UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO - UFERSA
CURSO DE CIÊNCIA DA COMPUTAÇÃO
Trabalho de Conclusão de Curso (2022).

DISTRIBUILDER: UMA PLATAFORMA DE DESENVOLVIMENTO DE APLICAÇÕES DISTRIBUÍDAS

Thomas Maikon dos Santos e Silva[1] e Paulo Henrique Lopes Silva[2]

Resumo: *A disseminação de conceitos com IoT está presente cada vez mais nos dias atuais devido ao avanço das Tecnologias da Informação e Comunicação (TICs), a utilização de sistemas embarcados viabiliza o uso de pequenos computadores que podem se conectar com a internet e possibilitam seu uso de forma distribuída. Todavia há carência no que diz respeito a ambientes que permitam ao usuário ter proveito das diferentes ações distribuídas, que os dispositivos presentes no meio disponibilizam para realizarem uma ação conjunta. Este trabalho apresenta o desenvolvimento de uma plataforma que possibilite aos usuários leigos ou avançados programar, de forma simplificada e distribuída, as diferentes instruções disponibilizadas pelos dispositivos. Para tanto, foi desenvolvido uma arquitetura que possibilite a transferência de dados entre os dispositivos sem que eles precisem se conhecer, o desenvolvimento dessa plataforma aborda temas como padrões de projeto, sistemas distribuídos e internet das coisas. Para validação do sistema foram feitos testes práticos na plataforma com alunos da UFERSA que apresentaram sugestões e opiniões quanto à ferramenta. A partir da validação chegou-se à conclusão de que a plataforma atingiu o objetivo disponibilizando uma plataforma que é capaz de compor e controlar diferentes dispositivos para trabalharem em conjunto e chegar a diferentes resultados realizando operações simples e mais avançadas.*

Palavras-chave: programação em blocos; programação distribuída; internet das coisas; padrões de projeto.

1 INTRODUÇÃO

Com o avanço das TICs conceitos, tais como computação distribuída emergiram, impondo novas estruturas trazendo consigo pontos positivos e negativos como o compartilhamento de recursos e concorrência, que naturalmente ocorrem no tocante a sistemas operacionais porém agora ampliado para o contexto de sistemas distribuídos, respectivamente (COULOURIS, 2013). Outra área que sofreu significativo avanço foi a de microcontroladores e microprocessadores, tornando-se mais eficientes, menores e mais acessíveis, viabilizando sua utilização nos mais variados contextos.

Toda essa evolução possibilitou que hoje um indivíduo tenha um computador em seu bolso, na geladeira, no microondas e demais quaisquer lugares alguns deles tendo capacidade se conectar à internet conseguindo trocar informações e serem manipulados remotamente, atualmente conhecidos como internet das coisas (IoT) e computação ubíqua (COULOURIS, 2013).

Na última década houve um grande aumento nas áreas de pesquisa e desenvolvimento de sistemas embarcados e IoT devido ao surgimento e avanço de plataformas e placas de prototipagem, que permitem a adição e programação de componentes para sua utilização. Sua aplicabilidade dar-se-á na utilização cotidiana mais simples como acender um LED até conceitos mais avançados como cidades inteligentes, robôs e demais, impactando significativamente a vida dos seres humanos (SILVA, 2017).

Conforme Mayr-Dorn et al. (2021) , a indústria 4.0 faz com o que haja aumento na quantidade de máquinas, robôs e ambientes interconectados. Como resultado da adição de múltiplos dispositivos atuantes no ambiente, o trabalhador necessita se aprimorar em outras áreas, como programação, a quem de sua formação, nesse contexto vê se a necessidade de ferramentas que abstraem a complexidade e proveem o mesmo nível manipulação ou o mais próximo possível quem uma linguagem de programação fornece e que auxiliem-no.

Apesar da existência de ferramentas que auxiliam e abstraem código, sua utilização é limitada, visto que dependem de especialistas de desenvolvimento (SILVA; BURLAMAQUI, 2016).

Portanto, este trabalho apresenta o desenvolvimento de uma plataforma que facilita a programação e a composição de instruções que realizam ações distribuídas em diferentes dispositivos conectados em rede utilizando estratégias de programação em bloco, por demonstrar ser mais efetiva e amigável para leigos, e padrões de projeto, trazendo uma base sólida e inteligente no desenvolvimento ao mesmo tempo que viabiliza a manutenção do código para futuras melhorias. A seguir, serão apresentadas as motivações para o desenvolvimento do trabalho, levando-se em consideração os pontos positivos e negativos das linguagens de programação bem como a programação de dispositivos distribuídos.

[1] *Discente do Bacharelado em Ciência da Computação da UFERSA - Campus Mossoró*
[2] *Professor Adjunto IV do Departamento de Computação da UFERSA - Campus Mossoró*

2 MOTIVAÇÃO

O ensino de linguagens de programação concentra-se na capacidade de entender e solucionar problemas de forma simples utilizando a programação, todavia existe baixa taxa de sucesso em seu ensino (GOMES; HENRIQUES; MENDES, 2008). Conforme Ko, Myers e Aung (2004), são apresentadas as "barreiras de aprendizagem" as quais representam dificuldades apresentadas por iniciantes na programação, sendo elas: barreiras de design, seleção, coordenação, uso, compreensão, e informação.

- *Design*: dificuldade cognitiva para saber solucionar um problema.
- Seleção: dificuldade em saber quais instruções devem ser utilizadas para a solução da problemática.
- Coordenação: dificuldade na composição de diferentes operações e instruções para uma solução.
- Uso: dificuldade em saber utilizar a sintaxe.
- Compreensão: falso ou pouco entendimento do funcionamento de funções, instruções e bibliotecas.
- Informação: dificuldade na obtenção de informação sobre o porquê algo não funcionou.

Tais dificuldades se apresentam, inclusive para os que já estão inseridos no contexto, devido ao crescimento e surgimento de muitas tecnologias, paradigmas e linguagens.

