Tri Logic

MANUAL *do usuário*

Guia de uso da plataforma

APRESENTAÇÃO *da plataforma*

O Trilogic é uma plataforma educacional desenvolvida com o propósito de facilitar o ensino e aprendizagem de lógica de programação por meio da Aprendizagem Tecnológica Ativa.

O conteúdo da plataforma é dividido em dez temas, cada um acompanhado por uma modal de início que consiste em uma breve explicação do conteúdo, seguida por, pelo menos, dez exercícios. É uma opção recomendada para alunos que desejam aprender, relembrar ou praticar lógica de programação.

Para acessar a plataforma, basta visitar o seguinte link: https://ambiente-trilogic.canoas.ifrs.edu.br/ e fazer login com seu e-mail*.

Para obter mais informações sobre o Trilogic, visite a página: https://trilogic.canoas.ifrs.edu.br/.

* alunos do Campus Canoas do IFRS devem usar o e-mail institucional.

ALGORITMO e descrição narrativa

Na fase de Algoritmo e Descrição Narrativa, você terá a oportunidade de explorar a arte de criar algoritmos e expressar suas ideias de forma lógica, coerente e compreensível utilizando a descrição narrativa.

Um algoritmo é uma sequência de instruções ou procedimentos que descrevem como alcançar um determinado objetivo. Ele representa um conjunto de passos finitos que devem ser executados de forma sistemática para realizar uma tarefa específica. Enquanto as pessoas se perguntam "o que fazer", os algoritmos respondem as pergunta "como fazer".

A descrição narrativa é uma abordagem que utiliza a linguagem natural para representar algoritmos. Ao utilizar palavras e frases do cotidiano, torna-se mais fácil compreender e comunicar os algoritmos, permitindo que sejam explicados de uma maneira acessível para um público mais amplo.

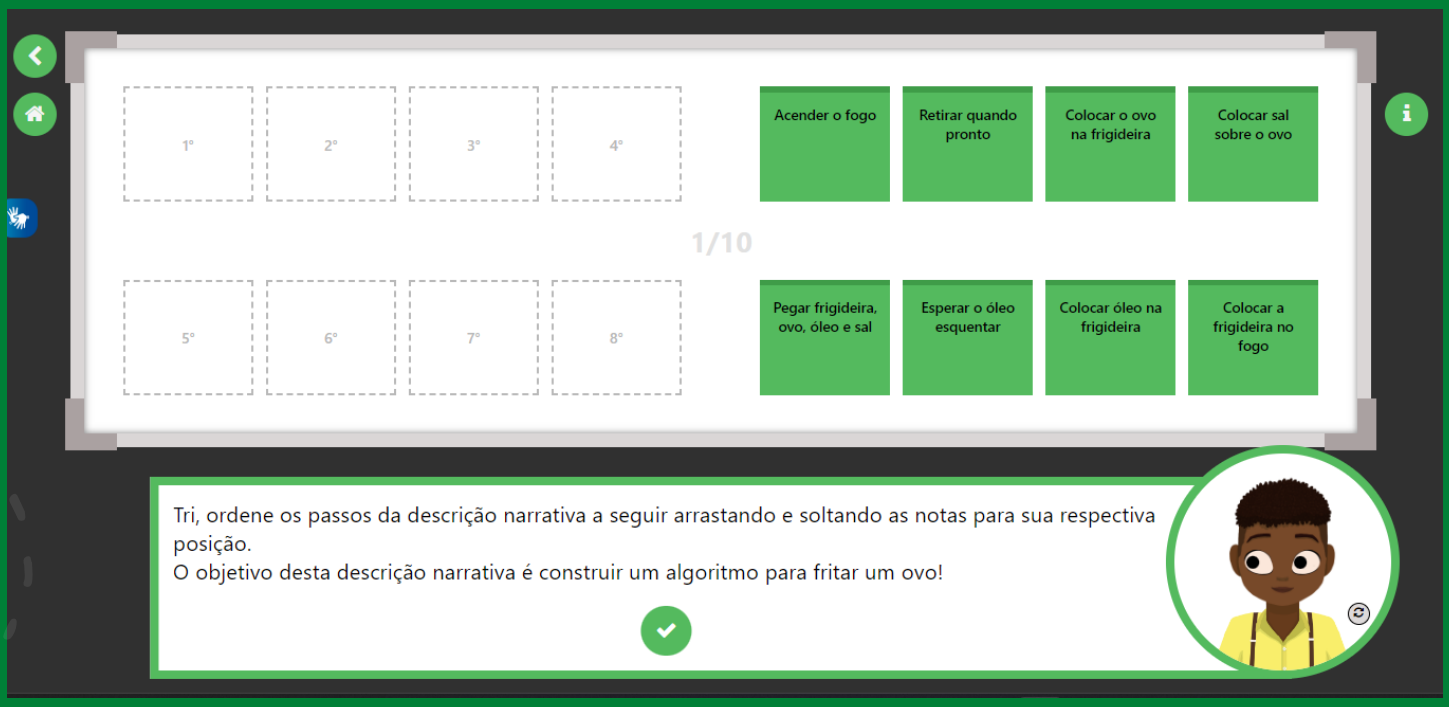
Nos exercícios, adotaremos a mecânica de arrastar e soltar para facilitar a visualização das respostas. Essa mecânica permitirá que você observe a execução das suas respostas de forma semelhante à estruturação de um programa. Em cada etapa, sua missão será completar a tarefa arrastando os elementos correspondentes para as colunas adequadas, seguindo a lógica do algoritmo e utilizando a descrição narrativa.

Através desses exercícios, você desenvolverá suas habilidades de análise, organização e expressão de ideias de maneira estruturada. Além disso, você aprenderá a pensar de forma algorítmica, o que é essencial para solucionar problemas complexos e criar soluções eficientes.

ALGORITMO e descrição narrativa

Aproveite essa fase, dedicando tempo para explorar os exercícios e praticar a criação de algoritmos com descrição narrativa. Lembre-se que essa habilidade é fundamental para a programação e resolução de problemas em diversas áreas.

Divirta-se organizando as notas de forma lógica e compreensível para construir um algoritmo para a atividade 'fritar ovo'.



