UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO GOMES FRANCO

FLUXER: DESENVOLVIMENTO DE UMA APLICAÇÃO GAMIFICADA PARA APRENDIZADO DE PROGRAMAÇÃO BÁSICA

BAURU
Dezembro/2020

FRANCISCO GOMES FRANCO

FLUXER: DESENVOLVIMENTO DE UMA APLICAÇÃO GAMIFICADA PARA APRENDIZADO DE PROGRAMAÇÃO BÁSICA

Trabalho de Conclusão de Curso do Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientadora: Profa. Dra. Andréa Carla Gonçalves Vianna

BAURU
Dezembro/2020

Francisco Gomes Franco Fluxer: Desenvolvimento de uma aplicação gamificada para aprendizado de programação básica/ Francisco Gomes Franco. – Bauru, Dezembro/2020- 39 p. : il. (algumas color.) ; 30 cm.
Orientadora: Profa. Dra. Andréa Carla Gonçalves Vianna
Trabalho de Conclusão de Curso – Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Ciência da Computação, Dezembro/2020.
1. Gamificação 2. Programação 3. Aplicação Móvel

Francisco Gomes Franco

Fluxer: Desenvolvimento de uma aplicação gamificada para aprendizado de programação básica

Trabalho de Conclusão de Curso do Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Profa. Dra. Andréa Carla Gonçalves Vianna
Orientadora
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Ma. Juliana Feitosa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 15 de dezembro de 2020.

Agradecimentos

Agradeço primeiramente aos meus pais, que sempre estiveram aqui por mim e me proporcionaram a melhor vida que puderam.

Muito obrigado também aos meus amigos, que me apoiaram nesta longa jornada e me impediram de desistir.

Por fim, estendo agradecimentos a todos os funcionários da Unesp que me deram o ambiente onde pude me desenvolver tanto.

In my opinion we are all gamification designers. At least in our childhoods.
Sebastian Deterding

Resumo

Com um número maior que o desejado de desistências em cursos de programação, alternativas são buscadas para o desenvolvimento pessoal das habilidades lógicas e de abstração. Este projeto propõe o uso de técnicas de gamificação no estudo de programação básica, através da construção de um aplicativo móvel que fornece uma experiência lúdica e acessível de ensino de conceitos introdutórios da área.

Palavras-chave: Gamificação, programação, lógica, aprendizado.

Abstract

With an increasing number of departure in programming courses, alternatives for personal development of logic and abstraction skills are being researched. This project purposes the use of gamification techniques in the study of basic programming, by presenting an mobile application that provides an accessible and ludic experience of introductory concepts of the area.

Keywords: Gamification, programming, logic, learning.

Lista de figuras

Sumário

1 Introdução

Atualmente é comum que cursos de programação tenham grandes margens de desistência e altas taxas de falha dos estudantes (AMBROSIO et al., 2011), onde muitos iniciantes falham em absorver os conceitos básicos do tema (COMBEFIS; BERESNEVICIUS; DAGIENE, 2016).

Ambrósio et al. (2011) definem que uma das causas da falta de compreensão dos alunos é o baixo desenvolvimento de certas capacidades cognitivas, em especial: abstração e sequenciamento de comandos, que são definidas pelos pesquisadores como as bases da criação de algoritmos.

O grupo de aspirantes a desenvolvedores se depara com um método de ensino consideravelmente cansativo, que exige dos mesmos uma capacidade que eles muitas vezes não possuem. Muitos alunos terminam frustrados e acabam deixando o curso nos primeiros semestres.

Este trabalho tem como objetivo produzir um aplicativo usando algumas técnicas de gamificação (também chamada ludificação), exemplificando como estes conceitos podem ser utilizados no aprendizado de programação básica.

Gamificação é um termo abrangente para um conjunto de técnicas com o objetivo de melhorar a experiência e engajamento do usuário em determinada atividade (DETERDING et al., 2011b). É esperado que as técnicas auxiliem diretamente nos aspectos de dificuldade cognitiva anteriormente mencionados e, de maneira geral, tornem a *User Experience* mais agradável.

O conteúdo desta monografia detalha o processo de desenvolvimento do trabalho, iniciando com a revisão bibliográfica no capítulo 2 e uma descrição das ferramentas utilizadas no capítulo 3, seguido pelo detalhamento da construção do projeto no capítulo 4 e sua conclusão com o capítulo 5. Por fim, o apêndice A apresenta uma transcrição direta de todo o material teórico produzido para este trabalho.

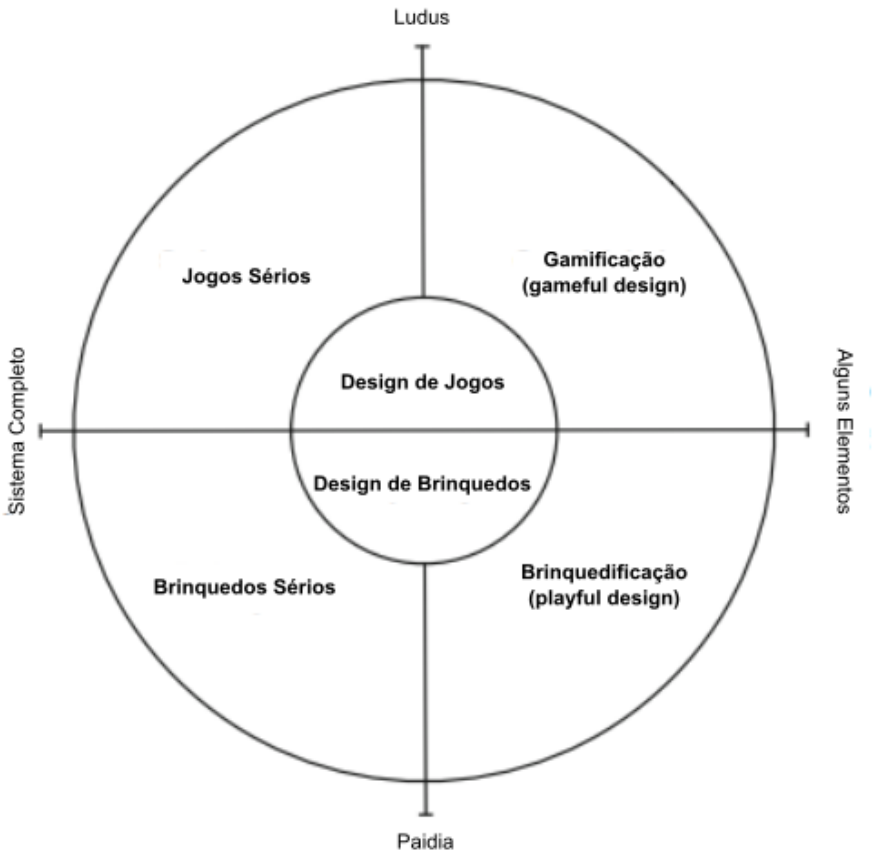
2 Fundamentação Teórica

Os tópicos deste capítulo apresentam uma definição do termo gamificação e suas principais técnicas, juntamente a uma análise do impacto da gamificação na educação.

2.1 Definição de Gamificação

Em poucas palavras, Gamificação é o uso de elementos do *design* de jogos em conceitos de não jogo (DETERDING et al., 2011a). É um termo abrangente para um amplo e variado conjunto de técnicas que não possuem uma definição específica (DETERDING et al., 2011b).

Figura 1 – Tipos de aplicação de elementos lúdicos em outros contextos.



Fonte: Make-Believe in Gameful and Playful Design. Link: http://eprints.whiterose.ac.uk/100127/ (Adaptada pelo autor)

Como representado na Figura 1, Sebastian Deterding (DETERDING, 2016) classifica a aplicação de elementos de jogo em contextos de não jogo dentro de dois eixos.

O eixo vertical define de que maneira o conceito de regras se aplica ao contexto, dividindo o *design* entre *paidia* e *ludus*:

- O *paidia* se refere a um ambiente de exploração livre e sem regras, como brincadeiras infantis de faz de conta ou teatro de improvisação. Um *design* baseado em *paidia* procura estimular a criatividade em suas funções e é associado à criação de brinquedos.

- Por outro lado, *ludus* significa que o sistema existe com objetivo definido e apoiado por um conjunto de regras, sendo o mais comum em *design* de jogos e esportes competitivos.

O eixo horizontal indica o quanto do *design* é produzido com o intuito de ser um jogo ou brinquedo. Um sistema mais próximo da extremidade esquerda do eixo pode ser classificado como um jogo ou brinquedo, enquanto os *designs* à direita são aplicações diferentes que incorporam alguns elementos de jogos.

