

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

VICTOR PASSOS DE PINHO

Assistente Virtual Personalizado para o iVProg

**São Paulo
2021**

VICTOR PASSOS DE PINHO

Assistente Virtual Personalizado para o iVProg

**Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo**

**São Paulo
2021**

VICTOR PASSOS DE PINHO

Assistente Virtual Personalizado para o iVProg

**Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo**

**Área de Concentração: Engenharia
Elétrica com Ênfase em
Computação**

**Orientadora: Professora Doutora
Anarosa Alves Franco Brandão**

**Coorientador: Mestrando Lucas
Mendonça de Souza**

**São Paulo
2021**

Pinho, Victor Passos de

Assistente Virtual Personalizado para o iVProg / V. P. Pinho -- São Paulo, 2021.

64 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Engenharia Elétrica com Ênfase em Computação 2.Engenharia de Software 3.Educação I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

RESUMO

Desejou-se desenvolver um sistema que se utilize de possibilidades que o ambiente virtual traz em um ambiente virtual de aprendizado, de forma que o ambiente seja mais do que um repositório de conteúdo e do que um local para realizar atividades avaliativas. A possibilidade escolhida foi a de permitir que os professores possam fornecer uma assistência personalizada para cada aluno mesmo sem ter o tempo disponível para isso. Portanto foi desenvolvido um assistente virtual personalizado para o sistema educacional iVProg, que é destinado ao ensino e à aprendizagem de algoritmos e programação para iniciantes, e que possui integração com o ambiente Moodle. O assistente apresenta diagnósticos sobre o conhecimento do aluno, com dicas e caminhos de como melhorar nos pontos fracos, ou seja, o assistente tem um comportamento semelhante ao de um tutor. Este trabalho apresenta o projeto, a implementação, os testes e o resultado do assistente desenvolvido, e apresenta algumas perspectivas de continuidade.

Palavras Chave: iVProg. Ambiente virtual de aprendizado. Assistente virtual de ensino.

ABSTRACT

The aim of this project was to develop a system that uses the possibilities that the virtual environment brings in a virtual learning environment, so that the environment is more than a content repository and a place to carry out evaluation activities. The option chosen was to allow teachers to provide personalized assistance to each student even without having the time available for this. Therefore, a personalized virtual assistant was developed for the iVProg educational system, which is intended for learning and teaching algorithms and programming for beginners, and which has integration with the Moodle environment. The assistant presents diagnoses about the student's knowledge, with tips and ways to improve on weaknesses, that is, the assistant has a behavior similar to a tutor. This work presents the project, the implementation, the tests and the result of the developed assistant, and presents some continuity perspectives.

Keywords: iVProg. Virtual learning environment. Virtual teaching assistant.

SUMÁRIO

1	INTRODUÇÃO	7
1.1	MOTIVAÇÃO	7
1.2	OBJETIVO	7
1.3	JUSTIFICATIVA	8
1.4	ORGANIZAÇÃO DO TRABALHO	8
2	ASPECTOS CONCEITUAIS	9
2.1	EDUCAÇÃO	9
2.2	SISTEMA DE APRENDIZADO VIRTUAL	10
2.3	ASSISTENTE VIRTUAL INTELIGENTE	11
2.4	PERSONALIZAÇÃO DE CONTEÚDO	12
2.5	EXPERIÊNCIA DO USUÁRIO (EU)	12
2.6	PROGRAMAÇÃO VISUAL	13
2.7	INTERAÇÃO HUMANO-COMPUTADOR (IHC)	13
2.8	AVALIAÇÃO AUTOMÁTICA	14
3	TECNOLOGIAS UTILIZADAS	15
3.1	IVPROG	15
3.2	MOODLE	15
3.3	IASSIGN	17
3.4	JAVASCRIPT	18
3.5	NODE.JS	19
3.6	PHP	20
3.7	MARIADB	20
4	METODOLOGIA DO TRABALHO	22
4.1	CONCEPÇÃO	22
4.2	FAMILIARIZAÇÃO	22
4.3	PROJETO	23
4.4	IMPLEMENTAÇÃO	23
4.5	TESTES	24
4.6	AJUSTES	24

5	ESPECIFICAÇÃO DOS REQUISITOS DO SISTEMA	25
5.1	SISTEMA	25
5.2	LINGUAGEM	26
5.3	INTEGRAÇÃO	26
5.4	INTERFACE	27
5.5	ENTRADAS	30
5.6	SAÍDAS	31
5.7	AGENTE	31
6	PROJETO E IMPLEMENTAÇÃO	33
6.1	INTERFACE	33
6.2	ENTRADAS	34
6.3	BANCO DE DADOS	38
6.4	AGENTE	42
7	TESTE E AVALIAÇÃO	45
7.1	ENTRADAS	45
7.2	BANCO DE DADOS	51
7.3	AGENTE	52
7.4	INTERFACE	54
7.5	INTEGRAÇÃO	55
8	CONSIDERAÇÕES FINAIS	57
8.1	CONCLUSÕES DO PROJETO DE FORMATURA	57
8.2	CONTRIBUIÇÕES	58
8.3	PERSPECTIVA DE CONTINUIDADE	58
	REFERÊNCIAS	60

1. INTRODUÇÃO

1.1 MOTIVAÇÃO

Existem diversos sistemas de aprendizado virtual em uso atualmente, no entanto a maioria deles são apenas uma espécie de repositório de conteúdo, e quando apresentam algumas funcionalidades extras elas consistem apenas em atividades avaliativas e um canal de comunicação com os alunos.

Outro problema observado é a dificuldade que os usuários enfrentam ao mexer nesses sistemas, muitas vezes tendo dificuldades em realizar tarefas simples e desconhecendo a totalidade das funcionalidades do sistema.

Difícilmente os sistemas de aprendizado virtual apresentam assistentes virtuais, e quando apresentam, funcionam basicamente como uma ligação entre o usuário e a página que contém as perguntas frequentemente feitas, tentando associar o que o usuário pergunta a uma das perguntas presentes na página.

1.2 OBJETIVO

O projeto visa desenvolver um assistente virtual personalizado para o sistema educacional iVProg em integração com o ambiente Moodle, de forma que o ambiente seja mais do que um repositório de conteúdo e do que um local para realizar atividades avaliativas, mas que permita que os professores possam fornecer uma assistência “personalizada” para cada aluno mesmo sem ter o tempo disponível para isso.

A ideia é que o assistente apresente diagnósticos sobre o conhecimento do aluno, com dicas e caminhos de como melhorar nos pontos fracos, ou seja, o assistente teria um comportamento semelhante ao de um tutor.

O assistente também fornecerá os dados consolidados sobre o conhecimento dos alunos aos professores, permitindo que seja visualizado de forma simples quais conteúdos devem ser reforçados aos alunos.

1.3 JUSTIFICATIVA

A educação é um caminho para o acesso a uma melhor qualidade de vida, além de contribuir para o progresso da sociedade como um todo, em todos os campos (social, ambiental, científico, saúde, etc).

Uma educação de qualidade exige um acompanhamento próximo do aluno, além de incentivo e acesso às informações, algo que é difícil de fornecer em ambientes com diversos alunos.

Portanto uma ferramenta que permita que o professor forneça isso aos seus alunos de forma personalizada seria de grande contribuição para uma educação de qualidade.

1.4 ORGANIZAÇÃO DO TRABALHO

Este documento detalha toda a construção deste projeto de criação de um assistente virtual personalizado para o iVProg.

Primeiramente realiza-se a contextualização dos conceitos e das tecnologias empregados no desenvolvimento do projeto.

Em seguida é apresentado todo o processo de desenvolvimento do projeto, especificando suas etapas, além de apresentar a especificação do projeto com seus requisitos, diagramas e documentos.

Após isso é relatado quais foram os resultados obtidos na implementação do projeto, especificando quais foram os procedimentos de testes aplicados.

Por último, são realizadas as considerações finais sobre o projeto, analisando os resultados alcançados, as contribuições que o projeto pode realizar e quais são as perspectivas futuras de continuidade do projeto.

2. ASPECTOS CONCEITUAIS

2.1 EDUCAÇÃO

A educação é um conceito importantíssimo na história do desenvolvimento humano, através do qual se foi possível chegar ao nível de desenvolvimento atual. Ela consiste na passagem de conhecimentos, habilidades, crenças, valores, hábitos de uma pessoa para a outra.

Ao longo da maior parte da história da humanidade este conhecimento era passado exclusivamente através da fala e do convívio, sendo perdido quando os detentores do conhecimento morriam.

Com a invenção da escrita por volta de 3000 AEC o conhecimento humano pôde ser difundido de forma mais fácil, permitindo a possibilidade de permanência do conhecimento mesmo após a morte dos detentores do conhecimento.

A invenção das escolas em diversas civilizações antigas também foi um ponto importantíssimo na história da educação, pois permitiu a existência de um local comum onde as pessoas pudessem se reunir para trocar conhecimentos e para produzir novos conhecimentos.

Junto com as escolas, surgiram discussões sobre o ato de ensinar, as quais deram início ao campo da pedagogia, que estuda o processo e as metodologias de ensino e aprendizado. Isso permitiu o desenvolvimento de diferentes métodos de passagem de conhecimento, garantindo uma maior eficiência no processo de ensino e aprendizagem.

A próxima invenção com grande impacto no campo educacional foi a prensa móvel, pois permitiu que a produção de cópias de um manuscrito fosse mais rápida e fácil, o que levou a um grande aumento na produção de livros, folhetos, manuais, dentre outros, permitindo uma maior difusão do conhecimento.

Atualmente a educação e a produção de conhecimento vêm passando por transformações que geram e devem gerar um grande impacto, elas se dão através de computadores e da *internet*, com a possibilidade do ensino a distância, com o início da produção de conhecimento por parte dos computadores e da inteligência artificial, com os diversos conteúdos de aprendizagem diferentes disponibilizados

por diferentes pessoas na *internet*, com o futuro cenário de personalização da educação, dentre outros diversos acontecimentos e possibilidades.

Além de importante no âmbito coletivo do desenvolvimento humano, a educação é também de grande importância na esfera individual, sendo ela o motor de ascensão e transformação social através do qual as pessoas podem alcançar melhor qualidade de vida.

A massificação da educação permitiu que a maior parte da população fosse alfabetizada e tivesse algum grau de escolaridade, no entanto uma atenção individual ao aluno facilita o aprendizado. Com o uso da tecnologia pode-se prover uma atenção individual, personalizada, ao aluno sem que haja uma sobrecarga nos professores.

Tendo em vista isso, o projeto visa participar dessas novas transformações que geram e devem gerar grande impacto no campo da educação, participando do futuro cenário de personalização da educação.

2.2 SISTEMA DE APRENDIZADO VIRTUAL

Nos anos 60 foi desenvolvido o Sistema PLATO (*Programmed Logic for Automatic Teaching Operations*, em português: Lógica Programada para Operações de Ensino Automatizadas), ele possuía 4 tipos de usuário: o tipo aluno, que podia assistir aulas que foram atribuídas a ele e se comunicar com os professores por meio de anotações; o tipo instrutor, que podia examinar o progresso dos alunos, se comunicar com eles e também assistir aulas; o tipo autor, que podia examinar o progresso dos alunos, se comunicar com eles, assistir aulas e criar novas aulas; e o tipo múltiplo, que era usado para realizar demonstrações no sistema.

O sistema tinha mais de um modo de aprendizado, apresentava uma espécie de assistente virtual que enviava sequências de ajuda diferentes para diferentes tipos de respostas erradas, e registrava informações sobre as respostas dos alunos às perguntas para futuras análises estatísticas.

Esse sistema pode ser considerado um dos primeiros sistemas de aprendizado virtual, também conhecidos como ambientes virtuais de aprendizagem. Apesar de terem se passado décadas desde o desenvolvimento do Sistema PLATO,

a maioria dos conceitos aplicados nele se mantiveram, a grande mudança se dá no meio de execução, deixando de ser um computador construído especificamente para isso, e se tornando uma aplicação web.

Atualmente os sistemas de aprendizado virtual são usados tanto para Educação à Distância (EaD) quanto para complementar aulas presenciais. Esses sistemas costumam apresentar os seguintes componentes: repositório de conteúdo (aulas, materiais de apoio), canal de comunicação (*chats*, fóruns), sistema de avaliação (testes, exercícios), sistema de notas.

Os sistemas também podem apresentar os seguintes componentes: estatísticas de desempenho, sala de aula virtual, *wiki*, assistente virtual, dentre outros.

2.3 ASSISTENTE VIRTUAL INTELIGENTE

Assistentes virtuais inteligentes são ferramentas de software que fornecem alguma espécie de auxílio ou assistência ao usuário. Podendo ser algo mais simples como um *chatbot*, ou algo mais complexo como um assistente pessoal.

Para gerar uma resposta o assistente virtual realiza uma análise do conteúdo recebido como entrada, essa análise é usada para selecionar a resposta mais adequada presente no banco de dados do assistente virtual. Pode-se utilizar inteligência artificial e aprendizado de máquina para refinar as respostas do assistente virtual.

A análise correta do conteúdo recebido como entrada e a construção de um banco de dados robusto são pontos críticos para o desenvolvimento de um assistente virtual de qualidade.

O uso de assistentes virtuais provê autonomia ao usuário, podendo torná-lo independente da ajuda de outro humano dependendo da situação. Pode também facilitar algumas tarefas e atividades para o usuário, automatizando-as, fornecendo dicas de como realizá-las, dentre outras possibilidades.

2.4 PERSONALIZAÇÃO DE CONTEÚDO

A personalização consiste em adaptar algo para que sirva de forma mais satisfatória aos requisitos de alguém. E por isso é algo que é de interesse das pessoas e que agrega valor a qualquer produto ou serviço.

No mundo digital e no mundo on-line não é diferente, a personalização também gera muito interesse e agrega valor a qualquer produto ou serviço, como pode ser visto em jogos, redes sociais, sites de vendas, aplicativos, *streamings*, dentre outros.

A personalização de conteúdo permite que o conteúdo exibido para o usuário no sistema, plataforma ou aplicação seja o mais adequado para esse usuário. Isso é muito usado para reter a atenção do usuário em *streamings* e em redes sociais, além de ser usado para gerar o engajamento do usuário nessas aplicações. É também muito usado para exibir ao usuário produtos e serviços que ele tem mais chances de adquirir ou comprar.

Poderia-se utilizar a personalização de conteúdo em um sistema de aprendizado virtual para exibir para o usuário o conteúdo de aprendizagem, ou de assistência de aprendizagem, que seja o mais adequado para o aprendizado deste usuário.

2.5 EXPERIÊNCIA DO USUÁRIO (EU)

Experiência do usuário é o conjunto de fatores e elementos relativos à experiência do usuário no uso de um sistema, produto ou serviço, podendo gerar uma percepção negativa ou positiva.

Na criação de um projeto voltado para a experiência do usuário, o projeto deve ter os usuários envolvidos em todo o desenvolvimento, com o projeto sendo desenvolvido e refinado por avaliações centradas no usuário, em um processo que é interativo, abordando toda a experiência do usuário.

Em um contexto de aprendizagem é importante que o aluno tenha uma experiência positiva, conseguindo absorver o que está sendo passado, e se mantendo motivado.

2.6 PROGRAMAÇÃO VISUAL

A programação visual permite criar programas manipulando os elementos do programa graficamente, em vez de especificar esses elementos textualmente. Para isso deve-se utilizar uma linguagem de programação visual.

Usa ícones, blocos, formas e diagramas para reduzir ou até mesmo eliminar completamente o potencial de erros sintáticos, ajudando no arranjo de primitivas de programação para criar programas bem formatados.

Pode fornecer alguns mecanismos para exibir o significado das primitivas de programação, isso pode incluir funções de ajuda que fornecem funções de documentação integradas às linguagens de programação.

Permite o estudo do comportamento dos programas em situações específicas, ou seja, permite que os usuários coloquem artefatos criados em um determinado estado para explorar como o programa reagirá a esse estado.

Costuma ser usada para tornar a programação mais acessível, sendo usada para introduzir o conceito de programação a crianças e a pessoas de áreas não relacionadas à programação.

2.7 INTERAÇÃO HUMANO-COMPUTADOR (IHC)

A interação humano-computador é a área de estudo da interação entre pessoas e computadores, estando relacionada ao conceito de usabilidade.

