Instituto de Matemática e Estatística - IME/USP
Bacharelado em Ciência da Computação - BCC

Uma Ferramenta de Simulações Interativas para Ensino de Computação para Crianças

Relatório Semestral de Iniciação Científica

Aluna: Marília Takaguti Dicezare
Orientadora: Profa. Dra. Kelly Rosa Braghetto

Resumo do Projeto Proposto

O projeto de pesquisa inicial propôs a criação de um protótipo de uma ferramenta web de simulações interativas de conceitos de lógica de programação para o ensino de computação no ensino fundamental. Junto a cada simulação, deve estar integrada a visualização do pseudocódigo associado a fim de propiciar aos usuários um contato com a estrutura real de um código de programação. O ambiente de cada simulação e a sua tradução em pseudocódigo devem permitir que os estudantes as explorem livremente.

Dessa forma, iremos desenvolver um arcabouço robusto que contenha pelo menos uma simulação de conceitos de lógica de programação, e que sirva como base para a implementação de novas simulações no futuro, juntamente com uma documentação.

O sistema está sendo arquitetado com o estilo arquitetural Portas e Adaptadores ou Arquitetura Hexagonal (*Ports and Adapters* ou *Hexagonal Architecture*), onde a comunicação entre drivers e a aplicação é feita através de adaptadores específicos em portas da aplicação (Cockburn, 2005). Além disso, o arcabouço está sendo desenvolvido com base no padrão arquitetural MVC (*Model-View-Controller*), o qual se apresenta em camadas (modelo, visão e controlador) e proporciona o isolamento entre as regras de negócio e a interface gráfica do usuário.

Pretendemos avaliar a usabilidade do protótipo proposto realizando validações com educadores e estudantes através de questionários de usabilidade, como o SUS (*System Usability Scale*) e o emoti-SAM (*Self-Assessment Manikin*).

Resumo das Atividades do Relatório Semestral Anterior

Inicialmente, foi realizado um breve estudo e comparação entre os três *frameworks* JavaScript de código aberto: Angular, React e Vue, este último escolhido para ser utilizado no desenvolvimento da ferramenta proposta. Escolhemos o Vue para o desenvolvimento do projeto por apresentar uma boa documentação, ser escalável e fácil de aprender.

Em seguida, criamos a arquitetura das classes do sistema. Para isso, definimos as abstrações que queríamos implementar e organizamos em módulos: estrutura do framework, simulações, gerador de pseudocódigo, entidades de simulação e controlador. O diagrama de classes foi projetado com base nessas diretrizes, e a arquitetura hexagonal do sistema foi

ajustada para melhor representar as separações das abstrações, mantendo a separação entre a lógica de negócio e as entidades externas.

Descrição das Atividades Realizadas a partir do Relatório Anterior

1. Modificação na Arquitetura do Sistema

Uma das funcionalidades que queremos desenvolver na ferramenta proposta é a geração do pseudocódigo correspondente a cada simulação. A ideia de tradução de linguagens, da representação visual em pseudocódigo, está contida na definição de um interpretador de linguagens de programação, como temos em computação. Um interpretador, de forma simplificada, traduz o código fonte em alguma representação intermediária e, em seguida, a executa.

Dessa forma, decidimos alterar a parte da arquitetura do arcabouço correspondente a geração do pseudocódigo de modo que o arcabouço contenha um interpretador de linguagens. A Figura 1 mostra o diagrama de pacotes e módulos UML da arquitetura do sistema. Nele, é possível observar um novo módulo que irá compor o *Core* da aplicação, o *AbstractSyntaxTree*, que corresponde a representação intermediária da linguagem. Além disso, é possível notar dois novos pacotes no *Framework*: o *Parser*, que será responsável pela tradução da representação visual inicial para a representação intermediária, e o *Operations*, o qual irá realizar operações sobre a representação intermediária para gerar a linguagem final (pseudocódigo).