Em contrapartida, o avanço dessas tecnologias, linguagens, paradigmas e técnicas auxiliam diariamente o desenvolvimento dos programadores. As *IDEs* (*Integrated Development Environment*) como Visual Studio Code (MICROSOFT, 2022), IntelliJ (JETBRAINS, 2022) e etc fornecem funcionalidades que auxiliam os desenvolvedores na identificação de erros de sintaxe, *bugs* com o depurador e aceleram no processo de programação com o *autocomplete*. Paradigmas como a orientação a objetos viabilizaram melhor qualidade software, uma vez que com o decorrer dos anos padrões que se repetiam foram identificados no desenvolvimento de software, e em sua identificação foram aperfeiçoados ao longo dos anos tornando-os conhecidos como padrões de projeto (GAMMA et al, 2000). Surgimento de livros como *Clean Code* e *Clean Architecture* que descrevem boas práticas para serem seguidas no desenvolvimento de software (MARTIN, 2009, 2019) e comunidades como *Stack Overflow* que ajuda seus membros a entender e solucionar problemas diversos sobre computação (STACK OVERFLOW, 2022).

Toda essa gama de recursos: padrões, práticas, comunidades e ferramentas, apesar de auxiliar significativamente, demanda tempo, prática e entendimento de conceitos mais avançados do que os vistos na introdução a programação, que é a lógica de programação.

Em comparação às linguagens convencionais (que utilizam formato de texto), as linguagens de programação em blocos são consideradas mais simples e fáceis para o entendimento da programação visto que elas reduzem a quantidade de carga cognitiva, erros básicos e facilitam no aprendizado do vocabulário (BAU et al, 2017).

Uma das ferramentas mais populares e disseminadas para introdução de pessoas na programação é o scratch (WEINTROP, 2019), ela possui uma interface gráfica que permite a utilização de instruções como declaração de variáveis, funções, iterações e condicionais que são comumente disponibilizados em linguagens de programação, em formato de texto, como Go (GOOGLE, 2022), Java (ORACLE, 2022a) e demais. O Scratch utiliza o formato de blocos de forma tal que possam ser aninhados e executar tal qual uma linguagem em formato de texto, nos estudos apresentados por Bau et al. (2017) estudantes que interagiram com os dois formatos, texto e blocos, obtiveram melhor desempenho em seu aprendizado.

2.1 Programação distribuída

Plataformas de prototipagem como arduino, permitem a programação e a adição de componentes de forma simplificada, todavia geralmente devido a forma como são programados e/ou montados há a necessidade de reprogramar e/ou alterar os componentes presentes.

Mesmo com a utilização de comunicação serial síncrona entre os dispositivos presente na arquitetura implementada para o sistema o modelo continua realizando causa de dependência entre os componentes, tornando-os inflexíveis mediante adversidades.

Em níveis mais avançados da programação são encontrados conceitos como programação distribuída, que possibilitam comunicação e o compartilhamento de recursos (COULOURIS, 2013). A utilização de arquiteturas distribuídas possibilita que cada componente tenha baixa granularidade e que unidos possam realizar atividades complexas. Tendo isto em vista, existe uma necessidade de estudo sobre modelos de sistema e arquiteturas distribuídas com o objetivo de listar e implementar uma solução que aborda, dentre outros aspectos, suas limitações de hardware e software.

No contexto de programação de times de robôs (SILVA, 2017) aponta trabalhos, assim como o próprio, que implementam soluções para o uso de estratégias distribuídas e possibilitar que robôs heterogêneos consigam unir esforços e entregar um resultado, todavia elas carecem de uma interface amigável e de fácil utilização para com o usuário leigo.

3 MÉTODO

Diante dos temas e desafios mencionados, este trabalho visa ampliar os horizontes apresentados por (SILVA, 2017) propondo uma plataforma que permite a usuários finais programar ou controlar dispositivos distribuídos em um dado ambiente como se comportassem como uma entidade única e bem definida. Para tanto, o desenvolvimento dessa plataforma de software foi dividida em etapas, que são descritas brevemente a seguir:

1. Revisão bibliográfica sobre modelos e arquiteturas de sistemas distribuídos para a programação de sistemas heterogêneos.
2. Revisão bibliográfica sobre placas de prototipagem, envolvendo suas características e aplicações.
3. Seleção de uma arquitetura de sistema para o desenvolvimento da plataforma, bem como as placas de prototipagem que foram interconectadas a ela.
4. Desenvolvimento da plataforma DistriBuilder:
    a. Seleção de tecnologias de programação.
    b. Definição de componentes do sistema:
        i. Utilização de Banco de dados: Serve para armazenar/cadastrar novas instruções de ordem simples e compostas, além de realizar a associação ao respectivo dispositivo.
        ii. Servidor *Back-End*: responsável por executar as diferentes solicitações enviadas pela aplicação *Front-End*, que incluem salvar, executar e agendar instruções.
        iii. Aplicação Front-End: Responsável por demonstrar as diferentes instruções e comandos presentes, assim como enviar solicitações para o servidor Back-End para compor novas instruções, agendar e/ouou executar as presentes na fila.
    c. Definição de tipos de instruções:
        i. Instruções de ordem: Representam as diferentes operações que os sistemas distribuídos cadastrados podem executar.
        ii. Instruções de comando: Os quais representam operações de loop ou condicionais.
5. Validação da plataforma DistriBuilder.

A seção seguinte apresenta os resultados do desenvolvimento da plataforma, com uma descrição mais detalhada sobre cada etapa proposta.

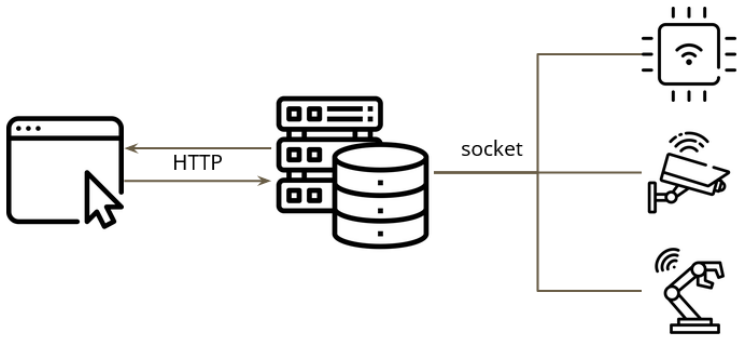
3.1 Arquitetura

Para desenvolvimento da plataforma a arquitetura utilizada, entre o cliente e o servidor, é modelo API (Application Programming Interface) RESTful, o qual permite maior flexibilidade, independência e escalabilidade (AMAZON WEB SERVICE, 2022). O formato para troca de dados escolhido foi o JSON, devido sua simplicidade na leitura e escrita tanto para usuário quanto para o software analisar (ORACLE, 2022b). A comunicação entre o servidor e os dispositivos se dá via socket, por permitir uma comunicação bidirecional entre os envolvidos.