FLUXOGRAMA

Na fase de Fluxogramas, você terá a oportunidade de explorar e compreender a ferramenta fluxograma, uma representação gráfica que permite visualizar e comunicar de forma clara e concisa as etapas de um processo, sistema ou algoritmo.

Os fluxogramas são amplamente utilizados em diversos setores e disciplinas para descrever processos complexos de maneira compreensível. Ao utilizar diferentes formas geométricas, como retângulos, oval, diamantes, setas conectoras, paralelogramos e círculos, é possível representar as atividades, pontos de tomada de decisão, entrada e saída de dados, fluxo sequencial e conexões entre etapas.

Essa fase tem como objetivo principal desenvolver suas habilidades de organização, análise e comunicação de processos por meio de fluxogramas. Além disso, os fluxogramas são uma ferramenta valiosa para documentar e depurar algoritmos, facilitando a identificação e solução de problemas.

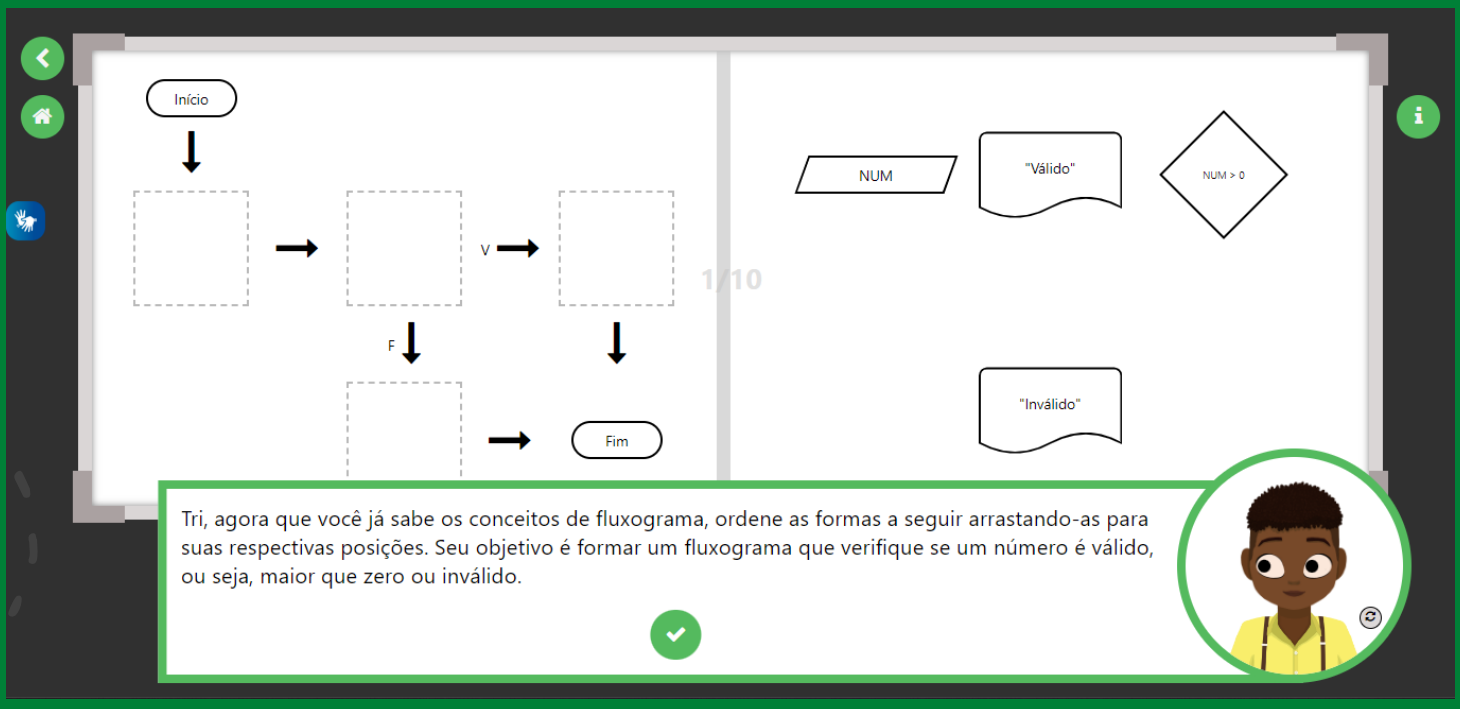
Durante os exercícios, você será desafiado a aplicar seus conhecimentos na criação de fluxogramas para desenvolver problemas específicos. O exercício inicial dessa fase apresentará um problema no qual você deverá organizar as informações de forma adequada em um fluxograma. Essa prática será fundamental para fortalecer sua compreensão e domínio na criação de fluxogramas.

Dominando fluxogramas, você estará preparado para enfrentar desafios mais complexos em programação, pois terá uma visão clara das etapas envolvidas em um processo. Além disso, você estará apto a comunicar suas ideias e soluções de forma eficiente, facilitando a colaboração em equipe e o entendimento de terceiros.

FLUXOGRAMA

Aproveite ao máximo essa fase, dedicando tempo para explorar os exercícios e a criação de fluxogramas. Lembre-se de que essa habilidade é uma ferramenta poderosa para aprimorar sua capacidade de programação e solução de problemas.

Agora você está pronto para utilizar seus conhecimentos em fluxogramas nos exercícios da plataforma.



VARIÁVEL

Uma variável é um local de armazenamento identificado por um nome, utilizado para armazenar um valor em um programa. Cada variável possui um tipo de dado específico que determina o tipo de valor que pode ser armazenado nela. Na linguagem de programação Portugol, existem cinco tipos de dados principais: inteiro, real, caracter, cadeia e lógico.

- O tipo "inteiro" é utilizado para armazenar números inteiros, como 1, 2, 3, etc.
- O tipo "real" é utilizado para armazenar números com casas decimais, como 1.4, 1.5, 1.6, etc.
- O tipo "caracter" é utilizado para armazenar um único caracter, como 'a', 'b', 'c', etc.
- O tipo "cadeia" é utilizado para armazenar um conjunto de caracteres que formam palavras ou frases, como 'Trilogic'.
- Por fim, o tipo "lógico" é utilizado para armazenar valores verdadeiros ou falsos.

