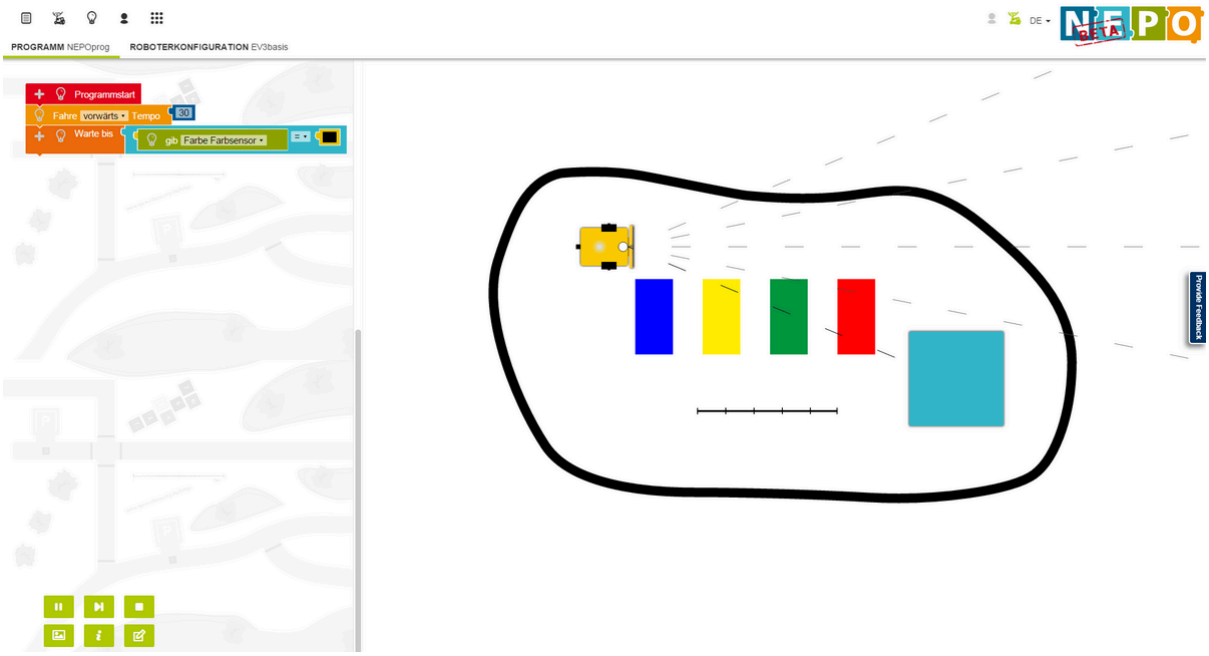
A principal proposta do *Open Roberta* é tornar o ensino de robótica e programação acessível a todos, independentemente do nível de conhecimento prévio. A plataforma oferece uma interface gráfica intuitiva onde os usuários podem programar diferentes tipos de robôs, como *LEGO Mindstorms*, *Calliope mini*, *micro*, entre outros.

Seguindo essa linha da acessibilidade, O *Open Roberta* utiliza uma abordagem baseada em blocos, semelhante ao *Scratch*[1]. O *Scratch* é uma linguagem de programação visual baseada em blocos, desenvolvida pelo MIT(massachusetts institute of technology), que permite criar animações, jogos e histórias interativas de forma intuitiva. Ele é amplamente usado na educação para ensinar conceitos de programação de maneira acessível e divertida. Isso facilita a compreensão dos conceitos de programação, especialmente para iniciantes e crianças. Por outro lado, caso o usuário queira um ambiente em que ele de fato escreva linhas de código em uma linguagem de programação específica e veja na prática a ação executada, a ferramenta não é recomendada.

Na figura 2, é exibido um exemplo de circuito na simulação do *Open Roberta*

Figura 2 - Simulação de um circuito no *Open Roberta*

[1] Disponível em: https://scratch.mit.edu/. Acesso em: 11 fev. 2025.



Fonte: Open Roberta Lab (c2024). Disponível em: https://lab.open-roberta.org/. Acesso em: 10 nov. 2024.

A plataforma fornece uma ampla gama de tutoriais, lições e exemplos de projetos que ajudam os usuários a aprender e explorar. Além disso, há uma comunidade ativa de educadores e entusiastas que compartilham conhecimentos, projetos e experiências, promovendo um ambiente colaborativo de aprendizado. Porém, a curva de aprendizado do *Open Roberta* é um pouco complexa. Embora a plataforma seja acessível para iniciantes, a criação de projetos mais avançados pode se tornar complexa e exigir conhecimentos adicionais que não são plenamente cobertos pela interface de blocos e não são explicados pelos tutoriais da ferramenta, além de o site não manter uma organização clara acerca de suas funcionalidades. Outro problema é o fato de que toda essa informação não está disponível totalmente na língua portuguesa, o que pode acabar dificultando o aprendizado de quem não está confortável com a utilização de outras línguas.

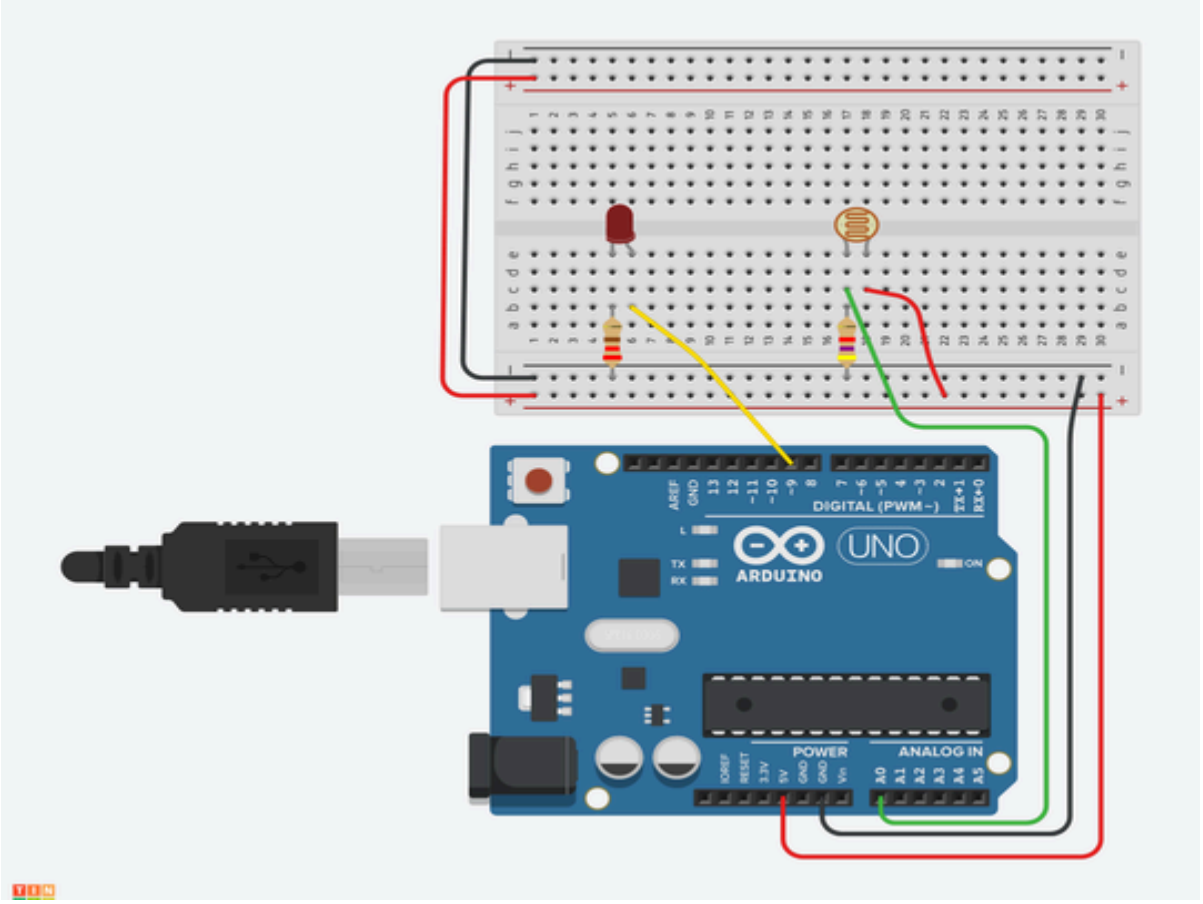
Com isso, o *Open Roberta* é uma ferramenta poderosa e acessível que desempenha um papel significativo na educação por meio da robótica e da programação. No entanto, suas limitações em termos de escrita de código e curva de aprendizagem, necessária e complexa, podem ser um empecilho para quem queira promover projetos de desenvolvimento rápido e dinâmicos e devem ser

consideradas ao planejar a sua implementação e utilização em ambientes educacionais.

2.3.2 Tinkercad

O *Tinkercad* é uma ferramenta online gratuita desenvolvida pela Autodesk que permite a criação de modelos 3D de maneira simples e intuitiva. Lançado em 2011, o *Tinkercad* tem sido amplamente utilizado em diversas áreas, desde a educação até a engenharia, por sua interface amigável e recursos acessíveis.

Figura 3 – Prototipação no *Tinkercad*



Fonte: Tinkercad (c2024). Disponível em: https://www.tinkercad.com/. Acesso em: 15 nov. 2024.

Uma das principais características do *Tinkercad* é sua acessibilidade. Como é uma aplicação baseada na *web*, não requer instalação de *software*, podendo ser

utilizada em qualquer dispositivo com acesso à internet. Isso facilita seu uso em ambientes educacionais onde o acesso a *softwares* específicos pode ser limitado.

No contexto educacional, o *Tinkercad* tem sido uma ferramenta valiosa para o ensino de conceitos de *design*, engenharia e programação. Professores utilizam o *Tinkercad* para introduzir os alunos ao mundo da modelagem 3D, permitindo que desenvolvam projetos de maneira prática e visual. Além disso, a ferramenta oferece tutoriais e lições integradas, auxiliando no aprendizado progressivo dos estudantes.

Para projetos de engenharia e *design*, o *Tinkercad* oferece funcionalidades que permitem a criação de peças complexas através de uma abordagem baseada em formas básicas que podem ser combinadas, subtraídas e manipuladas para obter o *design* desejado. Além disso, a integração com outras ferramentas da *Autodesk* permite que os modelos criados no *Tinkercad* sejam exportados e refinados em *softwares* mais avançados, como o *Autodesk Fusion 360*.

O *Tinkercad* também se destaca na área de eletrônica e programação, oferecendo um ambiente de simulação de circuitos eletrônicos. Usuários podem criar e testar circuitos utilizando componentes virtuais, programar microcontroladores (como o Arduino) e ver os resultados de suas simulações em tempo real. Esse recurso é especialmente útil para estudantes e pessoas que apenas desejam experimentar o uso da eletrônica sem a necessidade de equipamentos físicos.

O uso do *Tinkercad* em ambientes educacionais pode aumentar o engajamento dos alunos e facilitar a compreensão de conceitos complexos. A simplicidade da interface e a possibilidade de visualizar instantaneamente os resultados dos projetos tornam o aprendizado mais dinâmico e interativo.

Embora seja uma excelente ferramenta para conhecer e aprender programação, o *Tinkercad* tem alguns pontos negativos. Um desses pontos é a complexidade que um programador iniciante pode encontrar ao começar o desenvolvimento, já que hápossui muitos recursos que podem deixá-lo confuso. Outro ponto que pode ser insuficiente no *Tinkercad* é a falta de recursos de colaboração em tempo real em seus projetos. Para equipes que trabalham em conjunto em projetos de *design* ou engenharia ou até mesmo para estudantes que desejam aprender e desenvolver juntos, é necessário estar em um mesmo ambiente para que haja essa colaboração e troca de informações entre os envolvidos.

Em resumo, o *Tinkercad* é uma ferramenta versátil e poderosa que tem contribuído significativamente para a educação e para a prática de *design* e engenharia, sendo uma excelente ferramenta para iniciantes e entusiastas da área. Sua acessibilidade, sua interface intuitiva e sua integração com outras ferramentas fazem dele um recurso valioso tanto para iniciantes quanto para profissionais. Porém, é preciso avaliar melhor as suas necessidades de uso durante o desenvolvimento, pois a ferramenta é insuficiente quando queremos recursos mais avançados que nos tragam uma maior liberdade de criação de projetos e desenvolvimento.