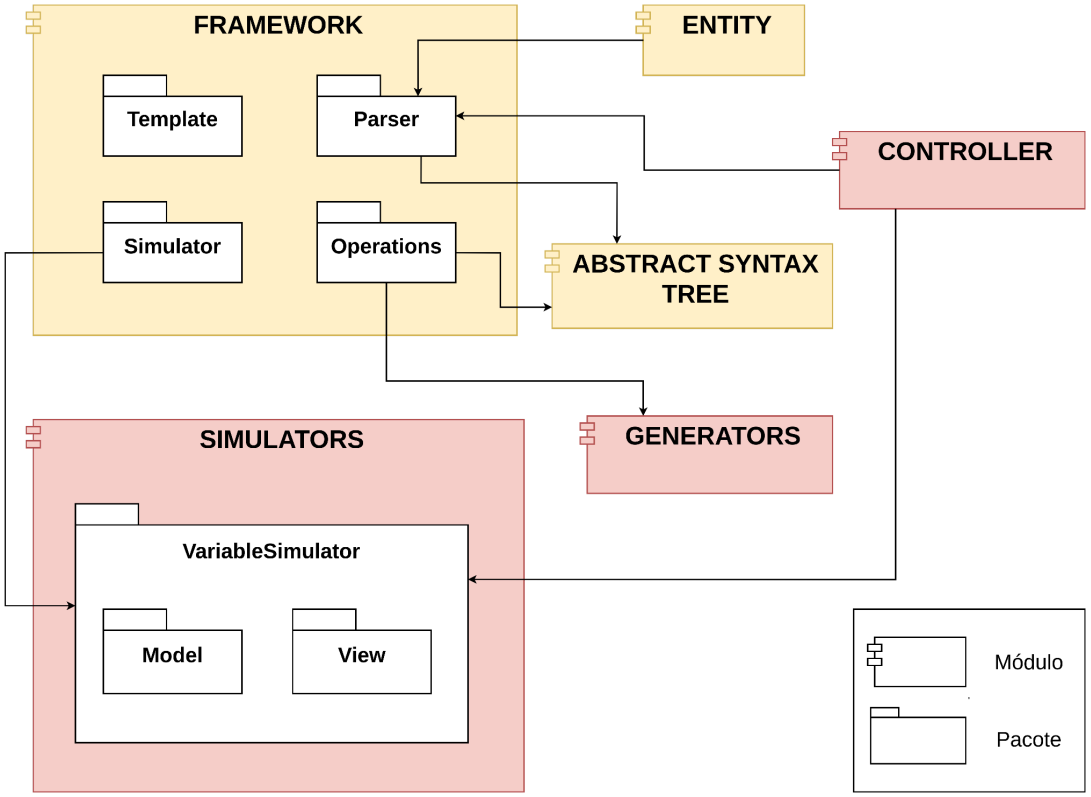

Figura 1 - Diagrama de pacotes e módulos UML da arquitetura do sistema.

De forma mais detalhada, a Figura 2 mostra o diagrama de classes UML das alterações realizadas na arquitetura. O módulo *Entity* implementa a representação visual das simulações que é recebida pelo *Parser* para ser traduzida para a representação intermediária. O *Parser* é modelado pelo padrão de projetos *Interpreter*, o qual define a gramática para a linguagem intermediária e um tradutor de expressões. Por sua vez, o módulo *AbstractSyntaxTree* é modelado pelo padrão *Composite*, o qual apresenta uma estrutura recursiva. Por fim, o padrão *Visitor* representa as operações realizadas sobre a linguagem intermediária, como a *SemanticAnalyser*, que irá validar a consistência semântica da linguagem, e o *CodeGenerator*, responsável por gerar o resultado final da tradução das linguagens.