O uso de computadores deve ser o mais simples, seguro e agradável possível, pois a criação de sistemas difíceis de usar pode inviabilizar o sucesso de softwares que poderiam ser bastante úteis.

Na implementação de uma linguagem de programação visual deve-se buscar seguir metodologias desta área de estudo, a fim de se garantir uma linguagem agradável e de fácil compreensão.

2.8 AVALIAÇÃO AUTOMÁTICA

A avaliação automática é um método de avaliação empregado quando a demanda de avaliações é maior que a disponibilidade dos avaliadores.

Nas avaliações automáticas digitais é checado se a resposta obtida se encontra dentro das respostas esperadas.

Existem diversos tipos de avaliações automáticas com complexidades diferentes, desde as mais simples que comparam se a resposta obtida é idêntica a esperada, até as mais complexas que podem analisar e interpretar a resposta completa dizendo o quanto a resposta se assemelha às possíveis respostas esperadas.

3. TECNOLOGIAS UTILIZADAS

3.1 IVPROG

iVProg (*Interactive Visual Programming on the Internet*, em português: Programação Visual Interativa na *Internet*) é um sistema educacional de código livre do LInE (Laboratório de Informática na Educação), destinado ao ensino e à aprendizagem de algoritmos e programação para iniciantes.

A versão atual (Versão IV) foi implementada em HTML5, usando JavaScript, sendo executada em navegadores, o que permite sua incorporação a qualquer página da *internet*. A primeira versão (Versão I) é de 2009 e foi implementada em Java.

Implementa o conceito de Programação Visual, que emprega ícones e simplifica a construção de códigos, reduzindo a necessidade do aluno conhecer detalhes de linguagens de programação, dando mais tempo para o aluno se dedicar ao aprendizado de algoritmos.

Dispõe de um avaliador automático para exercícios que pode ser facilmente integrado ao ambiente Moodle através do pacote iTarefa. Assim que o estudante finaliza o algoritmo, já consegue receber um parecer instantâneo, informando se sua solução resolve ou não o problema proposto.

O professor conta com uma área dentro do iVProg onde pode preparar as atividades, podendo reaproveitá-las em outros cursos. Possui integração completa com o Moodle, podendo gerar um relatório como todas as atividades realizadas pelos alunos (sendo possível examinar cada solução enviada) e podendo permitir que as notas dos exercícios sejam integradas ao quadro de notas do Moodle.

3.2 MOODLE

Moodle (*Modular Object-Oriented Dynamic Learning Environment*, em português: Ambiente Modular de Aprendizagem Dinâmica Orientada a Objetos) é um software livre e de código aberto, sendo ele um software de apoio à aprendizagem executado em um ambiente virtual.

É a plataforma de aprendizagem mais utilizada do mundo, com mais de 213 milhões de usuários. Possui uma interface simples, suporte a mais de 120 idiomas, é bem documentado, escalável, robusto, suporta diversos módulos de extensão, personalizável, seguro, atualizado constantemente, fornece privacidade.

A filosofia do Moodle é orientada pela pedagogia construcionista social, baseada em 4 conceitos principais relacionados: Construtivismo, Construcionismo, Construcionismo Social e Conectado e Separado.

Do ponto de vista Construtivista, as pessoas constroem ativamente novos conhecimentos à medida que interagem com seus ambientes.

O Construcionismo afirma que o aprendizado é particularmente eficaz ao construir algo para que outros experimentem.

O Construtivismo Social estende o Construtivismo aos ambientes sociais, nos quais os grupos constroem conhecimento uns para os outros, criando de forma colaborativa uma pequena cultura de elementos compartilhados com significados compartilhados.

O conceito de Conectado e Separado se aprofunda nas motivações dos indivíduos em uma discussão, o comportamento Separado é quando alguém tenta permanecer objetivo e factual, e tende a defender suas próprias idéias usando a lógica para encontrar buracos nas idéias de seu oponente, já o comportamento Conectado é uma abordagem mais empática que aceita a subjetividade, tentando ouvir e fazer perguntas em um esforço para compreender o outro ponto de vista. Há também o comportamento construído, que é aquele no qual uma pessoa é sensível a ambas as abordagens e é capaz de escolher qualquer uma delas conforme for apropriado para a situação atual.

Possui 3 tipos principais de usuários: administrador, professor e aluno. E pode ser configurado em 3 formatos diferentes: formato social, formato semanal e formato em tópicos.

Apresenta repositório de conteúdo (aulas, materiais de apoio), canal de comunicação (*chats*, fóruns), sistema de avaliação (testes, exercícios), sistema de notas, dentre outras componentes.

3.3 IASSIGN

iAssign (*interactive Assignment*, em português: Tarefas interativas) é um sistema educacional gratuito fornecido pelo Laboratório de Informática na Educação (LIInE), ele consiste em um módulo de extensão do Moodle que permite enriquecê-lo com mais ferramentas de aprendizagem interativas, esses módulos de extensão são chamados de iLM (*interactive Learning Modules*, em português: Módulos de Aprendizagem interativos).

O projeto iAssign foi iniciado em 2009, a sua primeira versão foi lançada no Moodle 1.9.

O objetivo do iAssign é aumentar a interatividade em atividades relacionadas a assuntos específicos (geometria, funções, programação, dentre outros) de forma flexível.

Para melhorar a interatividade, o iAssign utiliza o iLM, que consiste em qualquer ferramenta interativa que roda em um navegador. Eles usam o protocolo HTTP para se comunicarem com o Moodle, por exemplo, para obter a atividade do professor do servidor e enviar a resposta do aluno para o servidor.

Inicialmente, todos os iLM disponíveis foram implementados em Java (como "miniaplicativo"), mas a versão atual do iAssign permite integrar sistemas codificados em pilha HTML (com CSS e JavaScript). Isso significa que qualquer sistema Web, em princípio, poderia facilmente se tornar um iLM e ser integrado ao Moodle sob o pacote iAssign, por exemplo, se o iLM oferece funcionalidade de avaliação automática, o iAssign é capaz de lidar com isso.

Por motivos de segurança, apenas o administrador tem o privilégio de integrar um novo iLM ao iAssign. Uma vez integrado, um iLM pode ser usado por qualquer pessoa com acesso ao iAssign. Por exemplo, todos com privilégios de professores em diante têm permissão para usar as ferramentas de autoria iAssign para criar novas atividades para os alunos.

Os principais recursos providos pelo iAssign são: ferramentas de autoria para permitir que qualquer professor prepare facilmente as atividades para os alunos, as atividades podem ser um exercício, se o iLM tiver avaliação automática, está associado à avaliação automática, um teste, o aluno realiza a atividade, se o iLM

tiver avaliação automática, o aluno recebe um feedback imediato, mas nenhum dado é registrado no banco de dados do Moodle, ou um exemplo, nada é registrado; relatórios sobre as atividades dos alunos, os professores têm uma avaliação e algumas estatísticas sobre as respostas dos alunos e podem ter acesso rápido a qualquer uma de suas respostas, os professores também podem baixar um pacote com todas as respostas dos alunos, os alunos podem fazer um levantamento de suas atividades, incluindo suas notas; a integração com as notas gerais do Moodle; e um filtro que permite a inserção de conteúdo iLM em texto HTML no Moodle.

Além disso, como a maioria dos módulos de extensão do Moodle, o iAssign pode exportar qualquer conjunto de iActivitis.

3.4 JAVASCRIPT

JavaScript é uma linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma (protótipos, orientado a objeto, imperativo e funcional).

O JavaScript permite páginas interativas, o que o torna parte essencial dos aplicativos da web. A maioria dos sites utiliza o JavaScript, em função disso todos os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo.

Possui uma função que consegue executar comandos em tempo de execução. Suas funções são de primeira classe, suporta funções aninhadas e fechamentos.

Possui APIs (*Application Programming Interface*, em português: Interfaces de Programação de Aplicativos) para trabalhar com texto, datas, expressões regulares, estruturas de dados padrão e o *Document Object Model* (DOM).

Não inclui nenhuma entrada/saída (E/S), como rede, armazenamento ou recursos gráficos. Na prática, o navegador ou outro sistema de tempo de execução fornece APIs JavaScript para E/S.

Os mecanismos JavaScript eram originalmente usados apenas em navegadores, mas agora são componentes principais de outros sistemas de software, principalmente de servidores e de uma variedade de aplicativos.

Seu uso permite projetar aplicações que executam em qualquer dispositivo com acesso a navegadores, permitindo uma maior facilidade de acesso a essa aplicação.

3.5 NODE.JS

Node.js é um software de código aberto, multiplataforma, de back-end JavaScript baseado no interpretador V8 do Google que possibilita executar código JavaScript fora de um navegador.

Permite que desenvolvedores usem JavaScript para desenvolver ferramentas de linha de comando e scripts do lado do servidor, executando-os do lado do servidor para produzir conteúdo dinâmico de página da web antes que a página seja enviada ao navegador do usuário. Em decorrência disso, o Node.js representa um paradigma "JavaScript em todos os lugares", unificando o desenvolvimento de aplicativos da web em torno de uma única linguagem de programação, em vez de diferentes linguagens para scripts do lado do servidor e do lado do cliente.

É usado principalmente para a criação de servidores web e ferramentas de rede usando JavaScript e uma coleção de módulos que lidam com várias funcionalidades básicas. Os módulos são fornecidos para entrada e saída do sistema de arquivos, rede, dados binários, funções de criptografia, fluxos de dados e outras funções principais.

Tem uma arquitetura orientada a eventos capaz de entrada e saída assíncrona, isso visa otimizar o rendimento e a escalabilidade em aplicativos da web com muitas operações de entrada e saída, bem como para aplicativos de tempo real, com isso permite o desenvolvimento de servidores web rápidos em JavaScript. Os desenvolvedores podem criar servidores escalonáveis sem usar threading, usando um modelo simplificado de programação orientada a eventos que usa callbacks para sinalizar a conclusão de uma tarefa.

A principal diferença do Node.js para outras tecnologias é a execução das requisições e eventos em single-thread, onde apenas uma thread é responsável por executar o código Javascript, sem a necessidade de criar novas threads, o que

consumiria mais recursos computacionais, e sem necessidade de utilizar fila de espera.

3.6 PHP

PHP (*PHP: Hypertext Preprocessor*, em português: Pré-processador de Hipertexto, originalmente: *Personal Home Page*, em português: Página Inicial Pessoal) é uma linguagem interpretada, de código livre, orientada a objetos, portátil, de tipagem dinâmica, com sintaxe similar a de C/C++ e Perl.

Originalmente era utilizada apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, mas atualmente oferece funcionalidades em linha de comando e adquiriu características adicionais que possibilitam usos adicionais não relacionados à páginas web.

O módulo PHP interpreta o código no lado do servidor, gerando a página web que é visualizada no lado do cliente.

A diferença mais significativa entre Node.js e PHP é que na maioria das funções em PHP os comandos são executados apenas depois que os comandos anteriores terminam, enquanto nas funções do Node.js os comandos são executados simultaneamente ou mesmo em paralelo, utilizando callbacks para sinalizar conclusão ou falha.

3.7 MARIADB

MariaDB é um sistema de gerenciamento de banco de dados de software livre que surgiu a partir de uma ramificação do MySQL, tendo o seu desenvolvimento liderado por alguns dos desenvolvedores originais do MySQL.

Inicialmente era projetado para manter a alta compatibilidade com MySQL, garantindo uma capacidade de substituição com paridade binária de biblioteca e correspondência exata com APIs e comandos do MySQL, no entanto, o número de incompatibilidades tem crescido a cada nova versão.

Como todo sistema de gerenciamento de banco de dados, seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados.

4. METODOLOGIA DE TRABALHO

4.1 CONCEPÇÃO

Elaboração sobre o tema do projeto, escolhendo-se qual área deseja-se abordar e qual a ideia proposta.

A área escolhida para abordar foi a área educacional, com o intuito de desenvolver uma ferramenta que auxilie os alunos no seu processo de aprendizado.

Para alcançar esse intuito, concebeu-se a ideia de se desenvolver um assistente virtual personalizado, que pudesse guiar o aluno, fornecer dicas e observações sobre os conteúdos e desempenho, semelhantemente a um professor em sala de aula.

4.2 FAMILIARIZAÇÃO

Etapa onde deve ser realizada a familiarização com as tecnologias que serão usadas no desenvolvimento do projeto.

Inicialmente realizou-se um processo de utilização do iVProg, a fim de se familiarizar com o seu funcionamento. Após isso foi analisado o seu código, a fim de entender como se dá o seu funcionamento, e onde poderiam ser inseridas formas de se implementar o projeto.

Em seguida leu-se a documentação do Moodle, para entender a comunicação dos módulos de extensão com o Moodle e para entender quais são as possibilidades de funcionalidades a serem consideradas na implementação do projeto.

O próximo passo foi se familiarizar com o iAssign, lendo sua documentação para entender como utilizá-lo para integrar módulos de extensão com o Moodle, e para entender como se dá a sua comunicação com ambas as partes..

Por último, nesta etapa, estudou-se JavaScript para se preparar para as fases seguintes do processo de desenvolvimento do sistema, a fim de se estar consciente das funcionalidades possíveis desta linguagem.

4.3 PROJETO

Elaboração de como implementar a ideia concebida e definição dos requisitos do sistema elaborado.

Nesta etapa realizaram-se reuniões com a orientadora e com o coorientador para se mapear requisitos que deveriam ser definidos e os possíveis caminhos que poderiam ser tomados para a implementação da ideia concebida.

Com este ponto de partida e com o conhecimento adquirido na etapa de familiarização foram realizadas reflexões e análises sobre quais seriam essas definições e qual caminho se tomaria.

Levando isso em consideração, os requisitos para a implementação do sistema elaborado foram definidos e descritos.

Outra tarefa realizada foi a análise dos exercícios de iVProg presentes no curso “2021: Introdução à Programação (Diurno)” do SAW (Sistema de Aprendizagem via Web), um dos vários sistemas de aprendizado virtual da USP (Universidade de São Paulo).

Nessa análise foram levantados quais os temas principais de cada exercício, quais os erros esperados e quais foram os erros cometidos pelos alunos, de forma que pode-se criar categorias para os exercícios a partir dessa análise.

4.4 IMPLEMENTAÇÃO

Etapa na qual se implementará o projeto elaborado, através da construção de códigos que implementem os requisitos definidos.

Primeiramente se implementará as entradas e saídas adequadas ao sistema conforme definido no projeto.

O próximo passo é implementar o agente que realizará o cálculo de desempenho e fornecerá ao aluno um retorno sobre o exercício de acordo com o seu desempenho no exercício e de acordo com o seu histórico de desempenho.

Em seguida será implementada a interface do sistema, de forma que exiba corretamente as saídas ao usuário, e de forma que tenha uma interação adequada e satisfatória com o usuário.

Por último, se implementará os aspectos relacionados à integração deste sistema com o Moodle através do módulo de extensão iAssign.

4.5 TESTES

Testagem do sistema implementado para validar o seu funcionamento e tentar encontrar possíveis erros do sistema.

A cada etapa de implementação serão realizados testes relativos à etapa, a fim de se detectar de forma mais rápida os principais problemas e para validar o que foi desenvolvido na etapa antes dos testes finais.

Após a conclusão de todas as etapas de implementação se realizarão testes com o sistema completo integrado, a fim de se encontrar problemas que tenham passado despercebidos dos primeiros testes e possíveis erros de integração do sistema.

4.6 AJUSTES

Realização de ajustes relativos aos problemas detectados na etapa de testagem do sistema.