Combinando os eixos, temos quatro quadrantes do sistema:

- Um *design* mais próximos a *paidia* que incorpora um sistema completo é o que definimos como brinquedo sério, utilizando a composição de brinquedos para propósitos usualmente educativos e da área da saúde;

- Um ambiente *paidia* com alguns elementos de jogos é geralmente classificado como *design* lúdico, incorporando aspectos descontraídos à composição de um sistema;

- Características de *ludus* em um sistema composto totalmente por elementos do *design* de jogos classifica uma área próxima da gamificação, chamada jogos sérios ou *serious game design*;

- Por fim, um sistema de características lúdicas que incorpora alguns elementos de *game design* caracteriza a área de estudo deste projeto: Gamificação, também conhecida como ludificação ou *gameful design*.

2.2 Gamificação no Aprendizado

Em um levantamento literário de 2014 (HAMARI; KOIVISTO; SARSA, 2014), aplicações diversas de gamificação provocaram, em sua maioria, resultados positivos nas respostas comportamentais e psicológicas dos analisados.

Um estudo realizado em 2017 (COSTA; PEREIRA; APARICIO, 2017) identificou a falta de motivação dos estudantes como uma das principais causas de baixo desempenho em cursos de programação. Isto também foi apontado por Khaleel e colegas (KHALEEL et al., 2015), que cita altas taxas de desistência no primeiro ou segundo semestres.

A pesquisa de Costa et. al. (2017) sugere processos gamificados como forma de contornar a falta de motivação dos estudantes. Ainda que aplicada em baixa escala, os pesquisadores concluíram que a gamificação pode causar efeitos positivos no método de ensino, tornando

os alunos mais motivados. Analogamente, as implementações de Khaleel em seu curso de programação trouxeram aumento no engajamento e participação dos alunos durante as aulas.

Baseado em modelos e pesquisas como as citadas anteriormente, este trabalho propõe um método para afetar as habilidades essenciais que normalmente são deficientes em iniciantes da área de programação, como definidas por Ambrósio et. al. (2011):

- Abstração: O estudante precisa desenvolver a capacidade de passar os problemas concretos para um ambiente simbólico e semântico, ou seja, criar uma representação virtual e genérica dos elementos do mundo real. Sem abstração, é difícil traduzir problemas para a linguagem de programação.
- Sequenciamento de Comandos: O estudante também precisa aprender a não observar problemas buscando por uma solução geral e abrangente. Ao invés disso, um programador é favorecido pela habilidade de separar a solução em blocos menores e gerenciáveis, buscando compreender em que ordem eles devem ser organizados e como eles interagem entre si para resolver o problema generalizado.

Este processo de entender como os blocos independentes funcionam e organizá-los sequencialmente foi definido pelos professores entrevistados na pesquisa como "pensar como um computador" ou então "pensamento algorítmico".

As entrevistas de Ambrósio et. al. (2011) foram realizadas com dois grupos de alunos que tiveram facilidade e dificuldade no aprendizado de programação, respectivamente. Além disso, professores também comentaram sobre o rendimento dos alunos.

Um detalhe interessante é que os docentes identificaram um terceiro grupo de estudantes: aqueles que inicialmente apresentavam dificuldades, mas após um tempo conseguiram superá-las e aumentaram seu rendimento.

A pesquisa sugere que esta terceira categoria de alunos conseguiu inferir espontaneamente as capacidades básicas anteriormente mencionadas. Este trabalho propõe o uso das técnicas de gamificação para atingir a mesma intuição de tais habilidades.

2.3 Técnicas de *Game Design*

Como mencionado, gamificação envolve implementar em um sistema técnicas comuns do *design* de jogos. Hamari, Koivisto e Sarsa (HAMARI; KOIVISTO; SARSA, 2014) compilam um conjunto de elementos de jogo incorporados em diversos trabalhos, onde os mais comuns foram pontos, placares e medalhas, sendo também válido mencionar o uso frequente de níveis e avatares.

O trabalho de Piteira e Costa (PITEIRA; COSTA, 2017) apresenta mais detalhes sobre estes elementos e como eles influenciam a experiência do usuário:

- Pontos e placares: Quando atividades são realizadas dentro do ambiente, o usuário recebe uma quantidade de pontos virtuais que ficam registrados em seu perfil. Isto desencadeia a reação natural de uma pequena recompensa a cada atividade e incentiva a acumulação de tais pontos. Em algumas aplicações, pontuação também pode ser usada como moeda virtual para adquirir opções de customização;

  A quantidade total de pontos pode ser ainda comparada a de outros usuários se o ambiente for compartilhado (exemplo: uma sala de aula). Os estudantes podem ser organizados em um placar (também chamado de *leadeboard*), onde os melhores colocados podem inclusive receber algum tipo de premiação. O objetivo de placares é estimular uma competição saudável entre múltiplos indivíduos submetidos ao mesmo método;

- Medalhas: Um recurso comum em jogos modernos, as medalhas (também chamadas de *achievements*), marcam realizações importantes. Quando certo conjunto de atividades relacionadas são concluídas e o usuário está pronto para a próxima, o sistema pode garantir uma medalha além dos pontos, causando uma sensação maior de recompensa e marcando a evolução do usuário dentro do ambiente;

- Níveis: Uma forma comum de garantir a progressão controlada do usuário é por meio de níveis. Para contextos de ensino, o total de informações é dividido em pequenas partes e para cada uma delas se elabora níveis, que são ordenados por dificuldade. Com isso, o usuário só pode progredir para uma parte mais avançada depois que cumprir o nível anterior e cada nível é apenas um pouco mais avançado que o predecessor.

  Esta divisão tem o propósito de não sobrecarregar o usuário com muitas informações de uma vez, deixando-o compreender e assimilar o conteúdo em seu tempo. Além disso, os níveis mantém a progressão desafiadora o suficiente para as capacidades crescentes do usuário, uma vez que desafio também é uma tática que jogos utilizam para manter o engajamento; e

- Avatar: Avatares são representações de um jogador dentro do jogo. Quando se associa um perfil de usuário a um avatar customizável, cria-se uma aproximação da pessoa com o sistema, o que consequentemente pode aumentar o engajamento.

  Avatares são especialmente úteis em sistemas online com múltiplos membros, onde pode-se criar opções de interação entre os usuários por meio de suas representações.

Tendo em vista a natureza da aplicação aqui apresentada, as principais mecânicas escolhidas para o projeto foram a divisão em níveis e as medalhas.

O sistema de níveis foi implementado tanto em apresentação de conteúdo teórico quanto exercícios práticos. As medalhas foram utilizadas também como sistema de pontos, no sentido de que uma parte da medalha é liberada a cada nível e a cada grupo de níveis que compõe uma sessão (detalhado na sessão 4 deste projeto) uma medalha é completada.

3 Ferramentas

Este capítulo apresenta as ferramentas fundamentais para a execução do projeto.

3.1 *Unity Engine*

Unity Engine (UNITY, 1995) é um motor para desenvolvimento de jogos, animações e outros projetos gráficos, com foco em facilidade de uso para uma grande base de usuários. Nele é possível trabalhar com diversos objetos de jogo (designados no programa como *gameObjects*), vinculando-os a animações, eventos e simulações de física, iluminação, entre outros.

Para este projeto foi utilizada a versão pessoal e gratuita do programa, com foco nos componentes de animação e interface de usuário.

3.2 *Visual Studio Community 2017*

Visual Studio (VISUALSTUDIO, 1991) é uma IDE desenvolvida pela *Microsoft* para desenvolvimento de projetos em linguagens desenvolvidas pela empresa. O programa cria um ambiente robusto com diversos recursos como *plugins*, opções de *debug* e uma integração padrão com a *Unity Engine*.

Para este projeto a linguagem escolhida foi o *C#* (C#, 1991), uma linguagem de programação orientada a objeto na plataforma *.NET*, que é uma das opções pré-integrada aos *scripts unity*.

3.3 *Inkscape*

Inkscape (INKSCAPE, 2003) é um software livre e gratuito de *design* gráfico para construção de imagens vetoriais. Ele foi utilizado para construção de todas as imagens integradas na interface da aplicação e também do planejamento de fluxo de telas.

4 Desenvolvimento

Neste capítulo é detalhada a construção do aplicativo, desde os primeiros conceitos, passando pela revisão bibliográfica, planejamento do desenvolvimento e o processo de desenho e programação.

4.1 Planejamento

4.1.1 Ideias Iniciais

Inicialmente foi definido o desejo de trabalhar técnicas de *design* de jogos no projeto. O uso de jogos sérios foi considerado, mas gamificação foi escolhida como tema principal por mera questão de interesse pessoal.

Em seguida a questão seria definir em qual área aplicar a técnica escolhida. Com o passado do autor em diversas disciplinas relacionadas a programação, tanto em ensino superior quanto técnico, bem como observando discussões em fóruns, alguns problemas da área se tornaram aparentes.