Ao declarar uma variável em Portugol, é necessário especificar seu tipo e nome. Por exemplo, para declarar uma variável inteira chamada "idade", utilizamos a seguinte sintaxe:

```
inteiro idade
```

Após a declaração, podemos atribuir um valor específico a ela:

```
idade = 15
```

Dessa forma, a variável "idade" armazena o valor 15. Essa informação pode ser utilizada posteriormente para realizar cálculos, tomar decisões ou exibir resultados dentro do programa.

VARIÁVEL

A fase de Variáveis é uma etapa essencial no processo de programação, onde as variáveis desempenham um papel fundamental. As variáveis permitem armazenar temporariamente e manipular dados, possibilitando que os programas sejam flexíveis e capazes de lidar com diferentes valores e situações.

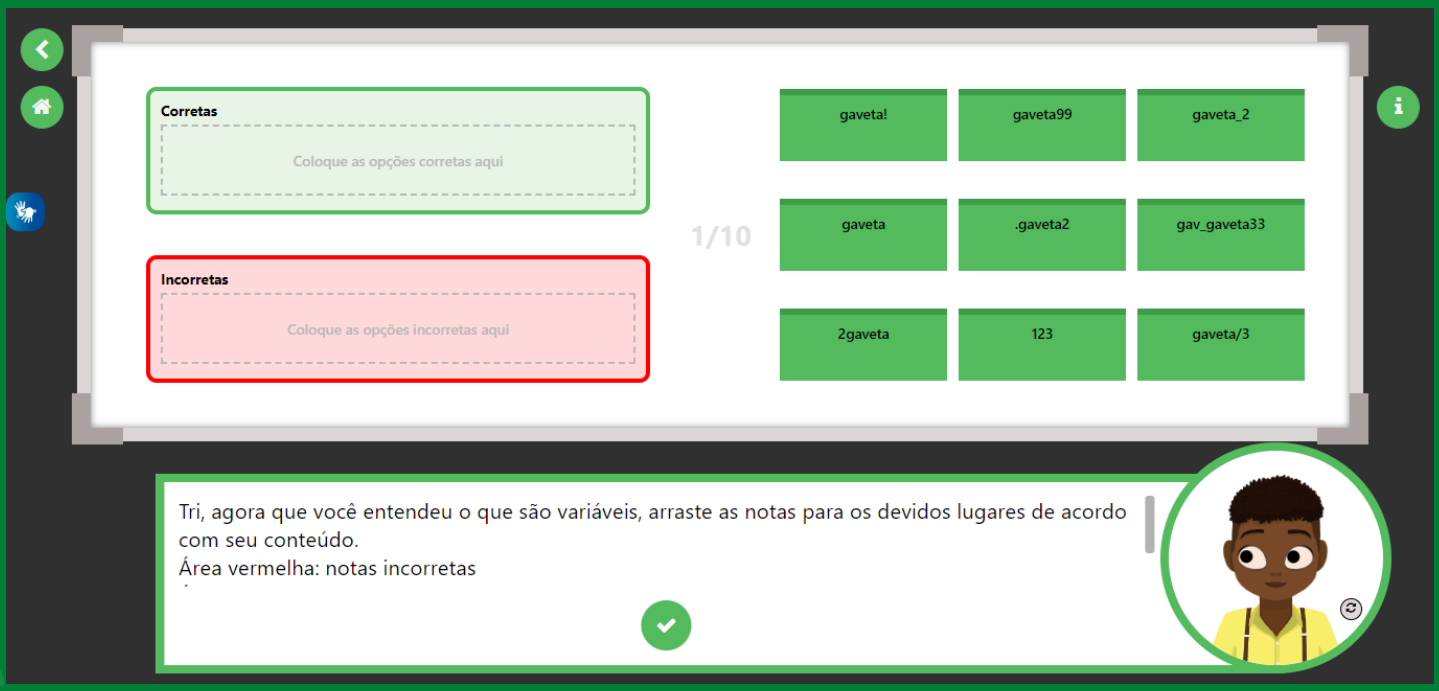
Durante essa fase, você aprenderá a nomear corretamente as variáveis e entenderá as regras gerais para essa prática. Aqui estão algumas diretrizes importantes:

1. Nome da variável: O nome da variável deve começar com uma letra do alfabeto. Evite o uso de caracteres especiais, pois alguns deles têm significados específicos em determinadas linguagens de programação, podendo causar erros ou comportamentos inesperados.
2. Espaços em branco: Evite o uso de espaços no nome das variáveis. Se for necessário separar palavras , recomenda-se utilizar o padrão "cameCase" ou underscores(_). Por exemplo, "nomeCompleto" ou "nome_completo".
3. Letras maiúsculas: É recomendado evitar o uso de letras maiúsculas no nome das variáveis.

Durante esta fase, serão apresentados dez exercícios para praticar nomenclatura e a declaração correta de variáveis. Cada exercício desafiará você a identificar a maneira correta de nomear uma variável, declara-la e associar o tipo de dado apropriado.

VARIÁVEL

Na fase de Variáveis da plataforma, você terá a oportunidade de aprimorar seus conhecimentos e habilidades em relação ao uso e manipulação de variáveis na programação. Essa fase é especialmente projetada para ajudá-lo a consolidar o aprendizado de toda a matéria, proporcionando exercícios de fixação abrangente, como o exercício um da fase de variáveis abaixo.



OPERAÇÕES

A fase de Operações é onde os operadores Aritméticos, Lógicos e Relacionais são explorados. Esses operadores desempenham um papel fundamental na definição das condições dos comandos, como "se então", "para" e "faça enquanto", além de serem utilizados em expressões para atribuições. Nesta seção. detalharemos cada um deles, fornecendo um guia abrangente sobre o seu uso e funcionalidades.

Operadores Aritméticos:
Os operadores Aritméticos são utilizados para representar as operações matemáticas. O computador segue uma ordem de prioridade específica para executar as operações, que inclui Exponencial e/ou Radiação,, Multiplicação e/ou Divisão, Soma e/ou Subtração. É importante observar que essa ordem de prioridade pode ser alterada, inserindo cálculos entre parênteses para indicar quais operações dever ser realizadas primeiro.