As ferramentas utilizadas para desenvolvimento da página web foi JavaScript com a biblioteca do React (REACT, 2022) e os *frameworks* NextJS (NEXTJS, 2022) e Bootstrap (BOOTSTRAP, 2022), a página web serve para compor e enviar os blocos a serem executados. O servidor, desenvolvido em Java e o *framework Spring* (SPRING, 2022a), é responsável por receber as solicitações do cliente (aplicação web) e iniciar novas comunicações com outros dispositivos, também ao lado do servidor foi escolhido o Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL (POSTGRESQL, 2022) devido a capacidade do modelo relacional possibilitar a construção de relações entre entidades presentes além de novos relacionamentos.

Nas subseções são apresentadas as ferramentas utilizadas para o desenvolvimento da plataforma, bem como foram utilizadas para implementar a arquitetura proposta.

Figura 1. Arquitetura da plataforma desenvolvida.

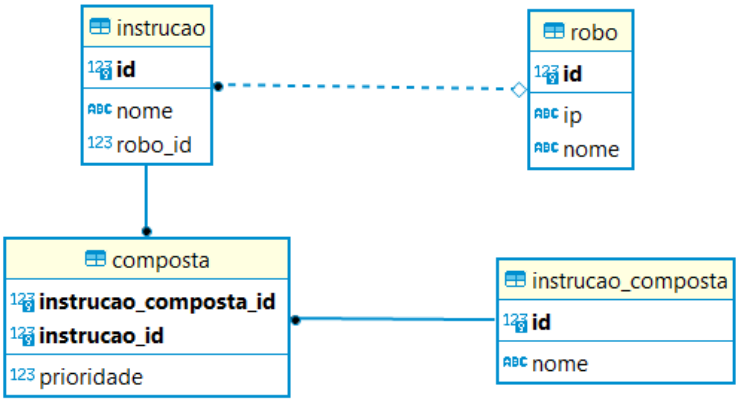


Fonte: Imagem criada pelo autor a partir de FREEPIK COMPANY S.L e DRAW.IO (2022).

3.2 Banco de dados

A modelagem do banco, como demonstrada pela Figura 2, exibe os relacionamentos realizados necessários para associar uma instrução a um dispositivo e gerar uma nova instrução a partir da associação múltiplas instruções (instrução composta) independente do tipo do dispositivo ao qual elas estão relacionadas.

Figura 2. Modelo Entidade Relacionamento entre os dispositivos, instruções e suas composições.



Fonte: DBEAVER (2022).

No banco as entidades `robo` e `instrucao` representam o dispositivo e a instrução do tipo ordem respectivamente e estão associadas em formato "1 para n", permitindo que diferentes tipos de instruções de ordem possam ser referentes a um único tipo de dispositivo.

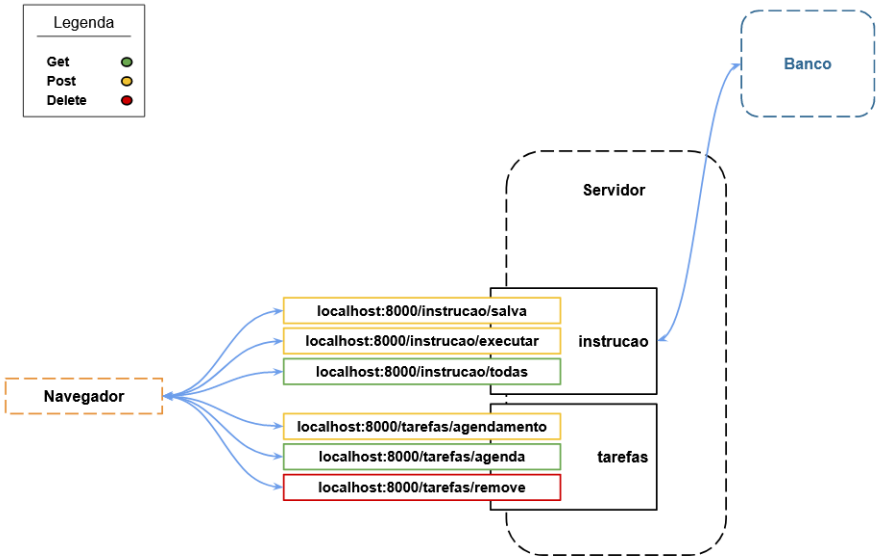
As entidades `composta` e `instrucao_composta` servem para realizar a associação (composição) de múltiplas instruções, na qual a primeira serve para armazenar as instruções necessárias, na devida ordem, já a segunda serve para demonstrar e associar as diferentes instruções que a compõem.

3.3 Servidor

O servidor é responsável por receber as solicitações do cliente e iniciar novas comunicações com outros dispositivos utilizando protocolo HTTP e socket respectivamente.

A linguagem Java fornece uma base sólida e viável para o desenvolvimento do projeto, uma vez que ela utiliza o paradigma orientado a objetos e possui uma ampla variedade de bibliotecas nativas. O Spring possui especificidades que auxiliaram no desenvolvimento da plataforma. A utilização de *notations* como *AutoWired*, *Component* e *Bean* fazem com o que classes sejam previamente instanciadas, durante a inicialização do projeto, além de outras bibliotecas, classes e interfaces fornecidas pelo mesmo.

Figura 3. Disponibilização de endpoints do servidor para a aplicação web.



Fonte: Imagem criada pelo autor a partir de GOOGLE DOCS (2022).

O servidor disponibiliza múltiplos *endpoints* para para as diferentes requisições realizadas pela aplicação web, como demonstrado na Figura 3. Estes endpoints utilizam métodos do protocolo *HTTP* como *get*, *post* e *delete* para listar/executar instruções, listar/criar/executar instruções compostas e agendar/remover tarefas.