4.1.2 Revisão Bibliográfica

Com os temas principais do trabalho em mente, diversos artigos foram pesquisados. Alguns foram selecionados por apresentarem discussões sobre gamificação aplicada a contextos de ensino, como os trabalhos de Costa, Pereira e Aparício (2017), Piteira e Costa (2017) e Khaleel et al. (2015), ou ainda relativos ao ensino de programação com métodos de jogo, mesmo que não especificamente gamificação (COMBEFIS; BERESNEVICIUS; DAGIENE, 2016).

Outro fator importante seria encontrar uma definição acadêmica para o termo "gamificação". Muitos artigos citavam os trabalhos de Sebastian Deterding, um dos nomes mais proeminentes na área. Suas publicações "*Gamification: Towards a Definition*" (DETERDING et al., 2011a) e "*Gamification: Using Game Design Elements in Non-Gaming Contexts*" (DETERDING et al., 2011b), foram essenciais para as definições apresentadas no início do trabalho (seção 2.1 definindo gamificação). Um segmento escrito pelo mesmo autor (DETERDING, 2016) também foi de grande ajuda no estabelecimento de questões específicas sobre a gamificação dentro da área de estudo da aplicação dos elementos de *game design*.

Mais uma seleção fundamental de artigos foram os relativos à quais as dificuldades sofridas por pessoas que ingressam em cursos de programação (COMBEFIS; BERESNEVICIUS; DAGIENE, 2016). O trabalho de Ambrósio et. al. (AMBROSIO et al., 2011) definiu duas capa-

cidades cognitivas que formam a base do pensamento algorítmico (abstração e sequenciamento de comandos).

Esta pesquisa também se estendeu a compreender o básico sobre como elementos de jogo podem afetar o usuário positivamente. Aqui é válido citar comentários sobre como elementos de jogo ajudam a gerar respostas positivas do usuário mesmo que ele falhe (RAVAJA et al., 2006), ou ainda relatórios de impactos majoritariamente positivos após a implementação de ambientes gamificados (HAMARI; KOIVISTO; SARSA, 2014) (MAIA; GRAEML, 2015).

4.1.3 Definição do projeto

Após a pesquisa descrita nos itens anteriores, foi elaborado o conceito do projeto: um aplicativo móvel, com foco em um público adolescente e jovem adulto, e com objetivo de desenvolver no usuário as qualidades básicas de um bom pensamento algorítmico e também ensinar conceitos básicos de programação.

O aplicativo não seria feito com objetivo de interações *online* e também não teria um tempo de uso muito longo devido à limitações de tempo para sua construção.

Em termos de funcionamento, ele seria dividido em um módulo teórico com pequenas lições que devem ser lidas e um módulo prático com exercícios. Ambos os blocos interagem entre si, de modo que exercícios são liberados conforme as lições teóricas são concluídas.

As principais técnicas de gamificação implantadas existem na divisão em níveis, tanto da parte prática quanto a teórica. Também existe uma tela de visualização de medalhas, liberadas conforme certos níveis são atingidos. Além disso, o *design* seria feito de maneira dinâmica e similar à jogos *mobile*.

4.2 Módulo Teórico

Com base no conteúdo inicial da matéria de Algoritmos I e mais referências de modelos de ensino encontrados na revisão bibliográfica (PITEIRA; COSTA, 2017), foram determinadas algumas lições abrangendo conteúdos básicos sobre o funcionamento de uma linguagem de programação.

Foi decidido que, assim como o *design* geral do aplicativo, as lições devem ser dinâmicas. Portanto elas são apresentadas em pequenos blocos de texto, um por vez, como pequenas páginas. Além disso, cada lição tem um baixo número de páginas e a maioria delas é seguida pelo desbloqueio de um exercício relacionado, para que o usuário possa imediatamente praticar o conteúdo.

Foi escrito um total de nove lições, divididas nos temas:

- Algoritmos: Definição de o que é um algoritmo e como ele se relaciona com situações

cotidianas;

- Entrada, Processamento e Saída: Explica o funcionamento padrão de um programa de computador, dizendo como dados são adquiridos, processados e então transformados em uma resposta;
- Fluxogramas: Comenta sobre como algoritmos são representados de forma gráfica e introduz o usuário ao ambiente de exercícios práticos;
- Variáveis: Define o conceito de variáveis, explicando como um programa armazena e acessa informações;
- Entrada de dados: Explica ao usuário sobre entrada de dados e como atribuir valores a variáveis dentro do ambiente prático;
- Exibindo informações: Explica sobre saídas de dados e demonstra o método de *output* do ambiente prático;
- Operadores aritméticos: Comenta sobre como os *softwares* realizam muitos cálculos e introduz o componente de operações matemáticas da interface;
- Comando condicional: Explica como muitas vezes é necessário que o programa realize decisões baseadas em informações que ele possui. Também apresenta como isso pode ser realizado na interface através dos comandos "Se-então" e "Senão"; e
- Comandos de repetição: Define o conceito de laços de repetição e como programas fazem tarefas repetitivas através de tais elementos, também ensina o usuário a utilizar os comandos "Para" e "Enquanto" da interface prática.

Uma lista contendo todo material teórico produzido está inclusa no apêndice A.

4.3 Elementos Gráficos

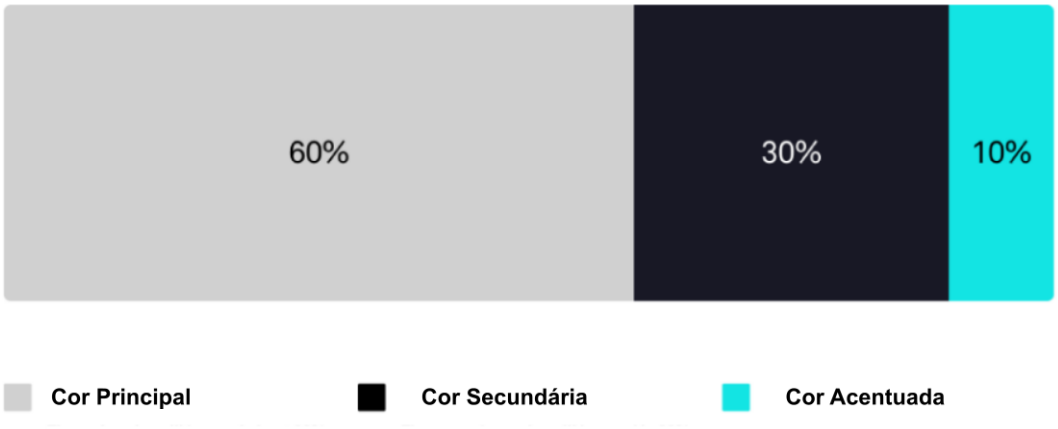
4.3.1 Escolha de Cores

Um *design* efetivo é um ponto importante para este projeto, portanto os elementos gráficos foram cuidadosamente planejados. Um ponto a se considerar é a paleta de cores, Renee Fleck (FLECK, 2018) recomenda uma distribuição como indicada na figura 2.

A distribuição de uma cor principal que ocupe 60% da interface, seguida por uma cor secundária próxima da principal em 30% e uma cor de destaque em 10% foi levemente alterada, resultando na paleta de cores representada na figura 3.

A cor principal é um tom escuro de cinza (RGB hexadecimal: 323232) e foi utilizada para o fundo da tela e dos elementos. Existem duas cores secundárias em tons de azul (RGBs

Figura 2 – Distribuição de cores recomendada em uma interface de usuário.



Fonte: https://dribbble.com/stories/2018/12/19/choosing-colors-for-web-design-a-practical-ui-color-application-guide (Adaptada pelo autor)

Figura 3 – Paleta de cores da aplicação.