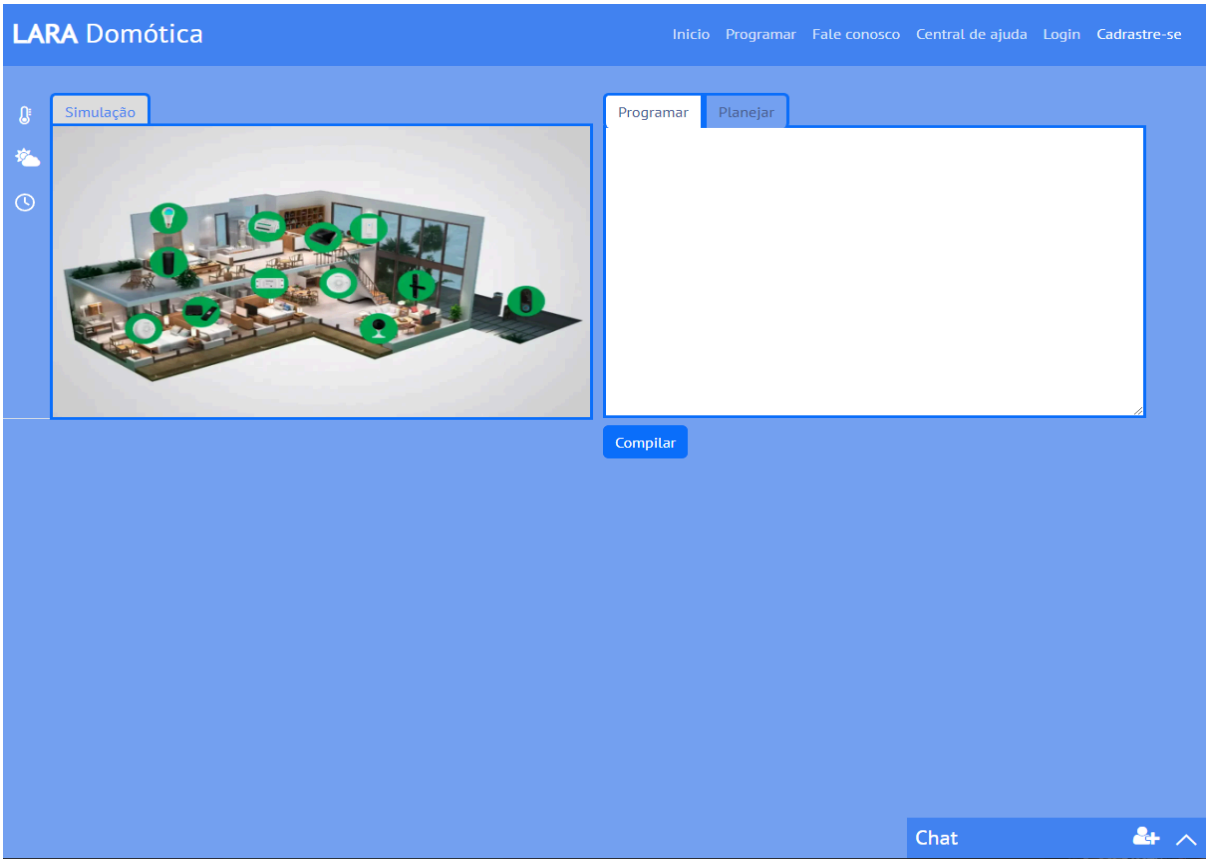
2.3.3 Trabalhos de conclusão de curso

Os trabalhos de conclusão de curso (TCC), são ótimas ferramentas de estudos, principalmente para a utilização como trabalhos relacionados. Por muitas vezes terem objetos parecidos de pesquisa, muitos destes trabalhos acabam se complementando ou, ainda, oferecendo informações cruciais para o desenvolvimento de novos projetos. Abordam-se, a seguir, três trabalhos que serviram de inspiração para o desenvolvimento deste projeto.

2.3.3.1 Lara Domótica - Protótipo de Laboratório virtual de Domótica

O trabalho de Lisboa (2023) apresenta o *LARA Domótica*, um protótipo de laboratório virtual voltado para o ensino de programação por meio da simulação de uma casa inteligente. A proposta baseia-se na utilização de um ambiente interativo em que os alunos podem programar dispositivos automatizados, como sensores de movimento e iluminação, e visualizar os efeitos de seus códigos em tempo real, como mostrado na figura 4. O sistema foi projetado para auxiliar no ensino de programação, tornando o aprendizado mais acessível e motivador, especialmente para iniciantes.

Figura 4 – Tela principal do Lara Domótica



Fonte: Lisboa (2023, p. 47)

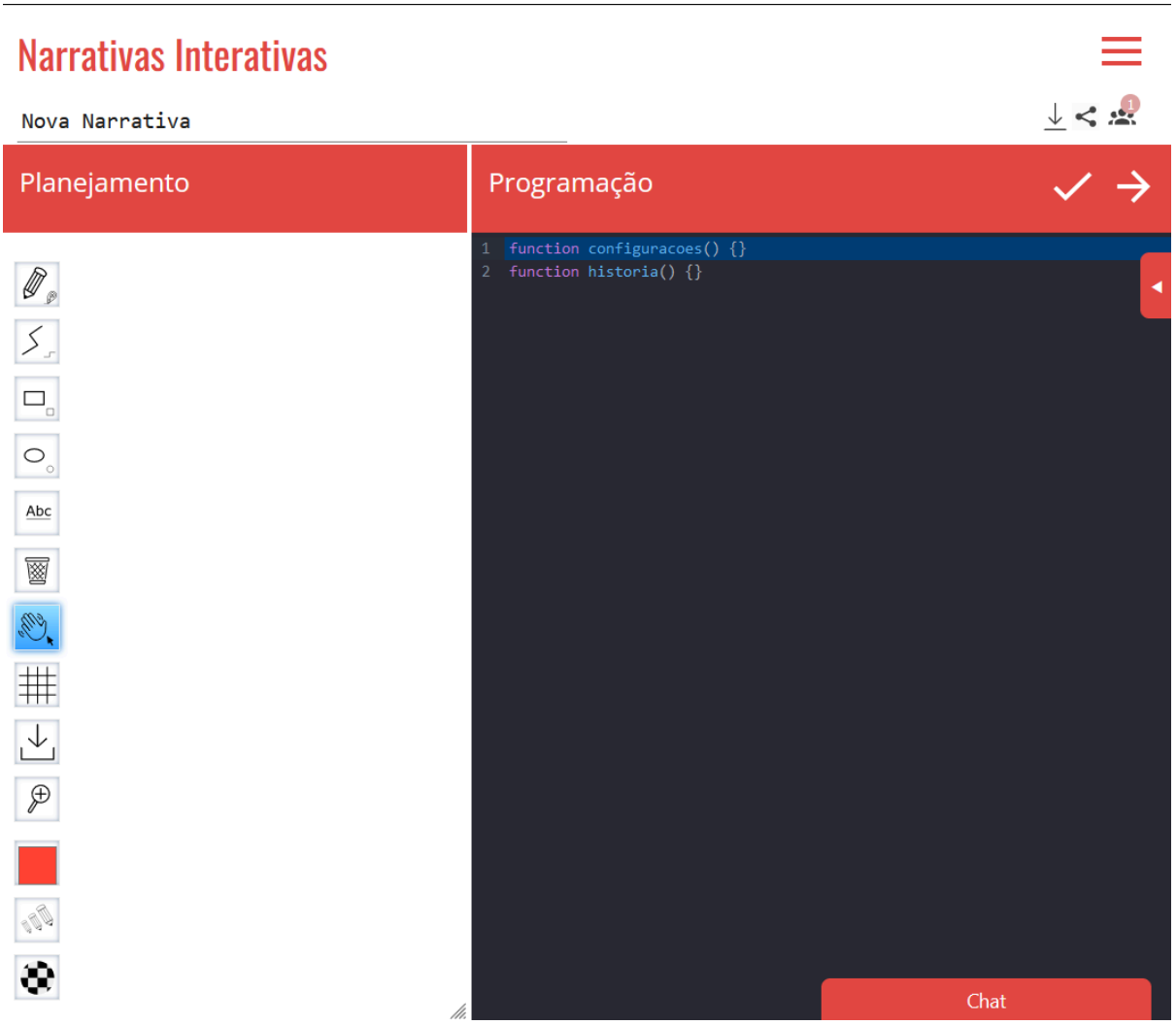
Além disso, a ferramenta promove a colaboratividade, permitindo atividades em grupo e interação entre alunos e professores. O estudo utilizou a metodologia Design Science Research (DSR) e avaliou a usabilidade do sistema por meio das heurísticas de Nielsen, aprimorando a interface com base no *feedback* dos usuários. Os resultados indicaram que a abordagem baseada em simulação e experimentação prática pode facilitar a aprendizagem e engajar os estudantes no desenvolvimento de suas habilidades de programação.

2.3.3.2 Desenvolvimento de um laboratório virtual para criação de histórias interativas

O trabalho de Pereira (2022) apresenta o desenvolvimento de um laboratório virtual para criação de histórias interativas como uma ferramenta de apoio ao ensino de programação. A pesquisa fundamenta-se na dificuldade enfrentada por alunos iniciantes na área, explorando o uso de narrativas interativas como estratégia

pedagógica. Para isso, foi desenvolvido o *Ecoden (*figura 5*)*, um ambiente virtual colaborativo baseado na biblioteca Nine.js, que permite a escrita e execução de histórias interativas em *JavaScript*. O sistema foi projetado com uma arquitetura de microsserviços, integrando funcionalidades como edição de código, planejamento com fluxogramas, *chat* e colaboração em tempo real.

Figura 5 – Tela principal do projeto Narrativas Interativas



Fonte: Pereira (2022)

A avaliação do sistema indicou que a utilização de histórias interativas pode tornar o aprendizado de programação mais envolvente e acessível, reforçando a importância de ferramentas tecnológicas no ensino de TI.

3.3.3 Lara Core Server – API: Implementação do LARA (Laboratório em Redes de aprendizagem) como um serviço

O trabalho de Sousa (2023) apresenta uma proposta de arquitetura orientada a serviços para o LARA (Laboratório em Redes de Aprendizagem), um ambiente virtual de aprendizagem. O objetivo principal foi substituir a arquitetura existente por uma abordagem mais modular e escalável, utilizando um *webservice RESTful* desenvolvido com Django, Django Rest Framework e MySQL. A metodologia adotada foi a Design Science Research, permitindo uma abordagem iterativa para o desenvolvimento. A arquitetura proposta foi baseada no modelo MVS (Modelo, Serviço, Visualização), visando melhorar a estrutura e integração do sistema com outras plataformas. Os testes realizados demonstraram que a nova implementação atendeu aos requisitos propostos, promovendo melhorias na comunicação entre aplicações, controle de dados e segurança nas trocas de informações.

3 SIMUCAR VIRTUAL

Nesta seção serão apresentados a proposta do trabalho, seus requisitos funcionais e não funcionais, os diagramas que auxiliam em sua construção, suas interfaces e o passo a passo de como funciona o programa e como ele foi construído.

3.1 Proposta

O trabalho desenvolvido neste projeto consiste na construção de um laboratório virtual em que o aluno tenha a liberdade de mover um desenho em forma de carrinho, em uma área pré-determinada que denominaremos de pista, por meio de comandos similares aos comandos de uma linguagem de programação, para que o auxiliem na aprendizagem e façam com que eles fiquem mais interessados em aprender. Por meio do laboratório, o aluno entenderá melhor como funcionam os comandos e conceitos básicos da programação (sequência de comandos, comandos condicionais IF, ELSE, VARIÁVEL, comandos de repetição, FUNÇÕES

etc.), pois será inserido em uma abordagem prática, fazendo com que sua aprendizagem seja mais dinâmica, simples e interessante.

Sendo assim, é possível definir as especificações do projeto em uma abordagem de construção de *software*, priorizando os requisitos do projeto em que são contextualizadas e definidas suas principais características.

3.2 Requisitos

Nesta seção serão apresentados os requisitos do sistema, funcionais e não funcionais. Esses requisitos foram pensados e discutidos durante o planejamento do projeto levando em consideração as necessidades do LARA.

3.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e comportamentos que o sistema deve possuir, detalhando as operações, ações e tarefas que o sistema deve ser capaz de realizar. Em essência, são os requisitos que especificam o que o sistema deve fazer para atender às necessidades dos usuários.

Neste projeto, os requisitos funcionais do sistema são:

- **Criação e movimentação do robô**: o sistema deve permitir que os usuários criem e movimentem o robô em um ambiente virtual, utilizando comandos de programação.
- **Inserção de linhas e obstáculos**: o sistema deve permitir a inserção de linhas e obstáculos no ambiente virtual, para que os usuários possam criar circuitos e cenários personalizados.
- **Detecção de cores**: o robô deve ser capaz de detectar cores no ambiente virtual, permitindo a implementação de algoritmos que dependem da identificação de cores para a execução de ações programadas.
- **Programação do robô:** o sistema deve suportar a implementação de programas, permitindo que os usuários criem comportamentos automáticos e condicionais para o robô

3.2.2 Requisitos não funcionais

Os requisitos não funcionais especificam os critérios de qualidade e as restrições do sistema. Eles descrevem como o sistema deve funcionar, em vez de quais funções ele deve realizar. Esses requisitos estão frequentemente relacionados a aspectos como usabilidade, confiabilidade, desempenho e manutenção do sistema. Os requisitos não-funcionais do sistema são:

- **Usabilidade**: o sistema deve ser intuitivo e fácil de usar, permitindo que os alunos interajam com o ambiente virtual de forma fluida e eficiente.
- **Desempenho**: o sistema deve ser capaz de processar e executar os comandos de programação em tempo real, sem atrasos ou falhas.
- **Disponibilidade**: o sistema deve ser acessível de qualquer lugar com conexão à internet, permitindo que os alunos estudem em seu próprio ritmo e horário.

3.3 Ferramentas e tecnologias utilizadas

A escolha da arquitetura e das tecnologias utilizadas no desenvolvimento do projeto foi fundamentada na experiência prévia do autor e na necessidade de adotar soluções eficientes para atender aos requisitos do sistema. A decisão envolveu a seleção de *frameworks*, linguagens e ferramentas que proporcionassem escalabilidade, desempenho e facilidade de manutenção. Além disso, levou-se em consideração a compatibilidade com padrões modernos de desenvolvimento, garantindo a integração com outras plataformas e a acessibilidade do sistema.