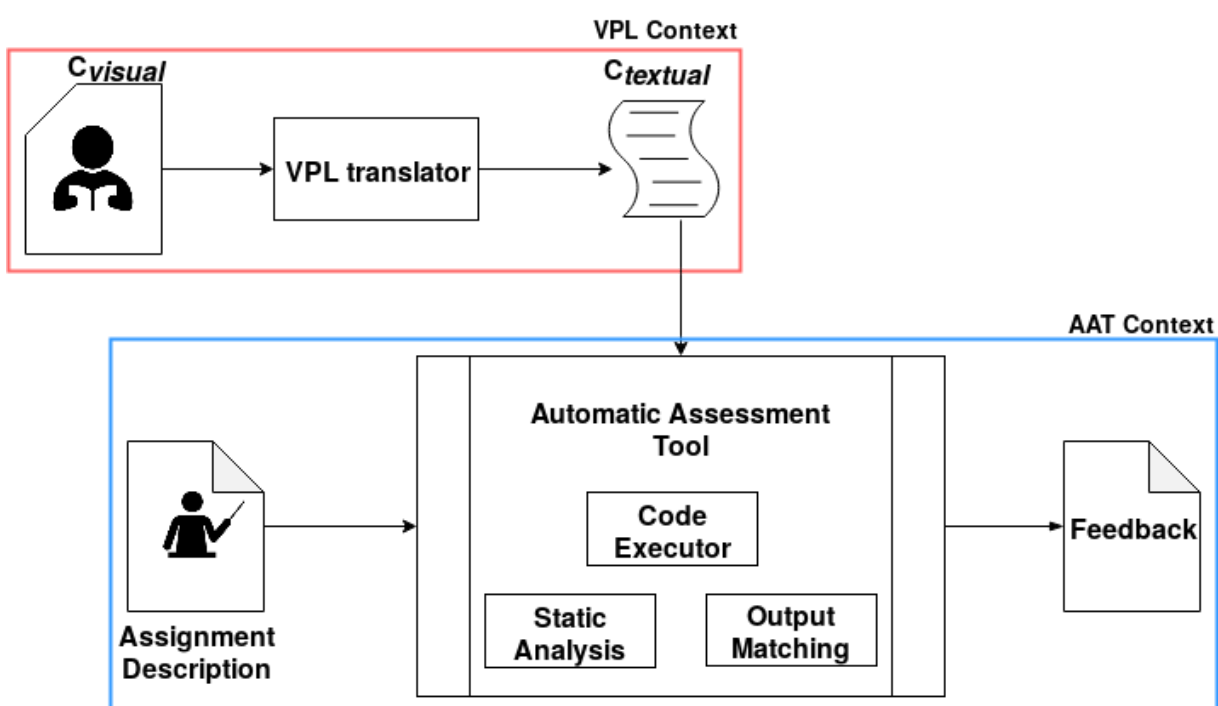
Inicialmente serão analisados os problemas detectados, para que se possa compreendê-los, e então realizar os ajustes necessários, obtendo-se a versão final deste projeto.

5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

5.1 SISTEMA

O iVProg versão IV possui sua estrutura composta por dois módulos, um relacionado ao contexto da programação visual e um relacionado ao contexto da avaliação automática.

Figura 1 - Estrutura da versão IV do iVProg



Fonte: Lucas Mendonça de Souza

No módulo relacionado ao contexto da programação visual ocorre a tradução do código elaborado em linguagem de programação visual para linguagem de programação textual.

O módulo relacionado ao contexto da avaliação automática recebe a tradução do código visual para textual e uma descrição da tarefa, com essas informações ele realiza a execução do código e então compara as saídas do programa com as saídas apresentadas na descrição da tarefa, e fornece um retorno sobre a tarefa ao aluno de acordo com o resultado da execução e da comparação.

O sistema desenvolvido neste projeto será um agente dentro do módulo relacionado ao contexto da avaliação automática, interagindo com os outros agentes que compõem o módulo.

Assim ele pode interagir de forma direta e interna com os elementos necessários para o seu funcionamento.

5.2 LINGUAGEM

A versão IV do iVProg está implementada em HTML5 com o uso de JavaScript, e é executada em navegadores.

Com base nisso e na escolha de implementar o sistema como um agente dentro do iVProg, foi definido que a linguagem a ser usada no desenvolvimento do sistema é JavaScript.

Desta forma pode se manter uma coesão no código do iVProg, além de evitar possíveis dificuldades e possíveis falhas na interação entre códigos de linguagens diferentes.

A versão atual do iAssign está implementada em PHP, e se utiliza do banco de dados MariaDB, em função disso é necessário adição de código em PHP no código fonte do iAssign para criar e manipular o banco de dados onde se armazenará o desempenho dos alunos.

5.3 INTEGRAÇÃO

A versão IV do iVProg possui integração com o Moodle através do iAssign, essa integração será aproveitada pelo sistema desenvolvido, com algumas adições em função da implementação de funcionalidades do assistente virtual.

A integração do sistema com um sistema de aprendizado virtual permite a associação dos exercícios de programação visual a usuários, permitindo que se possa realizar um acompanhamento dos usuários.

Com isso, professores podem atribuir exercícios de programação visual a alunos no sistema de aprendizado virtual de uma forma mais simples e mais acessível, aumentando a possibilidade de uso do sistema.

As adições visam possibilitar a personalização de conteúdo ao aluno, para que se possa fornecer assistência personalizada, e possibilitar o armazenamento do desempenho do aluno.

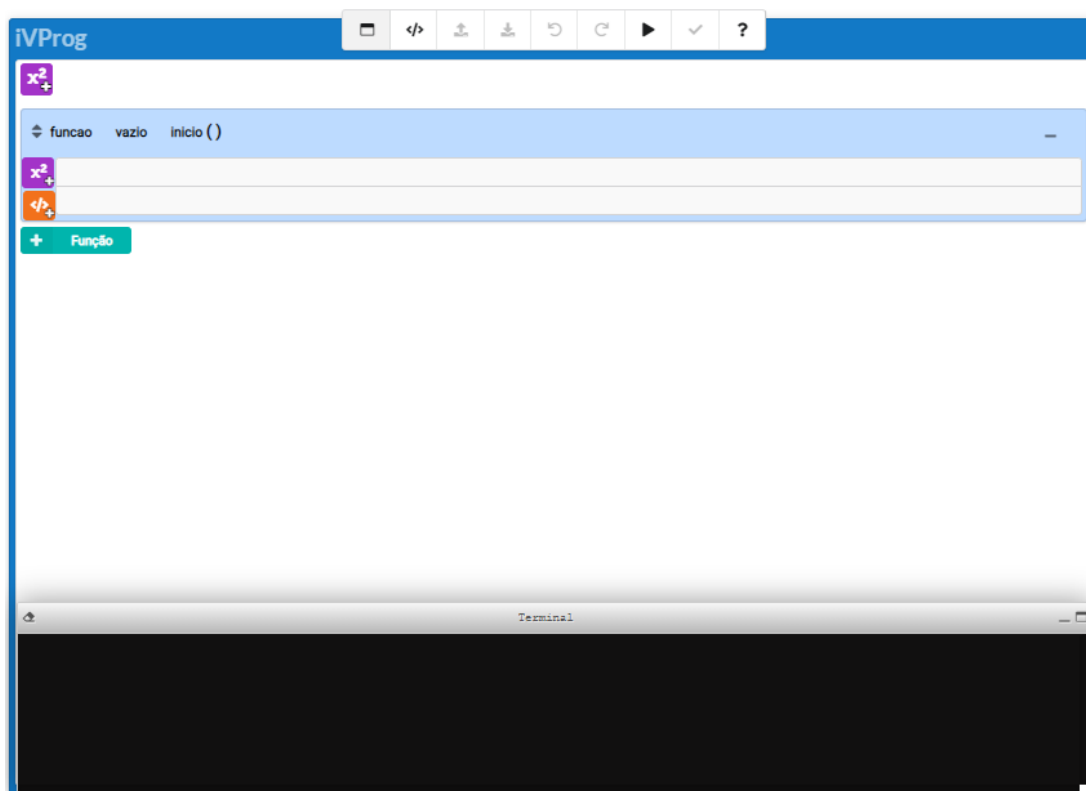
5.4 INTERFACE

A interface do iVProg versão IV possui duas janelas diferentes, a janela do código e a janela do terminal.

A janela do código é fixa na tela, possui duas abas (código visual e código textual), possui botões e em sua aba de código visual é possível ocultar os elementos da função. Nessa janela é possível construir um código visual, observar a sua tradução para código textual e executar esse código.

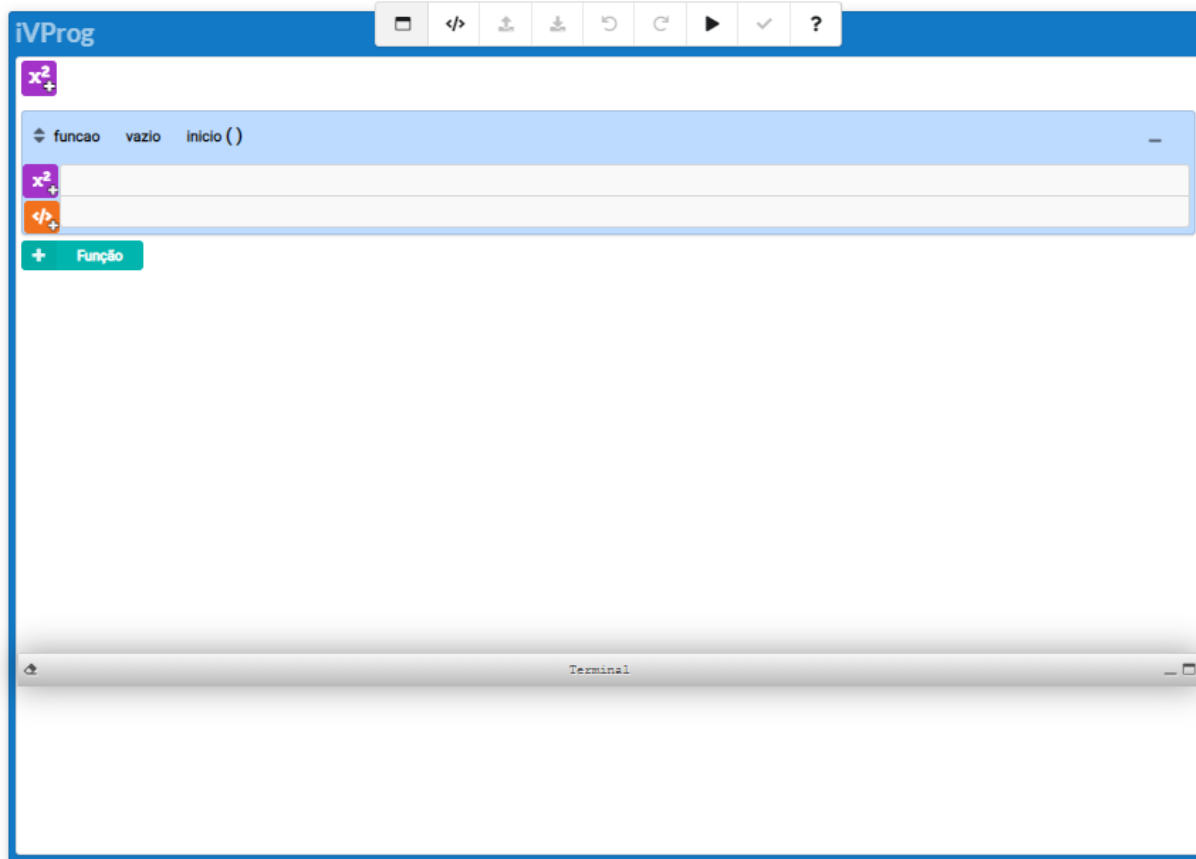
A janela do terminal pode ser movimentada pela tela, além disso ela pode ser minimizada e expandida. Nela são exibidos o resultado da execução do código e o resultado do avaliador automático.

Figura 2 - iVProg com o Terminal expandido



Fonte: Execução da versão IV do iVProg

Figura 3 - iVProg com o Terminal minimizado



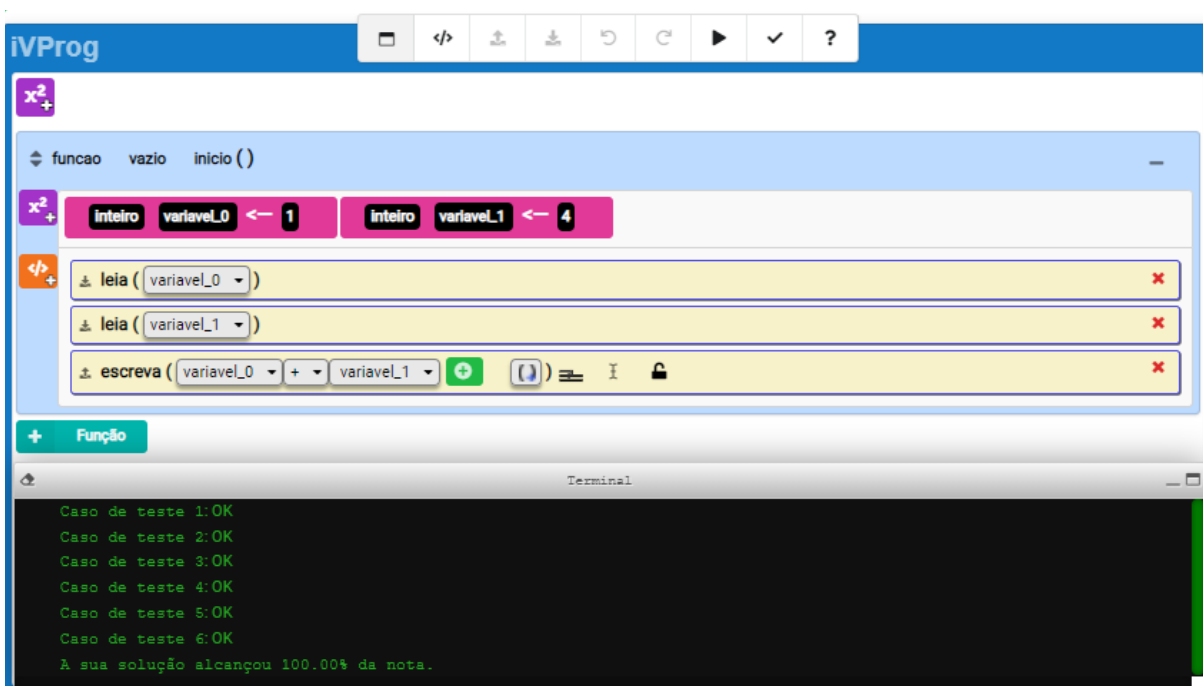
Fonte: Execução da versão IV do iVProg

Quando se executa o código elaborado no iVProg aquilo que é escrito pelo código executado é exibido na janela do terminal.

Quando se executa a correção automática do exercício o terminal exibe os resultados de cada caso teste e a nota final obtida. Os detalhes do resultado de cada caso teste podem ser visualizados ao se clicar em cada caso teste, o que resulta na abertura de uma nova janela do navegador contendo os detalhes do resultado do caso teste clicado.

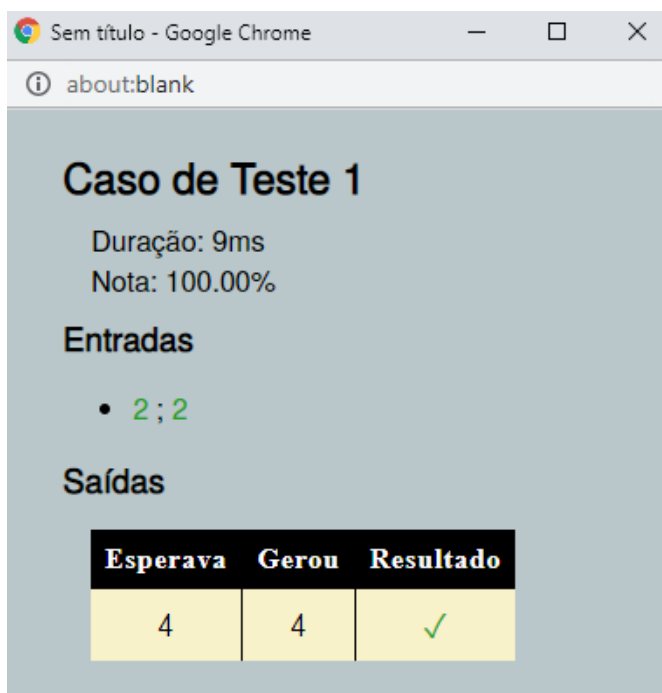
A implementação da interface do assistente virtual se dará da mesma forma que ocorre com os detalhes do resultado de cada caso teste na correção automática, onde será exibido no terminal uma mensagem indicando que caso o usuário queira visualizar o conteúdo retornado pelo assistente ele deve clicar nessa mensagem, que ao ser clicada abrirá uma nova janela do navegador contendo o conteúdo retornado pelo assistente.

