

Dilson José Lins Rabêlo Júnior

**Cosmo: Um ambiente virtual de aprendizado
com foco na Introdução de Algoritmos**

São Luís

2018

Dilson José Lins Rabêlo Júnior

Cosmo: Um ambiente virtual de aprendizado com foco na Introdução de Algoritmos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Orientador: Carlos de Salles Soares Neto

São Luís

2018

José Lins Rabêlo Júnior, Dilson.

Cosmo: Um ambiente virtual de aprendizado com foco na Introdução de Algoritmos/ Dilson José Lins Rabêlo Júnior. – São Luís, 2018-79p. : il. (algumas color.) ; 30 cm.

Orientador: Carlos de Salles Soares Neto

Dissertação de mestrado – Universidade Federal do Maranhão

Centro de Ciências Exatas e Tecnologia

Programa de Pós-Graduação em Ciência da Computação, 2018.

1. Algoritmos. 2. Cosmo. 3. Plataforma de ensino. I. de Salles Soares Neto, Carlos.

II. Título

Dilson José Lins Rabêlo Júnior

Cosmo: Um ambiente virtual de aprendizado com foco na Introdução de Algoritmos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Trabalho aprovado. São Luís, 31 de julho de 2018

Carlos de Salles Soares Neto
(Dr. em Informática - UFMA)

Samyr Béliche Vale, Prof. Dr.
(Dr. em Informática - UFMA)

Rômulo Martins França, Prof. Dr.
(Dr. em Informática na Educação - UFMA)

São Luís
2018

Este trabalho é dedicado a Deus, minha família e amigos.

Agradecimentos

Agradeço primeiramente a Deus pro me acompanhar em mais essa jornada. Agradeço aos meus pais, Dilson Rabelo e Rita Maria, por me apoiarem sempre em qualquer projeto que me envolvo. Eles podem ter certeza que esse sonho é deles também. A minha esposa amada Aldrea Rabelo, por não sair do meu lado mesmo nos momentos mais difíceis, sendo o meu pilar de sustentação.

Agradeço aos grandes amigos que sempre me incentivaram, que me aconselharam e me ajudaram durante o Mestrado. Em especial, Clidenor Neto, Hedvan Fernandes, Rômulo França, Geraldo Braz, Simara Rocha, Daniel, Jullyana Fialho, Higo Sampaio, Maurício Pessoa, Ruy de Oliveira, Profª Auxiliadora, Marcelo Monier e tantos outros, a lista é grande e não caberia aqui. Obrigado por se fazerem presentes.

Não posso deixar de agradecer também aos amigos do trabalho, que em minhas ausências puderam compreender a minha necessidade e me apoiaram. Obrigado a Alisson Gomes, Edilson Santos, Genilson Soares, Luis Filipe, Lázaro Henrique, Gabriel Castro, Vital Vital, Jefferson Paixão, Diego Figueiredo, Elza Bernardes (DINDA, minha afilhada), Regimarina Reis, Katherine Marjorie e mais uma listinha bem extensa. É claro que não posso esquecer, gratidão eterna, a Ana Emília Figueiredo, obrigado pelo apoio, compreensão, auxílio e reconhecimento.

Sei que não é comum colocarem isso em um agradecimento, mas não posso esquecerlos. Obrigado a Kika e Luffy, vocês fazem minha alegria até nos momentos mais estressantes.

Obrigado a todos os professores do Departamento de Informática na UFMA. São amizades que espero nunca perder.

Nas últimas semanas de escrita da dissertação tive o apoio de uma pessoal especial. Obrigado Alex Souza, seu auxílio foi crucial no desenvolvimento deste trabalho.

Um agradecimento super especial ao meu orientador, Carllos de Salles, que em nenhum minuto deixou de acreditar no meu trabalho, que me incentivou a prosseguir, teve uma paciência pra aguentar minhas "lengas lengas". Sem sua presença nada disso teria se concretizado. Espero trabalhar contigo novamente em outras empreitadas.

*"Ao infinito e além".
(Buzz Lightyear)*

Resumo

O Cosmo é uma plataforma de ensino multitarefa extensível por plugins, focada em atividades voltadas ao estudo de algoritmos. Esta plataforma possui uma camada de gerenciamento de *Logs*, que armazena diversas informações. A proposta deste trabalho é de analisar os *Logs* do Cosmo em busca de fatores que possam auxiliar melhorias de requisitos da plataforma e identificar a curva de aprendizagem diferenciada de alunos. No experimento é analisado o progresso de uma turma de 45 alunos de graduação de uma disciplina de Introdução a Algoritmos. Uma avaliação qualitativa individualizada e uma análise na turma foram efetuadas, mostrando que a coleta dos dados do Cosmo podem apoiar uma avaliação bem detalhada do progresso de aprendizes.

Palavras-chaves: Cosmo. algoritmos. plataforma de ensino.

Abstract

The Cosmo is a multitasking teaching platform extensible by plugins, focused in activities related to the study of algorithms. This platform has a layer of *Logs* management, which stores various information. The proposal of this work is to analyze the *Logs* of the Cosmo in search of factors that may help platform requirements and identify the differentiated student learning curve. In the experiment it is analyzed the progress of a class of 45 undergraduate students in a course of Introduction to Algorithms. An individualized qualitative assessment and a class analysis were carried out, showing that the collection of the Cosmo data can support the progress evaluation of learners.

Key-words: Cosmo. algorithms. teaching platform.

Lista de ilustrações

Figura 1 – O jogo <i>Lightbot</i>	19
Figura 2 – O jogo <i>The Foos</i>	20
Figura 3 – <i>Interface</i> do Code Combat	21
Figura 4 – Projeto Alice	22
Figura 5 – Organização da interface do Alice	23
Figura 6 – Plataforma CodeAcademy	24
Figura 7 – Projeto Euler - Área de Questões	24
Figura 8 – Projeto Euler - Área de Resposta	25
Figura 9 – <i>Code Hunt</i> - Seleção de fases	26
Figura 10 – <i>Code Hunt</i> - Caçando Bugs	27
Figura 11 – Exemplo de um aplicativo desenvolvido no <i>Scratch</i>	27
Figura 12 – Versão de MVP do Cosmo	33
Figura 13 – Pesquisa de Satisfação - Uso do Cosmo	34
Figura 14 – Como os alunos visualizam o Cosmo	35
Figura 15 – Nova <i>interface</i> do Cosmo área de Dashboard	35
Figura 16 – Fluxo de funcionamento do Cosmo	37
Figura 17 – Esquema do Cosmo	38
Figura 18 – <i>Dashboard</i> do Cosmo	39
Figura 19 – Caixa de Atividade	40
Figura 20 – Área de Resposta	40
Figura 21 – Histórico de Atividade	41
Figura 22 – Arquitetura do Cosmo	41
Figura 23 – Estrutura do Plugin	43
Figura 24 – Modelo de classe de configuração do <i>plugin</i>	44
Figura 25 – Estrutura de Dados de uma atividade	45
Figura 26 – Diagrama de classe do <i>plugin</i>	46
Figura 27 – Arquivo de Configuração do Plugin	47
Figura 28 – Classe <i>Hydrator</i> do <i>plugin</i> Problema	47
Figura 29 – Modelo do <i>plugin</i> Problema	48
Figura 30 – Implementação do método <code>__invoke</code>	49
Figura 31 – Estendendo layout	50
Figura 32 – Exemplo do código-fonte criado pelo Aluno Exemplar (à esquerda) e exemplo de código fonte criado pelo Aluno Ausente (à direita)	56
Figura 33 – Quadro de submissão do aluno Participativo	56
Figura 34 – Gráficos de aproveitamento de uso: Alunos Exemplares e Desatentos	59
Figura 35 – Termo de Consentimento Livre e Esclarecimento - Página 1	66

Figura 36 – Termo de Consentimento Livre e Esclarecimento - Página 2	67
Figura 37 – Questionário de Expectativa do Cosmo - Página 1	68
Figura 38 – Questionário de Expectativa do Cosmo - Página 2	69
Figura 39 – Resultado do Questionário de Expectativa do Cosmo - Página 1	70
Figura 40 – Resultado do Questionário de Expectativa do Cosmo - Página 2	71
Figura 41 – Resultado do Questionário de Expectativa do Cosmo - Página 3	72
Figura 42 – Resultado do Questionário de Expectativa do Cosmo - Página 4	73
Figura 43 – Wireframe versão atual Cosmo - Página 1	74
Figura 44 – Wireframe versão atual Cosmo - Página 2	75
Figura 45 – Wireframe versão atual Cosmo - Página 3	76
Figura 46 – Wireframe versão atual Cosmo - Página 4	77
Figura 47 – Wireframe versão atual Cosmo - Página 5	78
Figura 48 – Wireframe versão atual Cosmo - Página 6	79

Lista de tabelas

Tabela 1 – Requisitos Gerais do Cosmo - Visão do Professor	29
Tabela 2 – Catálogo de ferramentas - <i>Benchmarking</i> x <i>Plugins</i>	30
Tabela 3 – Visão Geral dos Requisitos Funcionais	30
Tabela 4 – Lista de histórias que compõem o RF01.	30
Tabela 5 – Lista de histórias que compõem o RF02.	31
Tabela 6 – Lista de histórias que compõem o RF03.	31
Tabela 7 – Lista de histórias que compõem o RF04.	32
Tabela 8 – Lista de histórias que compõem o RF05.	32
Tabela 9 – Lista de histórias que compõem o RF06.	32
Tabela 10 – Tabela de Eventos de <i>Log</i> do Cosmo	52
Tabela 11 – Agrupamentos Temáticos x Questões Respondidas	54
Tabela 12 – Alunos Selecionados para Avaliação Qualitativa	55
Tabela 13 – Perfis Alunos na Plataforma Cosmo	58
Tabela 14 – Quantidade de alunos agrupados por perfis	59

Lista de abreviaturas e siglas

AVA	Ambiente Virtual de Aprendizagem
HTML	HyperText Markup Language
IDE	Integrated Development Environment
MIT	Massachusetts Institute of Technology
MVP	Minimum Viable Product
SGBD	Sistema de Gerenciamento de Banco de Dados
WYSIWYG	What You See Is What You Get

Sumário

	Introdução	15
I	METODOLOGIA	17
1	FERRAMENTAS RELACIONADAS	18
1.1	Catálogo de Jogos Eletrônicos	19
1.1.1	<i>LightBot</i>	19
1.1.2	<i>The Foos</i>	20
1.1.3	<i>CodeCombat</i>	20
1.2	Catálogo de Plataformas de Ensino	21
1.2.1	Alice	21
1.2.2	Codecademy	23
1.2.3	Projeto Euler	23
1.2.4	<i>Code Hunt</i>	25
1.2.5	<i>Scratch</i>	26
2	REQUISITOS DO COSMO	28
2.1	Metodologia de desenvolvimento	28
2.2	Captura e Análise de requisitos	29
2.3	Protótipo do Cosmo	33
2.4	Validação do protótipo	34
II	COSMO	36
3	COSMO	37
3.1	Fluxo de funcionamento	37
3.2	<i>FrontEnd</i>	38
3.2.1	<i>Dashboard</i>	38
3.2.2	Área de Resposta	40
3.2.3	Histórico de Atividades	41
3.3	<i>BackEnd</i>	41
3.3.1	Arquitetura	41
3.3.2	Plugins	42
3.3.3	Árvore de Aprendizagem	42
3.4	<i>Plugins no Cosmo</i>	42

3.4.1	Arquitetura de um <i>Plugin</i>	43
3.4.2	Atividade Problema	46
3.4.2.1	Mapper	47
3.4.2.2	Model	49
3.4.3	<i>View</i>	50
III	RESULTADOS	51
4	UM EXPERIMENTO COM A PLATAFORMA COSMO	52
4.1	<i>Log</i> de Dados do Cosmo	52
4.2	Experimento Realizado	52
4.3	Resultados Obtidos com o Experimento	54
4.3.1	Análise Qualitativa Individualizada	54
4.3.1.1	Aluno Exemplar	55
4.3.1.2	Aluno Participativo	56
4.3.1.3	Aluno Ocasional	57
4.3.1.4	Aluno Ausente	57
4.3.1.5	Perfis dos Alunos no Cosmo	57
4.3.2	Análise da Turma	58
4.3.2.1	Como foi feita a Análise?	58
4.3.2.2	Conclusões da análise da turma	59
	Conclusão	61
	REFERÊNCIAS	63
	ANEXOS	65
	ANEXO A – TERMO DE CONSENTIMENTO LIVRE E ESCLA- RECIDO	66
	ANEXO B – QUESTIONÁRIO DE EXPECTATIVA DO COSMO	68
	ANEXO C – RESULTADO DO QUESTIONÁRIO DE EXPECTA- TIVA DO COSMO	70
	ANEXO D – WIREFRAME VERSÃO ATUAL COSMO	74

Introdução

A disciplina de Introdução à Programação tem um papel fundamental nos cursos de graduação na área de informática. É através desta que a maioria dos alunos tem o primeiro contato com programação. Cursos como Sistema de Informação e Ciência da Computação apresentam as maiores taxas de evasão dentre os cursos de educação superior no Brasil (CARVALHO; TAFNER, 2006). Segundo Palmeira e Santos, cursos de Ciência da Computação possuem uma taxa de ingressantes e concluintes de 14,3%. Santos e Costa afirma que um dos motivos para desistência nos cursos é a dificuldade em aprender os conceitos básicos de programação. As dificuldades são inúmeras, desde a entendimento do emprego da lógica até com erros de sintaxe. Estudantes gastam bastante tempo corrigindo erros de sintaxe (KIRAN; MOUDGALYA, 2015), de tal forma que esse conjunto de fatores desmotivam os alunos a buscarem conhecimento.