Dessa forma, a tecnologia escolhida não apenas reflete o conhecimento técnico do autor, mas também a busca por um equilíbrio entre inovação e confiabilidade para a construção de uma solução que atenda os critérios e os requisitos da aplicação.

3.3.1 Linguagens

As linguagens fundamentais *HTML*, *CSS* e *JavaScript* constituem a base do desenvolvimento *web*, cada uma desempenhando um papel essencial na construção de interfaces interativas e responsivas. O *HTML* (*HyperText Markup Language*) é responsável pela estruturação do conteúdo, definindo os elementos que compõem uma página, como textos, imagens e *links*. O CSS (*Cascading Style*

Sheets) complementa essa estrutura, permitindo a estilização e o *design* visual por meio de regras que controlam cores, *layouts* e animações, garantindo uma experiência de usuário mais agradável e intuitiva. Por fim, o *JavaScript* adiciona dinamismo e interatividade às páginas, possibilitando manipulação do DOM, requisições assíncronas e integração com APIs externas. Sua utilização, em conjunto com *frameworks* e bibliotecas modernas, contribui para o desenvolvimento de aplicações *web* avançadas e escaláveis.

3.3.2 Frameworks

A utilização de *frameworks* na programação *web* é um elemento crucial para o desenvolvimento eficiente e estruturado de aplicações online. Essas estruturas oferecem conjuntos de ferramentas, padrões e convenções que simplificam a construção de projetos robustos, permitindo aos desenvolvedores concentrarem-se mais na lógica específica da aplicação do que em questões infraestruturais.

Os *frameworks* na programação *web* tiveram suas raízes consolidadas na necessidade de lidar com a complexidade crescente das aplicações online. Desde o advento da *web* dinâmica, surgiram diversos *frameworks* que visavam simplificar a codificação, promovendo a reutilização de código e a padronização das práticas de desenvolvimento.

A diversidade de *frameworks* é notável, variando desde *frameworks front-end*, como *React*, *Angular* e *Vue.js*, até *frameworks back-end*, como *Django* e *Ruby on Rails*. Além disso, *frameworks full-stack*, como *Laravel* e *Spring*, oferecem soluções abrangentes para o desenvolvimento completo de aplicações.

3.3.3 Vue.JS

O *Vue.js* é um *framework* progressivo para a construção de interfaces de usuário, amplamente adotado por sua simplicidade, flexibilidade e desempenho. Baseado em *JavaScript*, o *Vue.js* utiliza um modelo de arquitetura baseado em componentes reativos, permitindo o desenvolvimento modular e reutilizável de aplicações *web*. Sua adoção no projeto foi motivada pela experiência prévia do

autor e pela necessidade de um *framework* que oferecesse uma curva de aprendizado acessível sem comprometer a escalabilidade.

Dentre suas principais características, destacam-se o *Virtual DOM*, que otimiza a renderização dos componentes, e o *two-way data binding*, que facilita a sincronização de dados entre a interface e a lógica de negócio. Além disso, o *Vue.js* possui um ecossistema robusto, incluindo ferramentas como *Vue Router* para roteamento e *Vuex* ou *Pinia* para gerenciamento de estado, proporcionando uma base sólida para o desenvolvimento de aplicações interativas e dinâmicas.

A escolha do *Vue.js* reflete a busca por uma tecnologia que alia desempenho, modularidade e produtividade, garantindo a construção de uma aplicação responsiva e eficiente, alinhada às necessidades do projeto.

3.3.4 CodeMirror

O *CodeMirror* é uma biblioteca *JavaScript* altamente customizável para a criação de editores de código embutidos em aplicações *web*. Ele oferece suporte a diversas linguagens de programação, realce de sintaxe, autocompletar, recuo automático e múltiplas seleções, tornando-se uma ferramenta essencial para o desenvolvimento de ambientes de edição de código interativos.

Sua arquitetura modular permite a adição de extensões e *plugins*, possibilitando a personalização conforme as necessidades do projeto. Além disso, o *CodeMirror* é amplamente utilizado em IDEs online, plataformas educacionais e sistemas de gerenciamento de código, devido ao seu desempenho otimizado e compatibilidade com navegadores modernos.

A escolha do *CodeMirror* para o desenvolvimento do projeto baseia-se na necessidade de oferecer uma experiência de edição fluida e intuitiva, alinhada com as melhores práticas do desenvolvimento *web*.

3.3.5 Supabase

O *Supabase*, que foi utilizado para fazer o sistema de autenticação do laboratório, é uma plataforma de desenvolvimento *back-end* como serviço (BaaS) que oferece um conjunto de ferramentas para facilitar o gerenciamento de banco de

dados, autenticação de usuários e armazenamento de arquivos. Ele é construído sobre o PostgreSQL, permitindo que os desenvolvedores utilizem um banco de dados relacional robusto com suporte nativo a consultas SQL e extensões avançadas. Entre seus principais recursos, destacam-se:

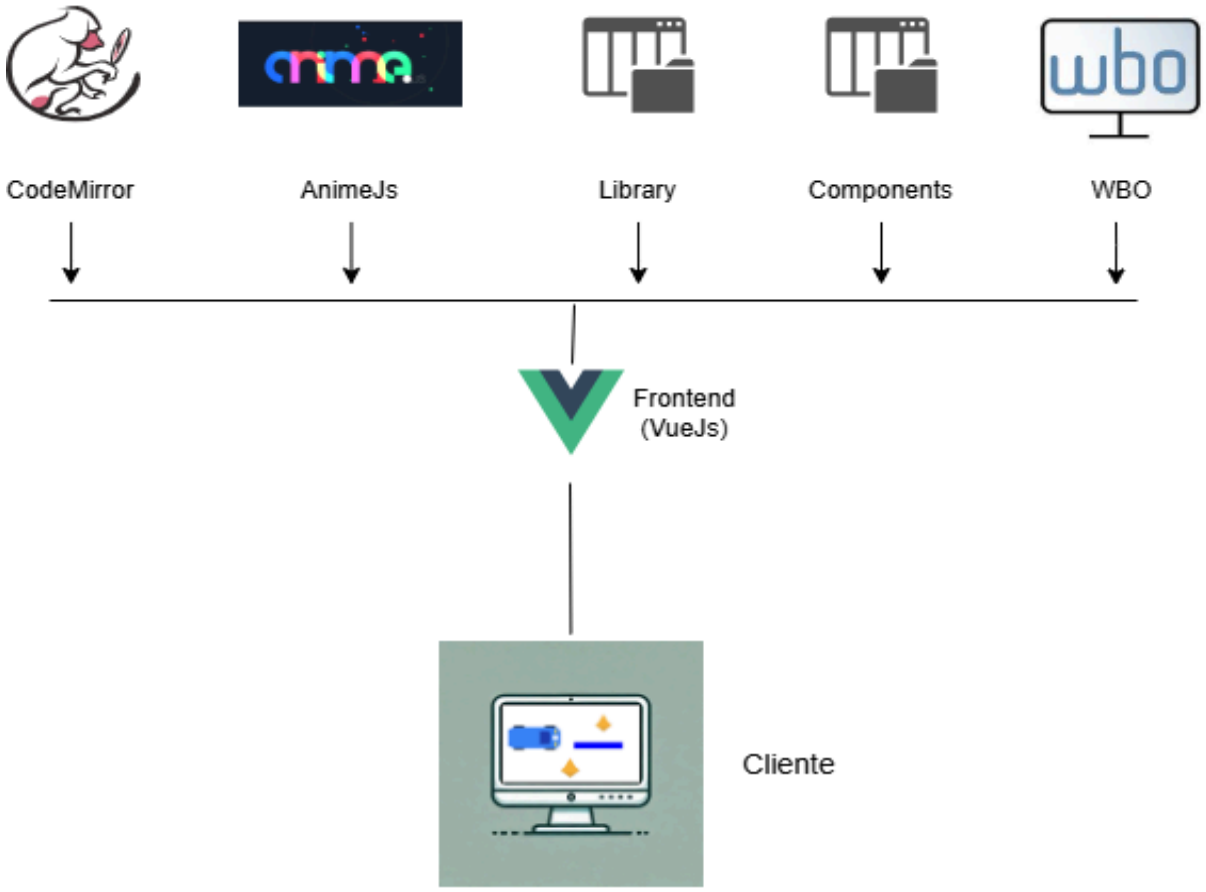
- **Banco de Dados PostgreSQL**: o *Supabase* fornece um banco de dados PostgreSQL gerenciado, permitindo consultas SQL diretas e suporte a *WebSockets* para atualizações em tempo real;
- **Autenticação e Autorização**: inclui suporte nativo para autenticação via e-mail/senha, provedores *OAuth* (*Google*, *GitHub*, etc.) e autenticação por terceiros, além do controle de permissões baseado em regras de acesso (RLS – *Row Level Security*);
- **Armazenamento de Arquivos**: possui um sistema de armazenamento de objetos compatível com S3, permitindo o upload e gerenciamento de arquivos como imagens, vídeos e documentos.

Por ser uma alternativa *open-source* ao *Firebase*, o *Supabase* se destaca por proporcionar mais flexibilidade e controle sobre os dados, além de ser compatível com diversas tecnologias *front-end* e *back-end*. Ele é especialmente útil para projetos que exigem um *back-end* escalável e com suporte a SQL, sem a complexidade de configurar servidores do zero

3.4 Arquitetura

A arquitetura do *Simucar Virtual* foi projetada para garantir uma experiência de usuário fluida e eficiente (figura 6), permitindo que os alunos interagissem com o laboratório virtual de maneira intuitiva e prática. A estrutura do sistema foi desenvolvida com base em uma abordagem modular, facilitando a manutenção, expansão e integração de novas funcionalidades no futuro.

Figura 6 – Arquitetura do Simucar



Fonte: autoria própria (2025)

A seguir, serão detalhados os principais componentes da arquitetura do sistema.

Para garantir a utilização eficiente e simplificada das bibliotecas no desenvolvimento do projeto, adotou-se o *Vue.js* como tecnologia principal para a integração de componentes. O *Vue* atua como a base estrutural que possibilita a comunicação e o funcionamento harmonioso entre os diferentes elementos do sistema.

Além disso, visando uma melhor organização e manutenção do código, foi aplicada a abordagem de componentização. Com a separação dos arquivos em componentes independentes, torna-se mais fácil identificar e corrigir eventuais erros, além de melhorar a reutilização de código. A estrutura do *Vue.js* é projetada para facilitar essa componentização, permitindo uma importação eficiente e promovendo a modularização do sistema.

Sendo assim, também foi possível implementar a biblioteca (*library*) para o sistema. A biblioteca consiste nas funções que o carrinho pode executar, e,

posteriormente, essa função pode ser exportada e utilizada em qualquer parte do código. Dessa maneira, também fica muito mais fácil a criação de novas funções ou a atualização de alguma função para o *Simucar* que modifique diretamente o carrinho, bastando apenas criar no local adequado com a sua característica e utilizara parte do código que for necessária.

Outro componente utilizado é o *WBO*. O *WBO* (*Web Whiteboard Online*) é uma ferramenta de quadro branco colaborativo com várias ferramentas de desenho, baseada na *web*, projetada para facilitar a interação em tempo real entre usuários. Com uma interface intuitiva e suporte para múltiplos participantes, o *WBO* permite a criação e a edição simultânea de diagramas, esboços e anotações, tornando-se uma solução eficaz para ensino, trabalho remoto e *brainstorming* em equipe. Por ser uma plataforma de código aberto, ele pode ser integrado a outros sistemas educacionais, como Ambientes Virtuais de Aprendizagem (AVA), proporcionando uma experiência colaborativa enriquecida. Seu uso no contexto acadêmico e profissional favorece a comunicação visual e a troca de ideias de forma dinâmica e acessível.

3.5 Laboratório

Nesta seção serão apresentados todos os conteúdos relacionados ao desenvolvimento do laboratório. Será feita a descrição da interface e a demonstração de todos os comandos disponíveis da biblioteca.

3.5.1 Descrição dos componentes da interface

A partir dos requisitos, foi iniciado o desenvolvimento do projeto. Primeiramente, foi desenvolvido uma página de *login* (figura 7), em que é possível criar uma conta, caso não tenha, e posteriormente efetuar o *login*, para que o estudante comece os seus experimentos.