Além disso, o operador '+' é usado para concatenar textos, valores inteiros e reais. Já os demais operadores são aplicados apenas a dados inteiros e reais.

Operadores Relacionais:
Os operadores Relacionais são usados para comparar variáveis ou expressões, resultando em um valor lógico (verdadeiro ou falso). Essas comparações são essenciais para estabelecer condições e tomar decisões dentro dos comandos. Os operadores relacionais incluem '>', '<', '>=', <=', '==', e '!='. É importante ressaltar que o operador '==' é o único que pode ser aplicado a todos os tipos de dados.

Tipos de dados e operadores correspondentes:

- Para dados lógicos e literais, apenas '==' e '!=' são aceitos
- Para dados inteiros e reais, todas as operações relacionais podem ser realizadas

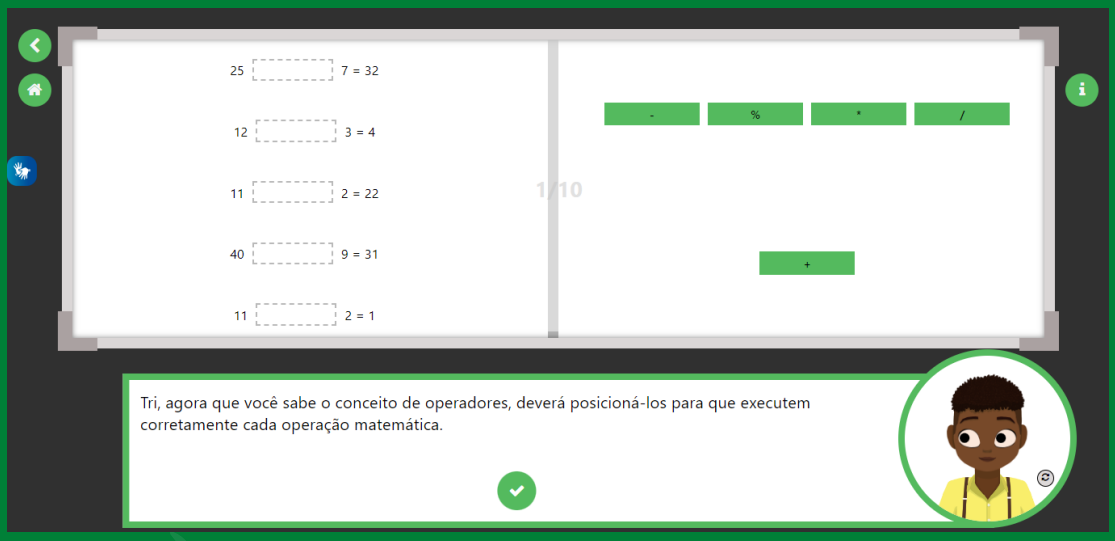
OPERAÇÕES

Operadores Lógicos:
Os operadores Lógicos são utilizados para avaliar expressões lógicas e são cruciais para estabelecer condições mais complexas. Os operadores lógicos incluem 'não', 'e', e 'ou'. O operador 'não' retorna o valor lógico, transformando verdadeiro em falso e vice-versa. O operador 'e' retorna verdadeiro apenas se todas as comparações forem verdadeiras, enquanto o operador 'ou' retorna verdadeiro se pelo menos umas das comparações for verdadeira.

Exemplos:

- No comando "se(media>6) e (frequencia>75) então..." ambas as condições dever ser verdadeiras para que o(s) comando(s) após "então" sejam executados.
- No comando "se(idade>18) ou (amigoDoDono=verdadeiro) então..." apenas uma das condições precisa ser verdadeira para executar o(s) comando(s) após o então
- No comando "se não(nome==\"Andreia\") então..." a condição é invertida: se o nome for "Andreia", retorna falso, caso contrário, retorna verdadeiro.
-

Agora você está pronto para utilizar esses operadores durante suas operações na plataforma.



LEITURA e escrita

Na fase de Leitura e Escrita, você vai explorar os recursos que permitem a comunicação entre o usuário e o programa, permitindo o intercâmbio de informações e dados.

Um recurso importante para essa comunicação é o comando "escreva()", que é responsável por exibir mensagens que o programa deseja emitir ao usuário. A sintaxe desse comando é simples e direta, como demonstra no trecho do código a seguir:

```
1   programa{
2           funcao inicio( ){
3                   escreva("Saudações!\nSeja bem-vindo(a)!")
4           }
5   }
```

Após a execução desse algoritmo, a mensagem "Saudações! Seja bem-vindo(a)!" será exibida no console. A presença do caractere especial "\n", que representa uma quebra de linha, equivale a pressionar a tecla Enter no teclado.

Essa simples adição de comandos de escrita permite ao programador criar uma interação com o usuário, fornecendo informações relevantes, solicitações ou orientações durante a execução do programa.

Nesta fase, você terá a oportunidade de praticar o uso adequado dos comandos de escrita para criar mensagens claras e informativas, tornando sua interação com o usuário mais eficiente e agradável