Fonte: Elaborada pelo autor

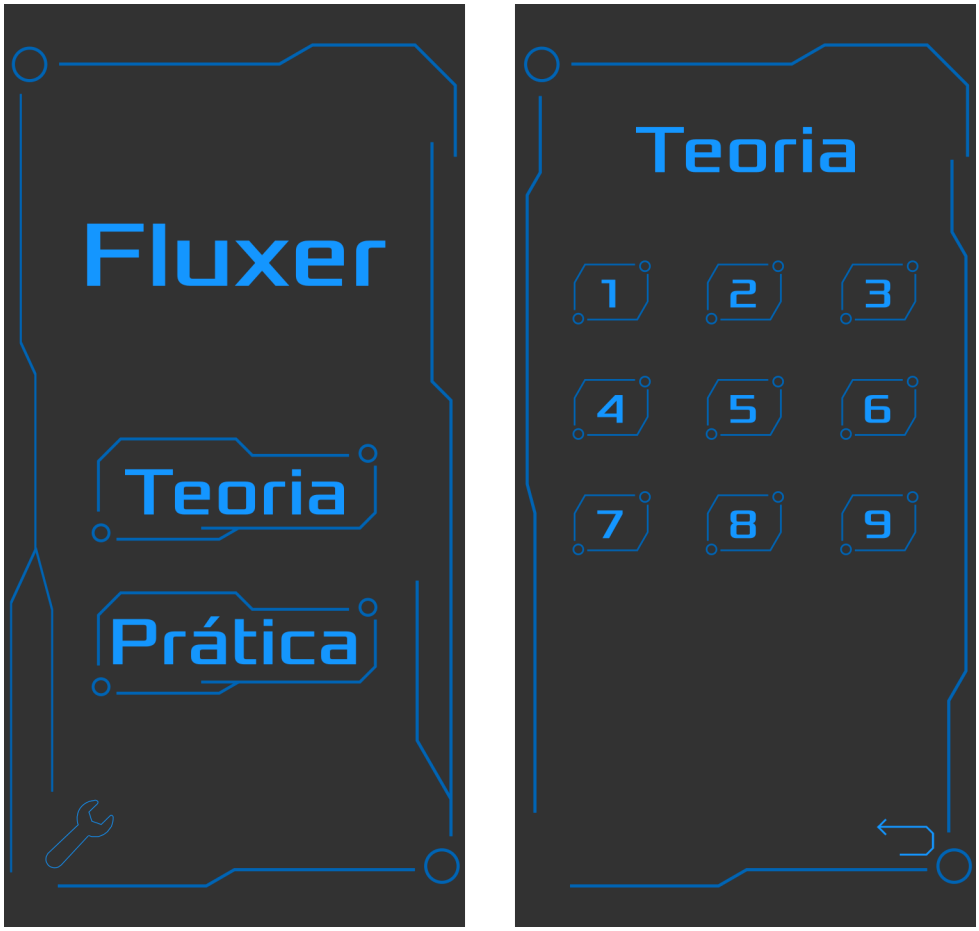
hexadecimais: 0064b4 e 1496ff, respetivamente), utilizados para contorno dos elementos e também a cor dos textos. Por fim uma cor de destaque em um tom muito claro de azul ciano (RGB hexadecimal: aaebff), utilizado para ressaltar alguns botões clicados e pequenos elementos de destaque na tela. A paleta foi escolhida esperando um bom contraste entre o fundo e os elementos, tendo em mente facilidade de visualização e uma distribuição agradável de cores.

4.3.2 Identidade Visual

Em seguida a identidade visual do aplicativo foi definida a partir da paleta e é demonstrada na figura 4. A aparência buscada para o aplicativo é um aspecto "tecnológico" e dinâmico, com linhas retas, mas ângulos acentuados quebrados por linhas menores. Além disso, linhas frequentemente não se tocam, criando um aspecto leve e flutuante.

Também foi necessária a determinação de uma fonte que acompanhe a ideia da identidade visual, sendo dinâmica, de fácil leitura e com aparência tecnológica. Para tal, foi escolhida a fonte gratuita e livre de *copyright* Xolonium, presente nos títulos das telas da figura 4 e também na figura 5.

Figura 4 – Exemplo das telas da aplicação.



Fonte: Elaborada pelo autor

Figura 5 – Exemplo da fonte escolhida.

Fluxer

Fonte: Elaborada pelo autor

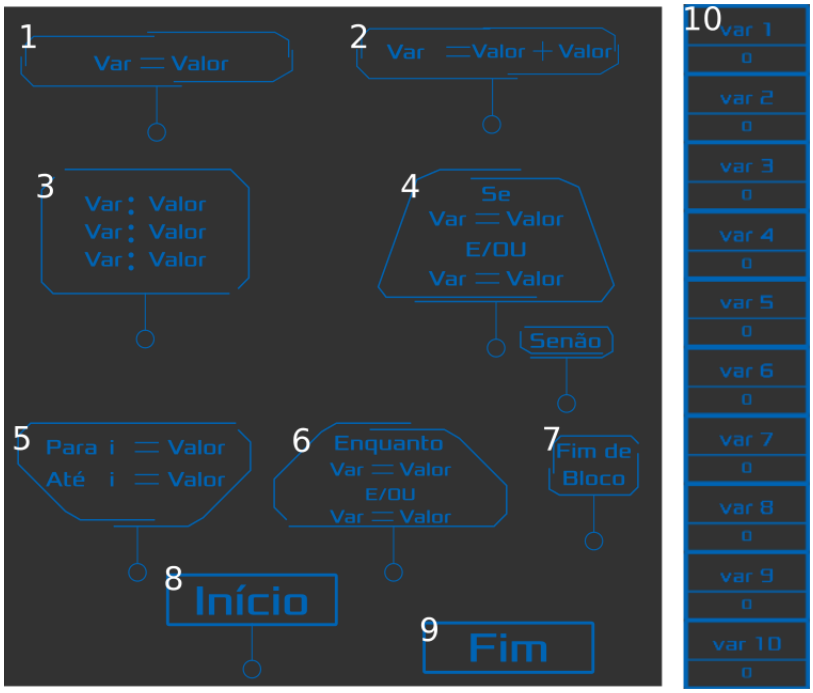
4.4 Módulo Prático

4.4.1 Exercícios

A sessão prática consiste em um conjunto de exercícios relacionados às lições teóricas. O primeiro exercício é desbloqueado quando o usuário termina de ler o respectivo texto. Em seguida, a conclusão de cada exercício libera o seguinte, até que todo o conjunto seja desbloqueado. A maioria das lições é associada a um grupo de três exercícios.

Os exercícios são realizados em um tipo de código criado especificamente para o aplicativo. É uma linguagem visual e interpretada, composta de blocos que correspondem a instruções e são distribuídos em uma linha reta que conecta os blocos sequencialmente, similar

Figura 6 – Componentes da linguagem



Fonte: Elaborada pelo autor

a um fluxograma.

Os componentes são exibidos na figura 6 e detalhados a seguir:

- 1 - Atribuição: Define uma variável* pelo valor** a direita;
- 2 - Operação: Define uma variável* por uma operação entre dois valores**. As operações possíveis são soma, subtração, multiplicação, divisão, potência e módulo (resto da divisão inteira);
- 3 - Saída: Exibe o valor contido em até três variáveis*. Também é utilizado pelo sistema para simular o *output* do programa.
- 4 - Se - Então - Senão: Age como comando condicional, sendo dividido em duas partes:

  O bloco principal possui duas comparações entre uma variável* e algum valor** (igual, diferente, maior que, menor que, maior ou igual a, menor ou igual a), sendo a segunda opcional. Se há duas comparações, elas devem ser conectadas por um comparador lógico (E ou OU).

  Se o resultado do bloco principal é determinado como verdadeiro, o programa executa as funções abaixo dele até encontrar o próximo "fim de bloco" (item 7). Se o resultado é falso, o programa pula os comandos até encontrar um "fim de bloco" ou um componente "senão", executando as funções abaixo dele até o próximo "fim de bloco";

- 5 - Para: Este comando é um tipo de laço de repetição, executando o conjunto de funções abaixo dele até o próximo "fim de bloco". O número de repetições é dado por sua variável interna i, na qual o usuário define seu valor inicial e seu valor final, e ela é incrementada em um a cada iteração;
- 6 - Enquanto: É um laço de repetição que funciona de maneira semelhante ao item anterior, porém a quantidade de repetições não é predeterminada. Neste caso, o bloco possui a determinação de uma ou duas condições (semelhante ao descrito no item 4) e os comandos são repetidos até que tal condição seja verdadeira;
- 7 - Fim de Bloco: Este comando é apenas um marcador utilizado para indicar onde termina a região de comandos afetados por algum dos comandos apresentados nos itens 4, 5 e 6;
- 8 - Início: Indica onde o fluxograma começa. Este componente é especial e não pode ser selecionado ou apagado pelo usuário, ao invés disso, ele é automaticamente colocado no começo de cada exercício;
- 9 - Fim: Indica onde o fluxograma termina. Este bloco é o único a não possuir um ponto abaixo, indicando que nenhum componente pode ser conectado depois dele;e
- 10 - Caixa de Variáveis: Conjunto de variáveis que podem ser utilizadas, cada uma com o nome var X, onde X é um número de um a dez. O número de variáveis disponíveis é predeterminado em cada exercício.

  As variáveis armazenam um valor inteiro de zero a 99, onde zero é o valor padrão.