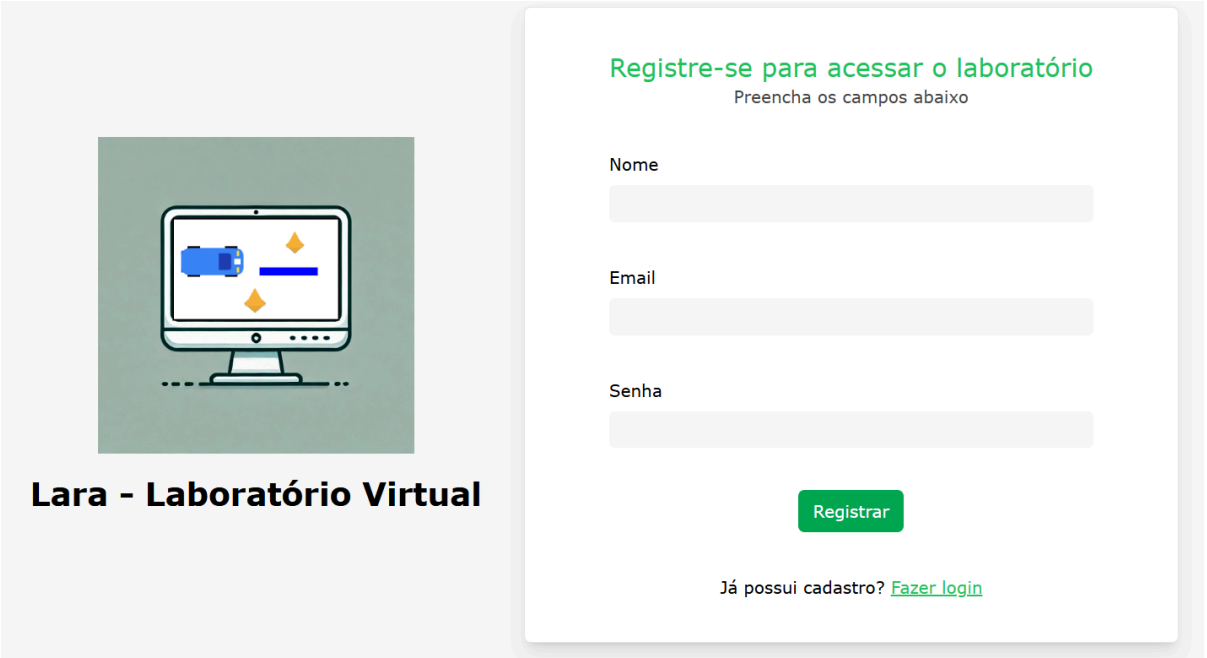
Figura 7 – Tela inicial de login

Fonte: autoria própria (2025)

O sistema de autenticação, que foi desenvolvido com o *Supabase*, permite uma facilidade na implementação desse requisito, uma vez que ele oferece todo o suporte de *back-end* como um serviço. Essa escolha foi agradável, pois permitiu alocar mais tempo no desenvolvimento do laboratório, que é o principal objetivo deste trabalho.

Como primeiro acesso, é necessário criar uma conta. Para isso, o usuário deve clicar em “Criar uma conta” no menu inferior direito, que o redirecionará para a página. O processo é simples, basta preencher com e-mail, nome e senha e a conta será criada, como mostrado na figura 8.

Figura 8 – Tela de criação de conta



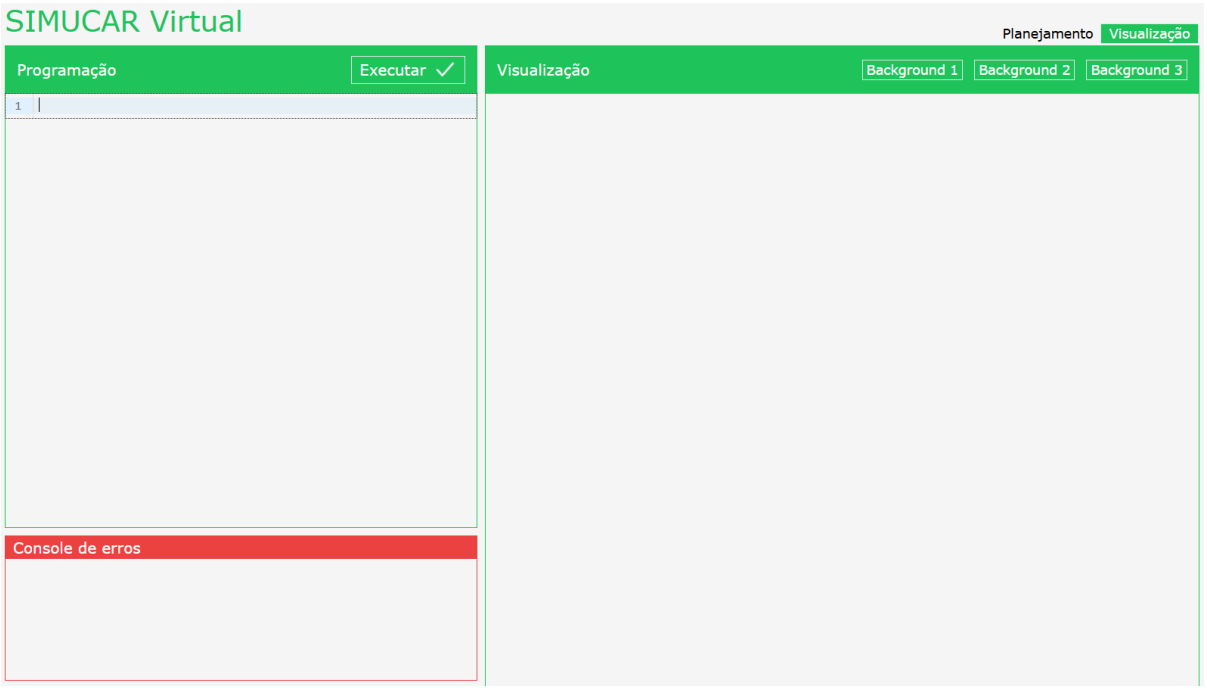
Fonte: autoria própria (2025)

Depois de criar a conta, o usuário pode efetuar o *login* com suas credenciais. Se tudo estiver correto, ele é redirecionado para a página principal do site (figura 7), onde é possível fazer todo o processo de aprendizagem.

Na página principal, o aluno tem acesso ao ambiente de programação, onde ele digita um programa e, a partir disso, é possível ver a execução daquilo que foi escrito. Do lado direito, está a aba denominada de v*isualização* (figura 9), onde será mostrada toda a ação do robô que for implementada corretamente. Nessa área, é possível ver o carrinho se locomover de acordo com as instruções recebidas.

Na parte superior da aba de visualização, há 3botões: “Background 1”, “Background 2” e “Background 3”. Esses botões servem para criar automaticamente um cenário pré-definido pelo sistema, para que o usuário não se preocupe com a criação de linhas e cones e seu posicionamento, mas apenas com a movimentação do carrinho.

Figura 9 – Tela inicial de visualização do laboratório

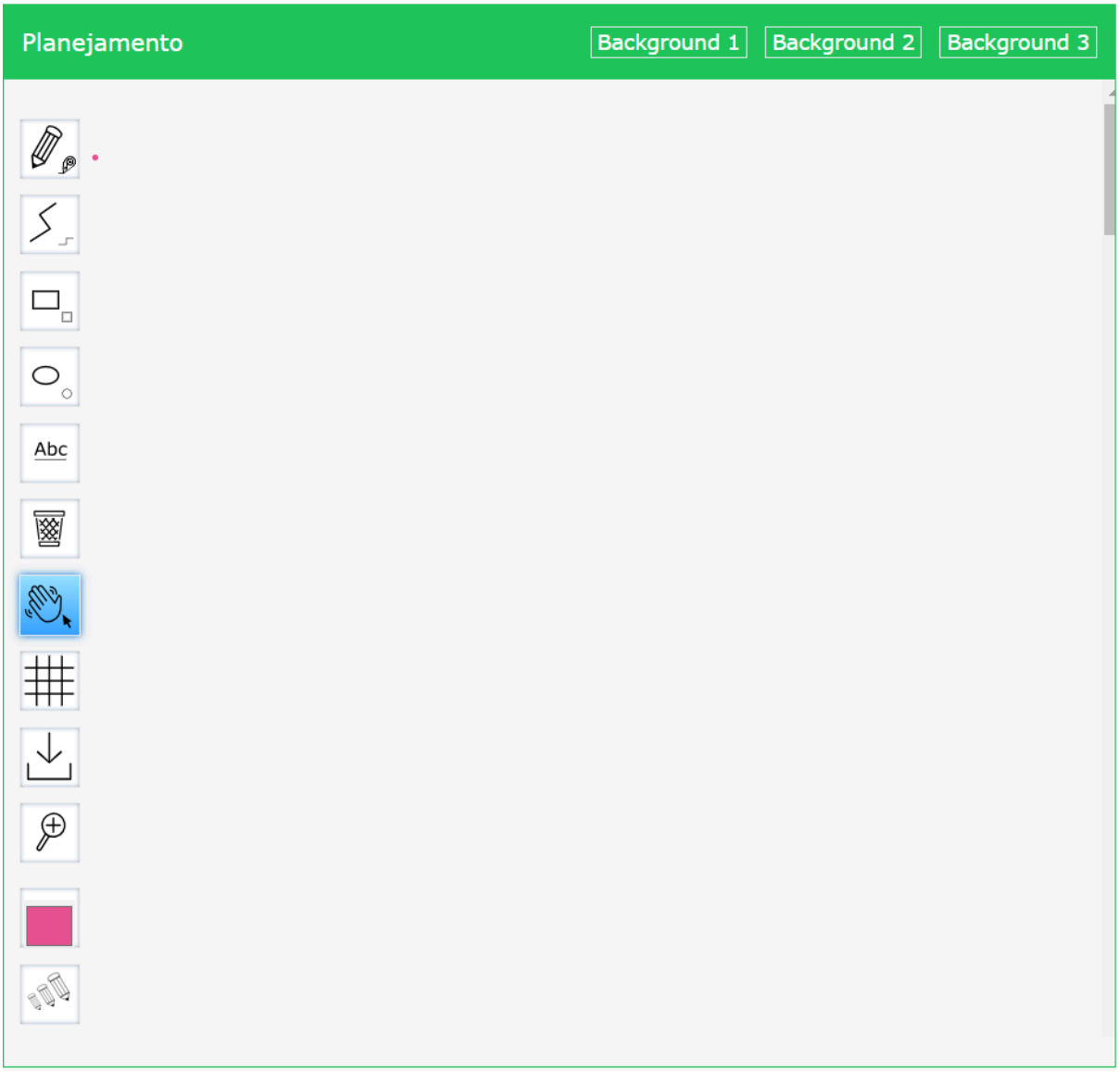


Fonte: autoria própria (2025)

Alternando entre abas, está a tela de *planejamento* (figura 10). Nessa tela, o usuário será capaz de fazer o planejamento do seu algoritmo utilizando formas, desenhos, cores etc. É importante realizar um planejamento de algoritmo, principalmente os novos estudantes, para que fique mais fácil o entendimento e consequentemente a resolução dos problemas. Para a execução dessa etapa, foi utilizada a biblioteca Wbo[2].

Figura 10 – Tela de Planejamento

[2] Disponível em: https://wbo.ophir.dev/. Acesso em: 20 nov. 2024.



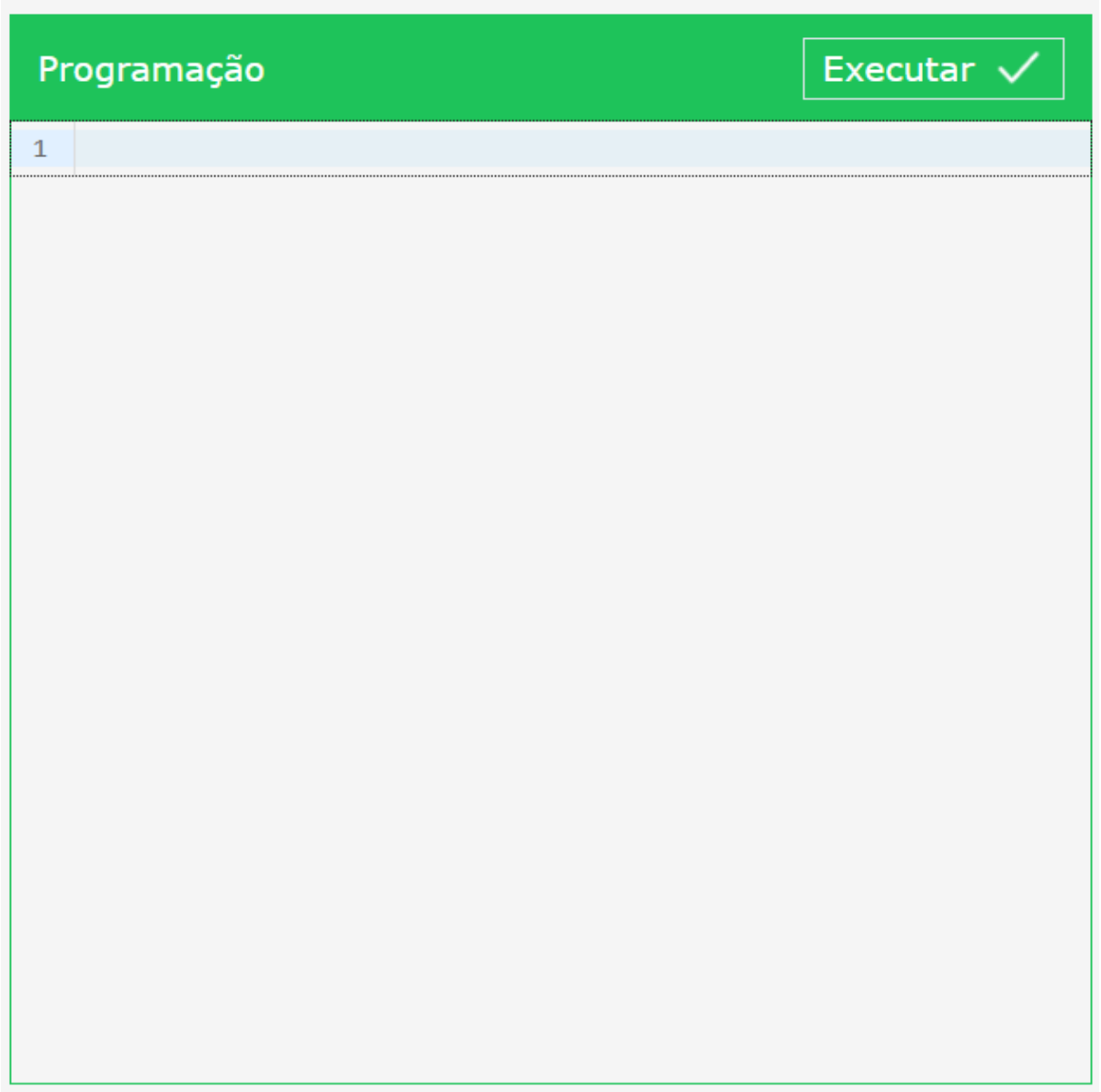
Fonte: autoria própria (2024)

Na área da programação, foi utilizado o *CodeMirror*[3], uma biblioteca *JavaScript* que fornece um editor de código em texto, que pode ser embutido em páginas *web*. Ele é amplamente utilizado em aplicativos *web* que requerem a edição de código, oferecendo uma experiência rica e semelhante a um IDE (Ambiente de Desenvolvimento Integrado). Nessa etapa, o aluno escreverá um código, que será explicado posteriormente, e que será interpretado pelo sistema quando ele clicar em "Executar". A partir dessa interpretação, o carrinho moverá, seja para esquerda, direita, cima ou baixo, sempre respeitando a área delimitadora em que ele se encontra. Também é possível seguir a linha e desviar dos obstáculos. Esses desvios

[3] Disponível em: https://codemirror.net/. Acesso em: 20 nov. 2024.

serão feitos pelo usuário, fazendo com que ele aprenda os comandos básicos de programação.