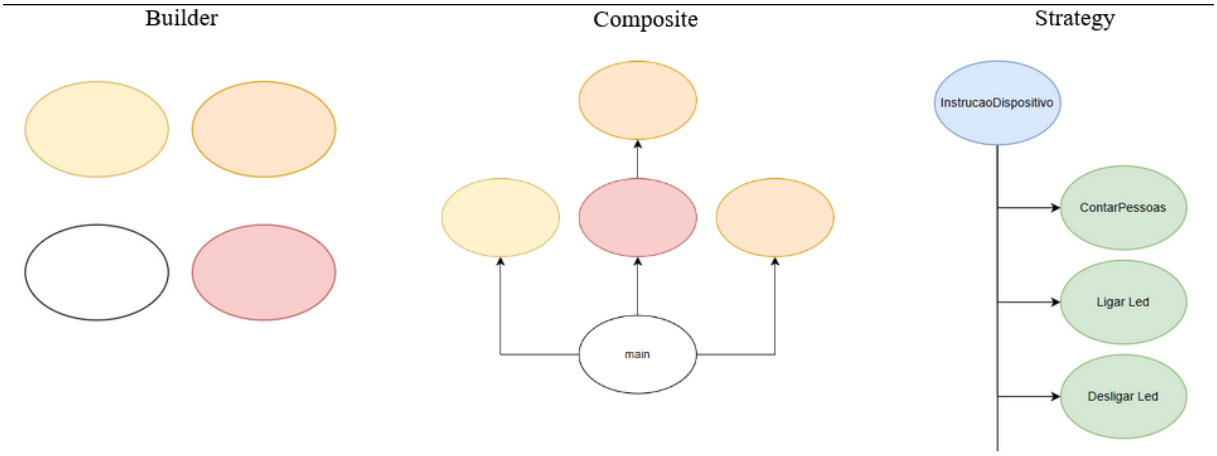
Em especial, o endpoint de `instrucao` possui conexão com o banco de dados, possibilitando salvar novas instruções ou executar as enviadas. Como demonstrado na Figura 2, a entidade instrução, contém o atributo nome o qual serve para identificar o objeto responsável por executar a instrução do dispositivo cuja entidade se refere no Spring.

Para realizar o gerenciamento dos objetos o Spring utiliza a estrutura de inversão de controle (do ingles *Inversion of Control - IoC*), a interface *BeanFactory* é responsável por prover configurações avançadas capazes de gerenciar os objetos presentes (SPRING, 2022b), por ser sua sub-interface a *ApplicationContext* contém todas as funcionalidades necessárias e a partir dela é possível utilizar métodos para receber os objetos manipulados pelo *framework* a partir do nome previamente definido pelo desenvolvedor.

Ainda no lado do servidor, foram implementados três padrões de projetos para possibilitar o recebimento, a interpretação e execução das instruções. O *Strategy*, padrão comportamental, permite que haja variação no comportamento de uma mesma instrução (método), reduzindo a quantidade de comandos e condicionais no código. Os padrões comportamentais se preocupam com algoritmos e a atribuição de responsabilidades, afastando o foco do fluxo de controle para permitir que o desenvolvedor concentre-se somente na maneira como os objetos são interconectados (GAMMA et al, 2000).

Ainda no lado do servidor, foram implementados três padrões de projeto para interpretar os dados vindos do *front* e ter o comportamento tal qual o usuário enviar através da ordem das instruções Figura 4.

Figura 4. Padrões de projeto utilizados no servidor para o recebimento dos dados e execução das instruções.



Fonte: Imagem criada pelo autor usando DRAW.IO (2022).

O primeiro padrão, *builder*, serve para identificar e construir os objetos a partir dos dados recebidos. O padrão *composite* atua juntamente com o *builder*, devido sua característica de atuar como uma estrutura do tipo árvore é possível inserir instruções dentro de outras de forma que seja possível ter controle no comportamento delas. Por fim, o padrão *strategy* serve como uma interface em comum que provê de o método, *run*, para que nele seja implementado a forma como se deseja realizar a comunicação com o dispositivo, à instrução propriamente dita.

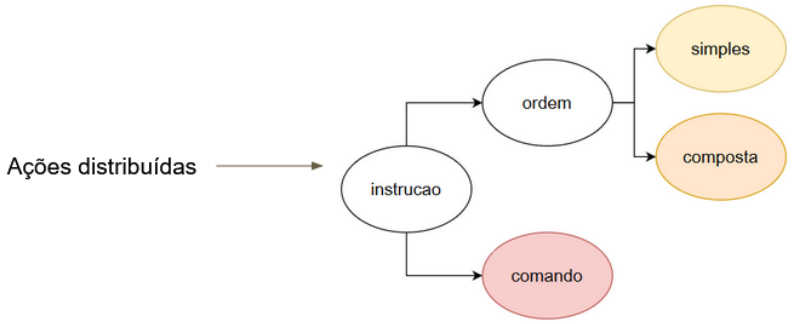
A partir da junção dos três padrões foi possível implementar uma estrutura capaz interpretar e gerar uma árvore, que ao ser percorrida gera o comportamento descrito pelo usuário a partir da sequência em que as instruções estão presentes.

3.3.1 Execução de instruções do tipo ordem e comando

As ações distribuídas são utilizadas no contexto desse trabalho como instruções que podem ser utilizadas para comunicação com o(s) dispositivo(s) sequencialmente, não só mas também podem ser utilizadas como ações de comando. A figura 4 demonstra a hierarquia dos diferentes tipos de instruções que a plataforma disponibiliza.

As instruções de comando são: condicional e iteração, a primeira serve para validar se o valor recebido por ela atende ao requisito necessário para executar suas sub-instruções, podendo ser ignorada quando a condição não for satisfeita, a segunda possui comportamento igual ao de um laço de repetição cujas iterações baseiam-se na quantidade de que é informada pelo usuário, esse tipo de instrução ocorre em tempo de execução, ou seja, suas classes são instanciadas sempre que uma nova requisição de execução ou agendamento for feita.

Figura 5. Os diferentes tipos de instruções e como elas funcionam.