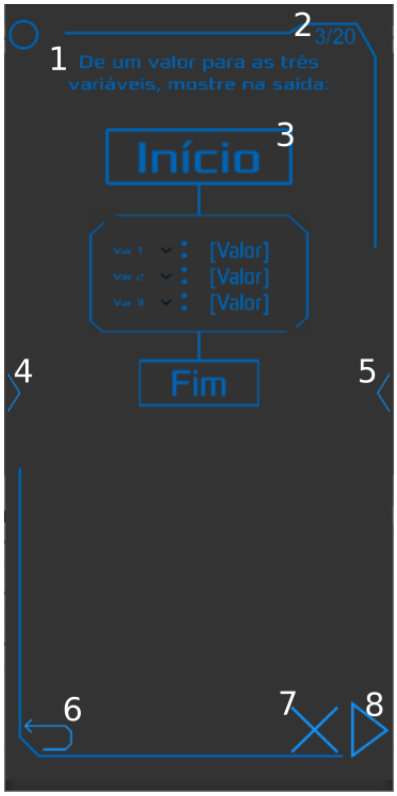
*Variável: Se refere à opção de selecionar qualquer uma das variáveis disponíveis.

**Valor: Significa que pode ser colocado o valor de alguma variável ou um número inserido pelo usuário.

O ambiente geral de um exercício é representado na figura 7, que utiliza como exemplo o exercício 1. Os elementos numerados representam:

- 1 - Enunciado: Determina o objetivo do usuário no exercício. São textos curtos (máximo de três linhas) que comunicam uma objetivo direto;
- 2 - Contador: Exibe o número de componentes no fluxograma, onde a maior quantidade possível de elementos é 20;
- 3 - Área do fluxograma: Neste ponto o usuário pode conectar blocos de comandos (descritos na figura 6);
- 4 - Botão da caixa de variáveis: Exibe a caixa de variáveis (mostrada no item 8 da figura 6), em alguns exercícios as variáveis podem ser editadas por este menu;

Figura 7 – Ambiente de exercício



Fonte: Elaborada pelo autor

- 5 - Botão da caixa de componentes: Exibe a caixa dos componentes descritos na figura 6 (itens 1 a 7). A partir dela, é possível arrastar blocos para a montagem do fluxograma.

  O número de componentes disponíveis é relacionado às necessidades específicas de cada exercício;

- 6 - Botão "retornar": Volta à tela de seleção de exercícios;

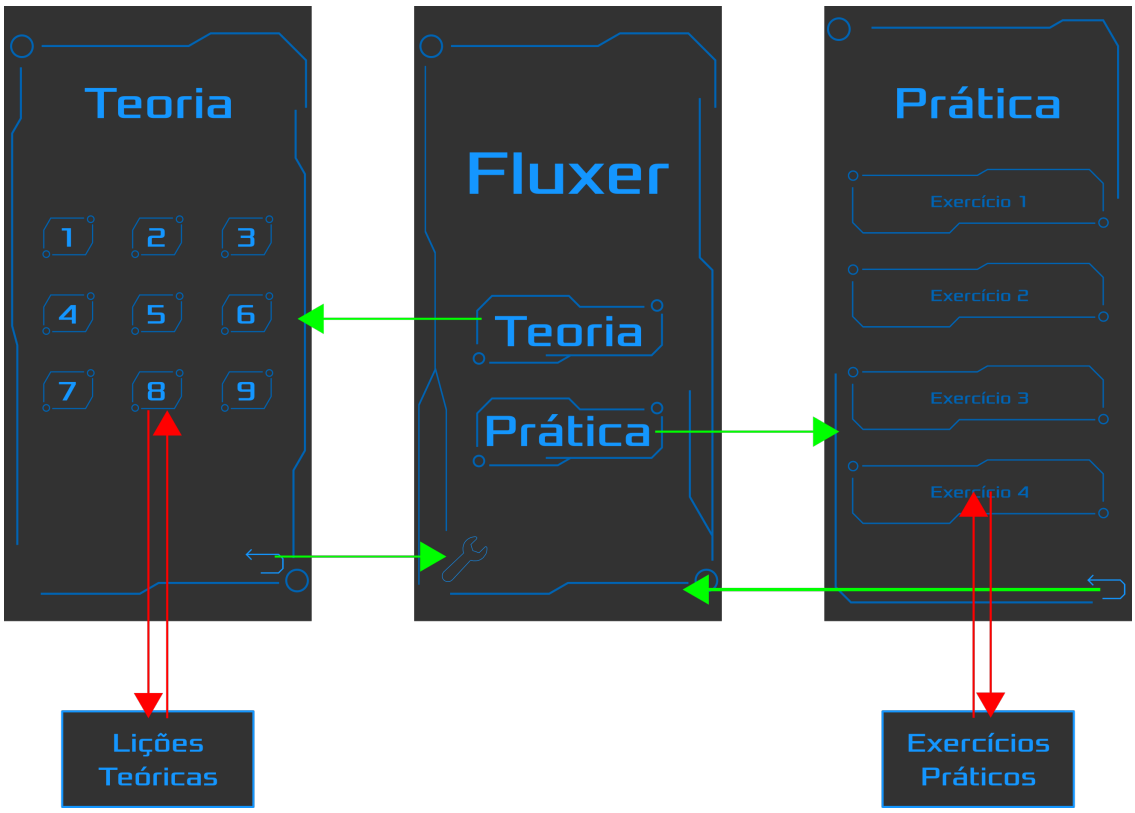
- 7 - Botão "limpar": Excluí todos os blocos conectados no fluxograma, exceto o componente "início"; e

- 8 - Botão "executar"/"parar": Quando clicado, inicia a execução. Enquanto o programa está neste estado não é possível interagir com nenhum outro elemento além deste botão até que ele termine. Se o botão é tocado novamente, a execução é interrompida.

4.5 Interfaces

O fluxo de telas do aplicativo é demonstrado na figura 8. Iniciando na tela de menu, o botão "teoria" leva para a página de seleção de aulas enquanto "prática" leva à lista de exercícios, ambas possuem um botão para retornar ao menu.

Figura 8 – Fluxo de telas



Fonte: Elaborada pelo autor

4.5.1 Progressão

A tela de teoria, no primeiro uso da aplicação, possui apenas a lição 1 disponível e ainda não há nenhum exercício aberto. A conclusão da primeira aula libera a lição 2 e em seguida a 3.

As lições 2 e 3 ainda são puramente teóricas e não liberam exercícios, mas o usuário desbloqueia a lição 4.

A quarta lição trabalha variáveis e desbloqueia o primeiro exercício, que trabalha as capacidades correspondentes.

A conclusão do exercício 1 libera um terço da primeira medalha e a lição 5, sobre entrada de dados.

A lição 5, por sua vez, desbloqueia o segundo exercício, cuja conclusão resulta no segundo terço da medalha e na lição 6.

A sexta lição explora saídas de dados e sua conclusão libera o terceiro exercício, que pratica todas as capacidades adquiridas até agora e finalmente libera a primeira medalha completa.

Em seguida, as lições 7, 8 e 9 tem uma progressão semelhante: O tópico trata

de um assunto teórico (Condicional, Repetição com "Para" e Repetição com "Enquanto", respectivamente) e libera o primeiro exercício relativo ao tópico. A conclusão de cada exercício libera outro subsequente e levemente mais difícil dentro do mesmo tópico, além de um terço de uma medalha. Após a terceira tarefa, o tópico está concluído e a próxima lição (se houver) é liberada.

Depois da conclusão das nove lições e dos doze exercícios, ainda há três exercícios extras. Este novo conjunto de tarefas trabalha os aspectos anteriores de maneira integrada e serve como desafio final para o usuário. Concluir os três exercícios finais garante a quinta e última medalha, imagens de todas as medalhas podem ser conferidas na figura 9.

Figura 9 – Coleção de medalhas

Fonte: Elaborada pelo autor

5 Conclusão

Gamificação é um conjunto de técnicas que proporciona experiências mais significativas e prazerosas aos participantes do contexto no qual ela é inserida. Por isso múltiplas áreas podem se beneficiar de métodos gamificados, como tratamento de pacientes em hospitais, interação com funcionários em ambiente administrativo, ensino alunos em salas de aula, ou o tópico abordado neste trabalho: métodos de aprendizado de programação.

Assim, o objetivo deste projeto foi alcançado. Com o levantamento bibliográfico foi possível se apropriar de conhecimentos de áreas diversas como gamificação, ensino de programação, impactos psicológicos dos métodos de ensino e como elementos lúdicos influenciam no aprendizado.

A aplicação desenvolvida utiliza algumas técnicas de gamificação pesquisadas, dotada de uma interface amigável e um método de ensino dinâmico que proporciona uma versão introdutória de cursos de algoritmos.