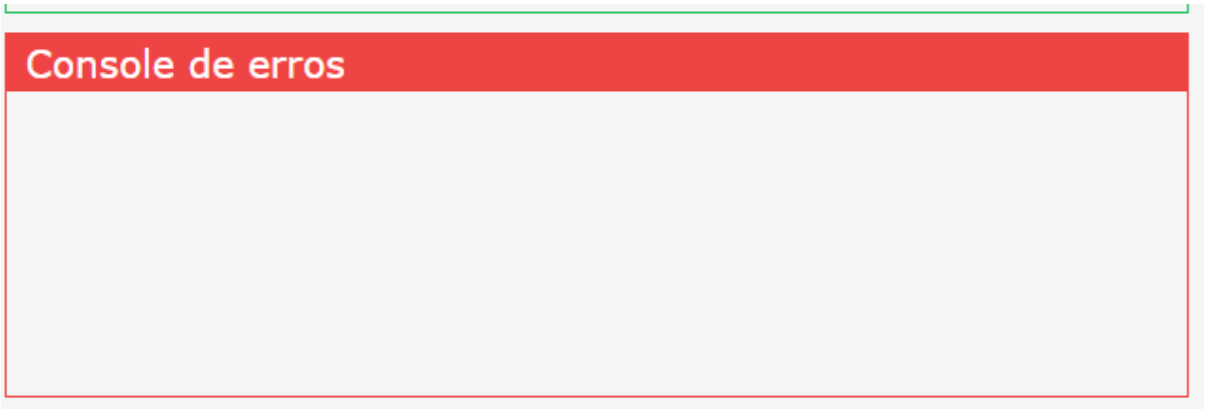
Figura 11 – Ambiente de programação



Fonte: autoria própria (2024)

Por fim, nessa tela inicial, há o console de erros (figura 12). Aqui, como o próprio nome aponta, será possível visualizar os erros que ocorrem durante a execução dos códigos, por meio de uma mensagem indicando qual foi o erro cometido pelo usuário.

Figura 12 – Console de erros da aplicação



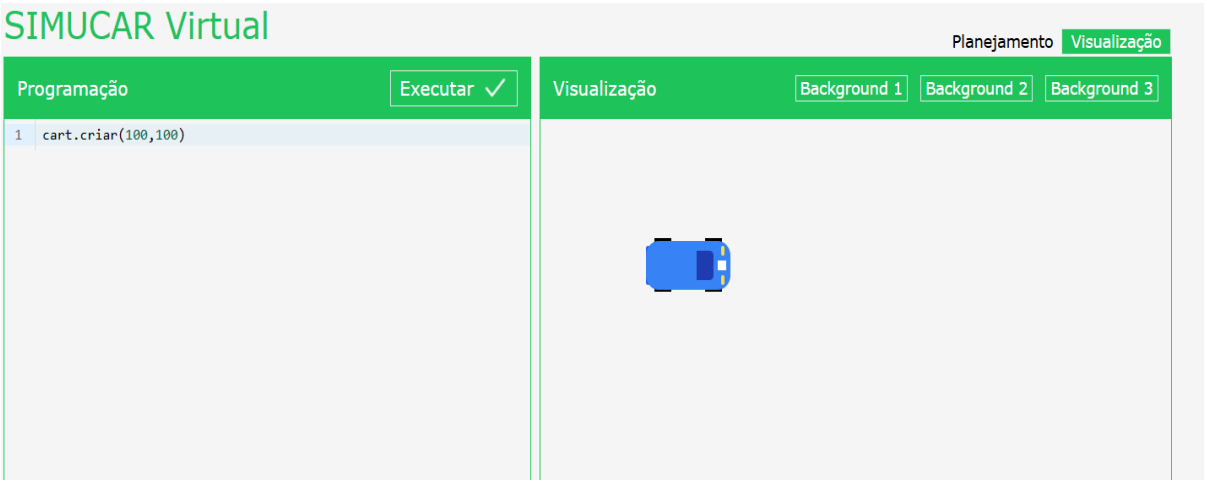
Fonte: autoria própria (2024)

A implementação deste console foi pensada tendo como inspiração os principais editores de códigos disponíveis para que o usuário já se familiarize com esse tipo de problema.

3.5.2 Iniciando a programação

Para iniciar no ambiente, é necessário digitar algum comando válido, que será interpretado pelo programa que fará o desenho na tela. Para começar, é possível iniciar com o comando para inserir o carrinho. Esse comando deverá ser escrito como: “cart.criar(x,y)”, com x sendo o valor inicial da coordenada X e Y o valor inicial da coordenada Y, ou seja, onde o usuário quer que o carrinho se posicione inicialmente. No exemplo abaixo, o carrinho foi criado com o comando “cart.criar(100,100)” na posição X = 100 e Y = 100

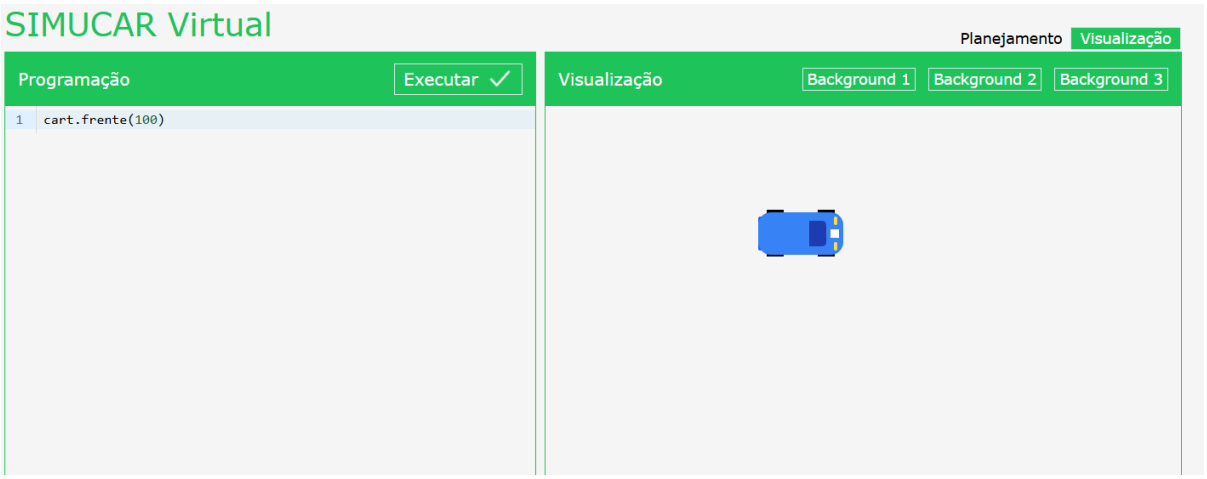
Figura 13 – Carrinho criado no Simucar Virtual



Fonte: autoria própria (2025)

Com o carrinho criado, pode-se fazer com que ele se mova para frente. Para que isso aconteça, no ambiente de programação, deve-se apagar a linha “cart.criar” e escrever, no lugar dela, “cart.frente(100)”, para que o carrinho se mova 100 *pixels* para frente. Ao término da escrita, basta clicar em “executar” que o carrinho se moverá para a frente, como mostra a figura 14.

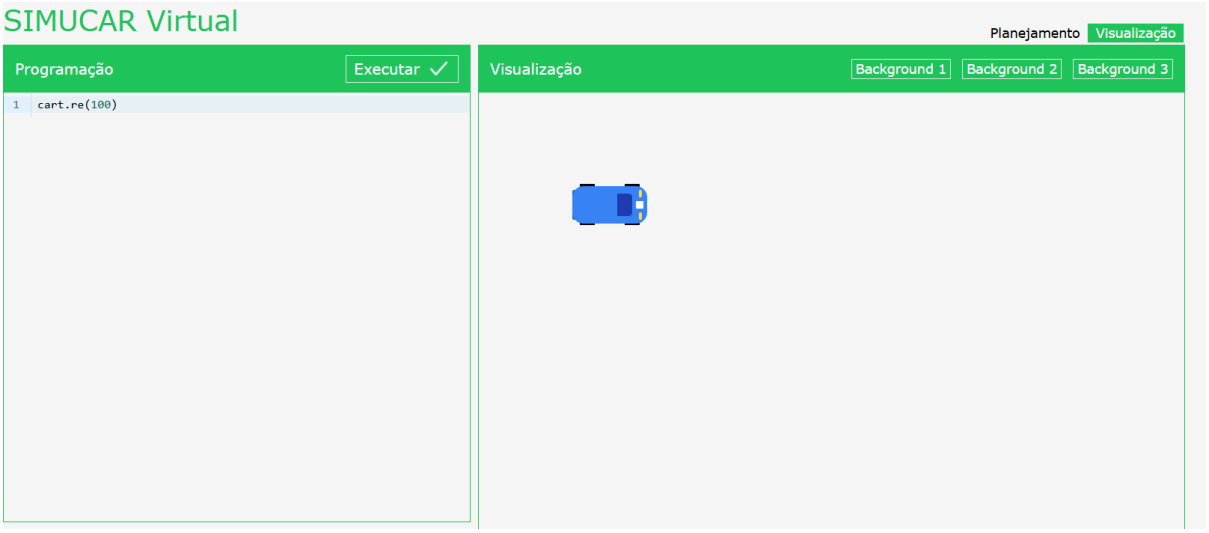
Figura 14 – Carrinho ao término do comando “cart.frente (100)”



Fonte: autoria própria (2025)

Para que o carrinho se mova para trás, o comando é parecido. Basta digitar “cart.re(100)” para que o carrinho se locomova para trás e volte à posição inicial do exemplo anterior, como mostra a figura 15.

Figura 15 – Carrinho após andar para trás



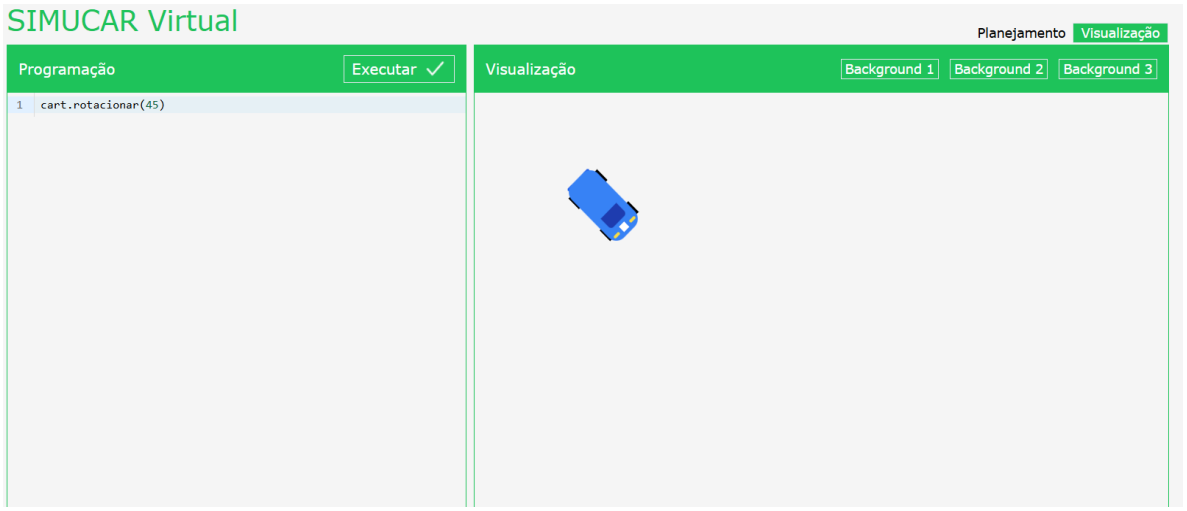
Fonte: autoria própria (2025)

Após a ação, o carrinho volta à posição inicial e está novamente disponível para novos comandos

3.5.3 Rotação

Uma outra funcionalidade implementada no sistema de controle do carrinho é a rotação. A rotação do carrinho é determinada pelo valor especificado pelo usuário no comando “cart.rotacionar(45)”, em que o número 45 representa o ângulo, em graus, que o carrinho deve rotacionar. Esse comando permite ajustar a orientação do carrinho de acordo com a necessidade da operação. Na figura 16, observa-se o carrinho posicionado em uma inclinação de 45 graus, resultado da execução do comando de rotação.