LEITURA e escrita

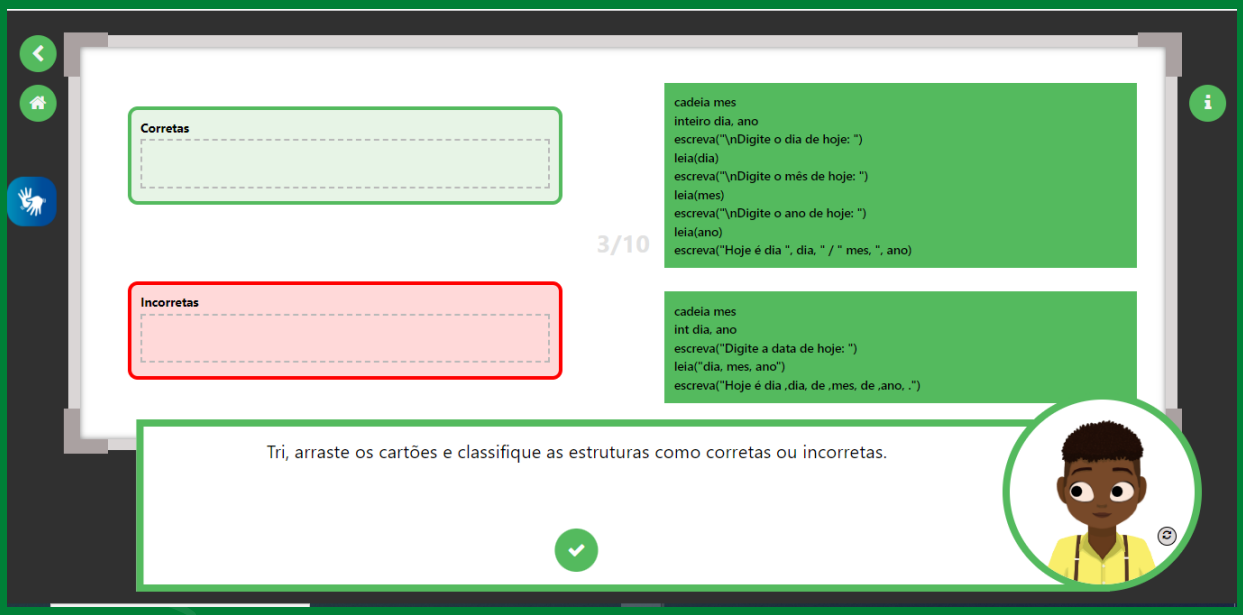
Além do comando "escreva()", temos também o comando "leia()", que é utilizado para capturar dados de entrada fornecidos pelo usuário durante a execução do programa.

```
leia(variavel);
```

Nesse comando, "variavel" representa a variável na qual o valor de entrada será armazenado.

Ao utilizar os comandos "escreva()" e "leia()", você poderá criar uma interação mais dinâmica e personalizada com o usuário, exibindo mensagens, solicitando entrada de dados e utilizando esses valores em cálculos ou tomadas de decisão dentro do programa

Veja mais sobre entrada e saída de dados na fase de Leitura e Escrita da plataforma.



ESTRUTURAS *condicionais*

As Estruturas Condicionais desempenham um papel fundamental na lógica de programação, permitindo que o programa tome decisões com base em determinadas condições. Essas estruturas garantem que o código execute ações diferentes dependendo do valor de uma expressão lógica.

Ao utilizar estruturas condicionais, os programadores podem criar programas mais inteligentes e dinâmicos, capazes de lidar com uma ampla variedade de situações. Elas são amplamente utilizadas em todas as áreas da programação, desde pequenos scripts até grandes sistemas de softwares.

As estruturas condicionais, também chamadas de estruturas de seleção, são responsáveis por deixar ou não que determinadas ações se concretizem, o que ocorre através das condições.

Há uma estrutura de seleção simples que utiliza o se, seguido da condição, e, após, a consequência. Isso quer dizer que, se essa condição for cumprida, então esse comando será executado.

Abaixo segue um exemplo:

```
programa{
    funcao inicio(){
        inteiro N = 1
        se (N == 1){
            escreva (N)
        }
    }
}
```

ESTRUTURAS *condicionais*

Além da Estrutura Condicional Simples, existem outras estruturas condicionais básicas comumente utilizadas na lógica de programação. São elas:

Estrutura condicional composta (if-else): permite que você especifique um bloco de código a ser executado caso a condição seja verdadeira e outro bloco de código a ser executado caso a condição seja falsa.

Estrutura condicional encadeada (if-else-if): permite testar várias condições consecutivas e executar diferentes blocos de código com base em qual condição é verdadeira.

Estrutura condicional aninhada (if aninhado): permite colocar uma estrutura condicional dentro de outra, criando uma hierarquia de condições.

Agora que você adquiriu conhecimentos sobre estruturas condicionais, está na hora de colocá-los à prova por meio dos exercícios disponíveis na plataforma.

ESTRUTURAS de respetição

As Estruturas de Repetição, também conhecidas como laços de repetição, são ferramentas que permitem automatizar cálculos e processos repetitivos. Elas são compostas por uma condição de execução no inicio, no final ou por meio de uma variável de controle.

Ao lidar com tarefas que envolvem uma grande quantidade de dados, como o cálculo da média de notas de uma turma com 30 alunos, seria difícil realizar manualmente esses cálculos. É aí que entram as estruturas de repetição, simplificando o processo.

É essencial definir uma condição que limite o número de vezes que o laço de repetição será executado, caso contrário, os comandos se repetirão infinitamente, resultando em um loop infinito.

Para uma melhor compreensão, vamos exemplificar as estruturas de repetição a seguir:

A estrutura de repetição com teste no início é realizada utilizando a palavra-chave "enquanto". Ela executará os comandos contidos no bloco de repetição enquanto a condição estabelecida pelo programador for verdadeira.

```
programa{
    funcao inicio{
        inteiro cont = 0
        enquanto (cont<=10){
            escreva (cont, " ")
            cont = cont + 1
        }
    }
}
```