Alguns fatores que causam a desmotivação no aprendizado de algoritmos é a dificuldade dos alunos em desenvolver raciocínio lógico, em conjunto da dificuldade em compreender as abstrações envolvidas no processo de programação e também a abordagem pedagógica (RAPKIEWICZ et al., 2007). Diante disso, faz-se necessário buscar meios que tornem o ensino de algoritmos mais prático e atrativo para este público.

A utilização de ambientes virtuais de aprendizagem no suporte ao ensino vem ganhando espaço no contexto educacional como ferramenta de apoio, devido a sua eficiência junto aos diferentes perfis de estudantes existentes. No ambiente virtual de aprendizagem (AVA), o processo de ensino-aprendizado pode ser mais dinâmico e personalizado (MEHLECKE; TAROUCO, 2003). Os Ambientes Virtuais de Aprendizagem (AVA) são plataformas que utilizam ferramentas eletrônicas no desenvolvimento de cursos virtuais para disponibilizar conteúdos, permitir interação entre os atores do ambiente educacional e oferecer ao professor a possibilidade de acompanhar e avaliar o aluno no processo de ensino-aprendizagem (ALMEIDA, 2003).

Assim, em um mesmo AVA, podem ser oferecidos aos alunos diferentes mídias e atividades para auxiliá-los na assimilação do conteúdo. Essa possibilidade do aluno aprender sob diferentes formas é bem vista, levando em consideração a existência de diferentes estilos de aprendizagem. Isso também potencializa a oferta de um ensino cada vez mais eficaz, dado que o conteúdo da disciplina passa a ser adaptado ao estilo de aprendizado de cada discente.

Diante disso, uma forma de auxiliar o aprendizado dos alunos é construir um ambiente que o permita conseguir estudar de acordo com suas características e preferências. Assim, é apresentado o Cosmo, que é um ambiente virtual de aprendizado, focado em

atividades voltadas ao estudo da disciplina de Introdução a Algoritmos. O ambiente é uma plataforma multitarefa extensível a *plugins* que disponibiliza ao aluno uma experiência de aprendizado diversificada.

Objetivos

Apresentar a plataforma Cosmo e o seu processo de desenvolvimento. Para tanto são apresentados:

- **Catálogo de ferramentas similares a proposta do Cosmo:** O estudo das ferramentas tem o intuito de expor requisitos e *plugins* que possam ser implementados para o Cosmo.
- **A implementação de um protótipo da plataforma:** O protótipo serve para validar o projeto e coletar novos requisitos a partir de uma pesquisa de uso e satisfação.
- **Experimento controlado com uso de *Logs* do Cosmo:** A análise dos dados obtidos do *Log* da plataforma Cosmo permite detectar aspectos que auxiliem no processo de aprendizagem de alunos em disciplinas de Introdução a Algoritmos.

Organização da Dissertação

Esta dissertação se encontra organizada da seguinte forma:

- Capítulo 1 - **Ferramentas relacionadas:** apresenta um catálogo de ferramentas com enfoque para o ensino de algoritmos. Nele são listados um catálogo de jogos eletrônicos e de plataformas de ensino;
- Capítulo 2 - **Requisitos do Cosmo:** apresenta a metodologia de desenvolvimento utilizada, os requisitos funcionais do Cosmo, o protótipo e sua validação;
- Capítulo 3 - **Cosmo:** apresenta a plataforma Cosmo e suas funcionalidades.
- Capítulo 4 - **Um experimento com a plataforma Cosmo:** apresenta um experimento controlado utilizando o *Log* do Cosmo.
- Capítulo 5 - **Conclusão:** apresenta as considerações finais desta dissertação.

Parte I

Metodologia

1 Ferramentas Relacionadas

Neste capítulo são apresentadas ferramentas que objetivam auxiliar no aprendizado da disciplina de Introdução a Algoritmos. Estas ferramentas estão agrupadas em dois catálogos que especificam sua modalidade de uso. O primeiro catálogo refere-se ao uso de jogos eletrônicos, visto na seção 1.1, e o segundo diz respeito às plataformas de ensino, visto na seção 1.2.

Cada vez mais os usos de jogos eletrônicos têm sido inseridos no processo de ensino aprendizagem a Introdução de Algoritmos. Isso se dá principalmente pelo apelo lúdico apresentado por esta modalidade. Outros fatores favoráveis ao uso de jogos são o aspecto visual que a ferramenta propõe, o empoderamento que alguns estilos de jogos sugerem e o universo virtual atrativo que o aluno interage durante o processo de aprendizagem (RAJARAVIVARMA, 2005).

Segundo Pontes, a utilização de jogos no contexto educacional oferece ao professor uma interessante estratégia para o desenvolvimento da criatividade, aptidão e autonomia do aluno, propiciando a construção do aprendizado em um cenário interdisciplinar. Esses jogos exploram diversos aspectos educativos, como o desenvolvimento de habilidades funcionais, a oferta de atividades sociais, a aquisição de condutas afetivas, a exploração da ludicidade e a aquisição de condutas cognitivas.

Ainda com Pontes, o autor reforça que os jogos em si não podem ser considerados responsáveis por todo o processo de aprendizado do aluno, mas o professor é responsável pelo acompanhamento das atividades e evolução do aluno.

Em uma outra esfera, têm-se as plataformas de ensino com inúmeros aplicativos que abordam o tema aprendizado de Algoritmos de forma bem diversificada.

Exemplificando uma dessas plataformas, observa-se a proposta do *Scratch*, desenvolvido pelo MIT (*Massachusetts Institute of Technology*) que se defende a universalização da programação através das suas funcionalidades. Em seu artigo, *Scratch: programação para todos*, Resnick et al., ressalta a capacidade de programar, assim como fornece benefícios como a expansão de processo criativo, expressivo junto ao computador e melhoria no processo de solução de problemas.

O propósito do desenvolvimento deste catálogo de aplicações é o de permitir observar, através de um processo de mapeamento sistemático das tecnologias, o que existe disponível, para assim elencar requisitos que possam nortear o desenvolvimento do Cosmo.

Algumas das várias propostas existentes são vistas nas próximas seções.

1.1 Catálogo de Jogos Eletrônicos

1.1.1 *LightBot*

Um jogo educativo com temática de robôs, ensina conceitos de lógica de programação através comandos estruturados em blocos. O seu ambiente aborda tópicos introdutórios de lógica, tais como a construção de algoritmo, procedimentos, estrutura condicional e repetição.

O objetivo do jogo é programar o robô através dos blocos e fazer com que ele acenda as luzes durante o percurso. A medida que os níveis são superados, a dificuldade aumenta, novos obstáculos são apresentados no caminho e novos conceitos sobre Algoritmos são introduzidos para o jogador. Sua mecânica está centrada em organizar os blocos de movimentos para que o robô possa se movimentar e acender as luzes durante o caminho.



Figura 1 – O jogo *Lightbot*

A figura 1, apresenta o jogo e a sua interface principal. Na barra inferior são destacados os comandos para manipulação do robô. Na direita tem-se a estrutura que o jogador utiliza para comandar o robô. E na parte superior tem-se os comandos de manipulação do jogo, como exemplo o botão de ação *play* que executa a programação elaborada pelo jogador.

A mecânica do *Lightbot* é amplamente utilizada em disciplinas de Introdução a Algoritmos, como pode ser visto no trabalho de Duarte e Pearce que avalia positivamente a eficácia do jogo em turmas iniciantes em Algoritmos.

1.1.2 The Foos

O The Foos é um jogo similar ao Lightbot, tendo sua diferenciação principal na definição do público alvo. O jogo de classificação livre, destinado a crianças de 6 a 10 anos. Uma das semelhanças com o Lightbot, está associado ao fato de que ambos buscam introduzir a programação de uma forma simples e lúdica.



Figura 2 – O jogo *The Foos*

A figura 2 apresenta a interface principal do jogo. O uso de cores vibrantes e alegres são um ponto forte, tipicamente atrativo e focado para o seu público alvo.

1.1.3 CodeCombat

CodeCombat é uma plataforma para estudantes aprenderem programação através de um jogo real *CodeCombat*¹. Trata-se de um jogo gratuito que também oferece fases pagas.

O jogo possui um estilo de aventura medieval e oferece uma IDE (*Integrated Development Environment*) diretamente no navegador. O ambiente de desenvolvimento oferece suporte ao jogador a 6 tipos de linguagens de programação para implementar seus códigos: Python, Javascript, Lua, CoffeScript, Clojure, e IO (ZANCHETT; VAHLIDICK; RAABE, 2017).

¹ <https://br.codecombat.com/>

O jogo é dividido em 6 mundos. Cada mundo aborda um assunto específico sobre programação. É possível encontrar mundos que abordam assuntos como sintaxe e atribuições, e outros que pautam assuntos mais complexos.

A Figura 3 apresenta a interface do jogo. Neste exemplo, o jogador tem como objetivo levar seu avatar até o outro lado da sala utilizando comandos de programação.

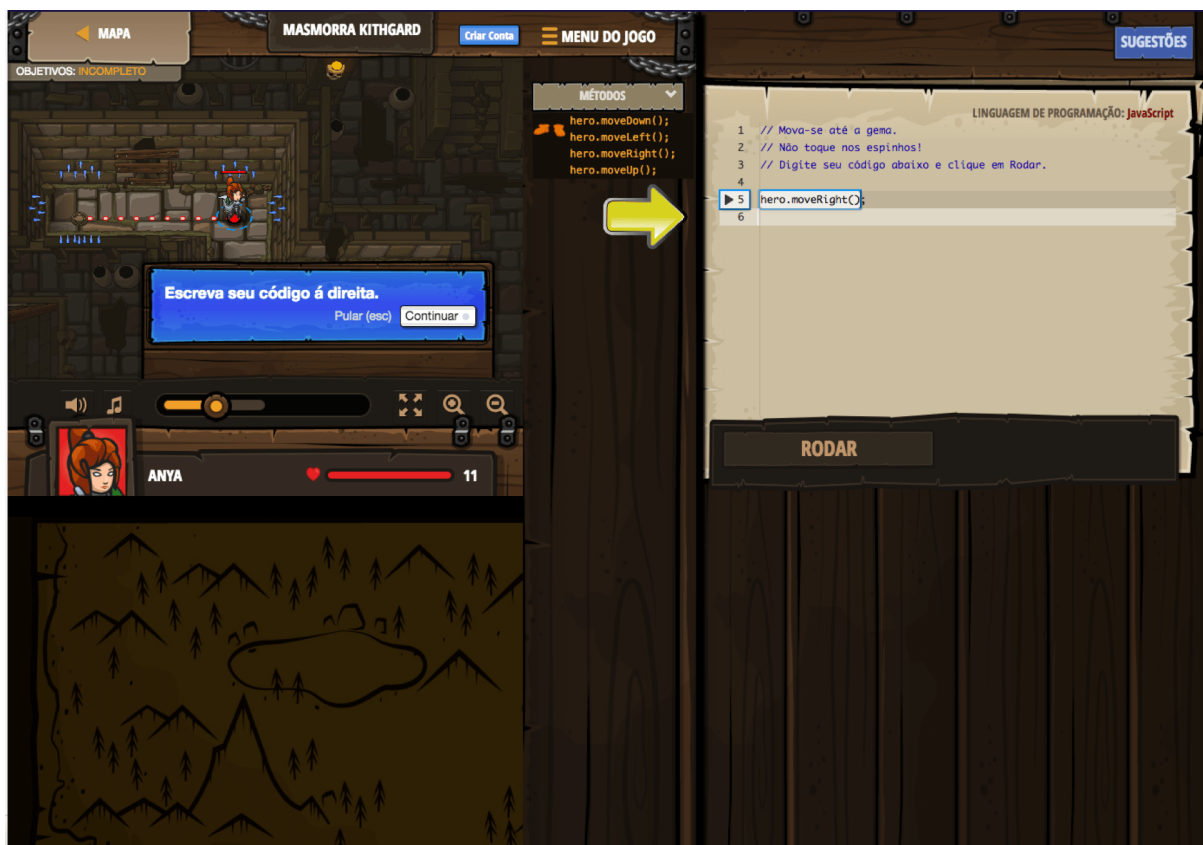


Figura 3 – Interface do Code Combat

1.2 Catálogo de Plataformas de Ensino

1.2.1 Alice

O Alice é um ambiente de programação projetado como uma ferramenta de ensino no qual permite os alunos criarem animações e jogos que utilizam elementos 3D (DANN et al., 2012).

Os alunos aprendem a programar utilizando estruturas pré-definidas a partir de uma interface gráfica que segue o padrão WYSIWYG (*What You See Is What You Get*). Toda nova estrutura adicionada ao projeto, seu resultado poderá ser percebido pelo desenvolvedor pressionando o comando de ação *play*, assim como apresentado na figura 4.