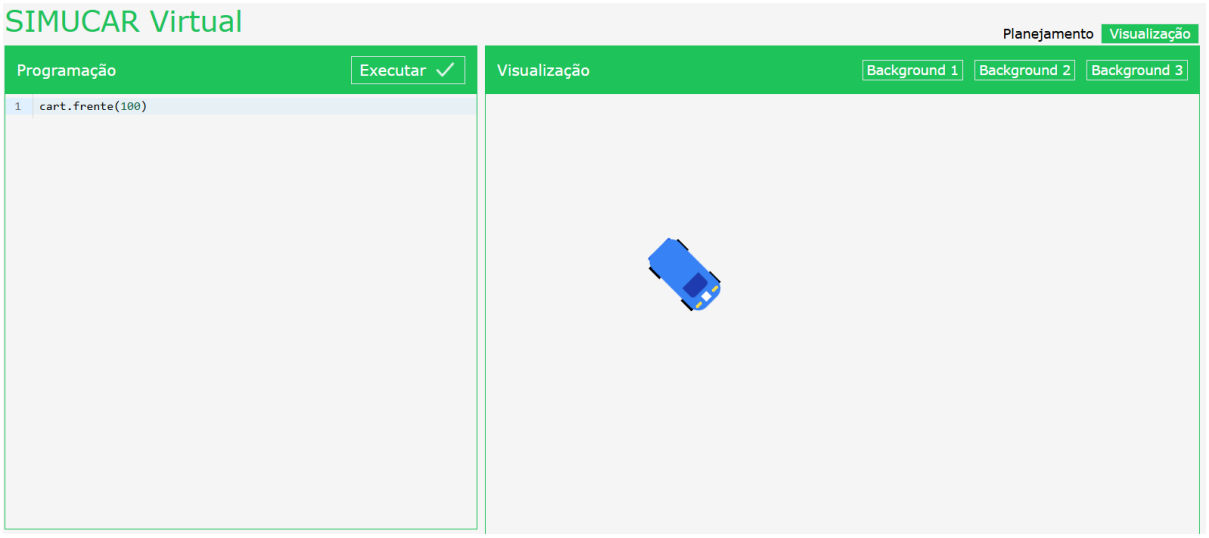
Figura 16 – Carrinho rotacionado em 45 graus



Fonte: autoria própria (2025)

Com o carrinho previamente rotacionado, se forem realizados novamente os movimentos para frente ou para trás utilizando os comandos “cart.frente(100)” ou “cart.re(100)”, respectivamente, esses deslocamentos ocorrerão seguindo o ângulo configurado anteriormente, garantindo que o carrinho se mova na direção correspondente à sua orientação definida, como pode ser visto na figura 17.

Figura 17 – Carrinho após andar para frente e com ângulo de 45 graus



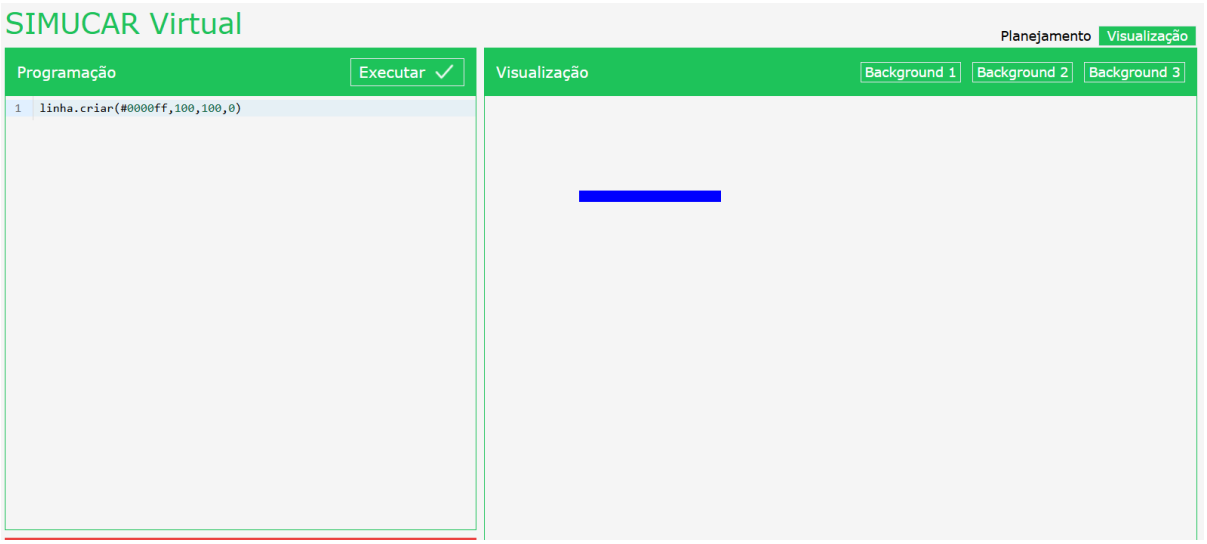
Fonte: autoria própria(2025)

Também é possível fazer com que o carrinho tenha uma rotação negativa, para que ele volte à posição original, basta passar como parâmetro da função um número negativo. Assim, o carrinho irá girar no sentido anti-horário.

3.5.4 Inserir linha

Outra funcionalidade disponibilizada no laboratório é a inserção de linhas. Essas linhas permitem a criação de circuitos variados no programa, possibilitando a personalização de atributos como tamanho, cor e inclinação. Dessa forma, o usuário possui total liberdade para desenvolver circuitos personalizados que atendam às suas necessidades específicas de programação. Para colocar uma linha em seu circuito, deve-se escrever “linha.criar(#0000ff,100,100,0)”, em que #0000ff é a cor da linha, que obrigatoriamente deve ser um número escrito em hexadecimal. Nesse caso, foi escolhidaa cor azul, como mostrado na figura 18. O primeiro e segundo parâmetro dessa função correspondem às posições *left* e *top*, respectivamente, de acordo com o contêiner pai em que está inserido. O quarto parâmetro corresponde a inclinação da linha, ou seja, ao seu ângulo. Para uma linha horizontal, foi usado como parâmetro o número 0. Para uma linha na vertical, utilizou-se como parâmetro o número 90, e assim por diante, podendo colocar qualquer número nesse parâmetro.

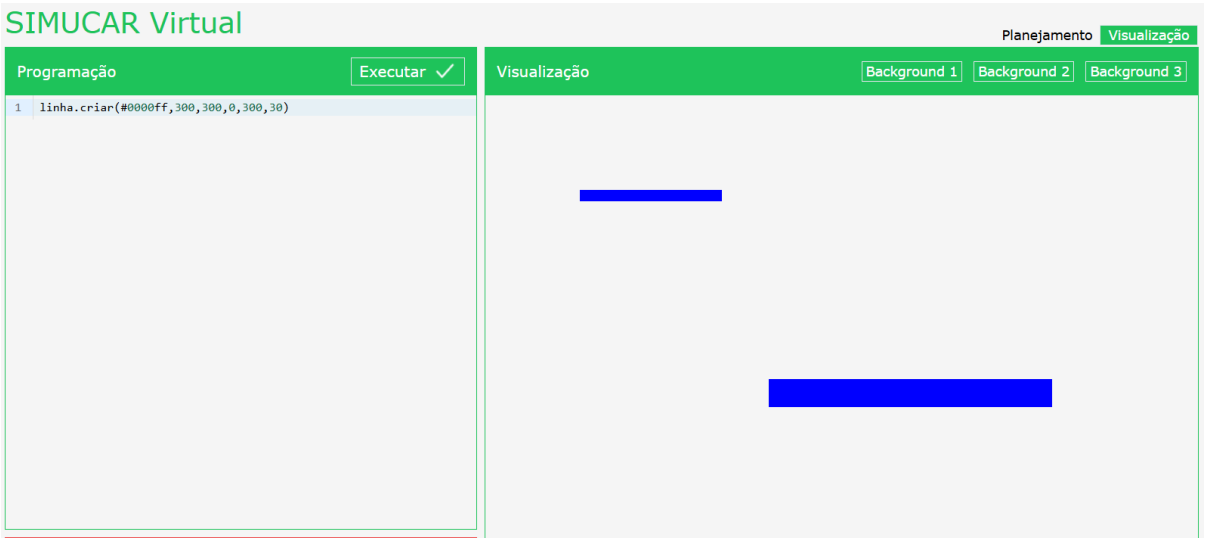
Figura 18 – Linha inserida na horizontal



Fonte: autoria própria (2025)

Essa função ainda aceita dois parâmetros opcionais. São eles: a largura da linha e a sua espessura. Com todas essas opções, é possível garantir uma gama de variedades na criação de linhas e de cenários no laboratório virtual. No exemplo abaixo, na figura 19, esses dois novos parâmetros foram adicionados com uma posição diferente.

Figura 19 – Segunda linha, com uma espessura maior, no circuito.



Fonte: autoria própria (2025)

Para uma maior agilidade no desenvolvimento, é possivel criar várias linhas de uma vez, basta escrever mais de um comando em cada linha e clicar em

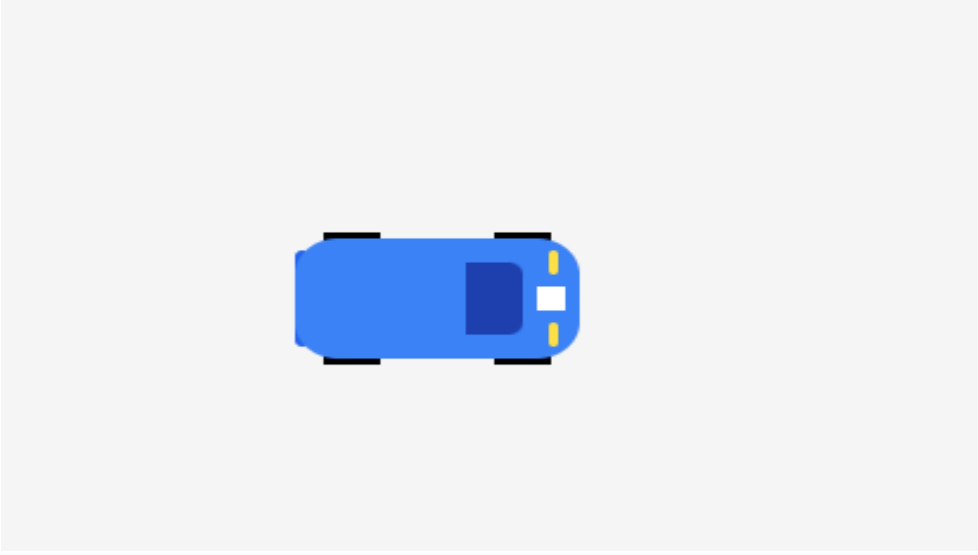
“executar”. Com isso o programa criará todas as linhas que foram digitadas corretamente

3.5.5 Detector de cores

Uma característica fundamental do carrinho presente no *Simucar* é a presença de um detector de cores, localizado estrategicamente em sua parte frontal, mais precisamente sobre o capô. Esse sensor desempenha um papel essencial na interação do carrinho com o ambiente virtual, permitindo que ele identifique a cor da superfície sobre a qual está se deslocando.

O funcionamento desse detector (figura 20) é dinâmico: sua cor se altera automaticamente conforme a cor da linha sobre a qual o carrinho está posicionado. Quando o carrinho é inicialmente criado no ambiente de simulação, pode-se observar que o detector não exibe nenhuma cor específica, permanecendo na cor branca. Esse estado indica que o carrinho não está sobre nenhuma linha detectável, encontrando-se, portanto, em uma área livre do ambiente de visualização.

Figura 20 – Carrinho com o detector na cor branca, localizado na parte frontal

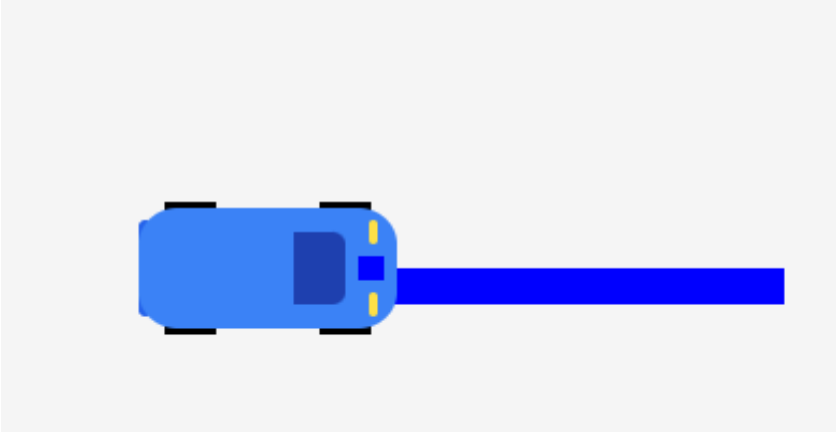


Fonte: autoria própria (2025)

Dessa forma, para observar o funcionamento do detector de cores, é necessário inserir uma linha no ambiente de simulação por meio do código e posicionar o carrinho sobre essa linha, como mostra a figura 21. Assim que o

carrinho estiver corretamente posicionado, o sensor de cores, localizado em seu capô, passará a exibir a mesma cor da linha inserida.