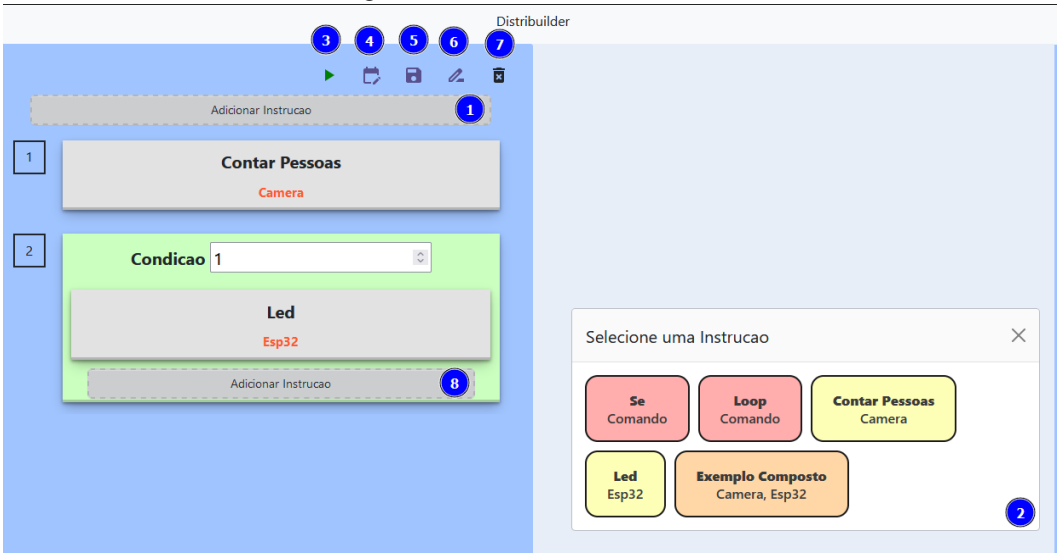
Fonte: Imagem criada pelo autor usando DRAW.IO (2022).

As instruções de ordem são classificadas como simples e composta as do tipo simples são chamadas diretas do método *run* do objeto que o invoca, nas instruções compostas a busca feita no banco utiliza a relação entre o id da instrução composta presente na tabela instrução composta com a tabela composta, a partir dos valores recebidos as instruções são identificadas e executadas sequencialmente passando os resultados obtidos de forma sequencial.

3.4 Aplicação web

A forma como o usuário se comunica com o servidor principal deve ser de forma amigável e intuitiva, visto que exige um certo grau de entendimento de lógica para compreender o *workflow* da sua fila de execução.

Figura 6. Interface Web do usuário.



Fonte: Autoria própria (2022).

As operações disponibilizadas pela aplicação web são descritas nas próximas subseções, assim como o envio dos dados e sua execução. As instruções são separadas por cores contendo abaixo de seu nome o(s) dispositivo(s) . As vermelhas representam instruções de comando, portanto abaixo delas informa que elas são do tipo comando, as amarelas são instruções simples contendo o nome da instrução e o respectivo dispositivo e as laranjas representam as compostas, contendo a sequência dos dispositivos que a executarão.

3.4.1 Execução das instruções

Uma vez selecionada a(s) instrução(ões) desejadas, uma requisição é enviada contendo o aninhamento das instruções presentes na fila de execução

Para realizar o envio das instruções é necessário clicar na opção de enviar (Figura 6-3) e os dados presentes serão enviados em formato JSON para o servidor, que recebe os dados aninhados e os adapta para para serem executados como discutido nas últimas subseções.

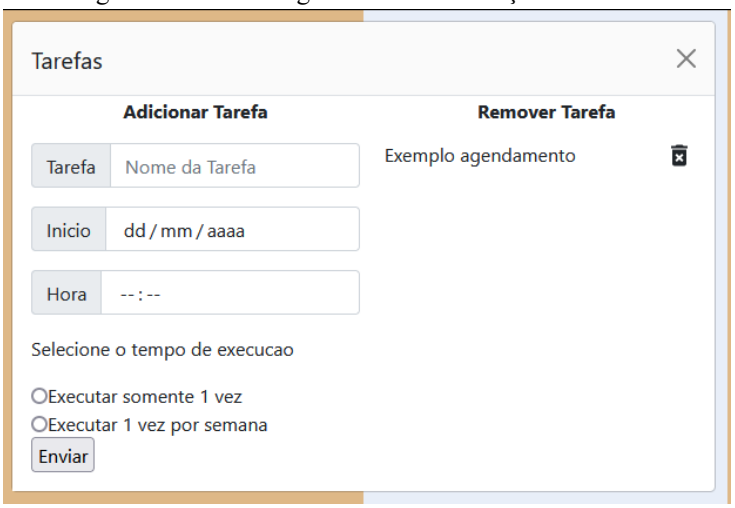
A adição de instruções na fila de execução é feita a partir da abertura do modal de instruções no componente que se deseja adicionar à instrução, podendo ser diretamente na instrução principal (fila) (Figura 6-1) como em sub-instruções(Figura 6-8).

Nas instruções de tipo comando é possível adicionar valores que são utilizados pelo servidor para determinar se suas instruções filhas executarão ou não, no caso do comando Se, ou quantas vezes as filhas executarão, no caso do comando *Loop*.

3.4.2 Agendar instruções

Com as instruções presentes na fila, ao selecionar a opção exibe na Figura 6-4, um modal é aberto solicitando os dados necessários para realizar o agendamento da tarefa responsável por executar as instruções na fila.

Figura 7. Modal de agendamento e remoção de tarefas.



Fonte: Autoria própria (2022).

Com o modal aberto, para efetuar o agendamento das instruções presentes na fila de execução todos os campos presentes são obrigatórios para que seja efetuada a requisição para o servidor e o agendamento realizado. No modal também é possível visualizar a existência de alguma tarefa previamente agendada, a qual é demonstrada pelo seu nome e um botão que permite a sua remoção.

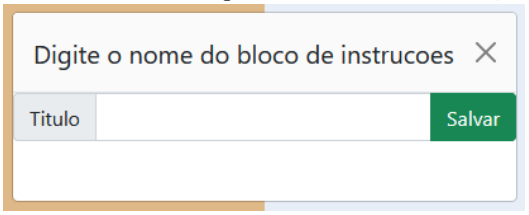
Esta versão do Distribuilder suporta o agendamento de uma instrução por vez, visto que o agendamento de muitas *tasks* gera a necessidade de implementação de sistemas como *threadpool* para o gerenciamento das *threads,* caso contrário a máquina em que o sistema estiver operando começa a ter problemas como concorrência por recursos, não contemplando o escopo deste trabalho até então.

No lado do servidor o agendamento se dá por meio da biblioteca *CRON* e *TaskScheduler* fornecidos pelo *framework* Spring. O primeiro serve como um gatilho com o tempo sendo determinado pelo usuário para ser acionado, já o segundo serve para executar uma *thread*, a qual nesse contexto contém as instruções a serem executadas.