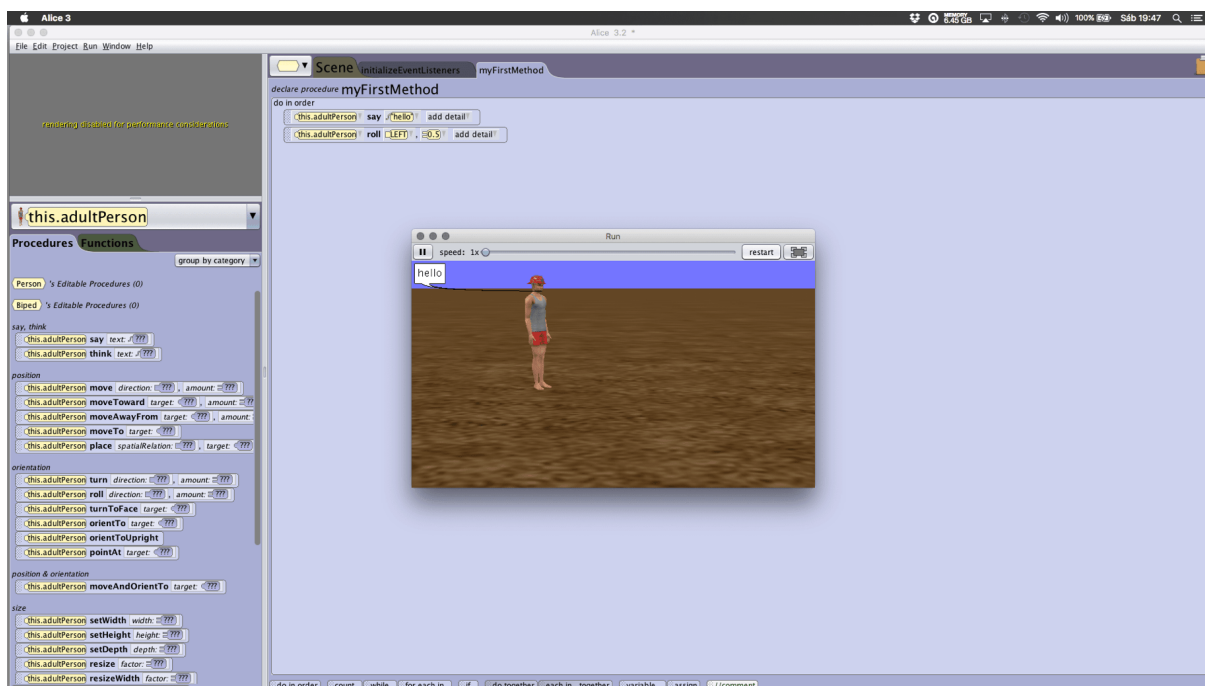


Figura 4 – Projeto Alice

Um projeto no Alice se organiza em uma Cena, no qual a interface do projeto está disposta em quatro subáreas, assim como apresentado logo abaixo na figura 5. (1) A primeira é denominada “área de cenário”, onde o aluno poderá visualizar o mundo 3D e adicionar recursos (personagens, itens de cenário, câmera, dentre outros). (2) A segunda área, chamado de “seletor de recursos”, facilita a manipulação da estrutura adicionada à área de cenários. (3) A terceira é chamada de “agrupamento de procedimentos e funções”, onde são apresentados os comportamentos de cada objeto selecionado. (4) A quarta e última corresponde à área de gerenciamento da cena, e é nela que o aluno elabora os métodos que irão manipular os objetos de cena e compor, por exemplo, uma animação. Tudo isso através do uso dos elementos em blocos.

Os blocos se encaixam, formando assim uma estrutura semântica da aplicação. Cada bloco possui um comportamento específico a um objeto, que pode ser adicionado com uso de um clique e arrastar. A sua execução é Top Down, onde a execução de interpretação dos blocos é feita de cima para baixo. Cada bloco pode ser parametrizado e conectado a funções.

Qualquer projeto no Alice é escrito na linguagem de programação Java. O Alice oferece uma opção de visualização para o código gerado. Essa opção permite que o aluno compreenda ou faça um comparativo de tudo o que foi definido em projeto, traduzido para linguagem Java. Outra forma de visualizar o código gerado é através do uso do ambiente de desenvolvimento integrado, o NetBeans ². O Alice oferece um *plugin* de integração

² <https://netbeans.org/>

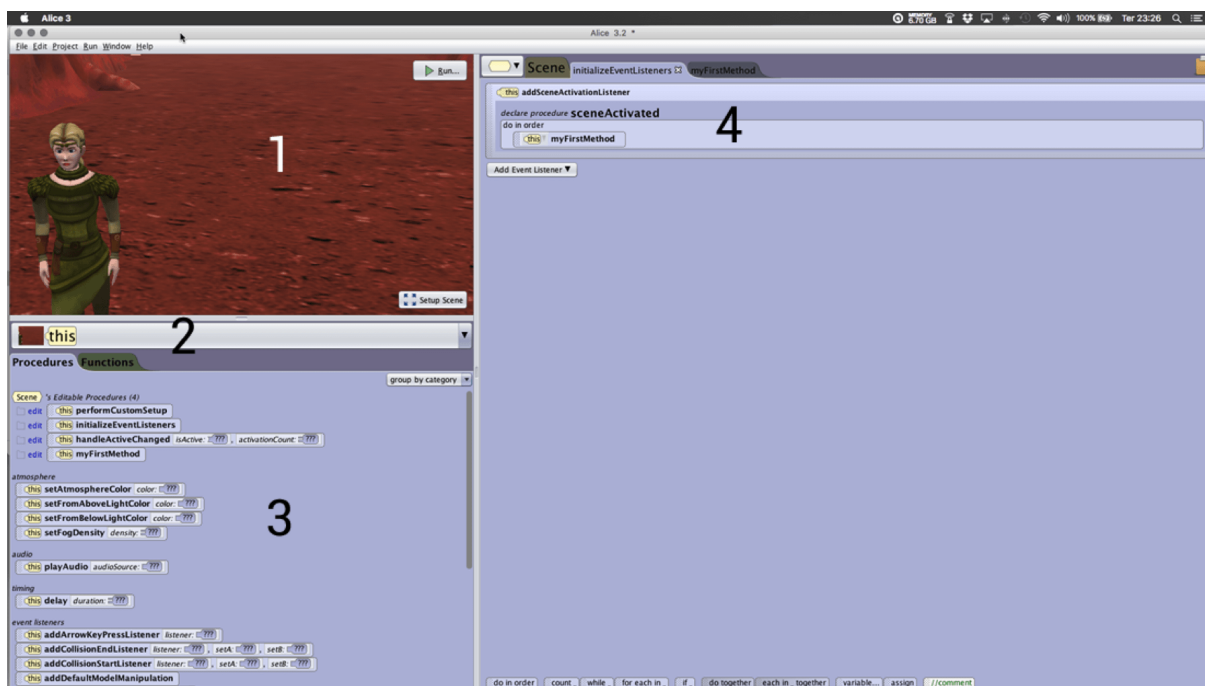


Figura 5 – Organização da interface do Alice

junto a essa IDE (Integrated Development Environment).

1.2.2 Codecademy

O Codecademy é uma plataforma de ensino com foco em diversas áreas de ensino de tecnologia. Os temas disponibilizados pela plataforma são de tendência de mercado com foco profissional.

O aluno consegue aprender de forma amigável, através de exemplos, vídeos, textos autoexplicativos e gamificação.

A empresa mantenedora da aplicação garante ser comprometida em construir a melhor experiência de ensino, dentro de sua plataforma on-line.

Outros recursos encontrados na plataforma são: IDE online, *highlight* e submissão de código, compartilhamento de dúvidas, entre outros recursos.

A figura 6 abaixo, apresenta um curso em andamento. No canto esquerdo temos o conteúdo de aprendizagem, no centro podemos perceber a prática do conteúdo em estudo e no canto superior em branco, temos o resultado em tempo de execução.

1.2.3 Projeto Euler

Uma série de problemas matemáticos e computacionais disponíveis para que a solução seja elaborada e submetida, esse é o Projeto Euler. Sua mecânica é objetiva,

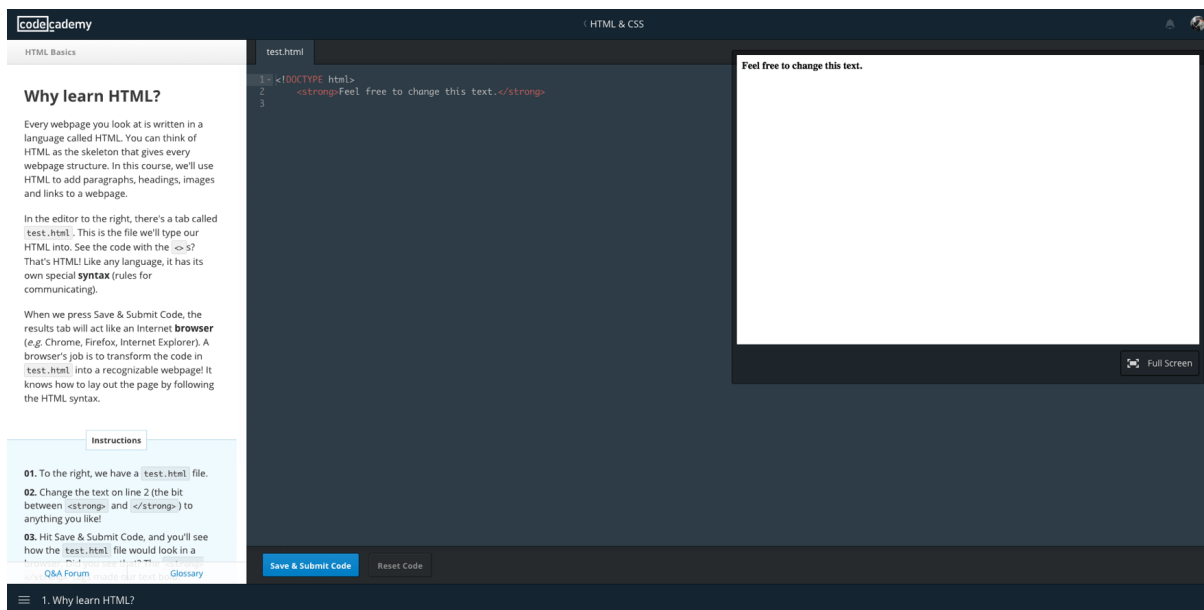


Figura 6 – Plataforma CodeAcademy

o usuário escolhe entre os 630 problemas disponíveis, lê seu enunciado e submete uma resposta. Um retorno imediato é informado ao usuário sobre a sua resposta submetida.

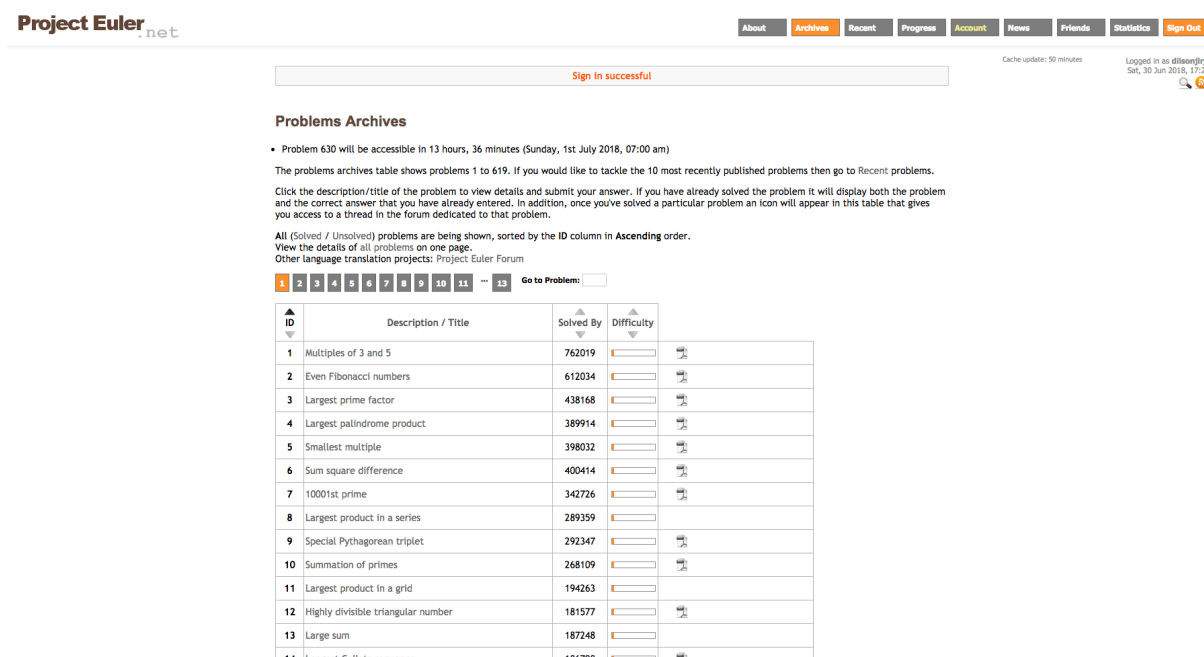


Figura 7 – Projeto Euler - Área de Questões

A figura 7 apresenta a página do projeto, onde é possível observar a lista de problemas disponíveis. Cada problema apresenta uma descrição, a quantidade de vezes que aquela questão foi resolvida e o seu grau de dificuldade. Além disso, a plataforma conta com um módulo estatístico para coleta de informações, como: quantidade de problemas resolvidos por países, quais linguagens de programação estão sendo utilizadas para resolução

dos problemas, dentre outras estatísticas. A área de resposta do projeto Euler é intuitiva. Ao selecionar um problema, o usuário recebe um enunciado, definindo o problema a ser resolvido, e um campo para informar a resposta. A área de resposta do projeto Euler pode ser vista na figura 8.