Trabalhos futuros no tema podem envolver a expansão da aplicação. Novas técnicas podem ser utilizadas como customizações de temas ou perfis com avatares personalizáveis. Com o trabalho de uma equipe profissional de *designers* é possível levar a experiência a outro nível de interatividade. Além disso, com mais tempo de desenvolvimento seria possível criar lições teóricas em elementos mais avançados, acompanhadas por exercícios que trabalhe estes novos recursos, entre os tópicos possíveis estão: tipagem de variáveis, manipulação de textos, *strings*, funções, entre outros.

Referências

AMBROSIO, A.; COSTA, F.; ALMEIDA, L.; FRANCO, A.; MACEDO, J. Identifying cognitive abilities to improve cs1 outcome. *Proceedings - Frontiers in Education Conference*, 10 2011.

C#. 1991. <https://docs.microsoft.com/pt-br/dotnet/csharp/getting-started/>. Accesso em: 2020-11-24.

COMBEFIS, S.; BERESNEVICIUS, G.; DAGIENE, V. Learning programming through games and contests: Overview, characterisation and discussion. *Olympiads in Informatics*, v. 10, p. 39–60, 07 2016.

COSTA, C. J.; PEREIRA, R.; APARICIO, J. T. The study of gamification application architecture for programming language course. In: *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.]: IEEE, 2017. p. 1–6.

DETERDING, S. Make-believe in gameful and playful design. In: _____. [S.l.: s.n.], 2016. p. 101–124. ISBN 978-3-319-29551-0.

DETERDING, S.; KHALED, R.; NACKE, L.; DIXON, D. Gamification: Toward a definition. In: *CHI 2011 Gamification Workshop Proceedings*. [S.l.: s.n.], 2011a. p. 12–15.

DETERDING, S.; O'HARA, K.; SICART, M.; DIXON, D.; NACKE, L. Gamification: Using game design elements in non-gaming contexts. In: *CHI 2011 Gamification Workshop Proceedings*. [S.l.: s.n.], 2011b. p. 2425–2428.

FLECK, R. *Choosing colors for web design: A practical UI color application guide*. 2018. <https://dribbble.com/stories/2018/12/19/choosing-colors-for-web-design-a-practical-ui-color-application-guide>. Accesso em: 2020-11-24.

HAMARI, J.; KOIVISTO, J.; SARSA, H. Does gamification work? – a literature review of empirical studies on gamification. In: *2014 47th Hawaii International Conference on System Sciences*. [S.l.: s.n.], 2014. p. 3025–3034.

INKSCAPE. 2003. <https://inkscape.org/pt-br/sobre/>. Accesso em: 2020-11-24.

KHALEEL, F. L.; ASHAARI, N. S.; MERIAM, T. S.; WOOK, T.; ISMAIL, A. The study of gamification application architecture for programming language course. In: *IMCOM '15: The 9th International Conference on Ubiquitous Information Management and Communication*. [S.l.]: Association for Computing Machinery, 2015. p. 1–5.

MAIA, R.; GRAEML, F. Playing and learning with gamification: An in-class concurrent and distributed programming activity. In: . [S.l.: s.n.], 2015.

PITEIRA, M.; COSTA, C. J. Gamification: Conceptual framework to online courses of learning computer programming. In: *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2017. p. 1–7.

RAVAJA, N.; SAARI, T.; SALMINEN, M.; LAARNI, J.; KALLINEN, K. Phasic emotional reactions to video game events: A psychophysiological investigation. *Media Psychology - MEDIA PSYCHOL*, v. 8, p. 343–367, 11 2006.

UNITY. 1995. <https://unity.com/pt>. Accesso em: 2020-11-24.

VISUALSTUDIO. 1991. <https://visualstudio.microsoft.com/pt-br/vs/>. Accesso em: 2020-11-24.

Apêndices

APÊNDICE A – Material Teórico

As seções a seguir apresentam uma transcrição literal das lições teóricas presentes no aplicativo, adaptações mínimas foram realizadas em relação a formatação e citação de figuras. A linguagem utilizada é de caráter informal para preservar a experiência que o texto objetiva proporcionar.

A.1 Algoritmos

Durante a vida cotidiana, é comum que você se depare com diversos problemas. Arrumar a cama, fazer uma prova ou preparar um lanche não soam como coisas similares, mas podem ser reduzidos a atividades realizadas com o objetivo de resolver algum problema.

Vamos analisar mais atentamente um problema comum: Suponha que você esteja com fome e não há nada já preparado em casa. Há diversas maneiras de resolver esta questão, pode-se pedir algo por delivery, sair para comprar comida ou preparar algo com os ingredientes disponíveis.

E dentro destas opções ainda há mais possibilidades, afinal existe uma infinidade de tipos de comida que podem ser pedidos/preparados.

Ao analisar todas as opções, vamos supor que você optou por preparar algo e escolheu um sanduíche. Desta forma o problema "Estar com fome" será solucionado por "Fazer um sanduíche e comê-lo".

Agora vamos destrinchar o processo de fazer um sanduíche. Primeiro definimos os ingredientes.

Por exemplo: Pão, alface, hambúrguer e queijo. Em seguida, para que se tornem o produto final, tais ingredientes devem passar por um processo: Cortar o pão em duas fatias, passar maionese dentro do pão, fritar o hambúrguer, colocá-lo na fatia inferior do pão, cobri-lo com uma fatia de queijo, depois uma folha de alface e colocar a fatia superior do pão por cima.

Note que o processo anterior é composto de uma série de instruções e depende de uma certa ordem. Cortar o pão é a instrução 1, passar maionese é a instrução 2 e obviamente não é possível realizar o passo 2 se o 1 não foi realizado.

Observe também que este processo está descrito em palavras, mais especificamente em português brasileiro. Mas também poderia ser escrito em inglês ou japonês, ou ser expresso através de áudio ou desenhos. Em outras palavras, ele pode ser representado através de linguagem.

Você também pode ter notado que nem todos os passos dependem estritamente do anterior. É possível colocar o alface antes do queijo, por exemplo, mas essa seria a descrição de um processo diferente. Além disso há outros passos que podem ser adicionados, retirados ou modificados a seu gosto, afinal existem várias formas de fazer um sanduíche. Mas este exercício se propõe a analisar apenas uma delas.

Toda a discussão sobre preparar o sanduíche poderia ser aplicada a qualquer outro problema diário e provavelmente chegaríamos a conclusões semelhantes sobre os mesmo pontos. Ou seja, se analisarmos um problema o suficiente podemos elaborar uma solução e representá-la em uma sequência de instruções ordenadas expressas através de linguagem.

Ou seja, somos capazes de produzir um algoritmo que resolva tal problema.

A.2 Entrada, Processamento e Saída

Agora, vamos trazer o conceito de algoritmos diretamente para o mundo dos softwares. . .

Tome como exemplo um aplicativo de calculadora. A utilização dele compreende três passos básicos:

- 1 - Um cálculo é informado;
- 2 - O aplicativo determina qual o resultado correspondente;
- 3 - O resultado para a conta é apresentado na tela.

Mas e para uma aplicação obviamente mais complexa? Pense em todas as vezes que você usou a barra de busca do Google. O processo também é bem simples se for resumido a:

- 1 - Palavras-chave são enviadas;
- 2 - Os sistemas da Google obtém tais palavras e decidem resultados;
- 3 - A lista dos resultados é exibida.

Em geral, qualquer software lida com as mesmas três fases básicas:

- 1 - Ele recebe um conjunto de dados, o que é chamado de entrada ou *input*;
- 2 - A entrada passa por uma série de processos que foram escritos em forma de código. O que é chamado de processamento;
- 3 - Por fim o resultado do passo anterior é manifestado, seja apresentando informações ao usuário, provocando algum evento, emitindo uma instrução, etc. O que é denominado saída ou *output*.

Como algoritmos se relacionam com todo esse conceito? Simples! Um algoritmo para um programa de computador é justamente a descrição do processamento.

Ou seja, o algoritmo é a maneira de elaborar todas as instruções que o computador deve realizar para transformar uma entrada em uma saída.

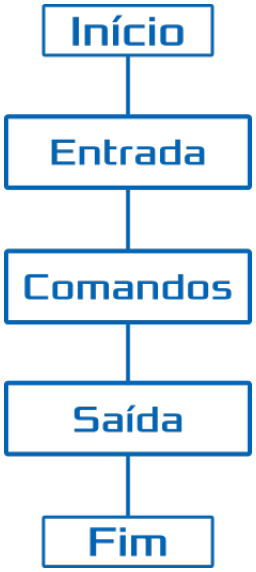
A.3 Fluxogramas

Na próxima lição começaremos com os conceitos de lógica e programação e com isso teremos exercícios práticos. Para que você visualize melhor os algoritmos na prática, os componentes serão representados de forma gráfica.