3.4.3 Salvar composição de instruções

Para salvar uma nova instrução é necessário adicionar um título; para isso é necessário selecionar a opção editar título (Figura 6-6) para abrir um modal que permite a adição de um novo título.

Figura 8. Modal de editar título para cadastrar uma instrução composta.



Fonte: Autoria própria (2022).

Após inserir o título e salvá-lo é necessário clicar no botão de salvar (Figura 6-5) e uma requisição será enviada para o servidor.

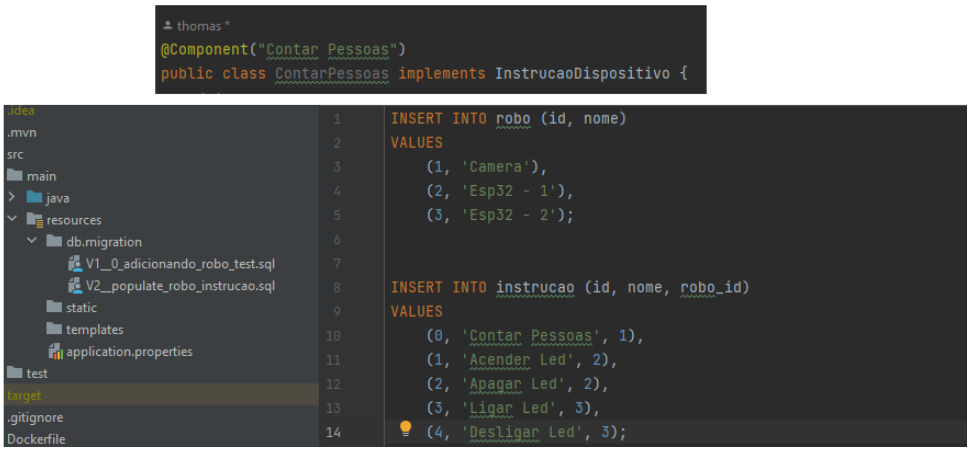
Somente é possível salvar uma instrução, se e somente se, todas as instruções presentes na fila forem do tipo ordem, juntamente do nome que a representará. O servidor coleta e as armazena no banco de dados fazendo os devidos relacionamentos como demonstrado na Figura 2.

3.5 Adição de dispositivos e instruções

Vale ressaltar que para realizar a adição de novos dispositivos e instruções é necessário um entendimento prévio sobre o *spring*, *hibernate* (HIBERNATE, 2022) e *flyway* (RED GATE SOFTWARE LTD, 2022) para melhor entendimento do que será descrito a seguir.

Para adicionar novos dispositivos e instruções na plataforma é necessário criar uma classe que implemente a interface `InstrucaoDispositivo` e que acima dela tenha a notação `@Componente` contendo um nome referente a como ele será chamado desde que não seja igual aos outros já presentes, dentro do método herdado da interface é onde deve ser implementado o código que realizará a comunicação com o dispositivo. Após criar a classe é necessário que na pasta de *migrations* seja adicionado um novo arquivo seguindo o seguinte padrão: `V(numero)__algumnome.sql`, é imprescindível que o número obedeça a ordem. Dentro do mesmo arquivo deve conter a inserção do dispositivo no banco e posteriormente a inserção da(s) instrução(ões) que o mesmo contém, o nome da instrução deve ser o mesmo nome que foi declarado na *notation* anteriormente durante a criação da classe como demonstrado pela Figura 9.

Figura 9. Adicionando novo dispositivo no servidor e no banco.



Fonte: Código elaborado pelo autor usando a plataforma IntelliJ (JETBRAINS, 2022).

Por fim, a plataforma disponibiliza um *endpoint*, localhost:3000/dispositivos, que demonstra todos os dispositivos presentes no banco bem como a possibilidade da adição/alteração do ip ao respectivo dispositivo. o ip que ele para a adição do ip que o

3.6 Validação do sistema

Para validação da plataforma, os alunos que frequentam o Laboratório de Ciência da Computação (LCC) da Universidade Federal Rural do Semi-Árido (UFERSA) no campus de Mossoró foram convidados a fazer testes de usabilidade no DistriBuilder. Foram registradas as informações, utilizando o método think aloud, quanto ao uso da plataforma durante o uso dos alunos. Ao todo foram registrados seis alunos, sendo quatro do curso de Ciência da Computação (1 deles mestrando) e 2 de Ciência e Tecnologia (C&T).

Para realização dos testes foram utilizados os seguintes dispositivos:

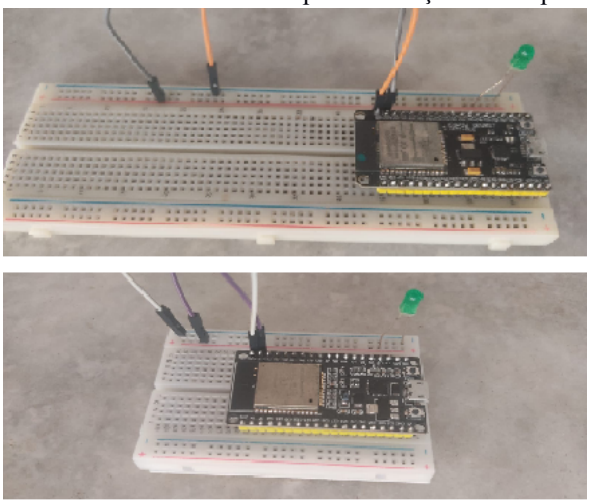
- Dois Esp32-WROOM-32 (Esp-WROOM-32) com processador ESP32-D0WDQ6 trabalhando de 80 a 240 MHz, memória rom de 448KB, placa wifi que implementa o TCP/IP e protocolo Wi-Fi MAC 802.11b/g/n trabalhando na frequência de 2.4 a 2.5GHZ, as placas operam entre 3.0V e 3.6V , dois leds 5mm verde e duas *protoboards* de 400 e 830 pontos Figura 10.
- Dois conectores USB Micro para conectar com uma fonte capaz de alimentar as placas.
- Um notebook com processador Intel® Core i5-10300H de 2.5Ghz com 8GB de memória RAM e placa de vídeo NVIDIA® GTX 1650 com 4GB.