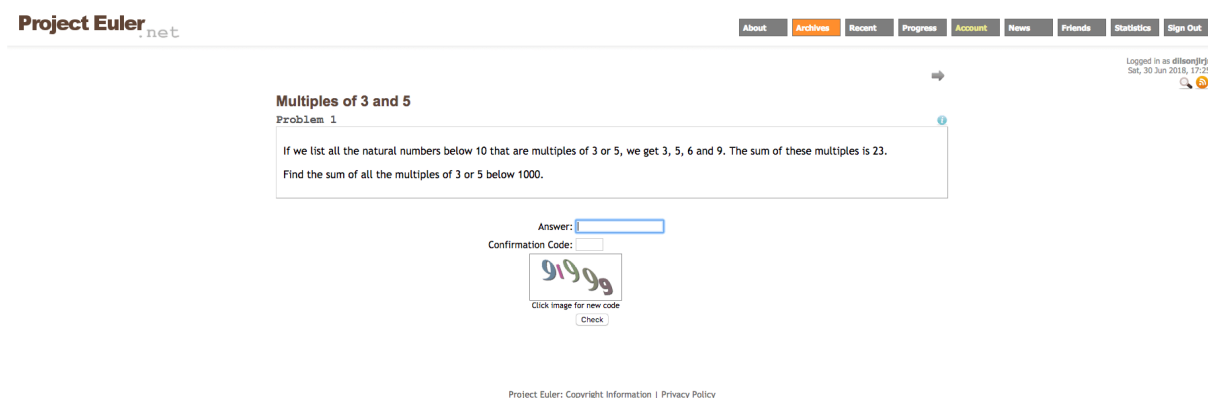


Figura 8 – Projeto Euler - Área de Resposta

1.2.4 Code Hunt

O *Code Hunt* é um jogo educacional estilo caça código. O jogador, ou caçador de código, busca fragmentos que faltam para completar, ou desenvolve uma solução mais elegante para um código desafio.

O jogador tem possibilidade de escrever códigos em C# ou Java, o que exige dele um conhecimento básico ou mediano em uma das linguagens.

O fluxo de execução do jogo se inicia quando o jogador se inscreve no site através de autenticadores como o Google³, Yahoo⁴, Facebook⁵ ou Microsoft.

Em seguida o jogador é encaminhado para o processo de seleção da linguagem. Após a seleção, uma lista de fases é apresentada, porém fica disponível somente a fase de treinamento. Como regra, o jogador só poderá seguir para o próximo desafio quando concluir 90% do desafio anterior. A figura 9 apresenta a lista de fases disponíveis para o jogador.

³ <http://www.google.com>

⁴ <http://www.yahoo.com>

⁵ <http://www.facebook.com>



Figura 9 – Code Hunt - Seleção de fases

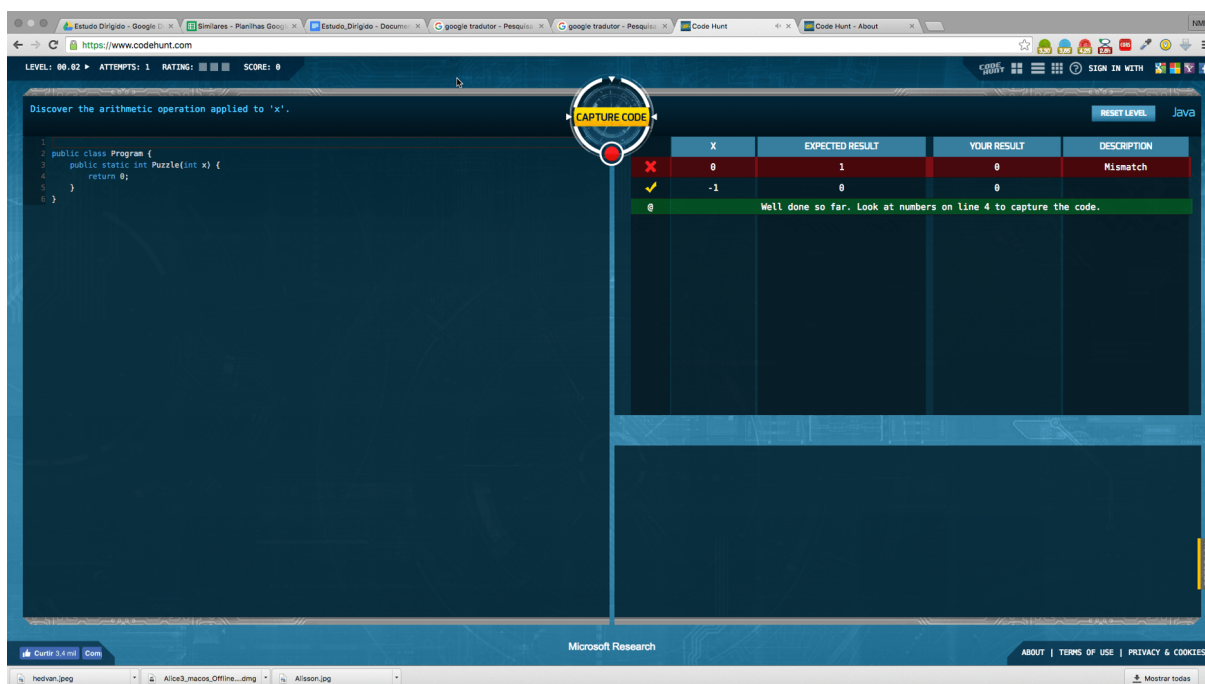
Cada fase aborda tópicos específicos sobre lógica de programação, como exemplo: estruturas de repetição, vetores, ordenação, entre outros. Ao selecionar um tema, o jogador é desafiado com um problema, apresentado à sua esquerda e uma bateria de testes de aceitação à sua direita, assim como apresentado na figura 10. Ao preencher o fragmento de código que falta, para satisfazer a bateria de testes, o jogador clica no botão "*capture code*", para que o código digitado seja analisado e um retorno seja apresentado. A cada novo desafio, a complexidade de jogo aumenta.

1.2.5 Scratch

O *Scratch* é uma linguagem de programação desenvolvida pelo MediaLab no MIT (*Massachusetts Institute of Technology*). É uma linguagem simples que não exige conhecimento prévio e aprofundado de programação, o que a torna ideal para públicos iniciantes. Como exemplo, os mantenedores a utilizam com crianças a partir de 8 anos para introduzir conceitos matemáticos e computacionais.

Criada em 2007, teve sua inspiração nos blocos de montar da empresa Lego.

[Resnick et al.](#) em seu artigo "*Scratch: programming for all*", cita que enquanto brincam e constroem no Scratch, os planos e objetivos evoluem organicamente, juntamente com as estruturas e histórias. A equipe queria que o processo de programação no Scratch tivesse uma sensação semelhante. A gramática *Scratch* é baseada em uma coleção de blocos gráficos de "programação" que as crianças juntam para criar programas. Assim como os tijolos de Lego, os conectores nos blocos sugerem como eles devem ser colocados juntos. As

Figura 10 – *Code Hunt* - Caçando Bugs

crianças podem começar simplesmente mexendo com os tijolos, juntando-os em diferentes sequências e combinações para ver o que acontece. Não há nada da sintaxe ou pontuação obscura das linguagens de programação tradicionais.

Com essa proposta o *Scratch* proporciona uma quebra de paradigma na forma de desenvolver aplicações e de propagar o conhecimento sobre programação. Essa afirmativa vai de encontro com os princípios de designs encontrados na ferramenta e definidos por seus autores. Existem três princípios básicos de design para o *Scratch*: torná-lo mais adaptável, mais significativo e mais social do que outros ambientes de programação (RESNICK et al., 2009).

A figura 11 apresenta um trecho código desenvolvido no *Scratch*.

Figura 11 – Exemplo de um aplicativo desenvolvido no *Scratch*

2 Requisitos do Cosmo

Este capítulo tem como objetivo apresentar o processo de captura, análise de requisitos e modelagem do protótipo do Cosmo. O capítulo apresenta também a aplicação de uma pesquisa para validação do protótipo e captura de novos requisitos.

2.1 Metodologia de desenvolvimento

O Cosmo foi implementado sob o uso da metodologia ágil, com auxílio do *framework Scrum*. *Scrum* é um framework Ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos. *Scrum* é embasado no empirismo e utiliza uma abordagem iterativa e incremental para entregar valor com frequência e, assim, reduzir os riscos do projeto (SABBAGH, 2014).

A adoção de tal metodologia foi escolhida pelo fato de apoiar o uso de uma documentação sucinta, transparência no processo de desenvolvimento e velocidade nas entregas de requisitos definidos pelo usuário. Todo processo de gerenciamentos das atividades foi acompanhado via Trello. O Trello é um organizador de atividades, totalmente colaborativo, onde é possível gerenciar projetos de pequena e larga escala (TRELLO, 2018).

Todos os seus requisitos foram documentados com uso de histórias de usuários. Uma história de usuário é uma especificação de uma ou mais sentenças na linguagem de negócio ou cotidiana do usuário final ou usuário do sistema que captura o que um usuário faz ou necessita fazer como parte de sua função de trabalho (BECK; FOWLER, 2001).

Uma história de usuário possui um formato de escrita que descreve: a quem a história se destina; a descrição da funcionalidade, aqui é possível descreve a funcionalidade sob a ótica do usuário levando em consideração como a funcionalidade deverá se comportar nas mãos do executor; e o motivo da construção daquela funcionalidade.

Uma história de usuário pode seguir o seguinte formato:

- COMO/SENDO <QUEM>, EU QUERO/GOSTARIA/DEVO/POSSO <O QUE>, PARA QUE/DE/PARA <PORQUE/RESULTADO>

Para cada requisito funcional levantado são detalhadas em formato de histórias de usuário.

2.2 Captura e Análise de requisitos

Nesta seção são apresentados os requisitos do Cosmo. É também apresentado como foi feita esta captura e análise de requisitos. Os requisitos funcionais de um sistema descrevem o que o sistema deve fazer. Esses requisitos dependem do tipo do software que está sendo desenvolvido, dos usuários a que o software se destina e da abordagem geral considerada pela organização ao redigir os requisitos (SOMMERVILLE, 2011).

A captura de requisitos para o Cosmo se deu através de uma entrevista com o professor idealizador do projeto e análise aprofundada das ferramentas citadas na Seção 1.

A partir da entrevista com o professor foi possível obter a concepção geral do Cosmo. Para tal *stakeholder*, o Cosmo deveria ser um plataforma onde o aluno pudesse conhecer os princípios da Introdução a Algoritmos, através de atividades que fosse expostas pela plataforma. Essas atividades poderiam ser como exemplo: a resolução de problemas computacionais; a apresentação de funções com erros de código para o aluno corrigir; permitir ao aluno assistir a um vídeo e ao final responder questões sobre o conteúdo; etc. Como resultado da entrevista foi possível identificar alguns requisitos que podem ser visualizados na Tabela 1.

Tabela 1 – Requisitos Gerais do Cosmo - Visão do Professor

Número	Descrição
RF01	Manter <i>Plugins</i> , o Cosmo deve ser uma plataforma extensível a <i>plugins</i> . Já que as atividades que o Cosmo irá receber são variadas;
RF02	Manter <i>Árvore de Aprendizagem</i> , deve existir uma área onde o professor crie o conhecimento a ser apresentado ao aluno;
RF03	Manter <i>Turma</i> , o Cosmo deve oferecer suporte a turmas;

Com os requisitos iniciais expostos pelo professor, alguns outros requisitos foram modelados a partir do catálogo de aplicações definido na Seção 1.

Os aplicativos catalogados foram classificados em duas modalidades: aplicativos que podem ser utilizados considerados *plugins* para o Cosmo; e aplicativos que a partir do uso de *benchmarking*, seja possível a extração de novos requisitos.

Para Spendolini, o *benchmarking* pode ser definido como “um processo contínuo e sistemático para avaliar produtos, serviços e processo de trabalho de organizações que são reconhecidas como representantes das melhores práticas, com a finalidade de melhoria organizacional”. A Tabela 2 representa os aplicativos categorizados a partir dessa classificação.

A análise do *benchmarking*, a partir das três ferramentas, permitiu a extração de novos requisitos e a consolidação dos requisitos já existentes. Nessa análise é possível

Tabela 2 – Catálogo de ferramentas - *Benchmarking* x *Plugins*

Benchmarking	Plugins
CodeAcademy	Lightbot
Projeto Euler	The Foos
Code Hunt	Code Combat
	Alice
	Scracth

perceber que os requisitos existentes agregam boas funcionalidades ao Cosmo. O quadro final com a visão geral dos requisitos funcionais pode ser observado na Tabela 3.

Tabela 3 – Visão Geral dos Requisitos Funcionais

Número	Descrição
RF01	Manter <i>Plugins</i> , O Cosmo deve ser uma plataforma extensível a <i>plugins</i> ;
RF02	Manter Árvore de Aprendizagem, Deve existe uma área onde o professor crie o conhecimento a ser apresentado ao aluno;
RF03	Manter Turma, o Cosmo deve oferecer suporte a turmas;
RF04	Manter Usuário, o Cosmo deve permitir que o usuário se inscreva na plataforma;
RF05	Escolher Atividade, o aluno pode escolher uma atividade para responder;
RF06	Escolher Atividade, o aluno pode escolher uma atividade para responder;

Para cada uma dos requisitos funcionais levantados, estes serão extrapolados e convertidos em Histórias de Usuários. Na análise das histórias ficaram definidos três atores que irão manipular diretamente o uso da plataforma: o Aluno, o Professor e o Administrador. Fica estabelecido que o administrador do sistema tem acesso a todas as funcionalidades especificadas como requisitos.