Figura 4 - Execução da correção automática de um exercício de soma de dois valores



Fonte: Exemplo 3 da página de exemplos do site do iVProg

Figura 5 - Nova janela exibida após clique no “Caso de teste 1” presente na figura 4



Fonte: Exemplo 3 da página de exemplos do site do iVProg

5.5 ENTRADAS

No iVProg versão IV se tem duas entradas, o código visual implementado pelo usuário e a descrição de tarefa do avaliador.

A descrição de tarefa do avaliador é elaborada pelo criador do exercício a ser avaliado, essa descrição possui apenas casos de teste compostos por conjuntos formados por entradas e por saídas esperadas para aquelas entradas.

Figura 6 - Descrição de Tarefa da soma de dois valores

```
"testcases" : [  
  {  
    "input": ["2", "2"],  
    "output": ["4"]  
  },  
  {  
    "input": ["1", "10"],  
    "output": ["11"]  
  },  
]
```

Fonte: Exemplo 3 da página de exemplos do site do iVProg

Esses casos de teste são comparados aos resultados obtidos pela execução do código visual implementado.

Serão adicionados novos campos nos casos de teste da descrição de tarefa do avaliador, a fim de se adicionar algum contexto do exercício às entradas do exercício.

Dentre os novos campos se tem um campo para indicar qual a categoria de erro associada àquele caso de teste, ou seja, a qual erro a falha nesse teste está associada. O outro campo que será adicionado é referente ao tema do exercício, a fim de se informar quais os assuntos principais tratados no exercício.

Será adicionada uma nova entrada referente ao histórico de desempenho do aluno no iVProg, esse histórico será fornecido pelo sistema de aprendizado virtual.

5.6 SAÍDAS

A versão IV do iVProg possui uma saída composta por duas partes, uma parte consiste no resultado da execução do código, e a outra parte consiste no resultado do avaliador automático, sendo ambas exibidas na janela do terminal do iVProg.

O resultado da execução do código fornece informações sobre a execução do código, informando sobre a ocorrência de erros e exibindo o resultado da execução do código.

O resultado do avaliador automático exibe informações sobre a comparação dos casos de teste com o resultado da execução do código, informando o resultado de cada caso de teste e a nota total obtida.

Será adicionada uma saída referente ao assistente virtual, ela será exibida ao usuário em uma nova janela do navegador que é gerada quando o usuário clica na mensagem presente no terminal referente a exibição do conteúdo retornado pelo assistente. Nessa saída serão fornecidos os conteúdos relativos aos erros e ao usuário. Este conteúdo pode ser composto por textos, documentos, imagens, vídeos, links.

Também se adicionará uma saída referente ao desempenho do usuário no exercício, essa saída será fornecida ao sistema de aprendizado virtual.

5.7 AGENTE

Para que o agente (assistente virtual) forneça conteúdos relacionados aos erros ocorridos, e que possam ser personalizados de acordo com o usuário é necessário que haja uma categorização dos possíveis erros.

O iVProg versão IV possui mensagens de erros relativas a erros sintáticos, essas mensagens podem ser utilizadas para realizar a categorização de alguns erros.

Outra informação que pode ser utilizada para realizar a categorização de erros é a descrição de tarefa do avaliador com a adição dos novos campos de entrada.

Algumas outras informações podem ser extraídas do código visual traduzido para textual, da execução do código textual e do resultado da comparação do avaliador automático.

O agente fornecerá o conteúdo personalizado ao usuário e fornecerá as informações referentes ao exercício realizado ao sistema de aprendizado virtual, desta maneira é possível armazenar informações referentes ao usuário de forma que seja possível utilizá-las na personalização de conteúdo.

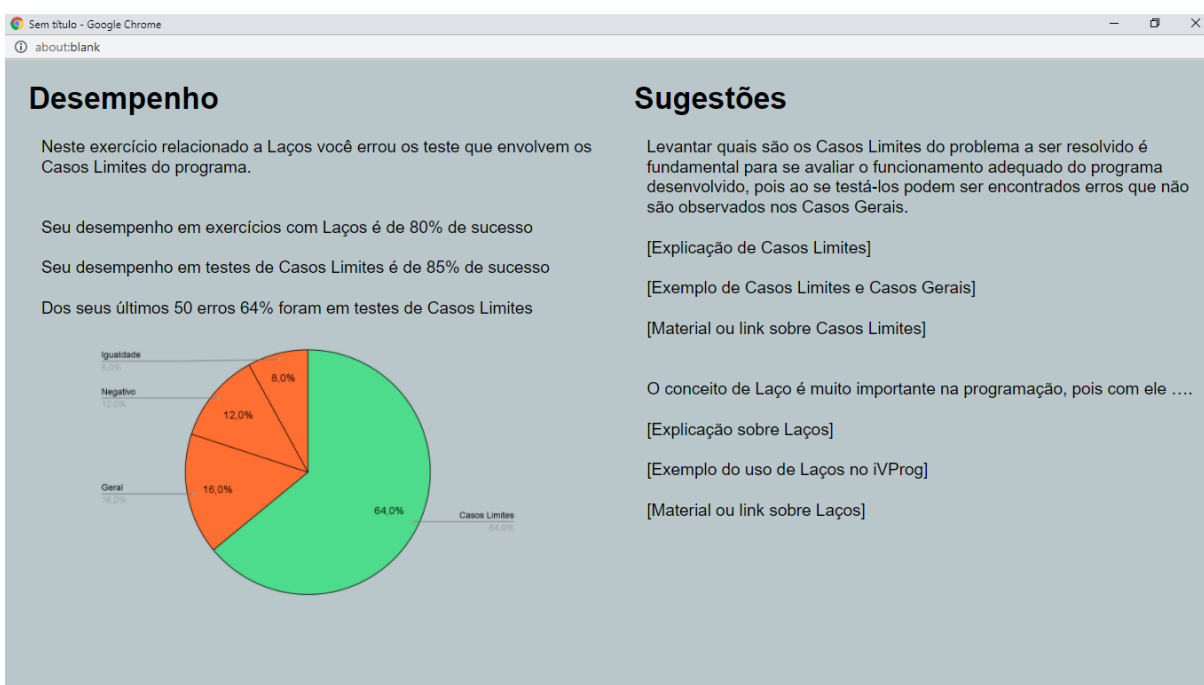
O conteúdo personalizado será gerado de acordo com os erros ocorridos e de acordo com o histórico de desempenho do usuário.

6. PROJETO E IMPLEMENTAÇÃO

6.1 INTERFACE

Inicialmente elaborou-se um protótipo da interface que o agente deve exibir, de forma que se pudesse ter uma visão mais clara do que será necessário para gerar essa interface e de como construir o agente (assistente virtual) de maneira que facilite a construção dessa interface.

Figura 7 - Protótipo da interface exibida pelo agente



Fonte: Autoria própria

A interface prototipada apresenta o desempenho do aluno em relação aos erros cometidos no lado esquerdo e as sugestões fornecidas a esse aluno no lado direito.

O desempenho do aluno é apresentado informando o tema ao qual o exercício é relacionado, em qual tipo de teste o aluno falhou, o desempenho geral deste aluno em exercícios deste tema, o desempenho geral deste aluno neste tipo de teste, a porcentagem correspondente de erros nesse tipo de teste em relação aos

últimos 50 erros e um gráfico mostrando a distribuição dos últimos 50 erros em relação ao tipo de teste.

As sugestões são apresentadas exibindo uma explicação da importância do tipo de teste em que o aluno falhou, uma explicação do que consiste esse tipo de teste, um exemplo de aplicação deste tipo de teste, um material relacionado a esse tipo de erro para caso o aluno deseje se aprofundar, uma explicação da importância do tema do exercício, uma explicação sobre em que esse tema consiste, um exemplo de aplicação deste tema no iVProg e um material relacionado a esse tema para caso o aluno deseje se aprofundar.

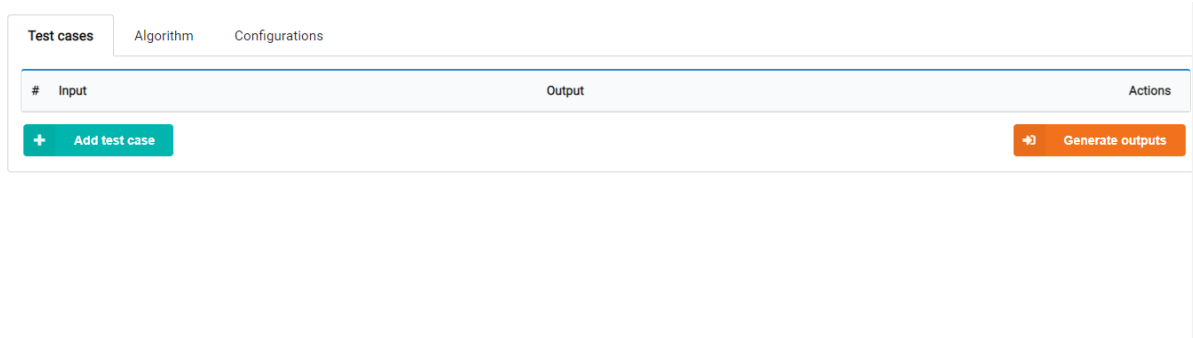
6.2 ENTRADAS

O arquivo “iassign-integration-functions.js” presente na versão IV do iVProg foi alterado com a finalidade de se adicionar novos campos nos casos de teste da descrição de tarefa do avaliador, a fim de se prover ao agente (assistente virtual) as entradas necessárias para seu funcionamento. Esses novos campos têm como objetivo adicionar algum contexto do exercício às entradas do exercício.

São dois novos campos, um referente aos temas presentes no exercícios chamado de “type”, e outro referente ao tipo de caso teste chamado de “tag”, para isso foi necessário alterar as seguintes funções presentes no arquivo citado acima: “addTestCase”, “getAnswer”, “prepareActivityCreation”, “prepareTableTestCases”, “prepareTestCases”. Também foi necessário criar as seguintes funções: “addType”, “prepareExerciseType”, “prepareTableExerciseType”, “updateTypeCounter”.

A interface atual de criação da descrição de tarefa do avaliador apresenta três abas. A primeira delas é referente aos casos testes, contendo entradas e suas saídas esperadas, sendo possível adicionar quantos casos testes forem necessários, removê-los e gerar as saídas esperadas para as entradas através de um algoritmo implementado na segunda aba.

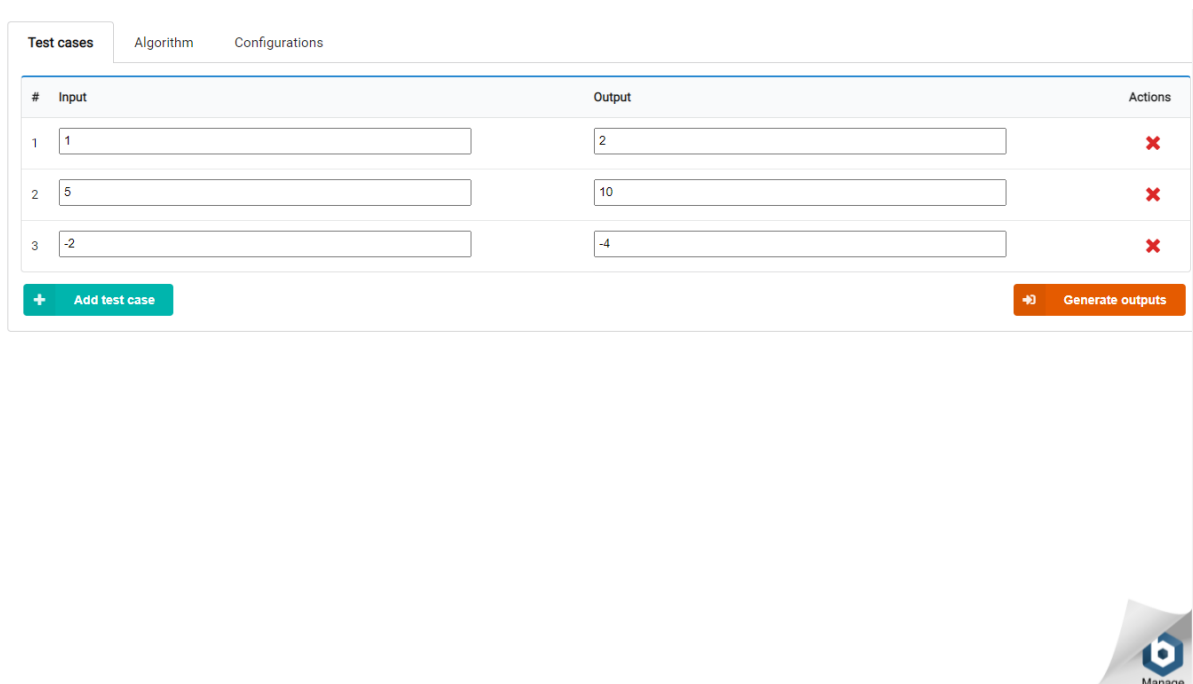
Figura 8 - Primeira aba da interface de criação da descrição de tarefa do avaliador



The screenshot shows the 'Test cases' tab of the iVProg interface. At the top, there are three tabs: 'Test cases', 'Algorithm', and 'Configurations'. Below the tabs is a table with four columns: '#', 'Input', 'Output', and 'Actions'. The table is currently empty. At the bottom left of the table area, there is a green button with a plus sign and the text 'Add test case'. At the bottom right, there is an orange button with a play icon and the text 'Generate outputs'.

Fonte: Execução da versão IV do iVProg

Figura 9 - Primeira aba da interface de criação da descrição de tarefa do avaliador preenchida



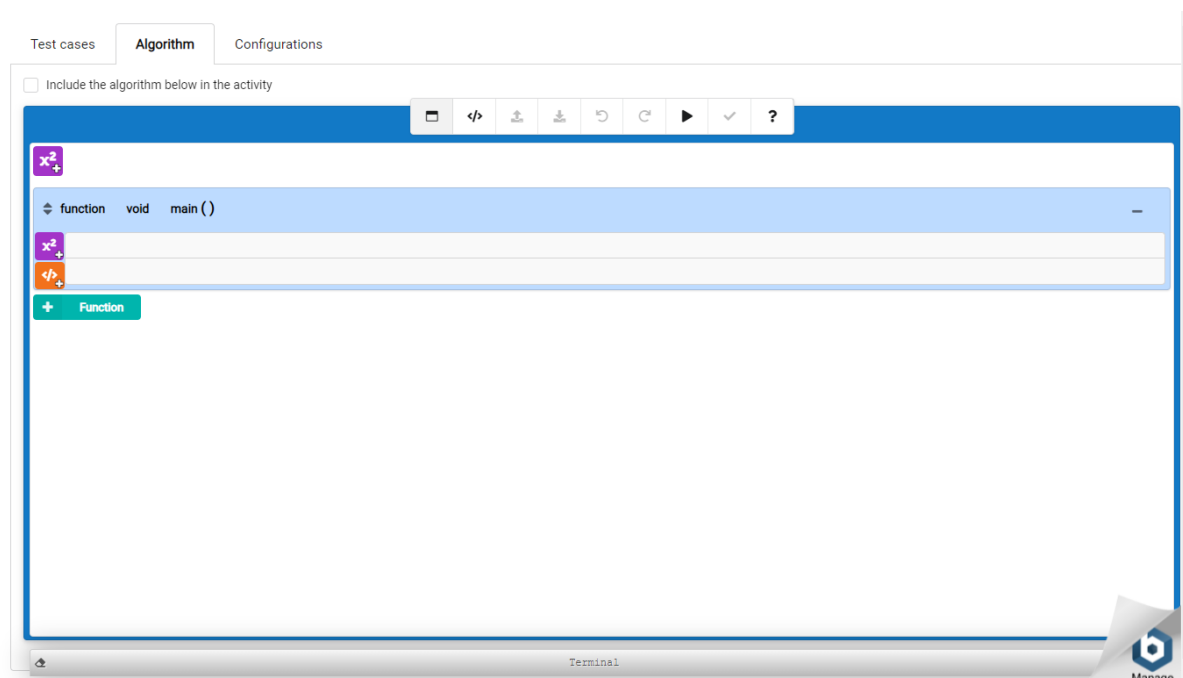
The screenshot shows the 'Test cases' tab of the iVProg interface, now filled with data. The table has three rows. Each row has an input field, an output field, and an 'Actions' column with a red 'X' icon. At the bottom left, there is a green button with a plus sign and the text 'Add test case'. At the bottom right, there is an orange button with a play icon and the text 'Generate outputs'. In the bottom right corner of the interface, there is a logo with a blue 'b' and the word 'Manage' below it.