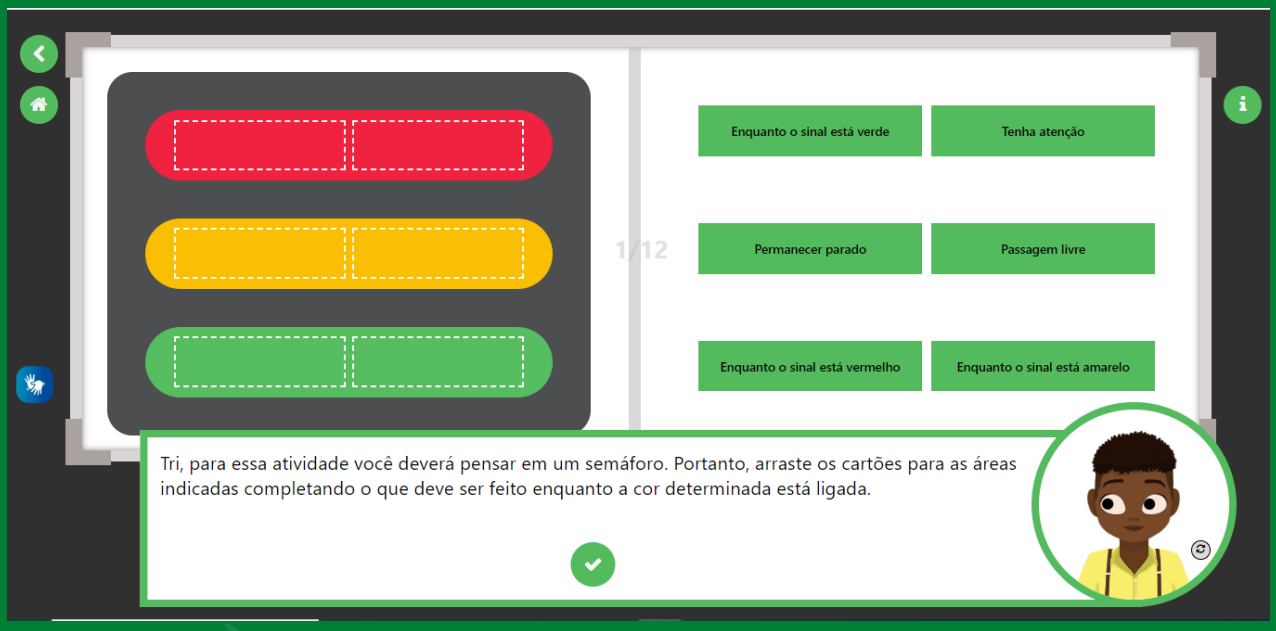
ESTRUTURAS de respetição

No exemplo acima, foi inicializada a variável "cont" com o valor zero e o laço de repetição continuará sendo executado até que "cont" seja maior que dez. A cada iteração do laço, o valor de "cont" é exibido no console por meio da função "escreva()", e em seguida, é incrementado.

Essa estrutura permite automatizar o processo de exibir os números de 0 a 10 no console, controlando a execução do laço com base na condição estabelecida.

As estruturas de repetição são poderosas ferramentas para lidar com tarefas repetitivas e lidar sobre uma quantidade de dados. Compreender e dominar essas estruturas é fundamental para otimizar o trabalho e evitar a repetição de código, permitindo que você automatize processos complexos de maneira eficiente.

Agora você está pronto para utilizar seus conhecimentos em estrutura de repetição nos exercícios da plataforma.



VETORES

Os Vetores (arrays unidimensionais) são variáveis que permitem armazenar múltiplos valores do mesmo tipo. Eles são especialmente úteis quando precisamos lidar com conjuntos de dados relacionados, como nomes de alunos ou notas em uma turma.

Cada elemento do vetor é acessado por um número chamado de índice, que começa em 0 e vai até o tamanho do vetor menos um.

A seguir, apresento um exemplo simples de leitura e impressão de 5 números utilizando vetores:

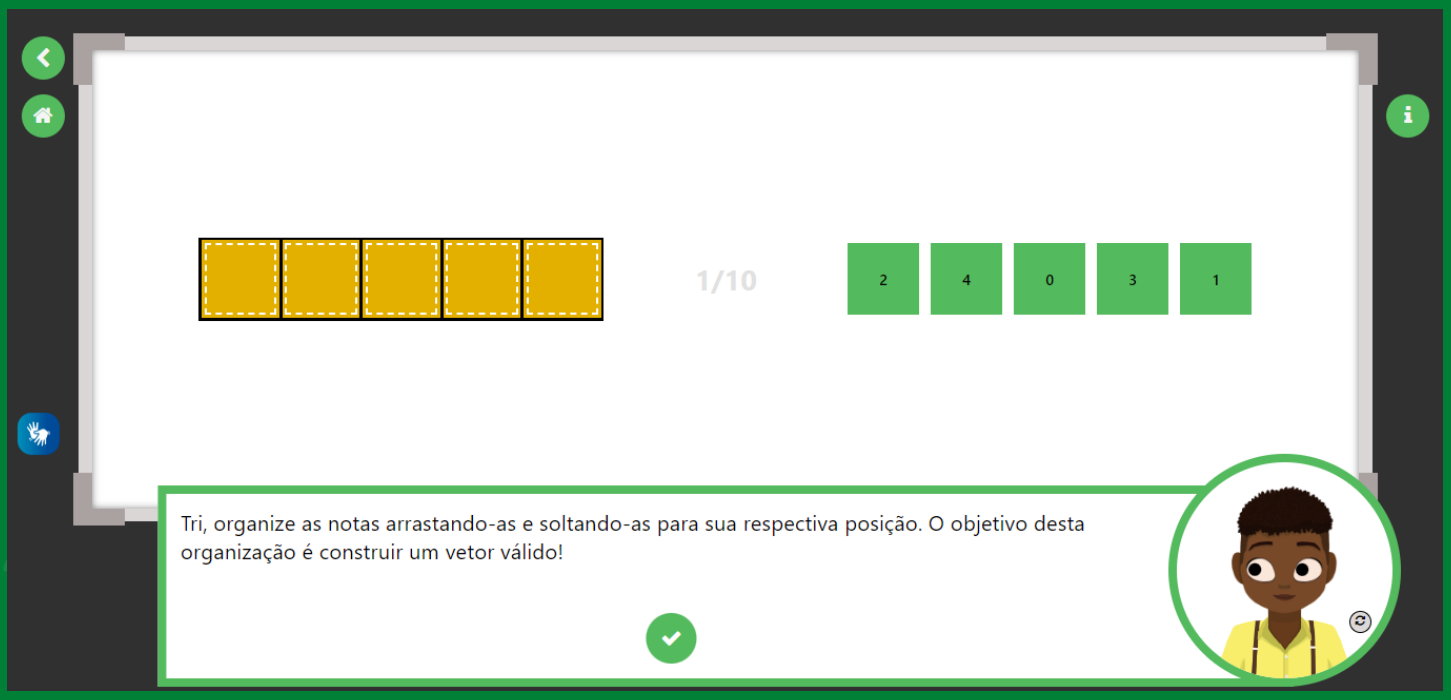
```
programa{
    funcao inicio(){
        inteiro vet[5]
        para (inteiro i = 0;  i < 5; i++){
            escreva ("\nDigite o ", i+1, "° número: ")
            leia (vet[i})
        }
        para (inteiro i = 0; i < 5; i++){
            escreva ("\nValor: ", vet[i])
        }
    }
}
```

No exemplo acima, foi declarado um vetor chamado "vet" com tamanho 5, capaz de armazenar 5 valores inteiros. Em seguida, utilizamos um loop "para" para iterar sobre o vetor e realizar a leitura de cada elemento. No código, o usuário pe solicitado a digitar o i+1-ésimo número, que será armazenado na posição i do vetor. Depois de concluir a leitura dos 5 números, outro loop "para" é usado para imprimir cada valor armazenado no vetor.