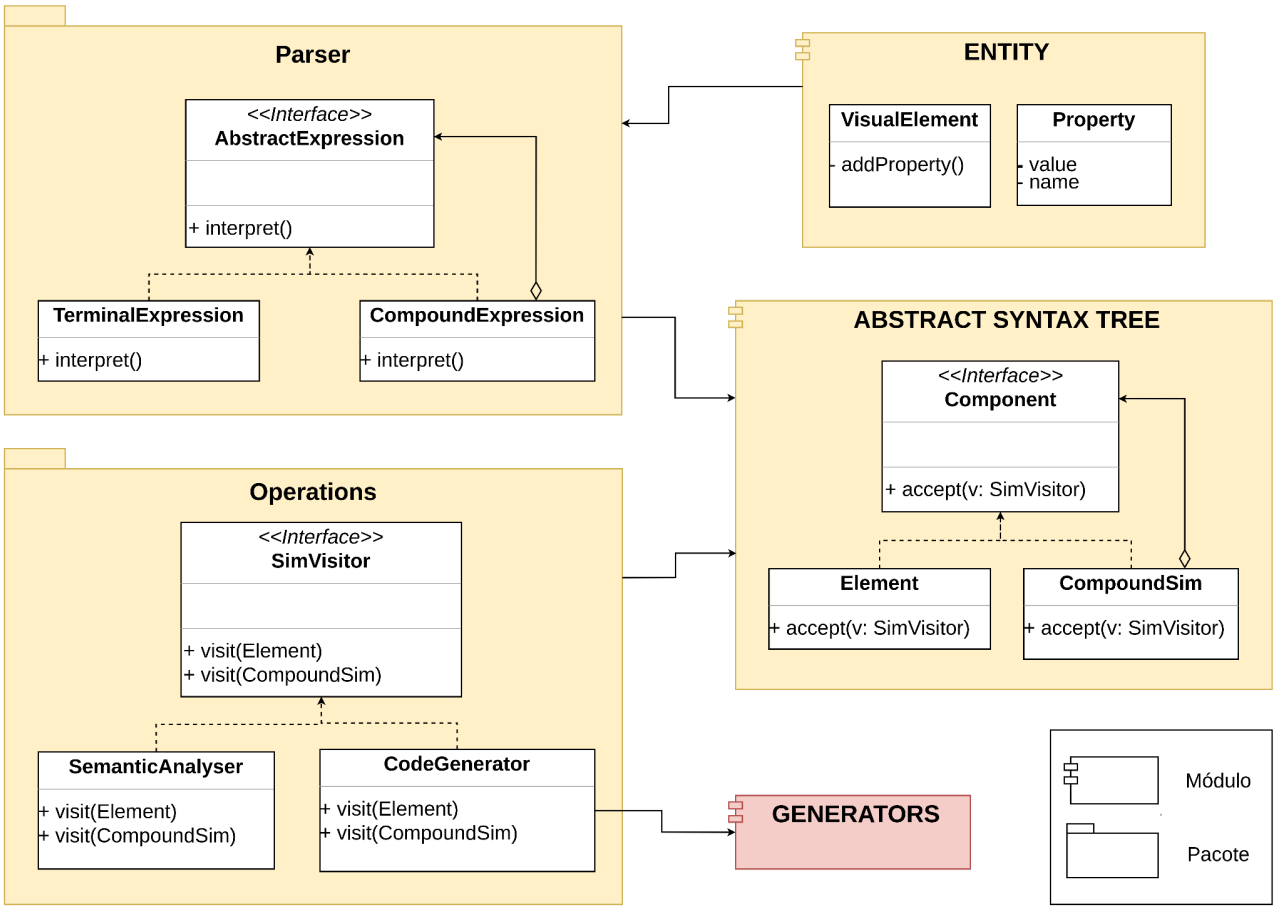

Figura 2 - Diagrama de classes UML da arquitetura do interpretador de linguagens do sistema.

É importante observar que a implementação da tradução da linguagem intermediária para a final é feita de fato no módulo *Generators* para garantir a separação da lógica de negócios central da aplicação. Isso possibilitará a implementação da geração não só do pseudocódigo, mas também a implementação futura da tradução em outras linguagens de programação, proporcionando escalabilidade à ferramenta.

2. Desenho do Protótipo de Alta Fidelidade no Figma

Para guiar o desenvolvimento da simulação de variáveis, foi criado um protótipo de alta fidelidade da ferramenta proposta a partir dos requisitos funcionais definidos no início do projeto. O protótipo foi construído na ferramenta de design de interface colaborativa Figma, onde foi

definida a interface do usuário e alguns aspectos de interação e fluxo de uso, e pode ser encontrado neste link.

A Figura 3 apresenta a interface da página inicial da ferramenta web com simulações interativas, intitulada no momento de "*Logic Sims*". Esta página permite que o usuário escolha uma simulação relacionada a um conceito de programação específico, além de contar com menus de informações, configuração e ajuda no canto superior direito dela.