#	Input	Output	Actions
1	1	2	✘
2	5	10	✘
3	-2	-4	✘

Fonte: Execução da versão IV do iVProg

A segunda aba contém o iVProg, nesta aba é possível construir um algoritmo que resolva a tarefa proposta a fim de se gerar automaticamente as saídas esperadas para as entradas inseridas, é possível também construir um código com o qual a tarefa será inicializada, funcionando como uma espécie de ponto de partida para o aluno.

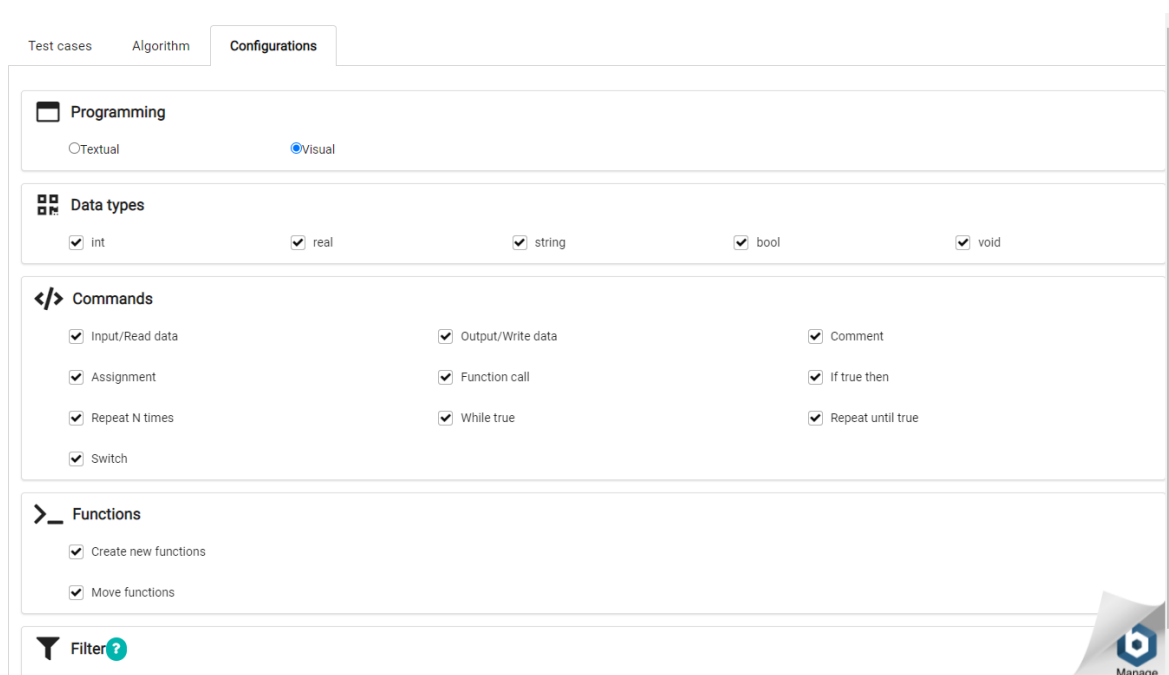
Figura 10 - Segunda aba da interface de criação da descrição de tarefa do avaliador



Fonte: Execução da versão IV do iVProg

A terceira aba da interface é referente às configurações da tarefa, nela é possível selecionar quais configurações deseja-se aplicar a tarefa.

Figura 11 - Terceira aba da interface de criação da descrição de tarefa do avaliador



Fonte: Execução da versão IV do iVProg

Ao finalizar a criação da descrição de tarefa do avaliador é gerado um arquivo .ivph que contém a descrição.

Figura 12 - Arquivo .ivph que contém a descrição de tarefa do avaliador

```
{
  "testcases" : [
    {
      "input": ["1"],
      "output": ["2"]
    },
    {
      "input": ["5"],
      "output": ["10"]
    },
    {
      "input": ["-2"],
      "output": ["-4"]
    }
  ],
  "settings_programming_type":
  [{"name": "programming_type", "value": "visual"}],
  "settings_data_types":
  [{"name": "integer_data_type", "value": "on"}, {"name": "real_data_type", "value": "on"}, {"name": "text_data_type", "value": "on"}, {"name": "boolean_data_type", "value": "on"}, {"name": "void_data_type", "value": "on"}],
  "settings_commands":
  [{"name": "commands_read", "value": "on"}, {"name": "commands_write", "value": "on"}, {"name": "commands_comment", "value": "on"}, {"name": "commands_attribution", "value": "on"}, {"name": "commands_functioncall", "value": "on"}, {"name": "commands_iftrue", "value": "on"}, {"name": "commands_repeatNtimes", "value": "on"}, {"name": "commands_while", "value": "on"}, {"name": "commands_dowhile", "value": "on"}, {"name": "commands_switch", "value": "on"}],
  "settings_functions":
  [{"name": "functions_creation", "value": "on"}, {"name": "functions_move", "value": "on"}],
  "settings_filter":
  []
}
```

Fonte: Execução da versão IV do iVProg

Foi criada uma nova aba para o novo campo referente aos temas do exercício, sendo esta a nova primeira aba, nela é possível adicionar quantos temas forem necessários, os temas são adicionados a partir de uma lista de temas predefinidos, também é possível remover os temas adicionados.

O novo campo relativo ao tipo de caso teste foi adicionado na aba referente aos casos testes, que se tornou a segunda aba nesta nova versão da interface de criação da descrição de tarefa do avaliador. Esta aba contém os mesmos campos e funcionalidades da versão anterior com a adição do campo relativo ao tipo de caso teste, o conteúdo deste campo é adicionado a partir de uma lista de tipos de caso teste predefinidos.

A aba que contém o iVProg se tornou a terceira aba nesta nova versão da interface e não sofreu alterações.

A aba referente às configurações da tarefa se tornou a quarta aba nesta nova versão da interface e também não sofreu alterações.

Assim como na versão anterior da interface, ao se finalizar a criação da descrição de tarefa do avaliador é gerado um arquivo .ivph que contém a descrição.

6.3 BANCO DE DADOS

Através de uma análise dos exercícios de iVProg presentes no curso “2021: Introdução à Programação (Diurno)” do SAW definiu-se quais os temas e os tipos de caso teste possíveis para exercícios aplicados atualmente no iVProg.

Nessa análise foram levantados quais os possíveis temas principais de cada exercício, quais os erros cometidos pelos alunos e quais os possíveis tipos de casos testes aplicáveis ao exercício.

Na tabela abaixo encontra-se o resultado da análise, a primeira coluna contém o nome do exercício no curso, na segunda coluna se encontram os temas principais do exercício, na terceira coluna estão os erros cometidos pelos alunos com a quantidade de alunos que cometeu o erro e os tipos de casos teste afetados pelo erro, na quarta coluna se encontram os tipos de casos teste que podem ser aplicados ao exercício.

Tabela 1 - Resultado da análise dos exercícios de iVProg presentes no curso “2021: Introdução à Programação (Diurno)” do SAW

Exercício	Temas	Erros dos alunos	Casos Teste
EF-3.1.a Introdução - Olá mundo!	Leitura e Escrita	1 - Errou um caractere (Todos) 1 - Número errado de entradas (Todos)	Caso Geral
EF-3.1.b Introdução - Ler um inteiro e imprimí-lo	Leitura e Escrita	-----	Caso Geral Caso Nulo Caso Negativo
EF-3.2.a Consumo médio	Matemática	1 - Mais de uma função e todas vazias (Todos)	Caso Geral Caso Nulo Tipo de Entrada
EF-3.2.b Troca de valores	Manipulação de Entrada	1 - Número errado de entradas (Todos)	Caso Geral Caso Nulo Caso Negativo
EF-3.2.c Antecessor e sucessor	Lógica Matemática	1 - Número errado de entradas e mais de uma função (Todos)	Caso Geral Caso Nulo Caso Negativo
EF-3.2.d Imposto em Marcianópolis	Matemática	1 - Número errado de entradas (Todos)	Caso Geral Caso Nulo Tipo de Entrada

EF-3.2.e Folha de pagamento	Lógica Matemática	1 - Errou uma das saídas/Não prestou atenção no enunciado (Todos) 1 - Número errado de saídas e Lógica/Falta de atenção (Todos)	Caso Geral Caso Nulo Tipo de Entrada
EF-3.2.f Total de dias	Matemática	-----	Caso Geral Caso Nulo Caso Negativo
EF-3.2.g Área do círculo	Matemática	1 - Tipo de variável errado (Tipo de Entrada) 1 - Uso errôneo da função do iVProg "raiz_quadrada" (Todos)	Caso Geral Caso Nulo Tipo de Entrada
Divisão da conta do restaurante	Lógica Matemática	1 - Lógica (Todos) 1 - Número errado de entradas (Todos)	Caso Geral Caso Limite Caso Nulo Tipo de Entrada
Inverter número inteiro de 3 dígitos	Manipulação de Entrada Lógica Matemática	3 - Número de entradas errado (Todos) 3 - Uso errôneo da função do iVProg "inverter_valor" (Todos)	Caso Limite Caso Nulo Caso Negativo Igualdade
Item 4 - Média ponderada	Matemática	1 - Ordem de leitura errada (Todos) 1 - Desistiu (Todos)	Caso Geral Caso Nulo Caso Negativo
EF-5.1.a Identificar se 2 números são iguais ou diferente	Condição	-----	Caso Geral Caso Limite Caso Nulo Caso Negativo
EF-5.1.b Imprimir o maior número dentre dois inteiros	Condição	-----	Caso Limite Caso Nulo Caso Negativo Igualdade
EF-5.1.c Verificar se um número é divisível por outro	Condição	1 - Condição (Caso Limite e Igualdade)	Caso Limite Caso Nulo Igualdade Tipo de Entrada
EF-5.1.d Soma ou subtração	Condição	2 - Sinal (Caso Limite)	Caso Limite Caso Nulo Caso Negativo Igualdade
EF-5.1.e Par ou ímpar	Condição	-----	Caso Geral Caso Negativo

Soma de inteiros representando ângulos internos de um triângulo	Condição	-----	Caso Geral Caso Limite Caso Nulo Caso Negativo Igualdade
6.1 Laço - Somatório dos primeiros naturais até um limite	Laço	2 - Lógica/Não entendeu o enunciado (Caso Geral) 1 - Inicialização de variável e Atribuição da variável pós laço (Caso Limite)	Caso Geral Caso Limite
6.2 Laço - Soma dos números ímpares (até digitar 0)	Laço Condição	2 - Atribuição de variável pré laço (Caso Limite e Caso Nulo) 1 - Condição de parada (Todos) 1 - Lógica/Não entendeu o enunciado (Todos)	Caso Geral Caso Limite Caso Nulo Caso Negativo
8.1 Vetor - Soma das posições x e y	Vetor Laço	1 - Lógica/Não entendeu como funciona atribuição dentro de laços (Todos)	Caso Geral Caso Limite Caso Negativo Igualdade

Como pode-se observar a tabela não possui muitos erros dos alunos, isso ocorreu porque o iVProg armazena apenas a última execução do exercício do aluno, o que limitou o acesso aos erros anteriores à última execução.

Na análise apresentada na tabela foram levantados sete possíveis temas de exercício:

- Leitura e Escrita: exercícios que são relacionados apenas a escrita e leitura de dados.

- Manipulação de Entrada: exercícios que tem como objetivo a manipulação da entrada recebida.

- Lógica: exercícios nos quais a lógica desempenha um papel importante.

- Matemática: exercícios onde se aplicam problemas matemáticos.

- Condição: exercícios que necessitam do uso de pelo menos uma condição para que possam ser resolvidos.

- Laço: exercícios em que a solução desejada necessita do uso de pelo menos um laço para que possam ser resolvidos.

- Vetor: exercícios em que a solução desejada necessita do uso de vetores para que possam ser resolvidos.

Foram levantados também seis possíveis tipos de caso teste:

- Caso Geral: testam o código executado com valores gerais, que não tem comportamento diferenciado, ou seja, os que não se encaixam nos outros tipos de caso teste.

- Caso Limite: testam o código executado com valores que se encontram no limite da mudança de comportamento do exercício.

- Caso Nulo: testam o código com valores nulos.

- Caso Negativo: testam o código com valores negativos.

- Igualdade: testam o código com valores de entrada iguais.

- Tipo de Entrada: testam se o código executado tem o tipo certo de variável como entrada.

Os temas e os casos testes levantados são aplicáveis a problemas simples de programação, que é o caso de uso atual do iVProg, para problemas mais complexos é necessário um estudo mais aprofundado para se definir temas e tipos de caso teste.

Com os temas e os tipos de caso teste possíveis para exercícios aplicados atualmente no iVProg definidos pode-se elaborar a estrutura do banco de dados.

Para que o aluno possa ter seu desempenho dentro do curso armazenado no banco de dados é necessário que haja um campo relativo à identificação do aluno e um campo relativo à identificação do curso no banco de dados.

Para os campos relativos ao desempenho do aluno definiu-se que o desempenho seria armazenado em forma de JSON (*JavaScript Object Notation*, em português: Notação de Objeto JavaScript), contendo um objeto JavaScript que contém todo o desempenho, sendo portanto necessário apenas um único campo para armazenar o desempenho. Foi definido dessa maneira para facilitar o gerenciamento do banco de dados, pois ocorrem diversas alterações em diversas áreas do desempenho em toda execução, o que traz complexidade para o gerenciamento individual de cada aspecto do desempenho.

Todo o gerenciamento e o acesso ao banco de dados é feito através do iAssign, para isso foi criado dentro do código PHP do iAssign uma classe chamada "Performance" que realiza o gerenciamento do banco de dados e foi criado também um ponto de acesso para identificar as requisições do agente.

6.4 AGENTE

O agente desenvolvido tem como entradas o histórico de desempenho do usuário, o resultado dos casos teste do exercício realizado e os temas desse exercício.

O desempenho é obtido através do banco de dados, convertendo o JSON armazenado em um objeto chamado "performance", que possui quatro propriedades: "type", "tag", "lastTypeErrors" e "lastTagErrors".

A propriedade "type" é um vetor de objetos que tem como função armazenar informações de desempenho relativas aos possíveis temas de exercício. Os objetos que compõem o vetor possuem três propriedades, "type", "total" e "errors", a primeira é um texto que tem como função identificar o objeto pelo nome do tema ao qual esse objeto é relacionado, a segunda é um número inteiro que armazena o total de exercícios desse tema realizados, e a terceira é um número inteiro que armazena em quantos desses exercício foram cometidos erros.

A propriedade "tag" é um vetor de objetos que tem como função armazenar informações de desempenho relativas aos possíveis tipos de casos teste de exercício. Os objetos que compõem o vetor possuem três propriedades, "tag", "total" e "errors", a primeira é um texto que tem como função identificar o objeto pelo nome do tipo de caso teste ao qual esse objeto é relacionado, a segunda é um número inteiro que armazena o total de exercícios desse tipo de caso realizados, e a terceira é um número inteiro que armazena em quantos desses exercício foram cometidos erros.

As propriedades "lastTypeErrors" e "lastTagErrors" são vetores que tem como função armazenar os últimos 50 erros cometidos pelo usuário relativos aos temas e aos tipos de casos teste respectivamente.

O resultado dos casos teste é obtido através do avaliador automático, ele contém os tipos de casos teste e os resultados. Os temas do exercício também são obtidos através do avaliador automático. O avaliador automático pode fornecer os tipos de casos teste e os temas do exercício devido as novas entradas que foram adicionadas, detalhadas nos itens relativos às entradas.