Uma visão gráfica de um algoritmo é chamada de fluxograma. Nele, as diferentes instruções são representadas como uma série de formas geométricas ligadas por linhas (exemplificado na figura 10). A execução será acompanhada por um pequeno ponto que indica qual passo do algoritmo está sendo realizado. Em geral a ordem de execução dos comandos vai de cima para baixo.

Controles gerais da interface no aplicativo são representados na figura 11.

Figura 10 – Exemplo de fluxograma.

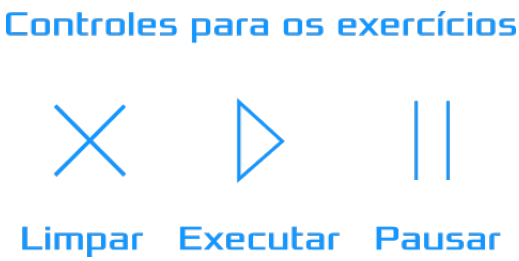


Fonte: Elaborada pelo autor

A.4 Variáveis

Como já vimos, a sequência de entrada, processamento e saída de um programa lida com um conjunto de dados.

Figura 11 – Controles da interface de exercícios.



Fonte: Elaborada pelo autor

O computador recebe estes dados e realiza várias transformações neles até obter a saída desejada. Mas para isto, ele precisa "se lembrar" destes dados. Em outras palavras, ele precisa reservar uma posição em sua memória para guardar qualquer tipo de informação importante.

É trabalho da pessoa que está fazendo o algoritmo dizer como e qual informação deve ser armazenada e o momento que isso deve ocorrer. Para tal, precisamos criar variáveis.

Uma variável é nada mais que uma posição reservada na memória do computador para guardar informações úteis. Pense nelas como se fossem caixas vazias e rotuladas, utilizadas para organizar papéis, diferentes peças, etc.

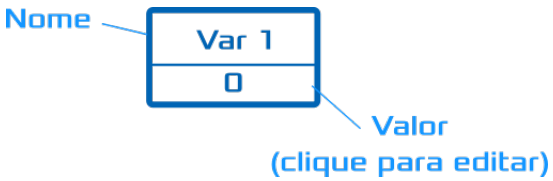
Quando você criar uma dessas caixas virtuais, ela sempre deve ser associada a um nome. Nunca deve existir duas variáveis com o mesmo nome, pois o computador não saberia diferenciar uma da outra.

Ao ser criada, a variável está inicialmente vazia. Você pode colocar qualquer informação dentro dela: um número, uma letra, uma frase, etc. Durante a execução do programa, o valor armazenado pode mudar, seja porque ela guardou uma entrada do usuário, o resultado de algum cálculo ou qualquer outra coisa.

O conteúdo varia com a execução, por isso chamamos de variável. Quando se faz o algoritmo, frequentemente queremos utilizar um valor que ainda não sabemos qual será, entretanto sabemos que ele estará na variável de nome x, por isso podemos trabalhar com ele.

Para nossos exemplos e exercícios, variáveis serão representadas como na figura 12.

Figura 12 – Representação das variáveis.



Fonte: Elaborada pelo autor

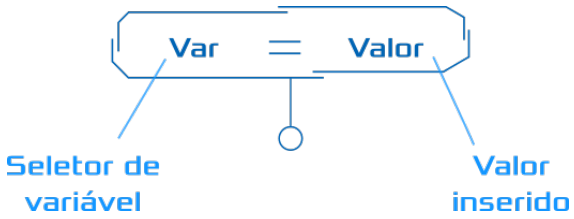
Para simplificar, os exercícios usam no máximo 10 variáveis. Elas são identificadas pelo nome "var" seguido de um número.

A.5 Entrada de Dados

Quando testamos um programa, é comum que nós mesmos cuidemos da inserção do input. Por isso, nos exercícios a seguir, podemos determinar qual a entrada do programa usando as variáveis

Além disso, podemos utilizar o elemento de atribuição (representado na figura 13) para colocar valores em variáveis durante a execução.

Figura 13 – Representação da entrada de dados.

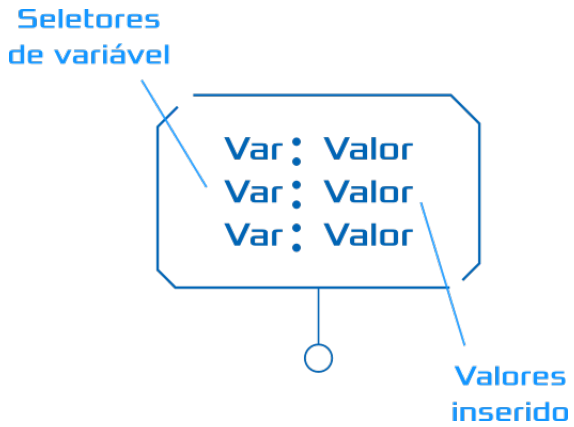


Fonte: Elaborada pelo autor

A.6 Exibindo Informações

Já vimos como cuidamos da parte de entrada de dados, agora vamos ver a saída. Para nossos exercícios, vamos exemplificar a saída com a escrita de alguma informação na tela como na figura 14.

Figura 14 – Representação da saída de dados.



Fonte: Elaborada pelo autor

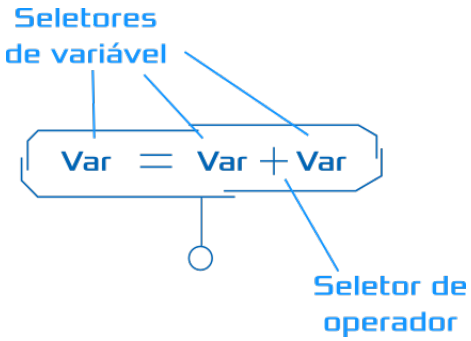
Lembre-se que para um programa a saída não precisa necessariamente ser uma resposta para o usuário. Pense que a saída de um caso real, poderia ser um dado armazenado, uma instrução para alguma máquina ou até mesmo a entrada de outro programa.

A.7 Operadores Aritméticos

Já vimos que o algoritmo é a representação de uma sequência de ações. Estas ações frequentemente envolvem cálculos: seja porque o programa precisa mostrar o resultado de uma conta, contar quantas vezes um comando foi repetido, etc.

Toda vez que o exercício exigir alguma operação, o fluxograma inclui uma imagem como na figura 15.

Figura 15 – Representação do comando operador.



Fonte: Elaborada pelo autor

As operações possíveis com este componente são descritas na figura 16.

Figura 16 – Representação das operações.

Fonte: Elaborada pelo autor

A.8 Comando Condicional

Além de fazer cálculos, muitas vezes devemos permitir que o computador faça decisões. Por exemplo, se estivesse programando um relógio digital, a cada segundo a variável que

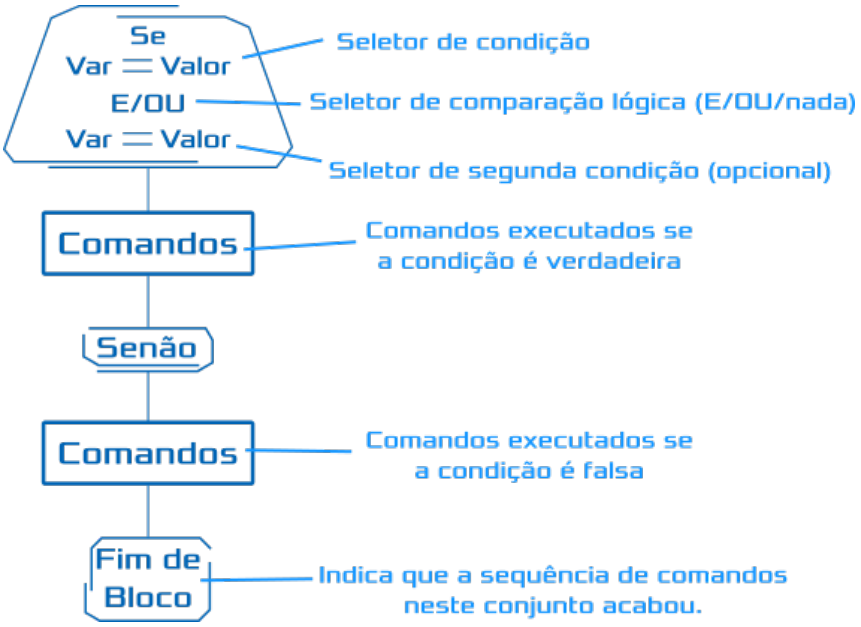
armazena os segundos aumentaria em 1. Porém o computador deve verificar que quando ela chegar em 60, deve voltar a zero e a variável dos minutos que precisa aumentar em 1.