A Tabela 4 apresenta a lista das histórias que compõem o RF01, relacionado a manter *plugins*.

Tabela 4 – Lista de histórias que compõem o RF01.

Número	História
HU01	Como administrador do sistema gostaria de instalar um <i>plugin</i> no Cosmo para poder utilizá-lo na modelagem do conhecimento na árvore de aprendizado e também como atividade para os alunos.
HU02	Como administrador gostaria de desinstalar o <i>plugin</i> do Cosmo.

A Tabela 5 apresenta a lista das histórias que compõem o RF02, o manter árvore de aprendizagem.

Tabela 5 – Lista de histórias que compõem o RF02.

Número	História
HU01	Como professor, gostaria de dar entrada em uma área de conhecimento, afim de vincular atividades.
HU02	Como professor, gostaria de vincular uma atividade a uma área de conhecimento, para que possa ser utilizado pelo alunos.
HU03	Como professor, gostaria de tornar ativa uma área de conhecimento, para que possa se tornar disponível ao alunos.
HU04	Como professor, gostaria de vincular a uma turma uma área de conhecimento.
HU05	Como professor, gostaria de excluir uma área de conhecimento.
HU06	Como professor, ao excluir uma área de conhecimento, gostaria que fosse mantido todo o histórico já efetuado por ela.
HU07	Como administrador, gostaria que existisse uma área onde pudesse administrar o conteúdo da árvore de aprendizagem, para pode executar algumas ações emergenciais.

A Tabela 6 apresenta a lista das histórias que compõem o RF03, o manter turma.

Tabela 6 – Lista de histórias que compõem o RF03.

Número	História
HU01	Como professor, gostaria de poder criar uma turma, para pode alocar alunos.
HU02	Como professor, gostaria de poder alocar alunos a uma turma.
HU03	Como professor, gostaria que quando um aluno fosse alocado fosse enviado um e-mail ao discente.
HU04	Como professor, gostaria de vincular a turma um conhecimento definido na árvore de aprendizagem, para que os alunos possam responder as atividades.
HU05	Como professor, gostaria de excluir uma turma.
HU06	Como administrador, gostaria que existisse uma área onde pudesse administrar as turmas, para pode executar algumas ações emergenciais.

A Tabela 7 apresenta a lista das histórias que compõem o RF04, o manter usuário.

A Tabela 8 apresenta a lista das histórias que compõem o RF05, o escolher tarefa.

Tabela 7 – Lista de histórias que compõem o RF04.

Número	História
HU01	Como Aluno, gostaria de me inscrever na plataforma Cosmo, para poder realizar minhas atividades.
HU02	Como professor, gostaria de me inscrever na plataforma Cosmo, para poder criar atividades, gerar turma e disponibilizar isso aos meus alunos.
HU03	Como administrador, gostaria que o processo de inscrição também pudesse ser feito a partir de redes sociais, para facilitar o processo de cadastro.
HU04	Como administrador, gostaria que existisse uma área onde pudesse administrar o usuários do sistema, para pode executar algumas ações emergenciais.
HU05	Como administrador, gostaria de poder manter um usuário a partir da área de administração.

Tabela 8 – Lista de histórias que compõem o RF05.

Número	História
HU01	Como Aluno, gostaria de ter uma área onde eu pudesse visualizar as tarefas do Cosmo, para poder selecionar e responder a qual desejar.
HU02	Como Aluno, gostaria de poder selecionar uma atividade, para poder respondê-la.
HU03	Como Aluno, gostaria de poder pular uma atividade, quando achar o difícil a atividade.
HU04	Como Aluno, gostaria de visualizar meu histórico de atividade de realizadas, para poder apresentar a meu professor.

A Tabela 8 apresenta a lista das histórias que compõem o RF06, o responder tarefa.

Tabela 9 – Lista de histórias que compõem o RF06.

Número	História
HU01	Como Administrador, gostaria de que quando o Aluno ao selecionar uma atividade o <i>plugin</i> fosse iniciado e a atividade apresentada, para o aluno responder sua atividade.

Todas as histórias criadas deram condições para a criação de um protótipo inicial da plataforma Cosmo.

2.3 Protótipo do Cosmo

A partir dos requisitos funcionais listados, foi possível desenvolver a primeira versão do Cosmo. Nesta versão foi desenvolvido o MVP (*Minimum Viable Product*), o Mínimo Produto Viável do Cosmo. O MVP é aquela versão do produto que permite a obtenção do produto, com o mínimo de esforço e o menor tempo de desenvolvimento (RIES, 2012).

A importância do MVP do Cosmo é garantir a validação da proposta da ferramenta junto ao usuário. Essa prática, auxilia na captura de novos requisitos, mapeamento de erros e também fluxo de regras de negócio.

Nessa versão continha: a área de *Dashboard*, local onde os alunos visualizam suas atividades; a Área de Respostas, onde o aluno responde suas atividades; e a área de Histórico de Atividades, local onde o aluno visualiza seu histórico de atividades já respondidas. Na Figura 12 é possível visualizar a versão utilizada como MVP do Cosmo.

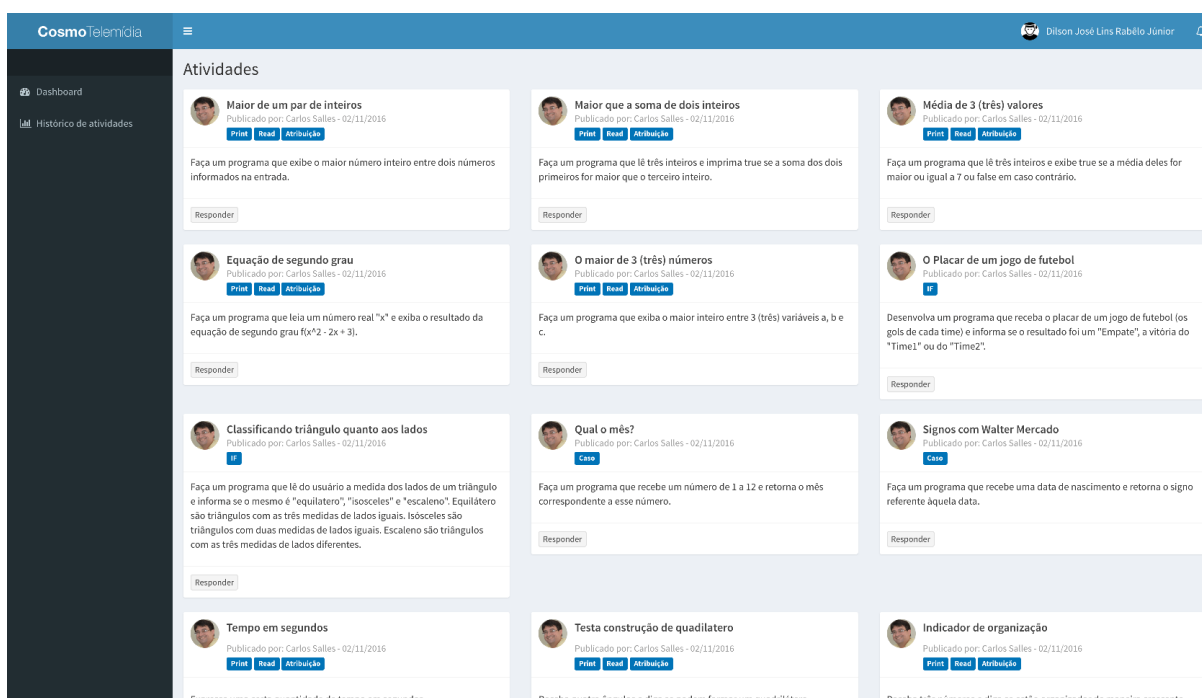


Figura 12 – Versão de MVP do Cosmo

Esta versão do Cosmo tem um *plugin* de atividade chamado de Problema, nele o aluno recebe um problema computacional e como desafio o aluno precisa desenvolver uma solução através de uma linguagem de programação. Para o *plugin* em questão, o Cosmo oferece suporte à linguagem Lua.

Todo protótipo foi construído com bases nos recursos que podem ser vistos na Seção 22, no capítulo 3.

2.4 Validação do protótipo

A validação do MVP do Cosmo se deu com uma turma de iniciantes no curso de Ciência da Computação na disciplina de Introdução a Algoritmos. Os alunos da disciplina foram convidados a utilizar a plataforma Cosmo e a responder um questionário de expectativas de uso.

O tempo de duração do teste era o equivalente a 1 hora e meia, onde de um total de 34 alunos, 8 responderam os questionários e, a partir dos resultados, algumas melhorias foram elencadas para a evolução do sistema.

Sobre os resultados coletados, é possível destacar que mais de 70% estão satisfeitos com o uso do Cosmo. A Figura 13 apresenta o gráfico da pergunta efetuada.

Com base na sua experiência até agora, você está satisfeito com o aplicação Cosmo:

8 respostas

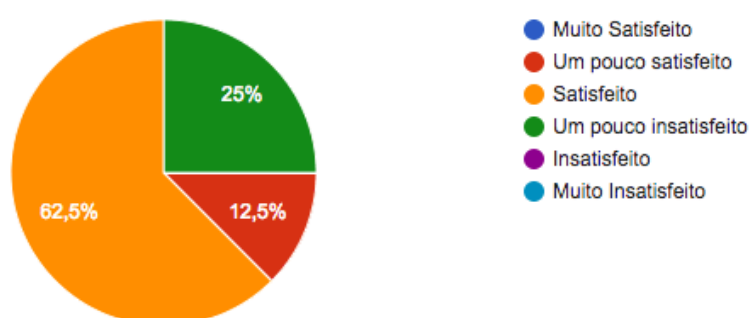


Figura 13 – Pesquisa de Satisfação - Uso do Cosmo

Outro aspecto interessante da pesquisa foi o de observar como o aluno vê o Cosmo. Cerca de 75% visualiza o Cosmo como uma área de treinamento, um local para fixar o conteúdo aprendido em sala de aula. A Figura 14, apresenta o resultado da pesquisa para esta questão.

Além desses aspectos já apresentados, outros requisitos foram elencados na pesquisa e que serão adotados em versões futuras da aplicação. Dentre os novos requisitos levantados podem ser destacados: uso de dicas para que as atividades se tornem mais interativas, espaço para tirar dúvidas com o professor, gamificação, indicação dos erros ao submeter o código-fonte na atividade Problema, dentre outros.

A partir dos requisitos coletados na questionário de validação foi possível se modelar uma nova *interface* para o Cosmo. Essa interface contém um novo modelo de *Dashboard*, Histórico e Área de Resposta. A nova proposta limitou o número de atividades na tela e

Como você considera o Cosmo?

8 respostas

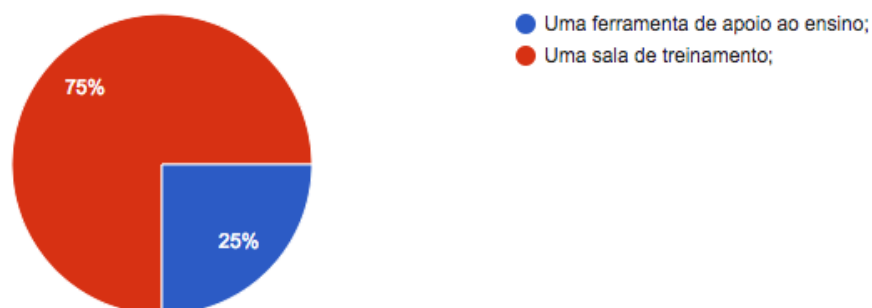


Figura 14 – Como os alunos visualizam o Cosmo

disponibilizou a opção de pular definida inicialmente na lista de requisitos. Na Figura 15, apresenta a nova interface do Cosmo.