Com as entradas recebidas o agente realiza a atualização do desempenho do usuário, formula as informações de desempenho que serão exibidas ao usuário em função do resultado no exercício executado, formula as sugestões que serão exibidas ao usuário em função do resultado obtido no exercício executado, e por último fornece o desempenho atualizado ao código PHP do iAssign responsável pelo gerenciamento e acesso ao banco de dados.

O código PHP desenvolvido para o gerenciamento do banco de dados compara o desempenho atualizado recebido com o desempenho atualizado calculado pelo próprio código PHP, caso sejam iguais, o banco de dados é atualizado, dessa forma é adicionada uma barreira à gravação de dados provenientes de ataques que alterem os dados enviados pelo agente ao código PHP responsável pelo gerenciamento do banco de dados.

As funções desenvolvidas para que o agente possua o funcionamento desejado são detalhadas a seguir.

A função “updateTags” analisa o resultado do exercício em relação aos tipos de caso teste do exercício, ela possui como entradas o resultado dos casos teste do exercício e o histórico de desempenho do usuário. Essa função atualiza o desempenho relativo aos tipos de casos teste do exercício e sinaliza se houve a ocorrência de algum erro no exercício.

A função “updateTypes” analisa o resultado do exercício em relação aos temas do exercício, ela possui como entradas os temas do exercício e o histórico de desempenho do usuário. Essa função atualiza o desempenho relativo aos temas do exercício.

A função “lastErrors” analisa o exercício em relação aos últimos 50 erros cometidos relativos aos temas e aos tipos de casos teste, ela possui como entrada o histórico de desempenho do usuário. Essa função contabiliza as ocorrências de cada tema e de cada tipo de caso teste dentro das filas de últimos 50 erros.

A função “findObject” encontra um objeto dentro de um vetor a partir do valor de uma de suas propriedades, ela possui como entradas o vetor, a propriedade e o valor da propriedade buscada. Essa função é uma função auxiliar desenvolvida para auxiliar no desenvolvimento das duas funções seguintes.

A função “addPerformance” formula as informações de desempenho que serão exibidas ao usuário em função do resultado no exercício executado, ela possui como entradas os temas do exercício, um vetor de tipos de casos teste do exercício em que ocorreu um erro e o histórico de desempenho atualizado do usuário.

A função “addSugestion” formula as sugestões que serão exibidas ao usuário em função do resultado obtido no exercício executado, ela possui como entradas os temas do exercício, um vetor de tipos de casos teste do exercício em que ocorreu um erro e o histórico de desempenho atualizado do usuário.

A função “analise” analisa o resultado do exercício por completo, ela faz isso através de três etapas: aquisição de dados, análise e retorno de dados. Na aquisição de dados, a função obtém as entradas necessárias para o funcionamento do agente: os temas do exercício, o resultado dos casos teste do exercício e o histórico de desempenho do usuário. Na análise, a função encadeia as funções detalhadas acima, fazendo com que o agente execute todas as funcionalidades projetadas. No retorno de dados, a função fornece à interface as sugestões e as informações de desempenho que foram formuladas para serem exibidas ao usuário, e fornece o desempenho atualizado ao código PHP do iAssign responsável pelo gerenciamento e acesso ao banco de dados.

7. TESTES E AVALIAÇÃO

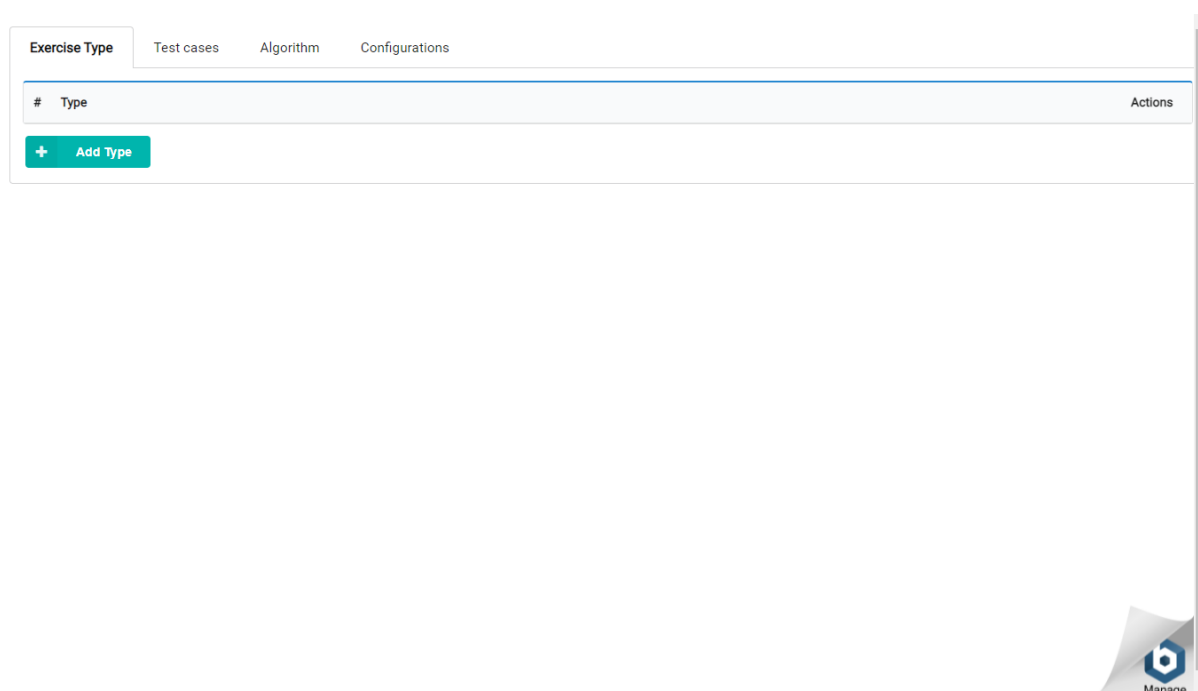
7.1 ENTRADAS

Para testar o funcionamento dos novos campos adicionados na nova versão da interface de criação da descrição de tarefa do avaliador, que foram adicionados com a finalidade de prover ao agente (assistente virtual) as entradas necessárias para seu funcionamento, é necessário testar a criação da descrição de tarefa do avaliador.

A fim de se realizar essa testagem, instalou-se em uma plataforma Moodle uma nova versão do módulo de extensão iAssign que contém a nova versão desenvolvida neste projeto do arquivo “iassign-integration-functions.js” presente no iVProg.

Inicialmente foram testadas todas funcionalidades de cada uma das abas da nova versão da interface de criação da descrição de tarefa do avaliador.

Figura 13 - Primeira aba da nova interface de criação da descrição de tarefa do avaliador



Fonte: Execução da versão deste projeto do iVProg

Figura 14 - Primeira aba da nova interface de criação da descrição de tarefa do avaliador preenchida

The screenshot shows the 'Exercise Type' tab selected in a navigation bar. Below the navigation bar, there is a table with three rows. Each row has a number in the '# Type' column and a red 'X' icon in the 'Actions' column. Below the table is a green button labeled '+ Add Type'. In the bottom right corner, there is a 'Manage' button with a gear icon.

#	Type	Actions
1	Type 2	X
2	Type 1	X
3	Type 3	X

+ Add Type

Manage

Fonte: Execução da versão deste projeto do iVProg

Figura 15 - Segunda aba da nova interface de criação da descrição de tarefa do avaliador

The screenshot shows the 'Test cases' tab selected in a navigation bar. Below the navigation bar, there is a table with columns for '# Input', 'Output', 'Tag', and 'Actions'. Below the table, there is a green button labeled '+ Add test case' and an orange button labeled 'Generate outputs'. In the bottom right corner, there is a 'Manage' button with a gear icon.

#	Input	Output	Tag	Actions
---	-------	--------	-----	---------

+ Add test case

Generate outputs

Manage

Fonte: Execução da versão deste projeto do iVProg

Figura 16 - Segunda aba da nova interface de criação da descrição de tarefa do avaliador preenchida

#	Input	Output	Tag	Actions
1	<input type="text" value="1"/>	<input type="text" value="2"/>	Tag 3 ▾	✖
2	<input type="text" value="5"/>	<input type="text" value="10"/>	Tag 1 ▾	✖
3	<input type="text" value="-2"/>	<input type="text" value="-4"/>	Tag 2 ▾	✖

+ Add test case + Generate outputs

Fonte: Execução da versão deste projeto do iVProg

Figura 17 - Terceira aba da nova interface de criação da descrição de tarefa do avaliador

Include the algorithm below in the activity

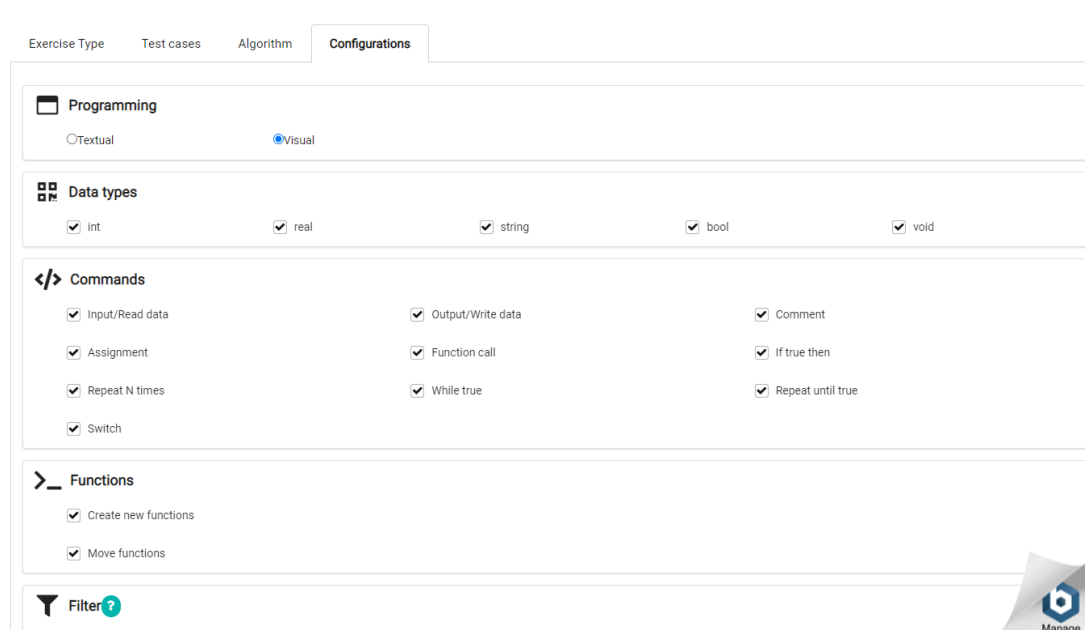
```
function void main ()
```

+ Function

Terminal

Fonte: Execução da versão deste projeto do iVProg

Figura 18 - Quarta aba da nova interface de criação da descrição de tarefa do avaliador



Fonte: Execução da versão deste projeto do iVProg

Todas as funcionalidades antigas de cada uma das abas da nova versão da interface de criação da descrição de tarefa do avaliador apresentaram o mesmo comportamento da versão anterior da interface.

Todas as novas funcionalidades de cada uma das abas da nova versão da interface apresentaram o comportamento desejado e esperado.

Com isso seguiu-se para a testagem da geração do arquivo .ivph que contém a descrição de tarefa do avaliador.

Para isso foram criadas diversas descrições de tarefa do avaliador, a fim de se observar se os campos preenchidos nas abas da nova versão da interface de criação da descrição de tarefa do avaliador são registrados de forma correta no arquivo .ivph gerado, ou seja, se apresentam o mesmo formato do arquivo .ivph gerado pela versão anterior da interface com o acréscimo dos novos campos que foram adicionados na nova versão da interface.

Figura 19 - Arquivo .ivph que contém a nova descrição de tarefa do avaliador

```
{
  "exercisetype" : ["Type 2","Type 1","Type 3" ],
  "testcases" : [
    {
      "input": ["1"],
      "output": ["2"],
      "tag": ["Tag 3"]
    },
    {
      "input": ["5"],
      "output": ["10"],
      "tag": ["Tag 1"]
    },
    {
      "input": ["-2"],
      "output": ["-4"],
      "tag": ["Tag 2"]
    }
  ],
  "settings_programming_type":
  [{"name":"programming_type","value":"visual"}],
  "settings_data_types":
  [{"name":"integer_data_type","value":"on"}, {"name":"real_data_type","value":"on"}, {"name":"text_data_type","value":"on"}, {"name":"boolean_data_type","value":"on"}, {"name":"void_data_type","value":"on"}],
  "settings_commands":
  [{"name":"commands_read","value":"on"}, {"name":"commands_write","value":"on"}, {"name":"commands_comment","value":"on"}, {"name":"commands_attribution","value":"on"}, {"name":"commands_functioncall","value":"on"}, {"name":"commands_iftrue","value":"on"}, {"name":"commands_repeatNtimes","value":"on"}, {"name":"commands_while","value":"on"}, {"name":"commands_dowhile","value":"on"}, {"name":"commands_switch","value":"on"}],
  "settings_functions":
  [{"name":"functions_creation","value":"on"}, {"name":"functions_move","value":"on"}],
  "settings_filter":
  []
}
```

Fonte: Execução da versão deste projeto do iVProg

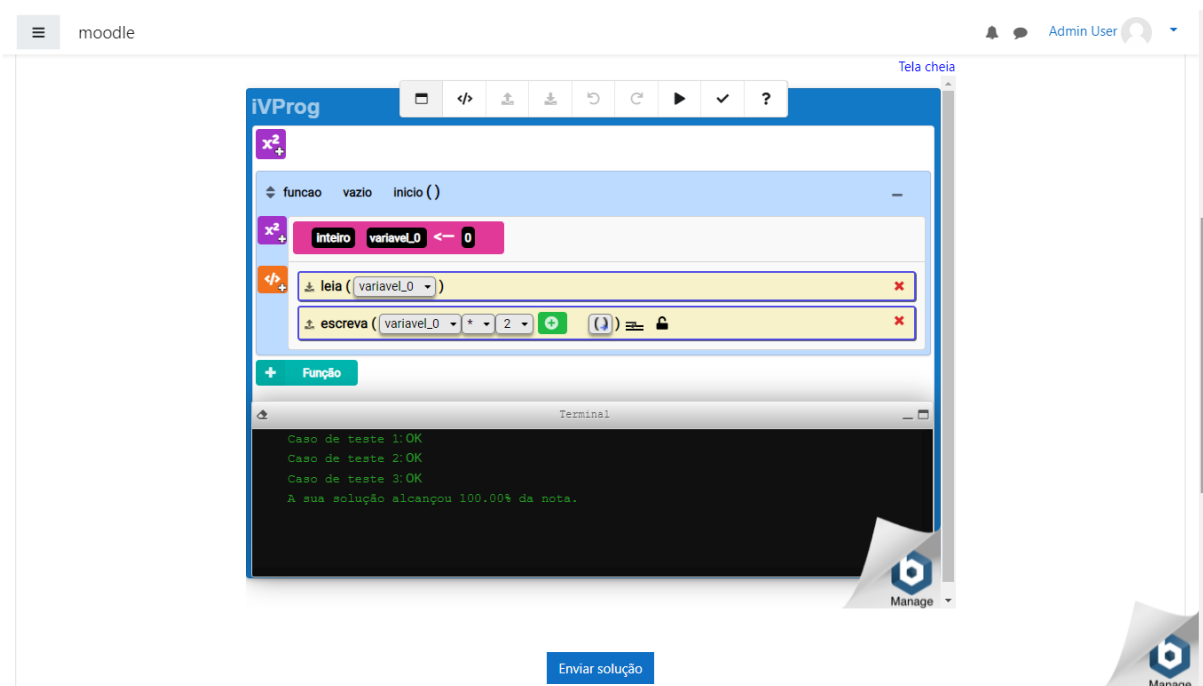
Observou-se que todos os arquivos gerados apresentaram o formato correto com o acréscimo dos novos campos que foram adicionados na nova versão da interface.

Por último, foi testado se o arquivo .ivph gerado é funcional, ou seja, se é possível executar tarefas geradas a partir de descrições de tarefa do avaliador geradas pela nova versão da interface de criação de descrição de tarefa do avaliador.