Uma forma simples de lidar com decisões é utilizar um comando condicional, composto por três partes: se, então e senão.

Este comando lida com o seguinte formato: Se uma condição é atendida, então faça uma ação. Senão faça outra ação. Exemplo: Se a variável de segundos é igual a 60, então variável de segundos se torna 0. Senão, variável de segundos aumenta em 1.

Para nossos fluxogramas a estrutura de condicional é demonstrada na figura 17.

Figura 17 – Representação do comando condicional.



Fonte: Elaborada pelo autor

Os operadores possíveis em uma condição estão na figura 18.

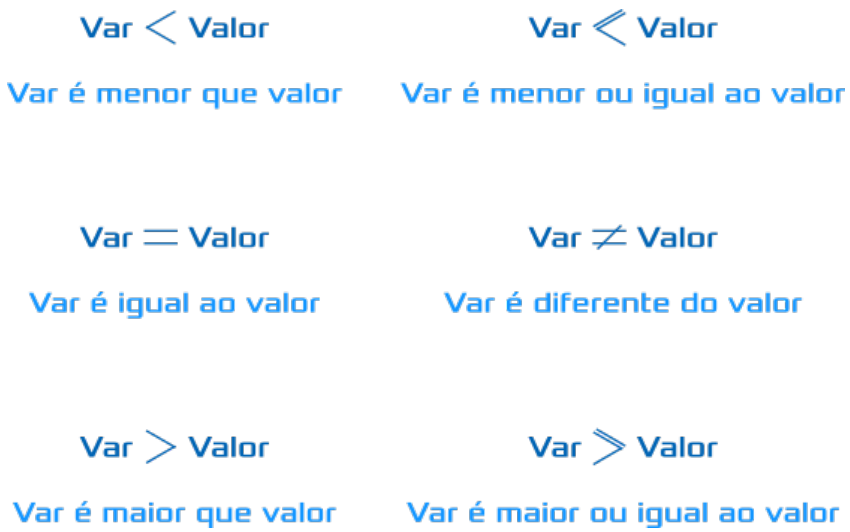
Os comparadores lógicos possíveis em uma condição são descritos na figura 19.

A.9 Comandos de Repetição

A este ponto já aprendemos vários comandos que podemos instruir ao computador, entretanto ainda há um ponto a ser discutido. Assim como qualquer máquina, computadores foram feitos para facilitar a execução de tarefas, ou mesmo automatizá-las totalmente.

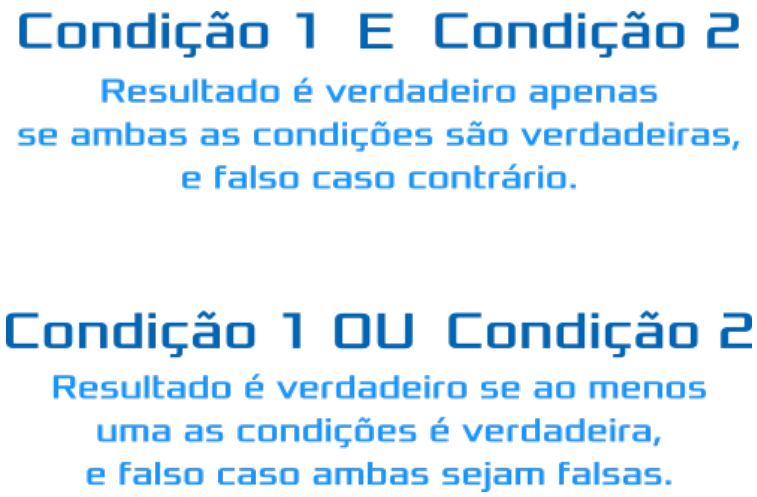
Imagine que você está fazendo um programa para calcular o fatorial para qualquer número inserido pelo usuário. Se a entrada for o número cinco, você poderia fazer 5 x 4 x 3 x 2 x 1. Simples, certo?

Figura 18 – Representação dos operadores condicionais.



Fonte: Elaborada pelo autor

Figura 19 – Representação dos comparadores lógicos.



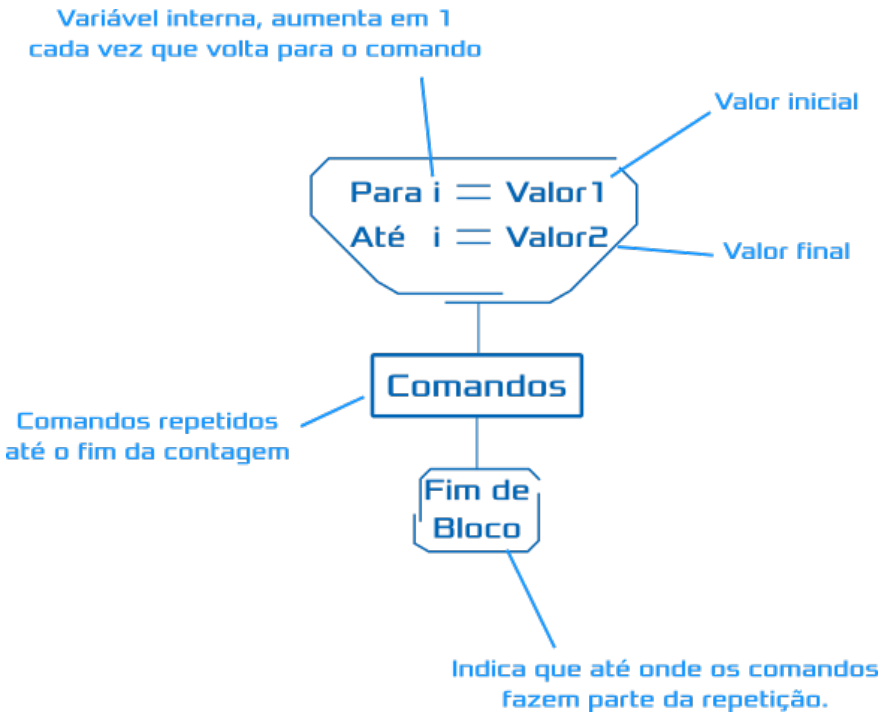
Fonte: Elaborada pelo autor

Mas o usuário pode digitar qualquer número no input. Supondo que a maior entrada possível seja 1000, você teria que fazer mil segmentos "se, então, senão" encadeados. O que não seria nada prático e consumiria uma quantidade desnecessária de tempo e memória.

Por isso, frequentemente utilizamos comandos de repetição. Ou seja, um comando que indica quantas vezes determinada parte do algoritmo deve ser repetida, baseado em alguma condição.

O primeiro comando que temos é o chamado "para" demonstrado na figura 20. Ele trabalha com uma variável de contador, utilizada para indicar quantas vezes o segmento deve ser repetido. A forma dele é algo semelhante a "para: contador iniciando em x; até que: contador seja igual a y; faça: [instruções];".

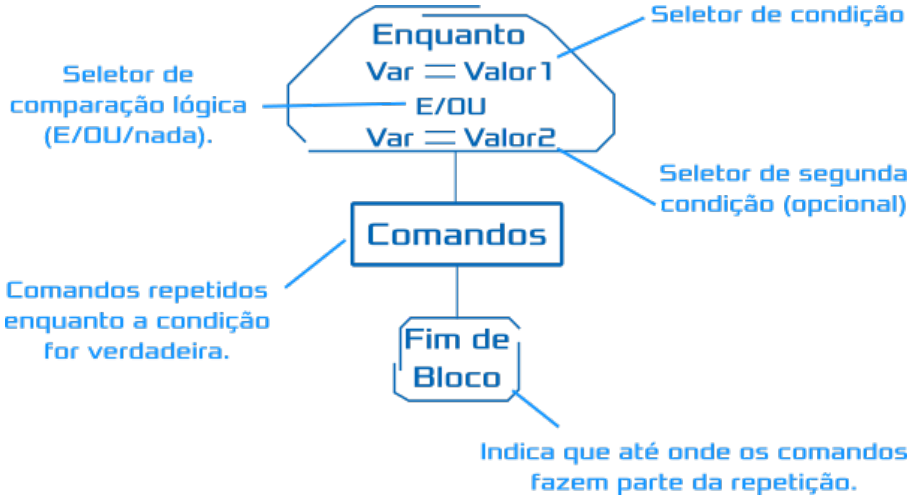
Figura 20 – Representação do comando "Para".



Fonte: Elaborada pelo autor

O segundo comando que temos é o "enquanto" (figura 21). Diferente do anterior, ele é utilizado quando não sabemos ao certo quantas repetições devem ser realizadas. Seu formato é: "enquanto: condição não for atendida; faça: instruções".

Figura 21 – Representação do comando "Enquanto".



Fonte: Elaborada pelo autor