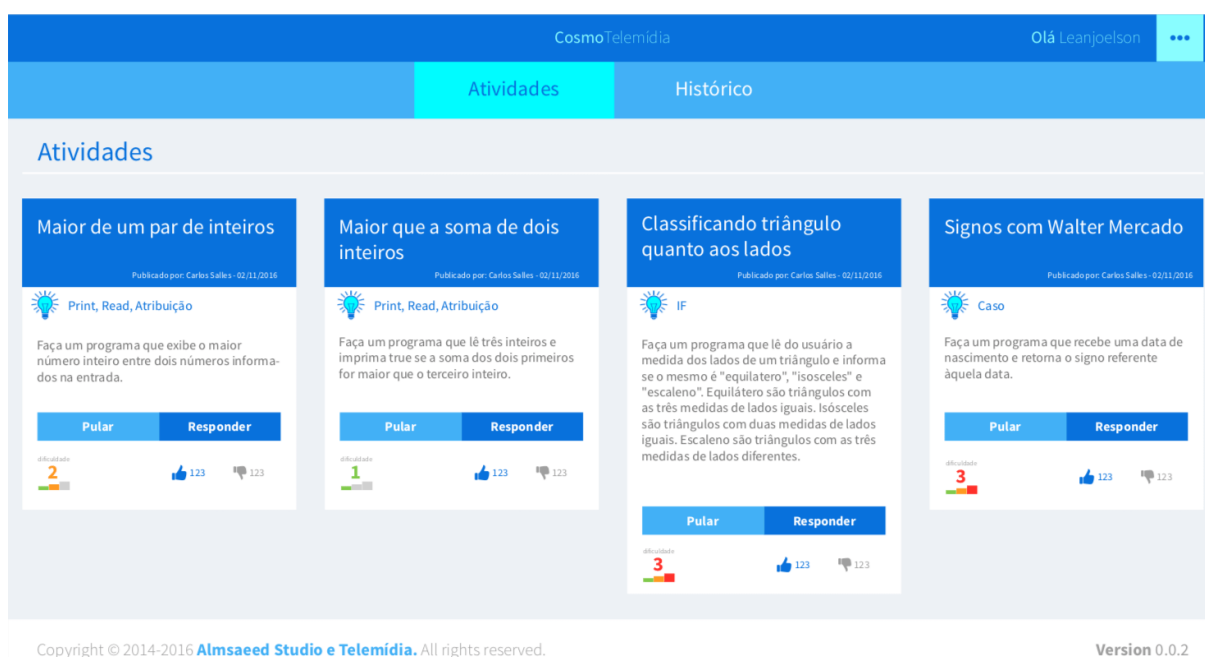


Figura 15 – Nova interface do Cosmo área de Dashboard

Parte II

Cosmo

3 Cosmo

O Cosmo é uma plataforma de ensino, multitarefa focado em atividades voltadas ao estudo de uma disciplina de Introdução a Algoritmos. Este capítulo objetiva-se apresentar a estrutura do Cosmo e suas funcionalidades. Sua arquitetura é extensível por *plugins*, os quais permitem acoplar ao ambiente diferentes tipos de atividades que possam ser criadas por professores no ensino de programação. Exemplos de tais atividades são: a resolução de problemas computacionais; a apresentação de funções com erros de código para o aluno corrigir; permitir ao aluno assistir a um vídeo e ao final responder questões sobre o conteúdo; etc.

O aluno interage basicamente com três áreas: o *Dashboard* de Atividades, área que apresenta as atividades a serem trabalhadas pelo aluno; Área de Resposta, local onde as atividades são respondidas; e o Histórico de Atividades, nessa área o aluno pode visualizar todas as suas atividades já respondidas. Sua estrutura é extensível a plugins, no qual permite que seja possível acoplar ao ambiente qualquer tipo de atividade que ofereça suporte ao HTML (*HyperText Markup Language*).

Todo conhecimento modelado pelo professor na ferramenta é mantido em uma Árvore de aprendizagem. Nela é especificada a trilha de conhecimento que o aluno irá conhecer, os assuntos que serão abordados e oferecidos como atividades pelo Cosmo.

3.1 Fluxo de funcionamento

Nesta subseção é apresentado o fluxo de funcionamento do Cosmo. A Figura 16, emblema o fluxo funcional da plataforma.

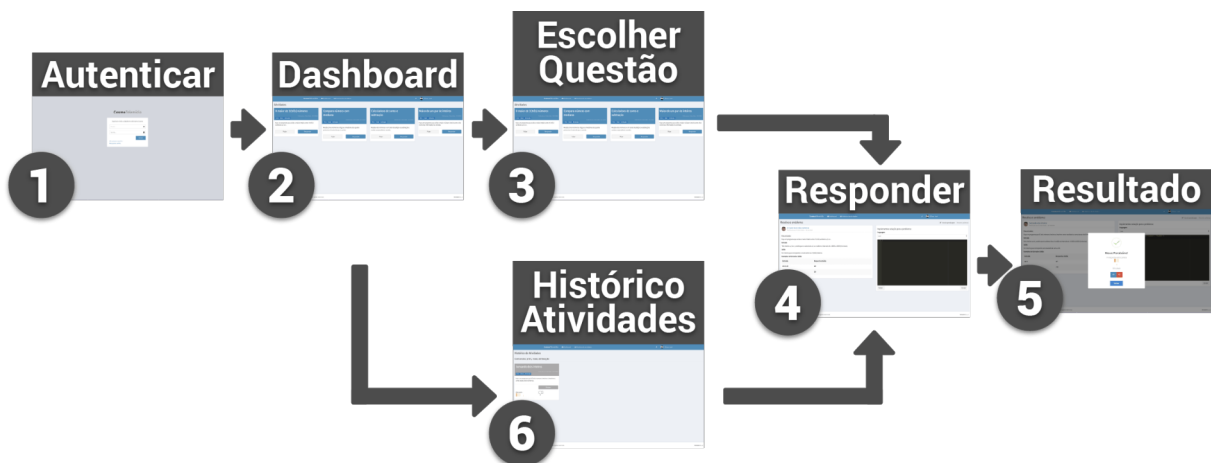


Figura 16 – Fluxo de funcionamento do Cosmo

Como pode ser observado no primeiro passo do fluxo, o aluno se autentica na plataforma. Caso já não faça parte da plataforma, existe a opção para criar o usuário ou recuperar senha.

Ao acessar o ambiente, o aluno é levado à área de *Dashboard* onde são listadas as atividades.

Nesse momento ele têm duas opções: escolher Responder uma Atividade ou acessar a área de Histórico de Atividades.

Caso o aluno clique em responder, este será levado à Área de Resposta e em caso de sucesso ele receberá um retorno positivo sobre a resposta de sua questão, caso contrário um retorno negativo será apresentado ao aluno.

Se o aluno optar em acessar a área de Histórico de Atividades, uma lista de atividades já respondidas pelo aluno é apresentada. Ao clicar em responder, o aluno segue para Área de Resposta e o fluxo segue normalmente.

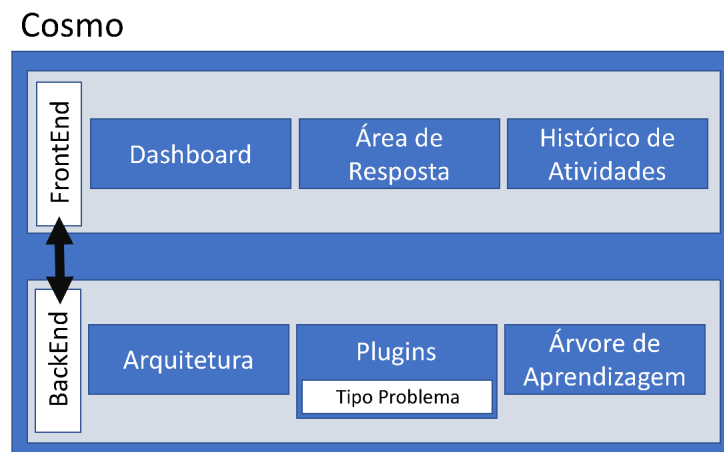


Figura 17 – Esquema do Cosmo

Assim como listado na Figura 17, o Cosmo pode ser apresentado descrevendo os itens da figura. O detalhamento de cada uma dessas funcionalidades são apresentadas nas subseções a seguir.

3.2 *FrontEnd*

Nesta seção são apresentados os elementos que compõe a camada de *FrontEnd* listado na Figura 17.

3.2.1 *Dashboard*

A área de *Dashboard* é o local onde são apresentadas as atividades que são respondidas pelo aluno.

São listadas 4 atividades, de forma aleatória sobre assuntos definidos na árvore de aprendizagem. A interface foi organizada em 5 pontos, conforme pode ser observado na Figura 18. Detalhando cada item pode-se observar:

- O item 1 refere-se às caixas de atividades. Os cards de atividades serão detalhados mais à frente;
- o item 2 apresenta a ação de pular atividade. O aluno pode pular uma atividade caso não seja de seu interesse responde-la. Nesse caso, o Cosmo encaminha uma nova atividade. Na versão atual do Cosmo, não existe limite para executar a ação de pular;
- No item 3 o Perfil do usuário, com as configurações do usuário e a ação de sair;
- O item 4 a Ação de responder, esta ação leva o aluno à Área de Resposta;
- E por fim o item 5, apresenta o menu do usuário, com os *links* para o Dashboard e Histórico de Atividade.

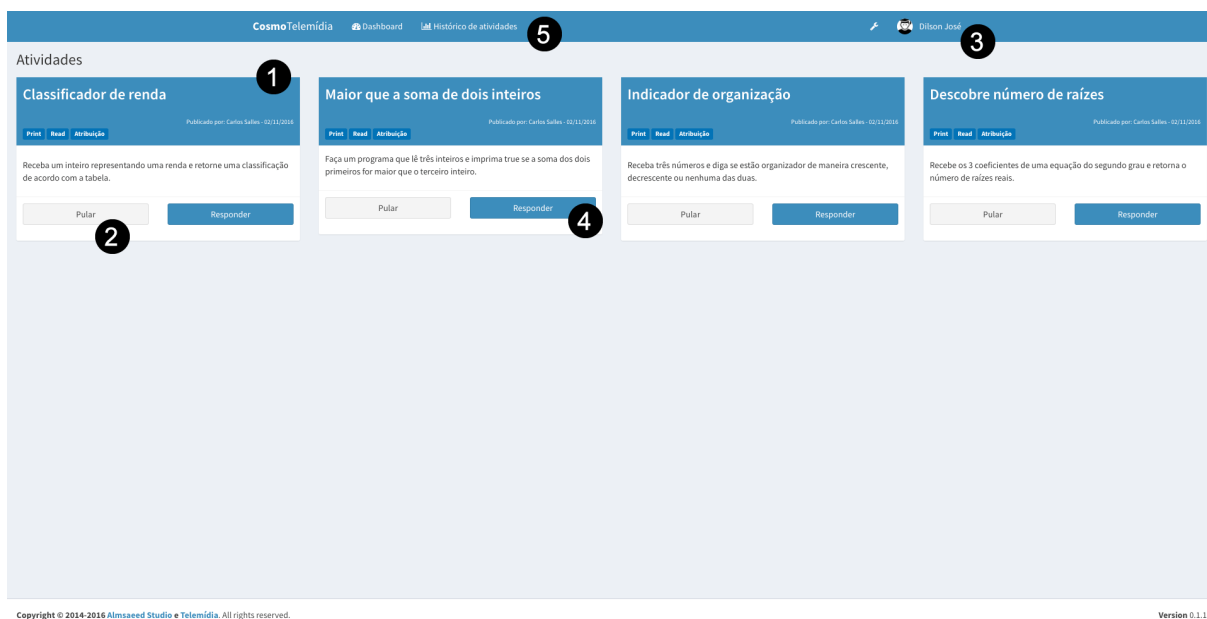


Figura 18 – *Dashboard* do Cosmo

A composição das atividades no *Dashboard* se dá de forma aleatória. Cada caixa de atividade possui um título, um breve conceito para expor previamente o tipo de atividade, quem a criou e *tags*, as quais indicam ao usuário de qual assunto a questão está abordando. Pode-se observar mais detalhadamente uma caixa de atividade na Figura 19.

Checa strings iguais

Publicado por: Carlos Salles - 02/11/2016

Print Read Atribuição

Faça um programa que lê duas strings e retorna True, se forem iguais, ou False se forem diferentes.

Pular Responder

Figura 19 – Caixa de Atividade

3.2.2 Área de Resposta

Ao clicar no botão "Responder", em uma caixa de atividade, o usuário é levado a uma tela chamada de Área de resposta, conforme apresentado na Figura 20. O conteúdo que o usuário visualiza nessa tela é dinâmico. As informações apresentadas nessa tela são originadas de um *plugin* que está sendo instanciado no momento. Mais detalhes sobre a organização dos *plugins* no Cosmo, são apresentados na seção 3.3.2. Na Figura 20, vemos o item 1 que corresponde o enunciado da questão e o item 2 que apresenta a forma como o *plugin* aceita a entrada de resposta para a atividade. Neste exemplo pode ser observado o *plugin* "Problema", mais detalhes sobre o *plugin* será visto na seção 3.4.2.

CosmoTelemidia Dashboard Histórico de atividades

Área de aprendizagem - Resolver problemas

Resolva o problema

1

Maior que a soma de dois inteiros
Publicado por: Carlos Salles - 02/11/2016

Enunciado:
Faça um programa que lê três inteiros e imprima true se a soma dos dois primeiros for maior que o terceiro inteiro.

Entrada
Três inteiros a, b e c, sendo que os valores de a, b e c estão no intervalo de -10000 a 10000 (inclusive).

Saída
Um boolean, true ou false.

Exemplos de Entrada e Saída

Entrada	Respectiva Saída
100 300 20	true
10 10 100	false

2

Implemente a solução para o problema:

Linguagem
C/C++

Voltar Enviar

Copyright © 2014-2016 Almsaeed Studio e Telemidia. All rights reserved. Version 0.1.1

Figura 20 – Área de Resposta

3.2.3 Histórico de Atividades

O Histórico de atividades possibilita o aluno a responder novamente uma atividade já respondida.