Nessa última testagem foram executadas diversas tarefas geradas a partir de diferentes descrições de tarefa do avaliador geradas pela nova versão da interface de criação de descrição de tarefa do avaliador.

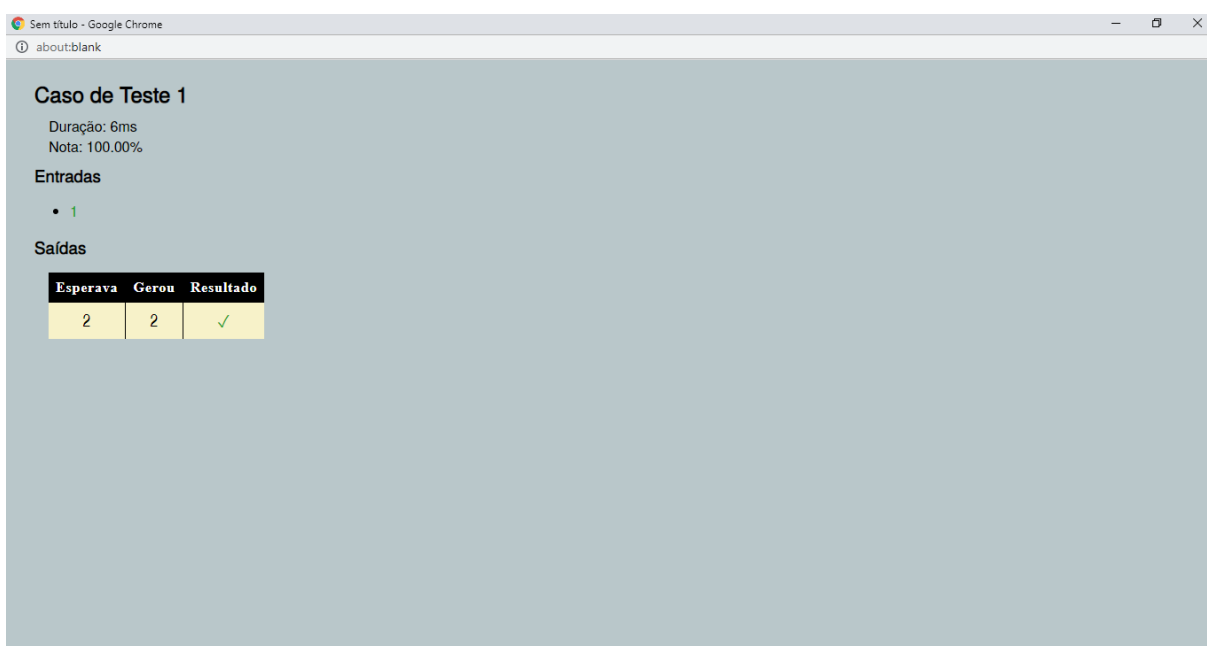
Essa testagem tem como finalidade observar se a execução das tarefas ocorre de maneira correta, ou seja, se apresenta o mesmo resultado apresentado pela versão anterior da interface, se os novos campos adicionados na nova versão da interface podem ser acessados e se esses novos campos apresentam os mesmos valores do arquivo .ivph usado para gerar essa tarefa.

Figura 20 - Execução de tarefa gerada a partir da nova descrição de tarefa do avaliador



Fonte: Execução da versão deste projeto do iVProg

Figura 21 - Resultado de caso teste da execução da tarefa gerada a partir da nova descrição de tarefa do avaliador



Fonte: Execução da versão deste projeto do iVProg

Todas as execuções de tarefas geradas a partir de diferentes descrições de tarefa do avaliador geradas pela nova versão da interface obtiveram o mesmo resultado que o apresentado pela versão anterior da interface.

Além disso, em todas as execuções, os novos campos adicionados na nova versão da interface puderam ser acessados e apresentaram os mesmos valores do arquivo .ivph usado para gerar a tarefa executada.

Com os resultados obtidos na execução de todos esses testes, as alterações feitas na interface de criação da descrição de tarefa do avaliador foram validadas.

7.2 BANCO DE DADOS

A testagem do banco de dados foi realizada ao longo do seu desenvolvimento utilizando um servidor PHP local e um banco de dados MariaDB local, com auxílio do programa livre e de código aberto HeidiSQL para visualizar o banco de dados em tempo real.

Inicialmente se testou se a tabela que armazena o histórico de desempenho do usuário no curso é gerada de maneira correta, com os campos com nomes corretos nos formatos corretos.

Observando o banco de dados exibido em tempo real pelo HeidiSQL foi constatado que a tabela está sendo gerada de forma correta.

Em seguida foi testado se é possível a adição de novos registros no banco de dados, isso foi feito através da adição de vários registros através do código PHP desenvolvido para realizar o gerenciamento do banco de dados.

Todos os registros adicionados puderam ser observados no banco de dados exibido em tempo real pelo HeidiSQL.

O teste seguinte visou verificar se é possível a remoção de registros do banco de dados, novamente o teste foi feito através do código PHP desenvolvido, realizando diversos comandos de remoção de registro do banco de dados.

Observando o banco de dados exibido em tempo real pelo HeidiSQL verificou-se que todos os registros que foram alvo de um comando de remoção no código PHP foram removidos do banco de dados.

O próximo teste consistiu em verificar se os dados do banco de dados podem ser acessados corretamente e se apresentam o formato correto. Para isso foram feitos diversos acessos a diferentes registros do banco de dados.

Todos os dados que foram acessados pelo código PHP desenvolvido para realizar o gerenciamento do banco de dados apresentaram os valores corretos no formato correto.

Por último foi testado se é possível atualizar os registros presentes no banco de dados, isso foi feito através do código PHP desenvolvido, realizando diversos comandos de atualização de registros do banco de dados.

Através do banco de dados exibido em tempo real pelo HeidiSQL se pode observar que todos os registros que foram alvo de comandos de atualização no código PHP foram atualizados de forma correta no banco de dados.

Com os resultados obtidos na execução de todos esses testes, o banco de dados e o código PHP desenvolvido para realizar o gerenciamento do banco de dados foram validados.

7.3 AGENTE

A testagem do agente foi realizada ao longo do seu desenvolvimento através de sua execução no Node.js.

Inicialmente foi testado se o agente é capaz de manipular dados de entrada que estão no formato dos dados de entrada que ele vai receber ao ser integrado com iVProg e com o iAssign. Para isso foram gerados dados de entrada no formato esperado, que foram alvos de comandos que manipulam os dados conforme manipulações que são necessárias para o funcionamento do agente projetado.

Esses testes foram validados através da exibição dos seus resultados no terminal após a execução do código pelo Node.js, os resultados exibidos mostraram que os dados foram manipulados da forma desejada com sucesso.

Após isso iniciou-se os testes das funções que foram detalhadas no projeto e implementação do agente.

Para a função “updateTags” foi testado se ela realiza a atualização do desempenho relativo aos tipos de casos teste do exercício de forma correta, e se ela sinaliza se houve a ocorrência de algum erro no exercício.

Na testagem da função “updateTypes”, foi verificado se a atualização do desempenho relativo aos temas do exercício está sendo feita de maneira correta.

Os testes da função “lastErrors” consistiram em verificar se ela está contabilizando corretamente as ocorrências de cada tema e de cada tipo de caso teste dentro das filas de últimos 50 erros cometidos relativos aos temas e aos tipos de casos teste.

Para testar a função “findObject” se checou se ela consegue encontrar um objeto dentro de um vetor a partir do valor de uma de suas propriedades.

Nos testes da função “addPerformance” foi verificado se ela formula de maneira correta as informações de desempenho que serão exibidas ao usuário em função do resultado no exercício executado.

Na testagem da função “addSugestion” foi checado se ela formula corretamente as sugestões que serão exibidas ao usuário em função do resultado obtido no exercício executado.

Para a função “analise” foram testadas duas de suas três etapas, a análise e o retorno de dados, já que a etapa de aquisição de dados será testada nos testes de integração. Nos testes da etapa de análise foi verificado se as funções são encadeadas de maneira correta, ou seja, se todas as funcionalidades projetadas são executadas corretamente. Nos testes da etapa de retorno de dados foi checado se as sugestões de desempenho e as informações de desempenho que foram formuladas para serem exibidas ao usuário são fornecidas no formato correto, e se o desempenho atualizado também é fornecido no formato correto.

Todos os testes de funções descritos acima foram realizados para diversas entradas diferentes, e foram validados através da observação dos seus resultados no terminal após a execução das funções pelo Node.js, os resultados observados em cada função foram os esperados.

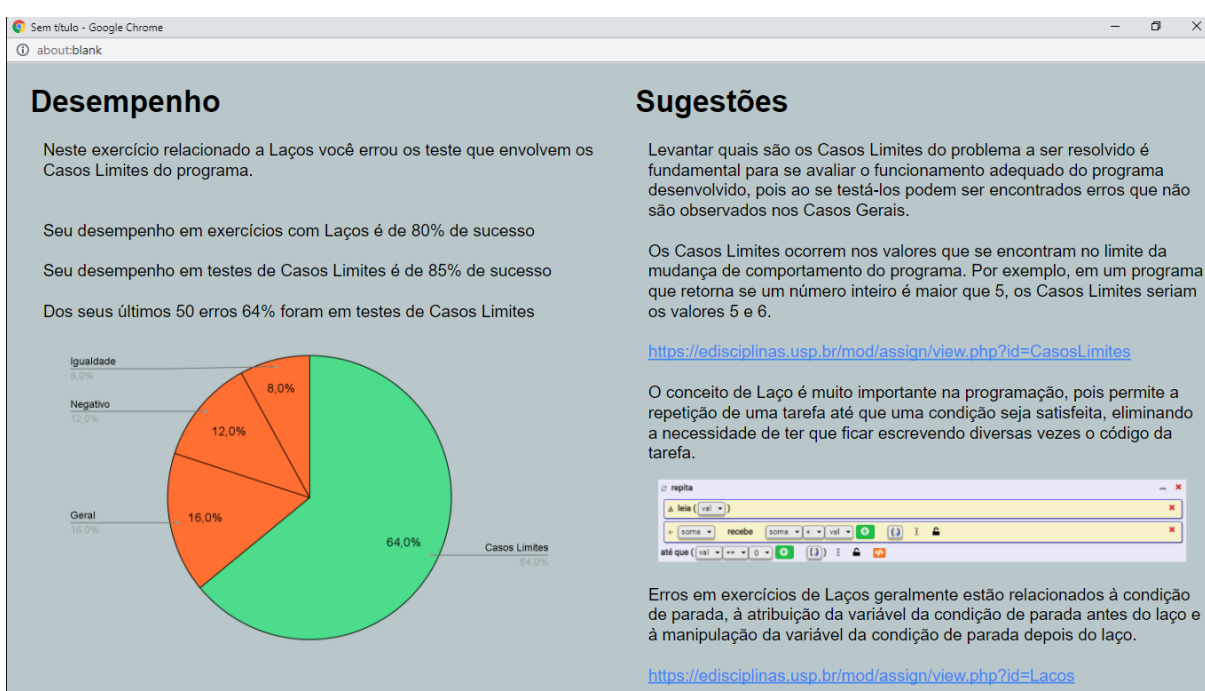
A execução e os resultados dos testes descritos acima possibilitaram validar o comportamento do agente projetado.

7.4 INTERFACE

Para testar o funcionamento da interface projetada é necessário testar se a interface apresenta o visual desejado.

A fim de se realizar essa testagem foram usadas diferentes entradas de desempenho e de sugestões para gerar janelas do navegador contendo o conteúdo dessas entradas.

Figura 22 - Interface exibida pelo agente



Fonte: Execução da versão deste projeto do iVProg

Todas as janelas de navegador geradas para as diferentes entradas de desempenho e de sugestões apresentaram o visual conforme o projetado para a interface do assistente.

A execução e os resultados do teste descrito acima possibilitaram validar o comportamento da interface projetada para o assistente.

7.5 INTEGRAÇÃO

As testagens de integração foram iniciadas após a validação de todos os testes descritos acima.

Inicialmente foi testada a integração do agente com o gerenciador do banco de dados, foi verificado se o agente consegue requisitar dados ao gerenciador do banco de dados, se o agente consegue enviar dados ao gerenciador de banco de dados e se esses dados requisitados e enviados podem ser utilizados corretamente pelo agente e pelo gerenciador de banco de dados.

Para verificar se o agente consegue requisitar dados ao gerenciador do banco de dados, foi testado se o gerenciador de banco de dados consegue identificar uma requisição do agente e se consegue enviar ao agente os dados desejados do banco de dados a partir da identificação dessa requisição.

As requisições de dados feitas pelo o agente ao gerenciador de banco de dados foram executadas com sucesso, o gerenciador de banco de dados identificou com sucesso a requisição de dados do agente, os dados foram recebidos de maneira correta no formato correto, e o agente conseguiu executar com sucesso todas as suas funcionalidades utilizando os dados recebidos do gerenciador de banco de dados via requisição.

Para verificar se o agente consegue enviar dados ao gerenciador de banco de dados, foi testado se o gerenciador de banco de dados consegue identificar o envio de dados do agente e se consegue acessar esses dados de maneira correta a partir da identificação desse envio de dados.

Os envios de dados feitos pelo agente ao gerenciador de banco de dados foram realizados com sucesso, o gerenciador de banco de dados identificou com sucesso o envio de dados, os dados foram recebidos de maneira correta no formato correto, e o gerenciador de banco de dados conseguiu executar com sucesso todas as suas funcionalidades que utilizam os dados recebidos do agente.

Com os resultados obtidos nos testes relatados acima, foi possível validar a integração do agente com o gerenciador do banco de dados.

Em seguida foi realizada a testagem da integração do agente com o iVProg, foi verificado se o agente consegue acessar os dados do iVProg necessários para

seu funcionamento, se o agente consegue enviar ao iVProg os dados necessários para seu funcionamento e se esses dados acessados e enviados podem ser utilizados corretamente pelo agente e pelo iVProg.

Os acessos aos dados do iVProg feitos pelo agente foram realizados com sucesso, os dados enviados pelo agente ao iVProg foram recebidos de maneira correta no formato correto, o agente conseguiu executar com sucesso todas as suas funcionalidades que utilizam os dados que foram acessados do iVProg, e o iVProg conseguiu executar com sucesso todas as suas funcionalidades que utilizam os dados que foram recebidos do agente.

Os resultados obtidos nos testes relatados acima possibilitaram validar a integração do agente com o iVProg.

Por último foi testada a integração entre o agente, o iVProg, o iAssign e o Moodle, ou seja, foi testado o produto final do projeto. Foi verificado se o iVProg manteve as funcionalidades da versão IV funcionando corretamente e se as novas funcionalidades adicionadas apresentaram o funcionamento projetado de forma correta.

Para verificar se o iVProg manteve as funcionalidades da versão IV funcionando corretamente, testou-se a criação da descrição de tarefa do avaliador, a criação de tarefa, a execução de código no iVProg e a avaliação automática.

A criação da descrição de tarefa do avaliador, a criação de tarefa, a execução de código no iVProg e a avaliação automática apresentaram o mesmo comportamento que é apresentado na versão IV do iVProg.

Para verificar se as novas funcionalidades adicionadas apresentaram o funcionamento projetado de forma correta, foi testado se o desempenho e as sugestões são gerados e exibidos de forma correta ao usuário e se o histórico de desempenho é atualizado corretamente.

O desempenho e as sugestões foram gerados e exibidos de forma correta, e o histórico de desempenho foi atualizado corretamente.

Com os resultados obtidos nos testes relatados acima, foi possível validar a integração entre o agente, o iVProg, o iAssign e o Moodle, ou seja, foi possível validar o produto final do projeto.

8. CONSIDERAÇÕES FINAIS

8.1 CONCLUSÕES DO PROJETO DE FORMATURA

O projeto final conseguiu alcançar os objetivos e os requisitos de sistema que foram definidos na elaboração e no planejamento do projeto, apresentando as funcionalidades propostas nos formatos planejados.

Foi desenvolvido um assistente virtual personalizado para o sistema educacional iVProg em integração com o ambiente Moodle, que permite que os professores possam fornecer uma assistência “personalizada” para cada aluno mesmo sem ter o tempo disponível para isso. O assistente desenvolvido apresenta diagnósticos sobre o conhecimento do aluno, com dicas e caminhos de como melhorar nos pontos fracos, um comportamento semelhante ao de um tutor.