Estes dispositivos foram utilizados devido a carência de *hardware* para implementações de ações mais

elaboradas que tivessem impacto visual e comportamental maior, todavia vale ressaltar que mesmo com essa carência isso não significa que a plataforma seja limitada a eles.

Cada dispositivo Esp32 foi programado utilizando *Arduino Programming Language* (ARDUINO, 2022a) na plataforma do Arduino IDE (ARDUINO, 2022b) cada dispositivo possui nativamente conexão sem fio. Os LEDs, conectados aos respectivos Esps, servem como forma representativa de demonstrar que a instrução percorreu o fluxo de dados partindo da aplicação Web passando pelo servidor e chegando ao dispositivo que a executa ligando ou desligando-o.

Figura 10. Circuitos construídos para utilização dos dispositivos



Fonte: Autoria própria (2022).

Foi implementado um algoritmo em Python para realizar a detecção de objetos através de imagens com o objetivo de demonstrar a troca de dados entre as instruções, para sua implementação foi utilizada a biblioteca de visão computacional OpenCV (OPENCV, 2022a) e a biblioteca YOLO que disponibiliza uma rede neural convolucional pré-treinada para tal (OPENCV, 2022b). O algoritmo de detecção de objetos foi utilizado para melhor exemplificação da transferência de dados entre as instruções.

Com o objetivo de nortear os voluntários foi apresentado o objetivo do trabalho e demonstrados exemplos quanto à utilização da plataforma em três diferentes níveis.

1. No primeiro nível foram demonstradas as funcionalidades da plataforma, no que tange à adição de instruções na fila de execução e sua execução.
2. No segundo nível foi apresentado o formato de composição de instruções, onde no primeiro momento foi demonstrada a execução de múltiplas instruções e no segundo momento sua conversão em uma instrução composta que quando executada obtém-se o mesmo resultado.
3. No último nível foram apresentadas as instruções de comando, o *workflow* que ocorre durante a execução e o agendamento de instruções. Para tanto foi utilizada a instrução de "contar pessoas" e em seguida a instrução de comando "Se", com um valor de condição 1 e uma instrução filha "ligar LED". Sua execução demonstra a passagem dos dados entre a primeira e a segunda instrução e realiza a validação dos dados recebidos comparando-os ao valor da condição determinada.

Apesar da validação ser baseada no teste de usabilidade, é de grande importância que mais tipos de testes, como unidade, integração e demais sejam realizados para garantir ainda mais que o comportamento individual, a integração e *workflow* entre os componentes do sistema e a execução sejam mais passíveis na identificação de falhas.

4 **RESULTADOS**

Posteriormente aos exemplos demonstrados, a plataforma foi disponibilizada para uso individual dos alunos solicitando que eles replicassem as instruções apresentadas ou criar novas a partir do que foi demonstrado, com o objetivo em primeiro momento testar a usabilidade e consequentemente o funcionamento dos componentes. A partir da utilização do DistriBuilder foi relatado um caso de *bug* na aplicação front-end, a qual não enviava corretamente os dados referentes à lista de execução, este por sua vez foi imediatamente resolvido. Durante a utilização da plataforma foram levantados questionamentos no que diz respeito às seguintes características: usabilidade, aparência e sugestões, devido às necessidades da plataforma como demonstrado na

introdução e na motivação deste trabalho . Abaixo são apresentadas as análises feitas a partir do relato feito pelos usuários.

Quadro 1. Análise realizada a partir dos relatos verbais dos voluntários

	Usabilidade	**Aparência**	**Sugestões**
Aluno Mestrando	Apresenta usabilidade simples	Apresenta aparência estática e interface pouco intuitiva	Informar, no momento de adição dos blocos como inserir eles
Aluno 1 de Computação	Apresentou dificuldades ao manusear as instruções.	Aparência não amigável	Possibilitar a remoção de uma instrução por vez, criar novas instruções de comando e/ou melhorias para as presentes como o else no comando “Se”
Aluno 2 de Computação	Apresenta interface simples, o que abstrai a necessidade de entendimento de conceitos mais avançados de programação	Necessita de Melhorias	Utilizar mais dispositivos diversificados para identificar novos casos de uso.
Aluno 3 de Computação	Contém o uso das instruções e agendamento de forma simplificada	Necessita de melhorias no quesito interação humano-computador	Possibilidade de remover uma instrução por vez
Aluno 1 de Ciência e Tecnologia	Apesar de ser uma interface de uso simples é pouco intuitiva	Não apresenta aparência amigável	Demonstrar graficamente o encaixe dos blocos de instrução e utilizar cores mais fortes na tela
Aluno 2 de Ciência e Tecnologia	Apresenta usabilidade simples	Necessita de melhorias quanto a utilização de cores e formas de deixar mais autoexplicativo	Ao cadastrar o nome da nova instrução composta o botão de salvar deve fazer diretamente a solicitação

Fonte: Conteúdo de autoria do autor, tabela criada usando GOOGLE DOCS (2022).

Com a observação do uso do DistriBuilder entre os voluntários, notou-se que o conceito e objetivo da plataforma foram bem compreendidos pois todos optaram por realizar implementações de instruções diferentes das apresentadas. A Figura 11 demonstra algumas das diferentes instruções realizadas pelos alunos.

Figura 11. Exemplos de instruções implementadas pelos alunos que utilizaram a plataforma.

Fonte: Autoria própria (2022).

5 CONCLUSÃO E TRABALHOS FUTUROS

Diante das motivações e objetivos para o desenvolvimento deste trabalho, foi implementado uma plataforma Web que possibilita a programação em blocos de instrução e um servidor responsável por identificar, montar, executar as instruções no formato recebido e possibilitar a comunicação indireta entre os dispositivos. Durante o desenvolvimento da plataforma foram encontrados desafios no que tange aos dispositivos devido ao pouco recurso de componentes de hardware o que impossibilitou realizar mais testes com diferentes interações entre eles e referente ao modelo de armazenamento de dados visto que não foi encontrada uma maneira viável de armazenar instruções de comando juntamente com as de ordem (instrução complexa).

O modelo inicial da plataforma demonstrou bons resultados, cumprindo o objetivo de abstrair complexidades quanto à programação e a programação distribuída, assim como simplicidade no uso das instruções em bloco e ampliando o leque de possibilidades em trabalhos como (SILVA, 2017), não limitando-se a robôs mas agora a quaisquer dispositivos que possuam conexão em rede. Os resultados obtidos por este trabalho apontaram potencial para uso acadêmico e comercial, desde que melhorias e mais testes sejam realizados.