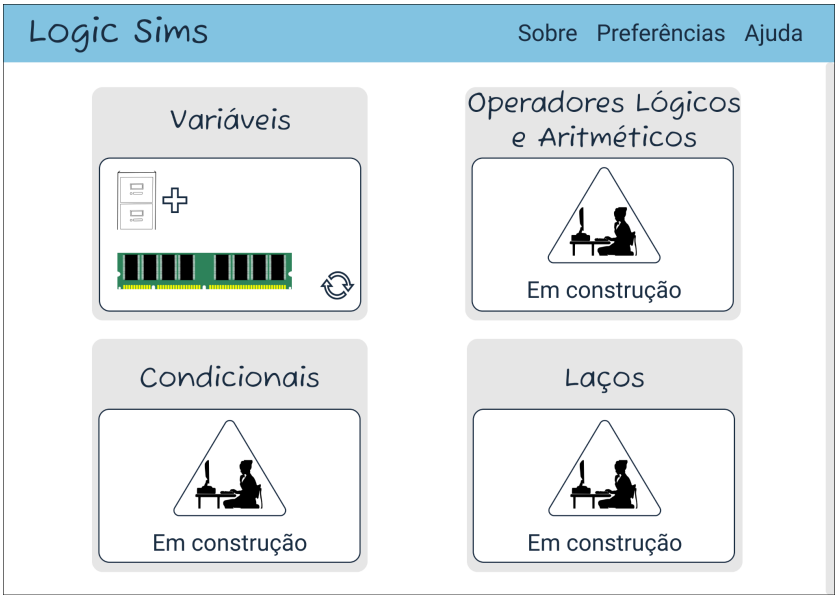

Figura 3 - Interface do usuário da página inicial da ferramenta web com simulações interativas de conceitos de lógica de programação.

Ao clicar em qualquer uma das opções de simulação disponíveis, o usuário será direcionado para a página correspondente. Todas elas irão apresentar um *template* similar (Figura 4a), com uma barra de navegação contendo o nome da simulação e menus de informações e configurações, uma área de trabalho para interação com a simulação e um painel opcional com o pseudocódigo correspondente. Em particular, a simulação de variáveis de programação será iniciada com uma representação de um objeto de armazenamento (armário com gavetas), com a opção de adicionar novos elementos, e uma representação visual da memória de um computador, como mostra a Figura 4b.

Ainda, selecionando uma das gavetas do armário, uma caixa de diálogo será aberta para que o usuário crie uma nova variável, digitando seu nome e valor (Figura 4c). Ao selecionar "OK", a gaveta selecionada para a criação da variável receberá em seu exterior um rótulo com o nome dela e seu valor será armazenado em seu interior. Ao mesmo tempo, esse valor será armazenado em um espaço "aleatório" da memória e o pseudocódigo correspondente à criação da variável será gerado no painel à direita da simulação (Figura 4d).

Outras possíveis interações com a simulação de variáveis é a adição de novos armários com gavetas, como já mencionado, e a reinicialização do estado inicial dela através do botão de *reset* no canto inferior direito da área da simulação.

É importante lembrar que as simulações devem permitir que os usuários as explorem livremente. Portanto, objetos clicáveis serão destacados através do ponteiro do mouse. Outro aspecto importante desse protótipo é a seleção de cores que compoẽm todos os elementos das

páginas da ferramenta, que foram escolhidas de acordo com as Diretrizes de Acessibilidade de Conteúdo Web (Web Content Accessibility Guidelines - WCAG), através da plataforma Coolors[1].