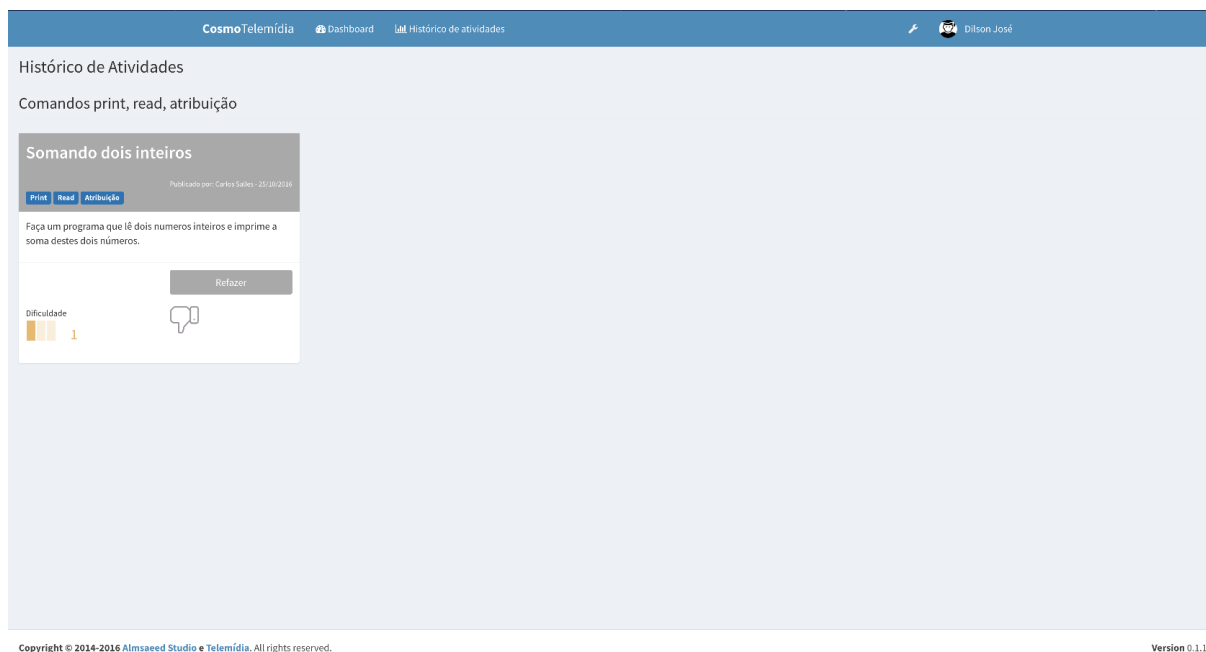


Figura 21 – Histórico de Atividade

Conforme apresentado na Figura 21, o aluno, ao clicar em refazer, será levado à Área de Resposta onde poderá refazer a questão selecionada.

3.3 *BackEnd*

Nesta seção são apresentados os elementos que compõe a camada de *BackEnd* listado na Figura 17.

3.3.1 Arquitetura

A arquitetura do Cosmo segue o modelo Cliente-Servidor. A Figura 22 apresenta a arquitetura do Cosmo.

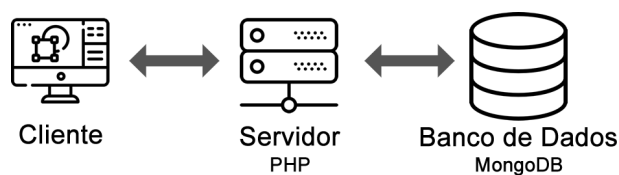


Figura 22 – Arquitetura do Cosmo

No seu *FrontEnd* foram utilizadas as tecnologias HTML5 (W3C, 2014), CSS3 (W3C, 2001) e Javascript (ECMAScript; ASSOCIATION et al., 2016). Sua interface foi desenhada com auxílio do AdminLTE ¹ e suporte do JQuery ².

Em seu *Back-End*, o Cosmo foi escrito com auxílio da linguagem de programação PHP ³, na sua versão 7.1.

A decisão de uso do MongoDB como SGBD (Sistema de Gerenciamento de Banco de Dados) para o projeto Cosmo, se dá ao fato da flexibilidade de como os dados são representados nesse tipo de SGBD. O uso do NoSQL maximizou a abstração de atividades do Cosmo. Mais sobre essa abstração é visto na seção 3.4.1. MongoDB é um banco de dados de código aberto, gratuito, de alta performance, sem esquemas e orientado à documentos (MONGODB, 2018).

3.3.2 Plugins

O Cosmo é uma plataforma extensível a *plugins* que tem a capacidade de incorporar qualquer tipo de atividade que ofereça suporte ao HTML5, Javascript e CSS3. Assim, existe a possibilidade de utilizar um jogo como atividade, um vídeo direto de um provedor de *stream*, a leitura de um documento, enfim, são inúmeras as possibilidades.

Mais detalhes sobre o desenvolvimento de *plugins* pode ser visto na seção 3.4.

3.3.3 Árvore de Aprendizagem

A árvore de aprendizagem é responsável por armazenar e gerenciar o conteúdo de atividades que são apresentadas aos alunos.

Exemplificando com um aluno que esteja cursando a disciplina de Introdução a Algoritmos. Em seu plano de aula, o professor abordará os assuntos como atribuição, estrutura de decisão e laços de repetição. Cada um desses assuntos são inseridos na árvore de aprendizagem e suas respectivas atividades. Após inserido, esse conteúdo é apresentado no *Dashboard* para que o aluno possa responder suas atividades.

Todo conteúdo formatado na árvore é modelado por um professor.

3.4 Plugins no Cosmo

Como já afirmado na seção 3.3.2, o Cosmo é uma plataforma extensível a *plugins*. Este capítulo se propõem a apresentar a arquitetura de *plugins* do Cosmo e a criação de uma atividade exemplo.

¹ <https://adminlte.io/themes/AdminLTE/index2.html>

² <https://jquery.com/>

³ <http://www.php.net/>

Por definição, um *plugin* é um programa de computador usado para adicionar funções a outros programas maiores, provendo alguma funcionalidade especial ou muito específica. Geralmente pequeno e leve, é usado somente sob demanda (RICE; FOEMMEL, 2018).

No Cosmo, as atividades são instâncias dos *plugins*. Quando um aluno acessa uma atividade, o núcleo da plataforma identifica e a apresenta ao aluno. Esta ação ocorre de forma transparente ao aluno, mas internamente o Cosmo busca em sua lista de *plugins* qual modelo deve ser instanciado.

A estrutura de *plugins* do Cosmo é capaz de abarcar qualquer tipo de aplicação que seja criada com uso do HTML5, CSS3 e Javascript. Com componentes bem isolados é simples incorporar jogos, vídeos, animações, regras de negócios tudo aqui que possa ser manipulado pelo HTML5 e o Javascript.

3.4.1 Arquitetura de um *Plugin*

A Área de Resposta, introduzida na seção 3.2.2, é onde o *plugin* é instanciado e exibido. Ao ser instanciado, o Cosmo monta a área de Enunciado e o *Content View*.

A Figura 23 apresenta a estrutura de *plugin* do Cosmo.

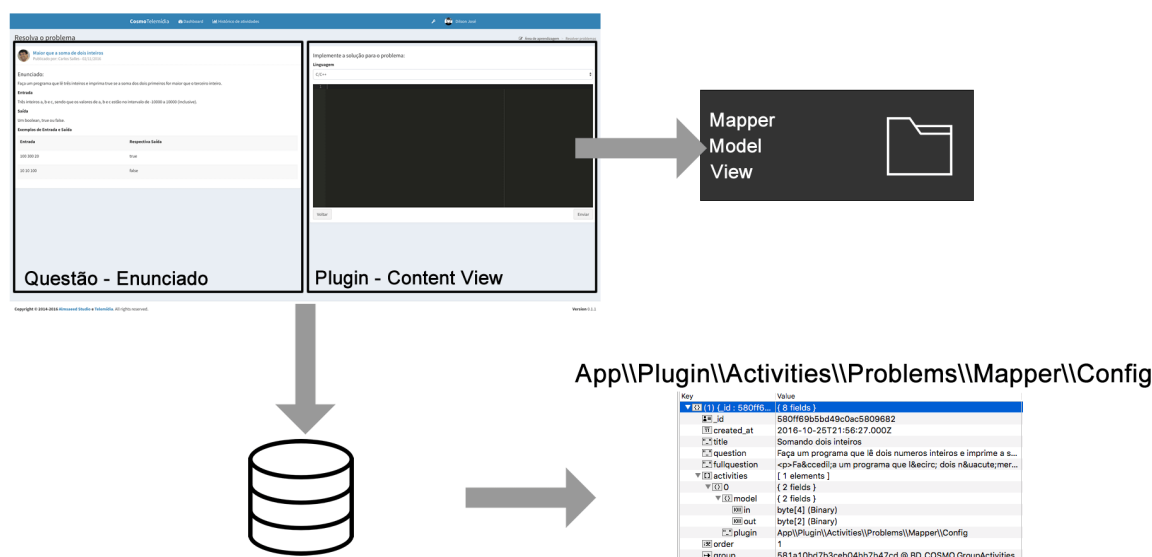


Figura 23 – Estrutura do *Plugin*

Cada *plugin* possui uma estrutura isolada do núcleo do ambiente e todo acesso ao núcleo é feito através de interfaces. Basicamente um *plugin* possui uma estrutura de três pastas: a *Model*, a *View* e a *Mapper*.

O diretório *Model* armazena todos os elementos que possam auxiliar a execução da tarefa. Nesse diretório é o local onde pode ser definida a regra de negócio que o *plugin*

irá apresentar. Nela podem ser aplicados elementos que não fazem parte da estrutura do Cosmo. Não é recomendável existir nesse diretório qualquer acesso à interface de banco de dados e outras chamadas de núcleo, para isso existe a camada *Mapper*.

A estrutura *Mapper* é responsável por definir as configurações do tipo de atividade para o Cosmo. A camada é organizada por 4 classes, uma de configuração, um *hydrator*, um modelo que representa o *plugin* e a classe de validação.

A classe configuração define o nome do *plugin*, o caminho absoluto da *view* do *plugin*, o pacote do *hydrator* e *validator*. Na Figura 24 é possível observar a estrutura da classe de configuração do *plugin*.

```
<?php
namespace App\Plugin\Activities\Problems\Mapper;

class Config
{
    public function __invoke()
    {
        return [
            'name' => 'Problems',
            'view' => 'Plugin/Activities/Problems/View/problems.twig',
            'hydrator' => 'App\Plugin\Activities\Problems\Mapper\ProblemsHydrator',
            'validate' => 'App\Plugin\Activities\Problems\Mapper\ProblemsValidate',
        ];
    }
}
```

Figura 24 – Modelo de classe de configuração do *plugin*

Essa configuração é a identificação do *plugin* junto à estrutura de dados. Ao submeter uma atividade, é necessário informar ao banco de dados que tipo de atividade está sendo criada. O Cosmo trabalha em sua estrutura de dados com o MongoDB, um tipo de banco de dados orientado a documentos. Esse tipo de banco permitiu uma ampla abstração na forma como dados são representados. Ou seja, é possível em uma mesma coleção de dados existir atividades com estruturas divergentes.

Cada atividade possui em sua estrutura campos comuns e um campo que representa o *plugin* da atividade. Detalhes sobre essa estrutura podem ser vistos na Figura 25.

Na Figura 25, pode-se observar que uma atividade é composta:

- id: Identificador da atividade, uma chave única criada pelo SGBD (Sistema de Gerenciamento de Banco de Dados);
- created_at: Data de criação da atividade;
- title: Título da atividade, esse campo é apresentado no *card* na área de *Dashboard* do aluno;
- question: Enunciado curto da atividade, visto também na área de *Dashboard* do aluno;

```

1 {
2   "_id" : ObjectId("580ff69b5bd49c0ac5809682"),
3   "created_at" : ISODate("2016-10-25T21:56:27.000+0000"),
4   "title" : "Somando dois inteiros",
5   "question" : "Faça um programa que lê dois numeros inteiros e imprime a soma destes dois números.",
6   "fullquestion" : "<p>Fa&ccedil;a um programa que lê dois números inteiros e imprime como resultado a soma desses dois",
7   "activities" : [
8     {
9       "model" : {
10        "in" : BinData(0, "MSAxCg="),
11        "out" : BinData(0, "Mgo=")
12      },
13      "plugin" : "App\\Plugin\\Activities\\Problems\\Mapper\\Config"
14    }
15  ],
16  "order" : NumberInt(1),
17  "group" : DBRef("GroupActivities", ObjectId("581a10bd7b3ceb04bb7b47cd"), "BD_COSMO")
18 }

```

Figura 25 – Estrutura de Dados de uma atividade

- `fullquestion`: Enunciado completo, visto apenas na Área de Resposta.
- `order`: Ordem de exibição da atividade, na versão atual do Cosmo esse campo foi depreciado.
- `group`: Grupo ao qual a atividade faz parte.
- `atividade`: Esse último item é a maximização da abstração citada no item 3.3.1. Esse *array* de elementos pode armazenar qualquer padrão da atividade. Isso permite criação de qualquer modelo, visto no campo *model*, e o indicativo de configuração do *plugin*, visto no campo *plugin*.

A classe de hydrator é responsável por injetar o modelo da atividade para o núcleo do Cosmo. Toda classe hydrator deve utilizar o método mágico do PHP chamado `invoke`. Um método mágico são métodos que podem ser implementados em uma classe com o propósito de ser chamado internamente pelo mecanismo PHP (DALL’OGLIO, 2015).

A classe de validação estende de uma classe abstrata chamada *AbstractValidateActivity*, esta classe contém dois métodos que precisam ser redefinidos para o *plugin* em desenvolvimento, o *saveAttempt* e o *saveHistory*. Além dos métodos redefinidos, a partir da classe abstrata, é necessário criar um método mágico `__invoke`, onde seu retorno deve ser do tipo booleano. O método `__invoke` irá informar ao núcleo do Cosmo se o aluno obteve sucesso ou insucesso ao responder a atividade.