VETORES

Esta é apenas uma introdução básica ao uso de vetores. Eles são amplamente utilizados na programação para armazenar e manipular conjuntos de dados de forma eficiente. Com vetores, é possível realizar várias operações, como cálculos, ordenação e busca, facilitando o trabalho com grandes quantidades de informações relacionadas.

Veja mais sobre vetores e continue praticando seus conhecimentos nos exercícios da plataforma.



MATRIZES

As Matrizes, também conhecidas como arrays multidimensionais, desempenham um papel fundamental na programação e na lógica, oferecendo uma estrutura eficiente para armazenar e manipular conjuntos de dados organizados. Ao utilizar matrizes, é possível lidar com informações de forma mais estruturada e simplificada, especialmente quando se trata de conjuntos de dados relacionados.

Uma matriz, pode ser visualizada como um vetor de vetores, permitindo que dados sejam organizados em linhas e colunas. Essa estrutura é extremamente útil em muitos cenários, como o exemplo de um professor que precisa armazenar as notas de 50 alunos de quatro turmas diferentes.

Ao declarar uma matriz, é necessário especificar o tipo dos elementos que serão armazenados, seguido pelo nome da matriz e suas dimensões, indicando o número de linhas e colunas. Por exemplo, para armazenar as notas dos 50 alunos em quatro turmas, podemos declarar uma matriz da seguinte forma:

```
tipo nome_matriz [n_linhas] [n_colunas]
inteiro matriz [50] [4]
```

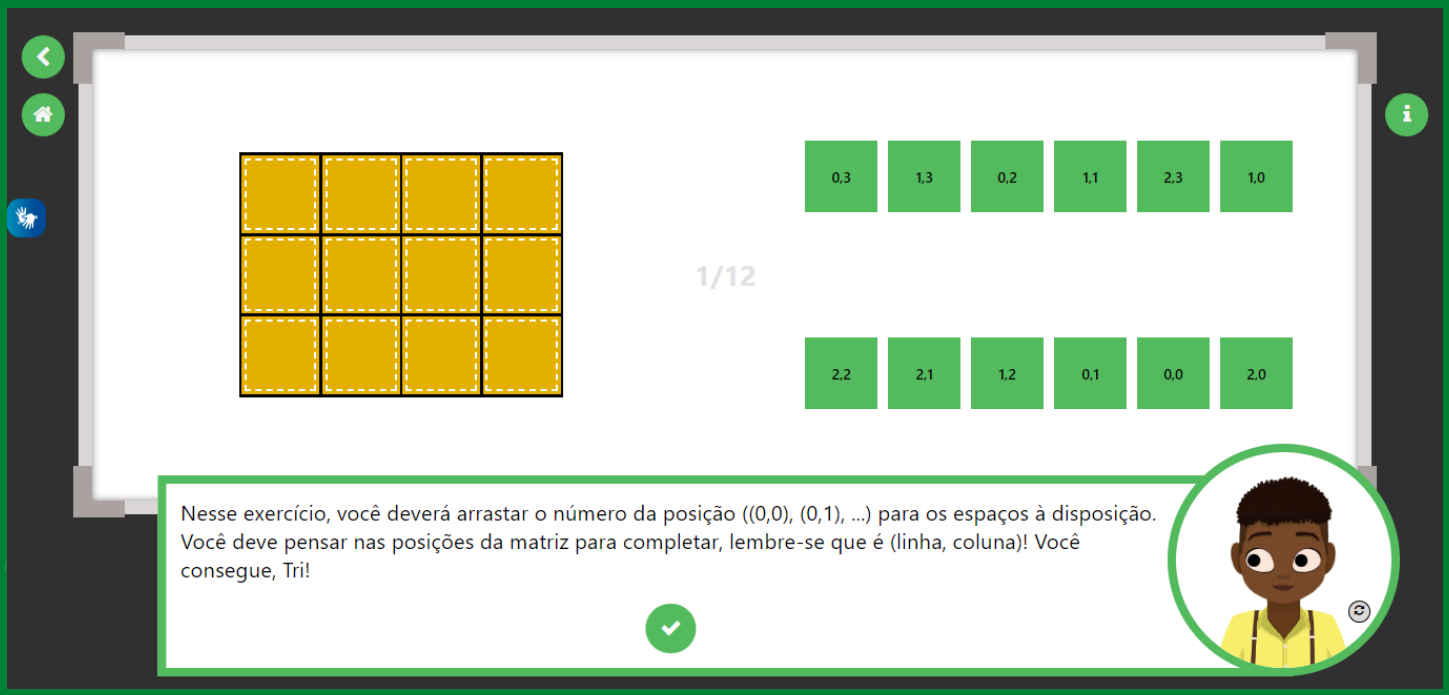
Nesse caso a matriz "matriz" é do tipo inteiro e possui 50 linhas (representando os alunos) e 4 colunas (representando as turmas). Essa estrutura de matriz permite que cada célula da tabela seja acessada de forma fácil e eficiente através de índices, facilitando a manipulação e o processamento dos dados.

Ao utilizar matrizes, podemos realizar diversas operações, como percorrer todos os elementos para realizar cálculos, buscar informações específicas, realizar ordenações, entre outras. Além disso, as matrizes também facilitam a implementação de algoritmos mais complexos, pois permitem a organização lógica dos dados e a manipulação de subconjuntos de informações.

MATRIZES

Portanto, a compreensão e o domínio das matrizes são essenciais para programadores e estudantes de lógica, pois essa estrutura de dados oferece uma forma eficiente de organizar, acessar e manipular conjuntos de informações relacionadas. Ao utilizar matrizes corretamente, é possível otimizar o desenvolvimento de algoritmos e obter resultados mais eficientes e precisos.

Agora você está pronto para utilizar seus conhecimentos em matrizes nos exercícios da plataforma.