Diante desses resultados, mais pesquisa e desenvolvimento podem ser realizados, tais como a utilização de inteligência artificial para a programação de instruções. Como trabalhos futuros vislumbra-se a implementação de melhorias apresentadas pelos voluntários na etapa de validação, em termos da interface do usuário utilizando técnicas e conceitos de interação humano-computador além de solucionar as dificuldades encontradas com relação ao modelo de banco de dados utilizado analisando propostas de bancos de dados diferentes como o não relacional (NoSQL) para armazenar as instruções complexas.

REFERÊNCIAS

AMAZON WEB SERVICE. **O que é API RESTful ?** Disponível em: https://aws.amazon.com/pt/what-is/restful-api/. Acesso em: 12 out. 2022.

ARDUINO. **Language reference**. Disponível em: https://www.arduino.cc/reference/en/. Acesso em: 19 nov. 2022a.

ARDUINO. **Arduino ide**. Programa. Disponível em: https://www.arduino.cc/en/software. Acesso em: 19 nov. 2022b.

BAU, D. et al. Learnable programming: Blocks and beyond. **Communications of the ACM**, v. 60, n. 6, p. 72–80, 2017.

BOOTSTRAP. Disponível em: https://getbootstrap.com/. Acesso em: 12 nov. 2022.

COULOURIS, G; DOLLIMORE, J; KINDBERG, T; BLAIR, G. **Sistemas distribuídos**: conceitos e projeto. Porto Alegre: Grupo A, 2013. 9788582600542. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788582600542/. Acesso em: 10 jul. 2022.

DBEAVER. Programa. Disponível em: https://dbeaver.io/. Acesso em: 15 nov. 2022.

DRAW.IO. Programa. Disponível em: https://drawio-app.com/. Acesso em: 01 dez. 2022.

FREEPIK COMPANY S.L. **Flaticon**. Disponível em: https://www.flaticon.com/. Acesso em: 01 dez. 2022.

GAMMA, E. et al. **Padrões de projetos**: soluções reutilizáveis de software orientados a objetos. Porto Alegre: Grupo A, 2000. E-book. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788577800469/. Acesso em: 07 set. 2022.

GOMES, A.; HENRIQUES, J.; MENDES, A. J. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, formação & tecnologias**, v. 01, n. 01, p. 93–103, 2008.

GOOGLE. **GO**. Disponível em: https://golang.google.cn/. Acesso em: 22 nov. 2022.

GOOGLE DOCS. Disponível em: https://www.google.com/docs/about/. Acesso em: 01 dez. 2022.

HIBERNATE. Disponível em: https://hibernate.org/orm/. Acesso em: 1 dez. 2022.

JETBRAINS. **IntelliJ IDEA – the Leading Java and Kotlin IDE**. Programa. Disponível em: https://www.jetbrains.com/idea/. Acesso em: 22 nov. 2022.

KO, A. J.; MYERS, B. A.; AUNG, H. H. Six learning barriers in end-user programming systems. *In*: SYMPOSIUM ON VISUAL LANGUAGES - HUMAN CENTRIC COMPUTING, 2004, Roma. **Anais**[...]. Roma: IEEE, 2004.

LUCIDCHART. Disponível em: https://www.lucidchart.com/pages/pt. Acesso em: 19 nov. 2022.

MARTIN, R. **Arquitetura limpa:** O guia do artesão para estrutura e design de software. [*S. l.*: *s. n.*], 2019.

MARTIN, R. **Código limpo**: Habilidades práticas do agile software. [*S. l.*: *s. n.*], 2009.

MAYR-DORN, C. et al. Considerations for using block-based languages for industrial robot programming - a case study. *In*: INTERNATIONAL WORKSHOP ON ROBOTICS SOFTWARE ENGINEERING (ROSE), 3, 2021, Madri. **Anais**[...]. Madri: IEEE/ACM, 2021.

MICROSOFT. **Visual Studio Code**. Programa. Disponível em: https://code.visualstudio.com/. Acesso em: 22 nov. 2022.

NEXTJS. Disponível em: https://nextjs.org/. Acesso em: 12 nov. 2022.

OPENCV. Disponível em: https://opencv.org/about/. Acesso em: 16 nov. 2022a.

OPENCV. Disponível em: https://opencv-tutorial.readthedocs.io/en/latest/yolo/yolo.html. Acesso em: 16 nov. 2022b.

ORACLE. **Java**. Disponível em: https://www.java.com/pt-BR/. Acesso em: 22 nov. 2022a.
ORACLE. **O que é JSON?**. [*S. l.*], 2022. Disponível em: https://www.oracle.com/br/database/what-is-json/. Acesso em: 8 nov. 2022b.

POSTGRESQL. Disponível em: https://www.postgresql.org/about/. Acesso em: 15 nov. 2022.

REACT. **Uma biblioteca JavaScript para criar interfaces de usuário**. Disponível em: https://pt-br.reactjs.org/. Acesso em: 12 nov. 2022.

RED GATE SOFTWARE LTD. **Flyway**. Disponível em: https://flywaydb.org/. Acesso em: 1 dez. 2022.

SILVA, P. H. L. **Uma Arquitetura de sistema para criação, programação e disponibilização de times de robôs para robótica educacional**. 2017. 125 f. Tese (Doutorado) - Curso de Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte, Natal, 2017.

SILVA, P. H. L; BURLAMAQUI, A. M. F. "System for creation, programming and availability of distributed robot teams using collective knowledge," **IEEE Latin America Transactions**, v. 14, n. 10, p. 4392-4401, Oct. 2016, doi: 10.1109/TLA.2016.7786321.

SPRING. Disponível em: https://spring.io/. Acesso em: 10 nov. 2022a.

SPRING. Disponível em:
https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-introduction. Acesso em: 19 nov. 2022b.

STACK OVERFLOW. **Empowering the world to develop technology through collective knowledge**. Disponível em: https://stackoverflow.co/. Acesso em: 10 nov. 2022.

WEINTROP, D. Block-based programming in computer science education. **Communications of the ACM**, v. 62, n. 8, p. 22–25, 2019.