Figura 21 – Detector na cor azul, pois o carrinho está sobre a linha de mesma cor



Fonte: autoria própria (2025)

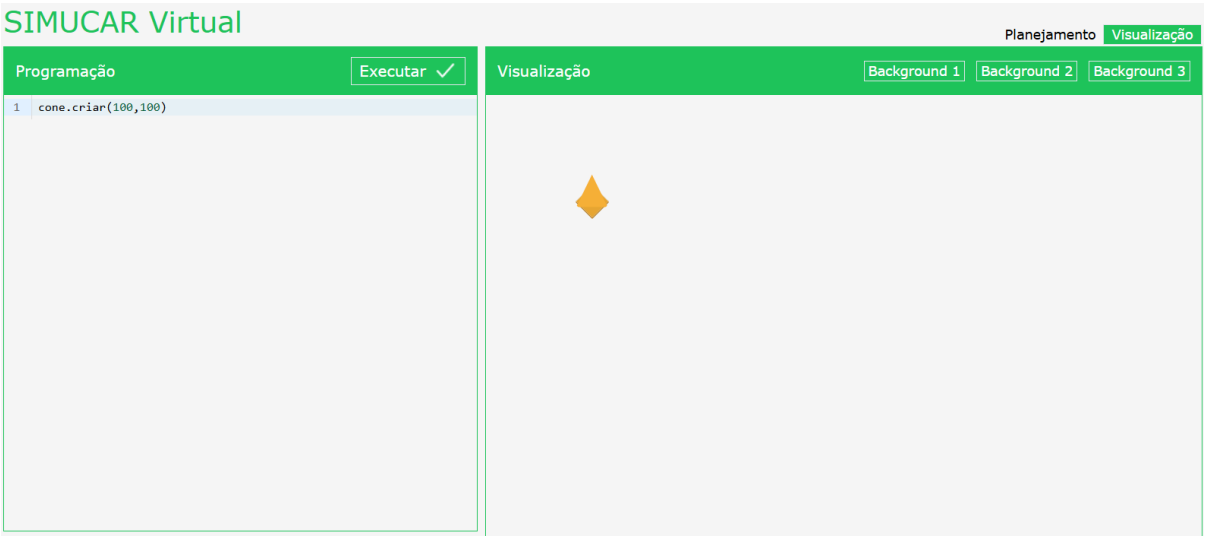
Esse comportamento confirma que o detector está operando corretamente e interagindo com o ambiente virtual. A mudança de cor do sensor permite que o carrinho reconheça diferentes padrões no percurso, possibilitando a implementação de algoritmos que dependem da identificação de cores para a execução de ações programadas. Essa funcionalidade é essencial para experimentos que envolvem navegação autônoma e seguimento de trajetórias pré definidas dentro do *Simucar*.

3.5.6 Inserir Obstáculos

Mais uma característica do laboratório virtual é a possibilidade de inserção de obstáculos, o que contribui para a simulação de um ambiente mais realista e didático. Para representar esses obstáculos, foi utiliaaa imagem de um cone, que ajuda a compor um circuito semelhante aos encontrados em cenários reais de treinamento e experimentação.

A inserção do cone, mostrada na figura 22, é realizada por meio do comando “cone.criar(100,100)”, no qual os parâmetros passados representam, respectivamente, as coordenadas **x** e **y** da posição onde se deseja posicioná-lo dentro do ambiente virtual. Esse comando segue a mesma lógica do “cart.criar(x, y)”, utilizado para a criação do carrinho no laboratório.

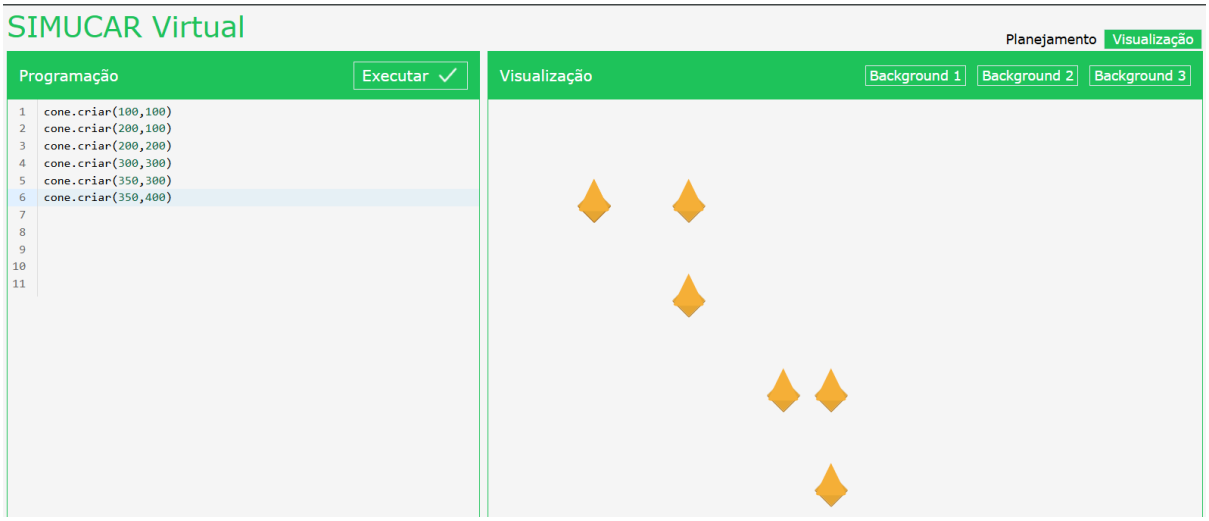
Figura 22 – Cone inserido no circuito



Fonte: autoria própria (2025)

Caso seja necessário adicionar múltiplos cones à simulação, basta modificar os valores das coordenadas x e y e criar novas linhas de código utilizando o comando “cone.criar(x, y)”. Após inserir os comandos desejados, ao clicar em "executar", os cones serão gerados e posicionados conforme especificado, enriquecendo ainda mais a experiência prática dentro do laboratório virtual. Na figura 23, é possível observar um exemplo da inserção de um cone dentro do ambiente simulado.

Figura 23 - Vários cones inseridos no circuito



Fonte: autoria própria (2025)

Essa funcionalidade contribui significativamente para a didática do laboratório, permitindo que os usuários configurem diferentes cenários de teste, recriem circuitos e experimentem diversas situações práticas, aprimorando o aprendizado e a interação com o ambiente virtual.

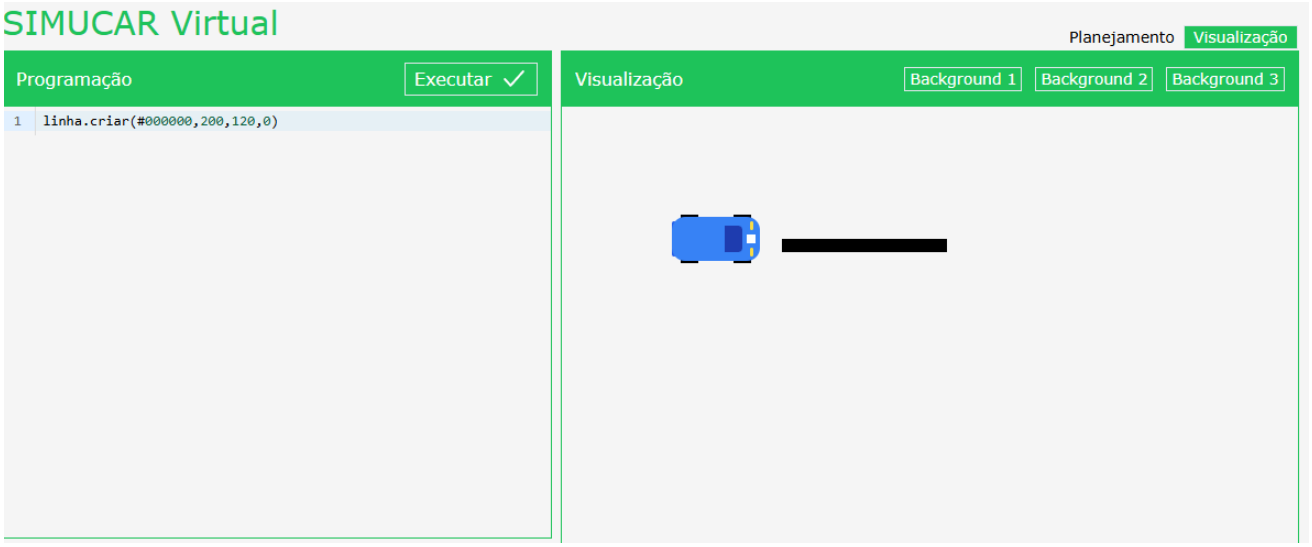
3.5.7 Laço de repetição

Outra funcionalidade relevante do *Simucar* é a implementação de laços de repetição, que permitem a criação de comportamentos automáticos e condicionais para o carrinho simulado. A utilização desse recurso possibilita que o carrinho receba uma condição específica, verificando-a continuamente. Caso a condição seja verdadeira, o *loop* será executado conforme os parâmetros definidos, proporcionando um comportamento dinâmico e ajustável às necessidades do usuário.

A sintaxe para criar um laço de repetição no *Simucar* difere um pouco das funcionalidades previamente discutidas. O comando necessário é “while(linha == #000000){cart.frente(20)}”, onde #000000 representa a cor em formato hexadecimal que se deseja verificar para o funcionamento do loop. Nesse exemplo, o comando “cart.frente(20)” é acionado sempre que a condição definida (detecção da cor especificada) for atendida.

Para que o laço de repetição opere corretamente, é fundamental que o carrinho esteja posicionado sobre a linha de cor indicada, de modo que seu sensor possa detectá-la. O processo para configurar essa condição inicia-se com a criação de uma linha na posição e na cor desejadas dentro do ambiente simulado.

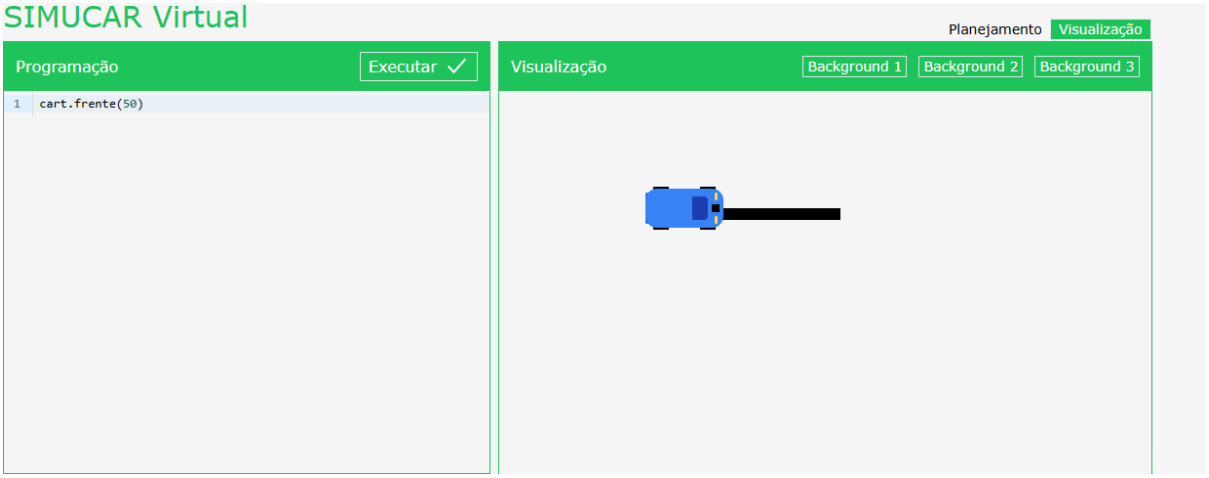
Figura 24 – Linha para laço de repetição inserida



Fonte: autoria própria (2025)

Após a criação da linha, ao escrever e executar o comando “while(linha == #000000){cart.frente(20)}”, é possível observar que o carrinho da figura 24 inicialmente não responderá ao comando. Isso ocorre porque o sensor do carrinho ainda não está sobre a linha, e, portanto, a condição linha == #000000 não é satisfeita. Para fazer com que a condição seja satisfeita, é necessário deslocar o carrinho até a linha desejada, utilizando um comando como “cart.frente(50)”. Esse movimento permitirá que o sensor localizado acima do capô do carrinho detecte a linha, conforme demonstrado na figura 25.