PSEUDOCÓDIGO

O pseudocódigo é uma forma de descrição de algoritmos que é independente de linguagens de programação específicas. Ele permite que uma pessoa leia e interprete o código sem se preocupar com detalhes técnicos e específicos de uma linguagem em particular.

O objetivo do pseudocódigo é fornecer uma representação clara e compreensível do algoritmo, eliminando informações desnecessárias que não são essenciais para entender o funcionamento do código.

Ao utilizar pseudocódigo, é possível descrever um algoritmo de forma mais simples e abstrata, sem a necessidade de se ater as regras sintáticas e de formatação de uma linguagem de programação real. Isso facilita a compreensão do algoritmo por parte de um ser humano, permitindo que ele se concentre nos princípios e na lógica subjacente, em vez de se preocupar com detalhes técnicos.

O pseudocódigo é amplamente utilizado no desenvolvimento de algoritmos, especialmente durante as fases iniciais do processo de programação. Ele permite que os programadores expressem suas ideias e soluções de forma mais clara e concisa, facilitando o planejamento e a estrutura do código.

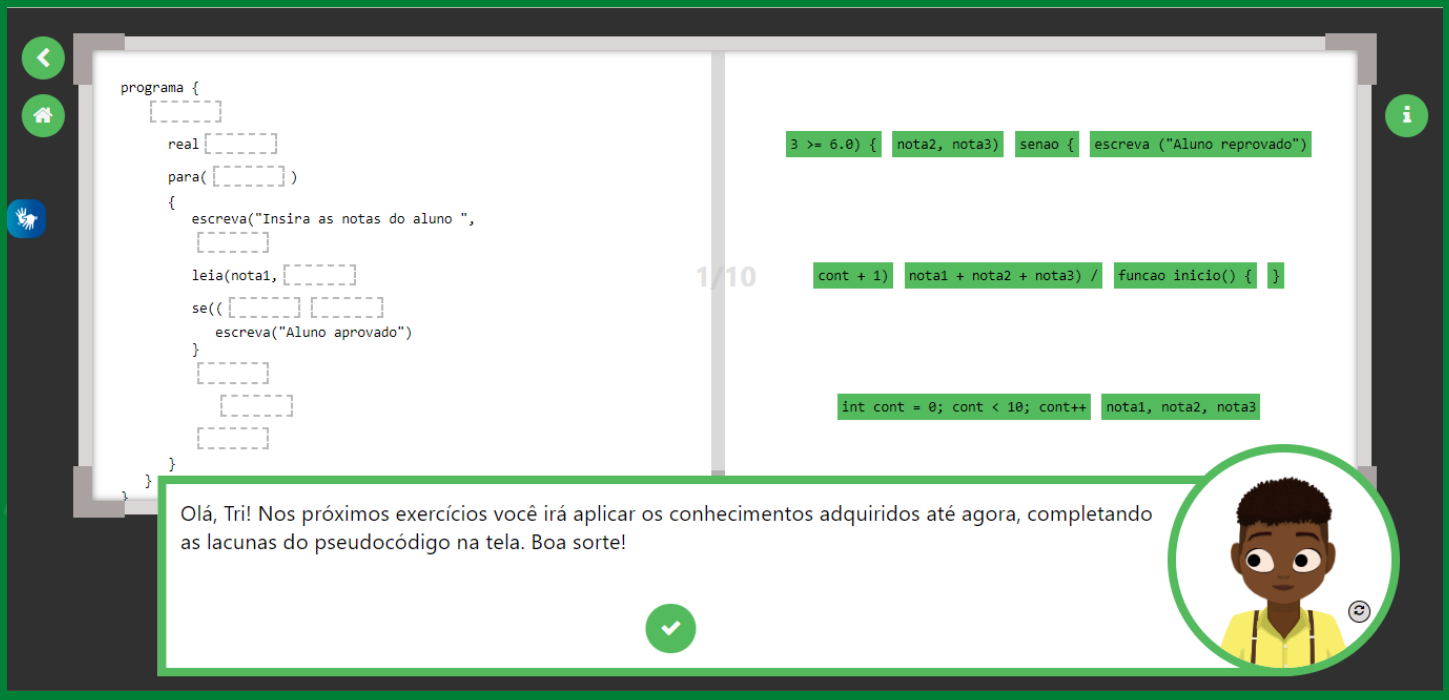
Além disso, o pseudocódigo também é uma ferramenta útil no processo de aprendizagem da programação, pois ajuda os iniciantes a entender os conceitos e os fluxos de um algoritmo antes de se aprofundarem nas complexidades de uma linguagem de programação real.

PSEUDOCÓDIGO

Portanto, o pseudocódigo desempenha um papel fundamental na programação, permitindo que os programadores expressem suas ideias e soluções de forma clara e concisa. Ele fornece uma representação abstrata dos algoritmos, tornando mais fácil a compreensão e a análise do funcionamento do código.

Dominar a habilidade de escrever e interpretar pseudocódigo é essencial para o desenvolvimento de algoritmos eficientes e de qualidades.

Agora você está apto a finalizar as atividades do último módulo da plataforma!



A EQUIPE DO TRILOGIC

COORDENAÇÃO DO PROJETO
Prof. Dr. Sandro José Ribeiro da Silva
sandro.silva@canoas.ifrs.edu.br
Profa. Dra. Carla Odete Balestro Silva
carla.silva@canoas.ifrs.edu.br

COLABORADORES
Prof. Dr. Márcio Bigolin
Prof. Dr. Igor Lorenzato Almeida
Vinícius Raupp Alves
Vitor Secretti Bertoncello

ALUNOS BOLSISTAS - ANO 2022/2023
Ana Laura da Silva Santos
Filipe Mallmann Siota
Carlos André Mielke Dutra
Henrique Gross Fantoni

DIAGRAMAÇÃO E TEXTO DO MANUAL
Ana Laura da Silva Santos

REVISÃO
Profa. Dra. Carla Odete Balestro Silva

DESENVOLVIMENTO DA PLATAFORMA 2022/2023
Filipe Mallmann Siota
Carlos André Mielke Dutra
Henrique Gross Fantoni

INSTITUTO FEDERAL
Rio Grande do Sul
Campus Canoas

TriLogic

CNPq