No entanto, acreditava-se que seriam alcançados objetivos extras, como funcionalidades adicionais, estudo mais aprofundado para definir temas e tipos de caso teste, otimização da comunicação entre o agente e o gerenciador de banco de dados. Esses objetivos extras são detalhados nas perspectivas de continuidade.

O primeiro empecilho encontrado foi em relação a definição de temas e tipos de caso teste, não foi encontrada uma bibliografia relevante sobre o tema, e os dados disponíveis sobre os erros cometidos por alunos no iVProg são escassos, com isso foi possível se definir apenas temas e casos testes aplicáveis a problemas simples de programação, o que foi suficiente já que esse é o caso de uso atual do iVProg.

Uma outra dificuldade, porém em menor grau, foi o aprendizado sobre as tecnologias utilizadas, havia se estimado uma menor curva de aprendizado, mas a falta de familiaridade com a maioria das tecnologias utilizadas estendeu a curva de aprendizado além do estimado.

Mas a maior dificuldade encontrada foi na documentação escassa do iVProg e principalmente do iAssign, ambas as aplicações possuem diversos arquivos e diversas funções que realizam uma grande variedade de tarefas, se familiarizar com elas sem foi extremamente difícil com a ausência de documentação, e no iAssign

existiu uma dificuldade extra devido a falta de boas práticas na organização do código.

8.2 CONTRIBUIÇÕES

O desenvolvimento de um assistente virtual personalizado para o iVProg é um projeto originado de uma ideia que pode ser aplicada em outros sistemas virtuais de aprendizado, de forma a permitir que professores possam fornecer uma assistência “personalizada” para cada um de seus alunos mesmo sem ter o tempo disponível para isso. Ao visualizar seu histórico de desempenho e as sugestões do assistente, o aluno tem uma informação baseada em dados do que ele precisa estudar e tem as sugestões de como fazê-lo, algo que possibilita estreitar a curva de aprendizado do aluno. Isso é algo que ainda não é aplicado na maioria dos sistemas virtuais de aprendizagem, mas que pode contribuir para a melhora desse formato de ensino via sistemas virtuais de aprendizagem.

Outra contribuição é o desenvolvimento do assistente virtual personalizado para o iVProg, que pode ter sua complexidade aumentada para fornecer melhor assistência aos alunos, pois como já existe um assistente com as funções básicas necessárias e que faz a comunicação entre o iVProg e a base de dados do Moodle via iAssign, facilita a aprimoração de funcionalidades já existentes e a adição de novas funcionalidades, já que não é necessário desenvolver um assistente com essas funções básicas e que faça essa comunicação.

8.3 PERSPECTIVAS DE CONTINUIDADE

Uma forma de dar seguimento a esse projeto é realizar um estudo mais aprofundado sobre temas e tipos de caso teste de exercícios de programação, acompanhando o uso do assistente em cursos ministrados, analisando o seu impacto no aprendizado e na quantidade de erros cometidos, colhendo opiniões dos alunos e professores sobre os temas e tipos de caso teste, estudando a fundo as aplicações dos conceitos de programação, dentre outros procedimentos, de forma

que se defina temas e tipos de caso teste que abranjam exercícios de programação de todas as complexidades.

Outra maneira de dar continuidade a esse projeto é focar no desenvolvimento de uma otimização na comunicação entre o agente e o gerenciador de banco de dados, nesse caso o desenvolvimento da maior parte do projeto seria no iAssign, aprimorando a segurança da comunicação feita com o Moodle, organizando e aplicando boas práticas no código existente, documentando o código existente, de forma que se facilite a manutenção e que aumente a segurança desse código.

Uma das perspectivas de continuidade desse projeto se dá através do desenvolvimento de novas funcionalidades para o assistente, como por exemplo a possibilidade de inserir um conteúdo associado ao erro em um caso teste na aba referente aos casos teste na criação da descrição de tarefa do avaliador, um outro exemplo seria inserir a possibilidade de se escolher utilizar um conteúdo específico em um tema ou tipo de caso existente ao invés do conteúdo padrão desse tema ou tipo de caso teste, o ideal seria elaborar as novas funcionalidades a partir do acompanhamento do uso do assistente em cursos ministrados, colhendo opiniões dos alunos e dos professores sobre o assistente, a fim de se desenvolver novas funcionalidades que terão impacto positivo na utilização do assistente.

Outra forma de dar seguimento ao projeto é aplicar o tipo de assistente desenvolvido em outros sistemas virtuais de aprendizado, de forma a permitir que professores possam fornecer uma assistência “personalizada” para cada um de seus alunos mesmo sem ter o tempo disponível para isso. É possível se utilizar desse tipo de assistente em matérias de ensino fundamental II e de ensino médio, em cursos pré-vestibulares, em cursos de concursos, em cursos específicos, dentre outras áreas de ensino que façam uso de sistemas virtuais de aprendizado.

REFERÊNCIAS

- SILVA, E. C.; VALORE, L. A. Educação e ascensão social: produção de sentidos nos discursos de egressos de um programa social da iniciativa privada. **Psicologia em Revista**, Belo Horizonte, v. 25, n. 1, p. 176-198, Jan. 2019. Disponível em <http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1677-11682019000100011>. Acesso em: 13 jun. 2021.
- ARANHA, M. L. A. **História da Educação e da Pedagogia: Geral e Brasil**. 4. ed. São Paulo. Moderna, 2020. 432 p.
- FISCHER, S. R. **História da Escrita**. 1. ed. São Paulo. Editora Unesp, 2009. 293 p.
- ENCYCLOPÆDIA BRITANNICA. **Education**. Encyclopædia Britannica, Inc. Disponível em: <<https://www.britannica.com/topic/education>>. Acesso em: 13 jun. 2021.
- CONNELL, W. F. **A History of Education in the Twentieth Century World**. 1. ed. Canberra. Curriculum Development Centre, 1980. 478p.
- MOODLE. **Online Learning History**. Disponível em: <https://web.archive.org/web/20070204052431/http://docs.moodle.org/en/Online_Learning_History>. Acesso em: 14 jun. 2021.
- BITZER, D.; LYMAN, E.; EASLEY, J. **The Uses of PLATO: a computer controlled teaching system**. Urbana: Coordinated Science Laboratory, University of Illinois, 1965. (Report R-268).
- DAVIS, C. **Fundamentals of PLATO Programming**. 1. ed. Urbana. Computer-based Education Research Laboratory, University of Illinois, 1980. 110 p.
- DEAR, B. **The Friendly Orange Glow: The untold story of the PLATO System and the dawn of cyberculture**. 1. ed. New York. Pantheon Books, 2017. 640 p.
- HAAG S.; CUMMINGS, M. **Management Information Systems for the Information Age Paperback**. 10. ed. New York. McGraw-Hill Education, 2019. 608 p.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 9241-210:2019 — Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems**. 2. ed. London. ISO/TC 159/SC 4 Ergonomics of human-system interaction, 2019. 33 p.
- JOHNSTON, W. M.; HANNA, J. R. P.; MILLAR, R. J. Advances in Dataflow Programming Languages. **ACM Computing Surveys**, New York, v. 36, n. 1, p. 1–34, Mar. 2004. Disponível em: <<http://www.cs.ucf.edu/~dcm/Teaching/COT4810-Spring2011/Literature/DataFlowProgrammingLanguages.pdf>>. Acesso em: 15 jun. 2021.

KUHAIL, M. A.; FAROOQ, S.; HAMMAD, R.; BAHJA, M. Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review. **IEEE Access**, v. 9, n. 1, p. 14181-14202, Jan. 2021. Disponível em: <<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9320477>>. Acesso em: 15 jun. 2021.

JACKO, J. A. **Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications**. 3. ed. Boca Raton. CRC Press, 2012. 1518 p.

CARROLL, J. Human-computer interaction: Psychology as a science of design. **Annual Review of Psychology**, v. 48, n. 1, p. 61-83, Fev. 1997. Disponível em: <https://www.researchgate.net/publication/5287803_Human-computer_interaction_Psychology_as_a_science_of_design>. Acesso em: 16 jun. 2021.

SOEGAARD, M.; DAM, R. F. **The Encyclopedia of Human-Computer Interaction**. 2. ed. Aarhus. Interaction Design Foundation, 2014. Disponível em: <<https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed>>. Acesso em: 16 jun. 2021.

BARBOSA, S.D.J.; SILVA, B.S. **Interação Humano-Computador**. 1. ed. Rio de Janeiro. Elsevier, 2010. 408 p.

BENYON, D. **Interação Humano-Computador**. 2. ed. São Paulo. Pearson Brasil, 2011. 464 p.

ROCHA, H. V.; BARANAUSKAS, M. C. C. **Design e Avaliação de Interfaces Humano-Computador**. 1. ed. Campinas. NIED/Unicamp, 2003. 242 p.

ROGERS, Y.; SHARP, H.; PREECE, J. **Design de Interação: Além da Interação Humano-Computador**. Tradução de Isabela Gasparini. Revisão de Marcelo Soares Pimenta. 3. ed. Porto Alegre. Bookman, 2013. 600 p.

KAMIYA, R. R.; BRANDÃO, L. O. iVProg - um Sistema para Introdução à Programação Através de um Modelo Visual na Internet. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 20., 2009, Florianópolis. **Anais do XX Simpósio Brasileiro de Informática na Educação**. Florianópolis: Sociedade Brasileira de Computação, 2009. Disponível em: <<https://www.ime.usp.br/~leo/artigos/artigo-sbie-2009-10-14c-iVProg.pdf>>. Acesso em: 17 jun. 2021.

BRANDÃO, L. O.; BRANDÃO, A. A. F.; RIBEIRO, R. S. iVProg – uma Ferramenta de Programação Visual para o Ensino de Algoritmos. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 1., 2012, Rio de Janeiro. **Anais dos Workshops do I Congresso Brasileiro de Informática na Educação**. Rio de Janeiro: Sociedade Brasileira de Computação, 2012. Disponível em: <<http://www.br-ie.org/pub/index.php/wcbie/article/viewFile/1668/1430>>. Acesso em: 17 jun. 2021.

RIBEIRO, R. S.; BRANDÃO, L. O.; RODRIGUES, P. A.; BRANDÃO, A. A. F.; ISOTANI, S. *iVProg e iTarefa: Aprimorando o Ensino de Algoritmos e Programação para Iniciantes*. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 1., 2012, Rio de Janeiro. **Anais dos Workshops do I Congresso Brasileiro de Informática na Educação**. Rio de Janeiro: Sociedade Brasileira de Computação, 2012. Disponível em: <<https://www.br-ie.org/pub/index.php/wcbie/article/view/1879/1644>>. Acesso em: 17 jun. 2021.

RIBEIRO, R. S.; BRANDÃO, L. O.; BRANDÃO, A. A. F. Uma visão do cenário Nacional do Ensino de Algoritmos e Programação: uma Proposta Baseada no Paradigma de Programação Visual. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 23., 2012, Rio de Janeiro. **Anais do XXIII Simpósio Brasileiro de Informática na Educação**. Rio de Janeiro: Sociedade Brasileira de Computação, 2012. Disponível em: <<https://www.br-ie.org/pub/index.php/sbie/article/view/1797/1558>>. Acesso em: 17 jun. 2021.

BRANDÃO, A. A. F.; BRANDÃO, L. O.; SOUZA, L. M.; FELIX, I.; FERREIRA, B. *iVProg: Programação Interativa Visual e Textual na Internet*. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 8., 2019, Brasília. **Anais dos Workshops do VIII Congresso Brasileiro de Informática na Educação**. Brasília: Sociedade Brasileira de Computação, 2019. p. 1164-1171. Disponível em: <https://www.researchgate.net/publication/337528860_iVProg_Programacao_Interativa_Visual_e_Textual_na_Internet>. Acesso em: 17 jun. 2021.

IVPROG. **iVProg - LinE (Free Educational Software and Contents)**. Universidade de São Paulo. Disponível em: <<https://www.usp.br/line/ivprog/>>. Acesso em: 17 jun. 2021.

IVPROG. **Repositório do iVProg**. Laboratório de Informática na Educação. Disponível em: <<http://200.144.254.107/git/LInE/iVProg>>. Acesso em: 17 jun. 2021.

MOODLE. **About Moodle**. Disponível em: <https://docs.moodle.org/311/en/About_Moodle>. Acesso em: 18 jun. 2021.

MOODLE. **Philosophy**. Disponível em: <<https://docs.moodle.org/311/en/Philosophy>>. Acesso em: 18 jun. 2021.

MOODLE. **Documentation**. Disponível em: <https://docs.moodle.org/311/en/Main_page>. Acesso em: 18 jun. 2021.

RODRIGUES, P. A. **iTarefa**: Componente Moodle para Incorporar Módulos de Aprendizagem Interativa em Cursos Web. 2011. 110 p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2011. Disponível em: <<https://www.teses.usp.br/teses/disponiveis/45/45134/tde-11042011-095825/publico/iAssignMoodle.pdf>>. Acesso em: 19 jun. 2021.

RODRIGUES, P. A.; BRANDÃO, L. O. iTarefa: Componente Moodle para Incorporar Módulos de Aprendizagem Interativa em Cursos Web. In: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 1., 2012, Rio de Janeiro. **Anais dos Workshops do I Congresso Brasileiro de Informática na Educação**. Rio de Janeiro: Sociedade Brasileira de Computação, 2012. Disponível em: <<https://www.br-ie.org/pub/index.php/wcbie/article/view/1669/1432>>. Acesso em: 19 jun. 2021.

IASSIGN. **iAssign**: Interactive Assignments. Universidade de São Paulo. Disponível em: <<https://www.matematica.br/ia/>>. Acesso em: 19 jun. 2021.

IASSIGN. **Repositório do iAssign**. Laboratório de Informática na Educação. Disponível em: <<http://200.144.254.107/git/LInE/iassign>>. Acesso em: 19 jun. 2021.

IASSIGN. **iAssign Plugin on Moodle**. Moodle. Disponível em: <https://moodle.org/plugins/mod_iassign>. Acesso em: 19 jun. 2021.

IASSIGN. **iAssign Moodle Documentation**. Moodle. Disponível em: <<https://docs.moodle.org/311/en/iAssign>>. Acesso em: 19 jun. 2021.

IASSIGN. **Tutorial Desenvolvendo iMA em HTML5 e JavaScript**. Instituto de Matemática e Estatística da Universidade de São Paulo. Disponível em: <<https://www.ime.usp.br/~igormf/ima-tutorial/>>. Acesso em: 19 jun. 2021.

JAVASCRIPT. **Tutoriais de JavaScript**. MDN Web Docs. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>>. Acesso em: 20 jun. 2021.

JAVASCRIPT. **Padrão JavaScript**. ECMA International. Disponível em: <<https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>>. Acesso em: 20 jun. 2021.

NODE.JS. **About Node.js**. Disponível em: <<https://nodejs.org/en/about/>>. Acesso em: 20 jun. 2021.

NODE.JS. **About Documentation**. Disponível em: <<https://nodejs.org/en/docs/>>. Acesso em: 20 jun. 2021.

PHP. **PHP**. Disponível em: <<https://www.php.net/>>. Acesso em: 21 jun. 2021.

PHP. **Documentation**. Disponível em: <<https://www.php.net/docs.php>>. Acesso em: 21 jun. 2021.

PHP. **The PHP.net Wiki**. Disponível em: <<https://wiki.php.net/>>. Acesso em: 21 jun. 2021.

MARIADB. **About MariaDB Server**. Disponível em: <<https://mariadb.org/about/>>. Acesso em: 22 jun. 2021.

MARIADB. **Documentation**. Disponível em: <<https://mariadb.org/documentation/>>. Acesso em: 22 jun. 2021.

HEIDISQL. **HeidiSQL**. Disponível em: <<https://www.heidisql.com/>>. Acesso em: 22 jun. 2021.

HEIDISQL. **Basic Help on Using HeidiSQL**. Disponível em: <<https://www.heidisql.com/help.php>>. Acesso em: 22 jun. 2021.

HEIDISQL. **Forum**. Disponível em: <<https://www.heidisql.com/forum.php>>. Acesso em: 22 jun. 2021.