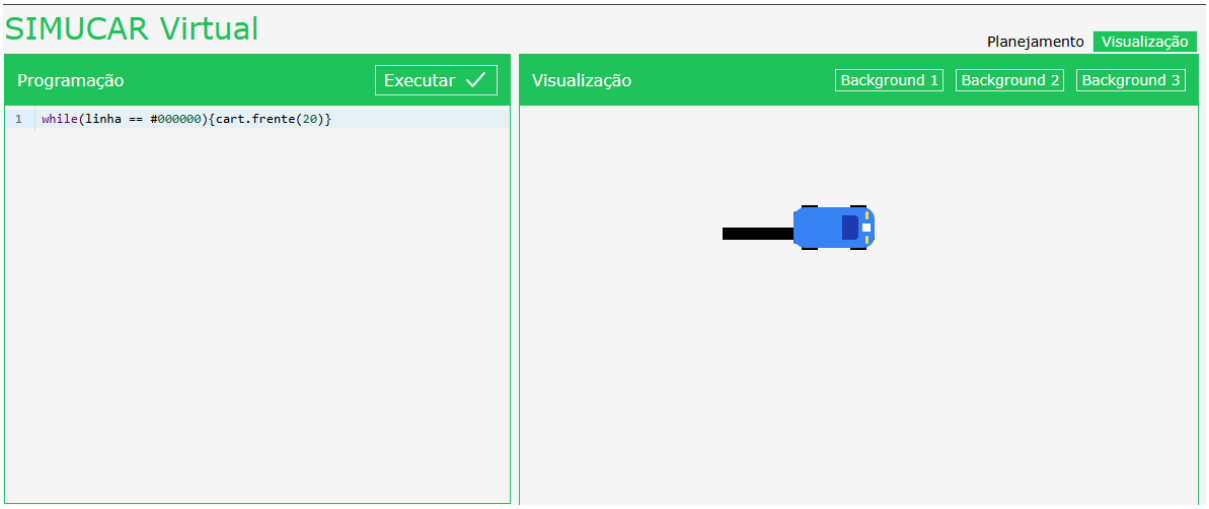
Figura 25 – Carrinho detectando uma linha, com seu detector na mesma cor preta



Fonte: autoria própria (2025)

Uma vez que o carrinho esteja detectando a linha, a execução do comando “while(linha == #000000){cart.frente(20)}” resultará na movimentação contínua do carrinho enquanto a condição se mantiver verdadeira. Na figura 26, pode-se observar o carrinho na sua posição ao final da execução do *loop*.

Figura 26 – Carrinho ao término do laço de repetição



Fonte: autoria própria (2025)

Com isso, essa funcionalidade proporciona uma maior interação com o ambiente virtual e permite a criação de circuitos complexos e desafiadores, enriquecendo o aprendizado e oferecendo aos usuários uma ferramenta eficaz para o desenvolvimento de habilidades práticas em lógica de programação e automação.

3.6 Considerações Finais

O desenvolvimento do *Simucar Virtual* proporcionou a criação de um ambiente de aprendizagem acessível, interativo e prático para o ensino de programação. Durante este capítulo, foram detalhadas as principais funcionalidades do laboratório virtual, demonstrando como ele permite que os alunos experimentem conceitos de programação de maneira visual e intuitiva, facilitando a assimilação do conteúdo.

Uma das grandes vantagens desse laboratório é sua disponibilidade. Como se trata de uma plataforma *web*, ele pode ser utilizado de qualquer lugar com conexão à internet, eliminando barreiras geográficas e permitindo que os estudantes pratiquem programação sem a necessidade de instalações complexas ou equipamentos específicos. Esse aspecto torna o *Simucar Virtual* uma ferramenta extremamente versátil para o ensino de programação, podendo ser integrada a diversos ambientes educacionais, como universidades, escolas técnicas e cursos online.

Além disso, o *Simucar Virtual* foi desenvolvido com foco na leveza e na facilidade de aprendizado. A interface intuitiva e as funcionalidades bem definidas garantem que alunos iniciantes possam interagir com o ambiente de maneira fluida, enquanto usuários mais avançados podem explorar suas capacidades para criar desafios e simulações mais complexas. O uso de tecnologias modernas, como *Vue.js* e *CodeMirror* contribui para a eficiência do sistema, permitindo uma experiência interativa sem comprometer o desempenho.

Portanto, o laboratório virtual desenvolvido neste capítulo representa um passo significativo na busca por soluções inovadoras para o ensino de programação. Ao proporcionar um ambiente acessível, leve e de fácil aprendizado, ele se mostra uma ferramenta valiosa para estudantes que desejam aprimorar suas habilidades de forma prática e dinâmica.

4. CONCLUSÃO

O laboratório virtual *Simucar* representa uma iniciativa para o ensino de programação e robótica, oferecendo um ambiente prático e interativo para os alunos desenvolverem suas habilidades de programação. O seu desenvolvimento permitiu a criação de um laboratório virtual voltado para o ensino de programação, utilizando a simulação de um robô móvel como elemento central da aprendizagem.

Durante o processo de desenvolvimento, foram analisados os requisitos essenciais para a construção do ambiente virtual, resultando na escolha de tecnologias adequadas, como *Vue.js* para a interface, *CodeMirror* para a edição de código e *WBO* para a colaboração e planejamento de algoritmos. A modelagem do sistema seguiu uma arquitetura modular, permitindo maior flexibilidade e escalabilidade para futuras expansões.

Além das funcionalidades essenciais, como a movimentação do robô, detecção de cores e manipulação de obstáculos, o *Simucar Virtual* foi projetado para oferecer uma experiência intuitiva e acessível, facilitando a interação com os conceitos de programação.

Como trabalhos futuros, pretende-se aprimorar a interface, tornando-a ainda mais interativa e intuitiva. Agregado a isso, pode-se acrescentar a inclusão da avaliação do uso do *Simucar*, para entender e avaliar os seus benefícios com os usuários na prática. Além disso, a inclusão de novos comandos disponíveis no laboratório e a personalização do carrinho podem ser um grande atrativo adicional, fazendo com que o estudante aprenda novos conceitos. Outro ponto importante a ser explorado é a integração com um banco de dados, para que fique salvo o progresso do usuário e a implementação de um sistema de *feedback* automático, permitindo que os usuários recebam sugestões e correções em tempo real para melhorar sua aprendizagem.

Dessa forma, este trabalho contribui para o desenvolvimento de soluções inovadoras na área de ensino de programação, demonstrando como o uso de laboratórios virtuais pode tornar o aprendizado mais dinâmico e acessível, além de abrir caminho para novas pesquisas e aprimoramentos no campo da educação computacional.

REFERÊNCIAS

BATISTA, M. C.; FUSINATO, P. A.; BLINI, R. B. Reflexões sobre a importância da experimentação no ensino de física. **Acta Scientiarum. Human and Social Sciences**, Maringá, v. 31, n. 1, p. 43-49, 2009. Disponível em: https://www.redalyc.org/articulo.oa?id=307325328006. Acesso em: 23 fev. 2025.

GOMES, A.; HENRIQUE, J.; MENDES, A. J. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias**, Caparica, v. 1, n. 1, p. 93-103, 2008. Disponível em: https://eft.educom.pt/index.php/eft/article/view/22. Acesso em: 11 nov. 2024.

HUANG, H. M.; RAUCH, U. LIAW, S. S. Investigating learner's attitudes toward virtual reality environments: Based on a constructivist approach. **Computers & Education (Elsevier)**, [*s. l.*], v. 55, n. 3, p. 1171-1182, 2010. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0360131510001466. Acesso em: 6 abr. 2025.

LISBOA, L. **Lara Domótica**: Protótipo de Laboratório Virtual de Domótica. 2023. 88f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Sudoeste da Bahia, Vitória da Conquista, 2023.

LOPES, M. S. S. **Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica**. 2017. 105f. Tese (Doutorado em Engenharia Industrial) – Universidade Federal da Bahia, Salvador, 2017.

MARCELA. **A Importância Da Componentização No Desenvolvimento Front-End.** Disponível em: <https://awari.com.br/a-importancia-da-componentizacao-no-desenvolvimento-front-end-5/>. Acesso em: 12 mar. 2025.

MORAES, R. O significado da experimentação numa abordagem construtivista: O caso do ensino de ciências. *In*: BORGES, R. M. R.; MORAES, R. (org.). **Educação em Ciências nas séries iniciais**. Porto Alegre: Sagra Luzzato, 1998. Vol 1, nº 3

PAPERT, S. **Logo**: computadores e educação. São Paulo: Editora, Brasiliense, 1985.

PEREIRA, C. **Desenvolvimento de um laboratório virtual para criação de histórias interativas**. 2022. 63f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Sudoeste da Bahia, Vitória da Conquista, 2022.

PIMENTEL, M; FILIPPO, D; SANTOS, T. M. - Design science research: pesquisa científica atrelada ao design de artefatos. **"RE@D – Revista de Educação a Distância e eLearning"** [Em linha]. ISSN 2182-4967. Vol. 3, nº 1 (março/abril 2020), p. 37-61

PIMENTEL, Mariano; FILIPPO, Denise; SANTORO, Flávia Maria. Design Science Research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação. In: JAQUES, Patrícia Augustin; PIMENTEL, Mariano; SIQUEIRA; Sean; BITTENCOURT, Ig. (Org.) **Metodologia de Pesquisa Científica em Informática na Educação: Concepção de Pesquisa**. Porto Alegre: SBC, 2020. (Série Metodologia de Pesquisa em Informática na Educação, v. 1) Disponível em: <https://ceie.sbc.org.br/metodologia/livro-1/>.

POTKONJAK, V.; VUKOBRATOVIĆ, M.; JOVANOVIĆ, K.; MEDENICA, M. Virtual Mechatronic/Robotic Laboratory – A Step Further in Distance Learning. **Computers & Education (Elsevier)**, [*s. l.*], v. 55, n. 2, p. 465-475, 2010. Disponível em: https://www.researchgate.net/publication/223244333_Virtual_MechatronicRobotic_laboratory_-_A_step_further_in_distance_learning. Acesso em 7 abr. 2025.

POTKONJAK, V.; JOVANOVIC, K.; PETROVIC, V. M.; HOLLAND, O.; UHOMOIBHI, J. Virtual Ambient for E-Learning in Engineering Sciences. *In*: CONFERENCE OF THE INTERNATIONAL JOURNAL OF ARTS AND SCIENCES, 2013, Valetta, Malta. **Proceedings** [...]. v. 6, n. 1, p. 7-14, 2013.

SANTOS, R. P.; COSTA, H. A. X. Methodologies Analysis and Environment of Teaching for Algorithms, Data Structures and Programming to beginners in Computing. **INFOCOMP Journal of Computer Science**, [*s. l.*], v. 5, n. 1, p. 1-10, 2006. Disponível em: https://www.cos.ufrj.br/~rps/pub/periodicos/2006/INFOCOMP.pdf. Acesso em: 23 jan. 2025.

SOUSA, Gabriel. **Lara core server - API**: implementação do Lara (laboratório em rede de aprendizagem como um serviço). 2022. 58f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Sudoeste da Bahia, Vitória da Conquista, 2022.

RESNICK, M.; MALONEY, J.; MONROY-HERNÁNDEZ, A.; RUSK, N.; EASTMOND, E.; BRENNAN, K.; MILLNER, A.; ROSENBAUM, E.; SILVER, J.; SILVERMAN, B.; KAFAI, Y. Scratch: Programming for All. **Communications of the ACM**, [*s. l.*], v. 52, n. 11, p. 60–67, 2009. Disponível em: https://dl.acm.org/doi/10.1145/1592761.1592779. Acesso em: 7 abr. 2025.

APÊNDICE A – COMANDOS DISPONÍVEIS NO LABORATÓRIO

Comando	Descrição	Parâmetros	Exemplo
cart.Criar(x, y)	Cria o carrinho na posição especificada	**x**: posição no eixo X **y**: posição no eixo Y	**cart.criar(100, 100)**
cart.frente(valor)	Move o carrinho para frente	**valor**: distância a percorrer	**cart.frente(100)**
cart.re(valor)	Move o carrinho para trás	**valor**: distância a percorrer	**cart.re(100)**
cart.rotacionar(graus)	Rotaciona o carrinho em um ângulo específico	**graus**: ângulo em graus	**cart.rotacionar(45)**
linha.criar(cor, x, y, angulo, largura, espessura	Cria uma linha no ambiente	**cor**: código hexadecimal **x**: posição no eixo X **y**: posição no eixo Y **angulo**: inclinação **largura**: comprimento (opcional) **espessura**: espessura (opcional)	**linha.criar("#0000ff", 100,100,0)**
cone.criar(x, y)	Insere um cone (obstáculo) na posição definida.	**x**: posição no eixo X **y**: posição no eixo Y	**cone.criar(200, 150)**

while(linha == cor) {cart.frente(valor) }	Cria um laço de repetição que move o carrinho enquanto ele estiver sobre uma linha de cor específica.	**cor**: código hexadecimal da cor da linha **valor**: distância a percorrer	**while(linha == "#000000") {cart.frente(20)}**

APÊNDICE B – EXEMPLO PARA DEMONSTRAÇÃO COMPLETA

```
cart.criar(50,50)
cone.criar(200,20)
cone.criar(200,120)
cone.criar(250,120)
cone.criar(250,20)
cone.criar(325,20)
cone.criar(325,120)
cone.criar(450,20)
cone.criar(450,120)
cone.criar(450,70)
cone.criar(450,180)
cone.criar(450,240)
cone.criar(325,180)
cone.criar(325,240)
cone.criar(520,240)
cone.criar(325,300)
cone.criar(325,350)
cone.criar(375,350)
cone.criar(425,350)
cone.criar(475,350)
linha.criar(#0000ff, 600,350,90,150,25)
linha.criar(#00ff00, 334,532,0,150,20)
cone.criar(200,520)
cart.frente(100) 2x
cart.frente(70)
cart.rotacionar(45)
```

```
cart.frente(45) 2x

cart.frente(125)

cart.rotacionar(-45)

cart.frente(45) 2x

cart.frente(50) 2x

cart.rotacionar(45)

cart.frente(45) 2x

while(linha == #0000ff) {cart.frente(65)}

cart.rotacionar(45)

cart.frente(45) 2x

cart.frente(20)

while(linha == #00ff00){cart.frente(40)}

cart.frente(20)
```

APÊNDICE C – LINK PARA REPOSITÓRIO NO GITHUB

https://github.com/ThiagoViana07/simucar-virtual