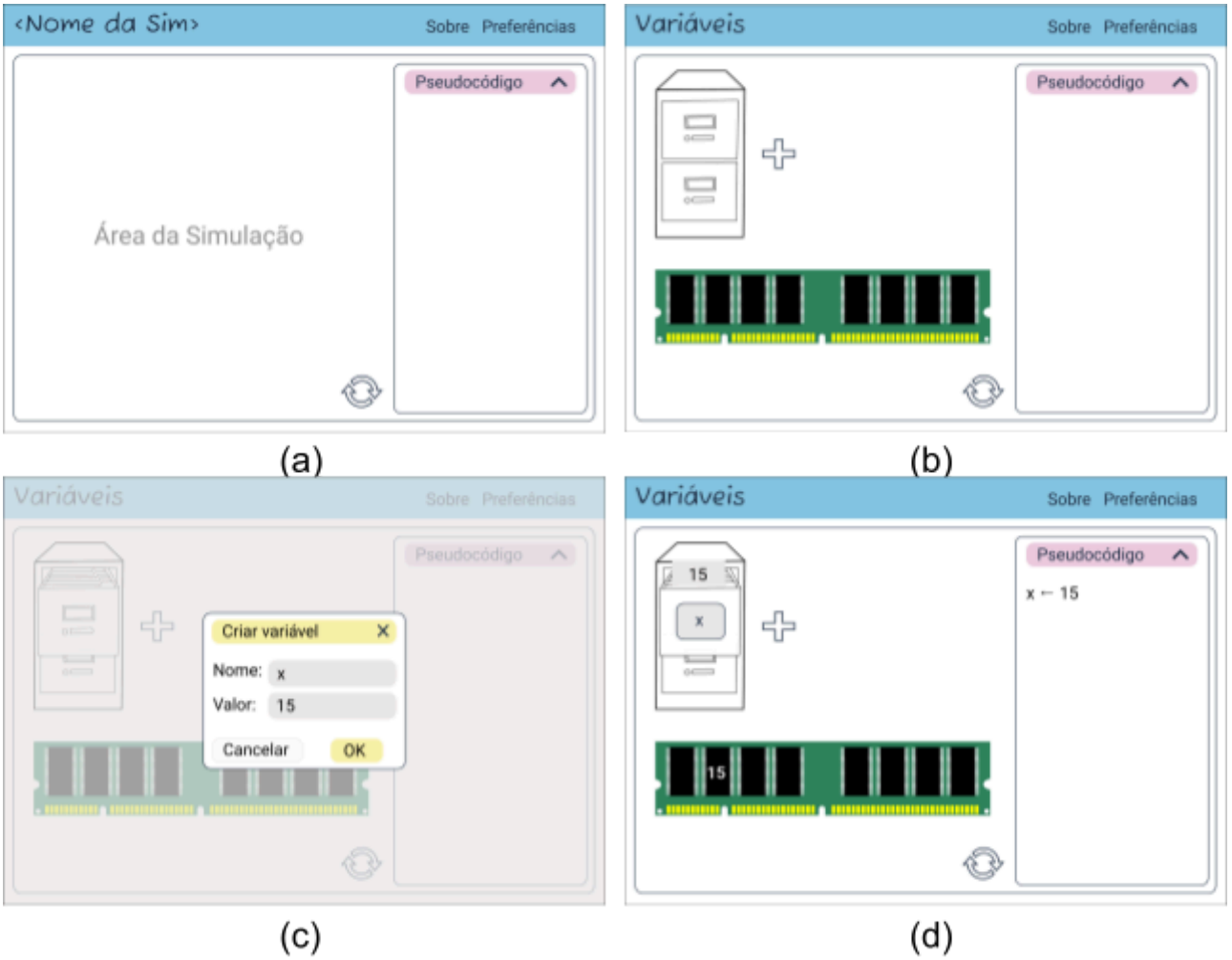

Figura 4 - (a) *Template* das páginas de simulação dos conceitos de lógica de programação. (b) Página da simulação de variáveis de programação. (c) Criação de uma nova variável de programação. (d) Armazenamento da variável de programação na representação visual da memória e geração do pseudocódigo correspondente.

Próximos Passos

Essa Iniciação Científica permitiu a realização de pesquisas sobre o ensino da computação na educação e as ferramentas de apoio pedagógico existentes, o levantamento dos requisitos funcionais e não-funcionais da ferramenta proposta, o desenvolvimento dos esquemas de arquitetura projetados e dos protótipos criados. A partir desse ponto de partida, pretendo seguir com o desenvolvimento do arcabouço proposto e das simulações no Trabalho de Conclusão de Curso (TCC) durante o ano de 2024.

Ademais, um aspecto importante do desenvolvimento das simulações se refere a como representar visualmente os conceitos de lógica de programação, o que também será pesquisado mais a fundo durante o desenvolvimento do TCC.

[1] https://coolors.co/contrast-checker/112a46-acc8e5

Referências Bibliográficas

Cockburn, A. (2005). Hexagonal architecture. Disponível em: https://alistair.cockburn.us/hexagonal-architecture/. Último acesso em: 26/09/2022.

Shvets, A. (2014-2023). Refactoring Guru. Composite. Disponível em: https://refactoring.guru/design-patterns/composite. Último acesso em: 11/11/2023.

Shvets, A. (2014-2023). Refactoring Guru. Visitor. Disponível em: https://refactoring.guru/design-patterns/visitor. Último acesso em: 11/11/2023.

SourceMaking.com. (2007-2023). Source Making. Interpreter Design Pattern. Disponível em: https://sourcemaking.com/design_patterns/interpreter. Último acesso em: 11/11/2023.

Toal, R. (2023). Introduction to Compilers. Disponível em: https://cs.lmu.edu/~ray/notes/introcompilers/. Último acesso em: 10/11/2023.