Todo o gerenciamento dos *plugins* fica a cargo de um controlador interno do Cosmo, chamado *ActivitiesController*. Esse controlador se responsabiliza por desenhar a *view* que o usuário irá visualizar e de gerenciar toda e qualquer submissão enviada ao Cosmo. O

Diagrama de classe apresentado na Figura 26, estrutura a arquitetura do *plugin* sob a ótica de um diagrama de classe.

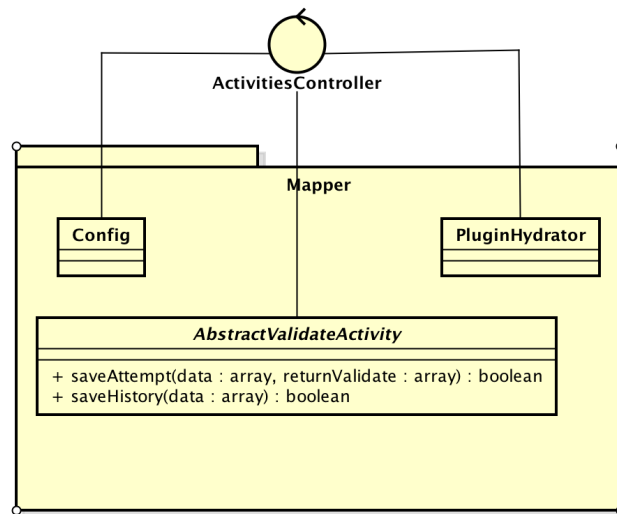


Figura 26 – Diagrama de classe do *plugin*

O diretório View armazena o HTML que será apresentado no *Content View*. Para auxiliar o desenvolvimento é utilizado um *template engine*, o Twig⁴. Esse tipo de template permite estender e isolar o uso de determinadas funcionalidade do PHP dentro da visão.

3.4.2 Atividade Problema

Um exemplo de tipo de atividade existente no Cosmo é chamada de Problema.

Esta atividade tem como foco descrever um problema e como desafio o aluno precisa desenvolver uma solução através de uma linguagem de programação oferecida pelo *plugin*. O *plugin* apresenta exemplos de entrada e saída para o código fonte submetido. Ao submeter o código-fonte, o sistema confronta a execução deste com uma bateria de testes. O resultado desta execução é comparado com um gabarito, retornando assim o indicativo de acerto ou erro ao aluno.

O *plugin* oferece suporte às linguagens Lua, C++ e Python.

Nesta seção é apresentada a implementação deste *plugin*, métodos e recursos.

Para o desenvolvimento do *plugin* foram utilizados recursos que operam no *frontend* e no *backend* do *plugin*. Seguindo a estrutura definida na seção 3.4.1, são apresentados detalhes das camadas de *Mapper*, *Model* e *View*.

⁴ <https://twig.symfony.com/>

3.4.2.1 Mapper

O arquivo de Configuração da camada Mapper do *plugin* Problema, foi configurado conforme a Figura 27.

```
<?php
namespace App\Plugin\Activities\Problems\Mapper;

class Config
{
    public function __invoke()
    {
        return [
            'name' => 'Problems',
            'view' => 'Plugin/Activities/Problems/View/problems.twig',
            'hydrator' => 'App\Plugin\Activities\Problems\Mapper\ProblemsHydrator',
            'validate' => 'App\Plugin\Activities\Problems\Mapper\ProblemsValidate',
        ];
    }
}
```

Figura 27 – Arquivo de Configuração do Plugin

Para o *plugin* Problema, o arquivo do *Hydrator* ficou simples, representa somente o retorno da instância do modelo que representa o *plugin*. A Figura 28 expõem o arquivo.

```
<?php
/** Created by PhpStorm. ... */
namespace App\Plugin\Activities\Problems\Mapper;

class ProblemsHydrator
{
    public function __invoke(array $model)
    {
        return new Problems($model['in'], $model['out']);
    }
}
```

Figura 28 – Classe *Hydrator* do *plugin* Problema

Para o arquivo do Modelo, ele tem a fiel representação do modelo na aplicação. Na Figura 29 é possível observar o modelo. Ele possui somente duas variáveis, uma que representa a entrada do código-fonte (*\$in*) e outra (*\$out*) que representa o resultado obtido a partir da execução do código-fonte junto à bateria de testes.

O último item da camada de *Mapper* é o arquivo de validação. três métodos são definidos nesta classe. O primeiro deles é o `__invoke`, o responsável pela execução das chamadas de negócios do *plugin*. Sendo assim é de sua responsabilidade instanciar os *plugins* de compilação e executar a bateria de testes. A Figura 30 apresenta a implementação do método `__invoke` para a classe *Validate* do *plugin*.

Os métodos *saveAttempt* e *saveHistory*, são métodos que fazem parte do núcleo do Cosmo, mas que podem ser redefinidos.

Na implementação, o *plugin* recebe os dados da atividade via parâmetro através da variável *\$data*. A variável *\$data* é do tipo *array* e armazena os dados: Identificador da


```
<?php
namespace App\Plugin\Activities\Problems\Mapper;

use MongoDB\BSON\Binary;

class Problems
{
    /**
     * @var string
     */
    private $in;

    /**
     * @var string
     */
    private $out;

    /**
     * Problems constructor.
     * @param string $in
     * @param string $out
     */
    public function __construct($in, $out)
    {
        $this->in = $in->bin;
        $this->out = $out->bin;
    }

    /**
     * @return mixed
     */
    public function getIn()
    {
        return $this->in;
    }

    /**
     * @param mixed $in
     */
    public function setIn($in)
    {
        $this->in = $in;
    }

    /**
     * @return mixed
     */
    public function getOut()
    {
        return $this->out;
    }
}
```

Figura 29 – Modelo do *plugin* Problema

atividade submetida, código-fonte submetido e linguagem de compilação. Dados pertinentes ao usuário logado no Cosmo ficam armazenados em sessão.

De posse do identificador da atividade, o núcleo do Cosmo busca informações sobre a atividade e executa o método *getActivities()*. Esse método irá preparar a variável *\$problem* para ser utilizada pelos modelos, que neste caso são representados pelas classes dos compiladores e interpretadores das linguagens (*CppExecute*, *LuaExecute* e *PythonExecute*). Outro comportamento do método é o de identificar as baterias de teste e gabaritos para o modelo, como pode ser visto no trecho que chama o método *getIn()* e *getOut()*.

Os modelos nessa implementação são classes externas que irão ser responsáveis por executar o código. São interfaces e possuem métodos comuns. A construção do objeto modelo está configurada conforme a seleção do usuário na Área de Resposta. Cada

```
function __invoke(array $data)
{
    $this->_dm = DatabaseFacilitator::getConnection();

    $idActivity = $data['id_activity'];
    $activity = $this->_dm->getRepository( documentName: Activities::class)->find($idActivity);
    $problem = $activity->getActivities();

    switch ($data['language']) {
        case "cpp":
            $obj = new CppExecute();
            break;
        case "lua":
            $obj = new LuaExecute();
            break;
        case "python":
            $obj = new PythonExecute();
            break;
        default:
            $obj = new LuaExecute();
            break;
    }

    $obj->criarEntradaCodigo($data['source_coude']);
    $obj->setRespostaInFile($problem->getIn());
    $obj->criaSaidaTeste($problem->getOut());

    $arrayReturn = $obj->runCode();
    $classReturn = new \stdClass();
    $classReturn->answer = $arrayReturn[0];
    $classReturn->payload = [
        'command' => $arrayReturn[2]
    ];

    return $classReturn;
}
```

Figura 30 – Implementação do método `__invoke`

modelo carrega 3 métodos: `criarEntradaCodigo`, este recebe o código-fonte feito pelo aluno; `setRespostaInFile`, este recebe a bateria de teste a ser aplicada pelo modelo; e `criaSaidaTeste`, este seta os gabaritos para comparação ao final da execução do modelo.

Por fim, o modelo executa a rotina de compilar ou interpretar o código fonte, rodar a bateria de testes e comparar o gabarito. O processo de compilação e execução do código roda em uma *sandbox* para garantir a integridade do servidor. A *sandbox* foi configurada com auxílio de *containers* Docker. Docker é uma plataforma aberta, criada com o objetivo de facilitar o desenvolvimento, a implantação e a execução de aplicações em ambientes isolados (GOMES, 2017).

O retorno encaminhado para o compilador é formado por um *array*, composto por: uma variável chamada *answer*, armazena o indicativo de certo ou errado; e o *array* *payload*, que retorna o resultado de execução do compilador ou interpretador.

3.4.2.2 Model

Como já definido na seção 22, o modelo serve de suporte para execução do *plugin*. No *plugin* atividade, foram definidos 3 modelos que compilam na linguagens Lua, C++ e Python. Todos os modelos utilizam um recurso de *sandbox* para executar os códigos-fontes.

3.4.3 View

Para o *View* do *plugin*, se fez uso um editor virtual de código o *Ace*⁵. Esse editor tem suporte a mais de 110 linguagens de programação, com uso de *highlights* e críticas de sintaxe.

O desenho do HTML contido na *View*, foi modelado utilizando a *engine* de *template* chamada Twig. Para iniciar qualquer *view* no Cosmo é necessário estender o arquivo de Layout conforme pode ser visto na Figura 31.

```
{% extends "View/layoutActivities.twig" %}
```

Figura 31 – Estendendo layout

No arquivo de Layout do *template* existem variáveis que podem ser alteradas para enquadrar o visual da página. São elas:

- *pagetitle* - Título da página no cabeçalho do navegador;
- *title* - Título da página no corpo da página;
- *script* - Adiciona *scripts* na página principal do *template*;
- *css* - Adiciona *css* na página principal do *template*;
- *content* - HTML que será exibido para o usuário.

⁵ <https://ace.c9.io/>

Parte III

Resultados

4 Um Experimento com a Plataforma Cosmo

Esta seção apresenta um experimento controlado efetuado com a Plataforma Cosmo. A proposta desse experimento é de analisar os *Logs* da plataforma e a partir dele analisar e aprender com os códigos-fontes submetidos pelos alunos. Com experimento espera-se obter conclusões que sejam promissoras, para assim auxiliar o professor ou até ser possível desenvolver ferramentas automáticas que auxiliem no processo de ensino em sala de aula.

4.1 Log de Dados do Cosmo

Em sua arquitetura, o Cosmo possui uma camada de gerenciamento de *logs*, os quais registram eventos relevantes que envolvam a manipulação de atividades no Cosmo.

Na Tabela 10 é possível observar todos os eventos gerenciados pela camada de *Log* do Cosmo e outros indicativos que compõem cada evento. Tais eventos contêm informações sobre onde e como ocorrem e, ainda, os dados que são armazenados.

Tabela 10 – Tabela de Eventos de *Log* do Cosmo

Evento	Onde ocorre?	O que é armazenado?
Submissão da Questão	Área de Resposta	Dados do Usuário, Código-fonte submetido, Tempo que o aluno demorou pra desenvolvimento do Código-fonte e Tentativa de resposta. A tentativa de resposta é um espelho da submissão da atividade.
Tentativa de Resposta da Questão	Área de Resposta	Dados do Usuário, Código-fonte submetido, Indicativo de resposta, Retorno do compilador
Satisfação e Recomendação	Área de Resposta e Histórico de Atividade	Grau de dificuldade e Classificação da questão.

4.2 Experimento Realizado

Este capítulo descreve um experimento controlado cujo objetivo central é demonstrar o emprego da plataforma Cosmo no centro do processo de coleta de informações sobre o progresso nas curvas de aprendizagem de uma turma de alunos de programação.

O experimento foi realizado com uma turma de 45 alunos no primeiro semestre de um curso de Ciência da Computação, na disciplina de Introdução à Programação. Foi

dado acesso à plataforma Cosmo a tais alunos de tal forma que eles poderiam responder o maior número de atividades disponíveis no período de uma semana.

Um total de 22 atividades estavam disponíveis aos alunos, as quais estavam agrupadas nas temáticas: "comando *print*, *read* e atribuição"(18 atividades), "comando *If*"(2 atividades) e "comando *Case*"(2 atividades). Todas as atividades foram elaboradas e validadas pelo professor da disciplina.

A plataforma Cosmo foi configurada no experimento para utilizar somente um tipo de *plugin*, denominado de Problema. Esta atividade tem como foco descrever um problema e como desafio o aluno precisa desenvolver uma solução através de uma linguagem de programação entre as disponíveis pelo *plugin*. A atividade Problema apresenta exemplos de entrada e saída de dados para ajudar o aluno produzir o código fonte da solução do problema. Ao submeter o código-fonte, o sistema confronta a execução deste com uma bateria de testes. O resultado desta execução é comparado com um gabarito, retornando assim o indicativo de acerto ou erro ao aluno.

O *plugin* de Problema foi limitado a utilizar apenas a linguagem de programação Lua, para atender a um requisito do professor, que a utiliza na disciplina. A apresentação da ferramenta foi feita por um tutor da disciplina, em sala de aula, e um Problema de exemplo foi respondido por este tutor a fim de exemplificar o uso do Cosmo. A interação inicial durou cerca de 100 minutos. Após este primeiro contato, a plataforma Cosmo estava disponível para os alunos responderem as 22 atividades de Problema disponíveis obedecendo o prazo de uma semana.

Após uma semana, foi feito um espelho da estrutura de dados coletados. A captura e análise dos *Logs*, foram feitas a partir dos eventos disponibilizados na Tabela 10.

A partir dos resultados obtidos, foi solicitado ao professor que definisse estereótipos para a turma, a fim de desenvolver uma análise qualitativa individual em alguns alunos específicos e também estender esta análise para uma ótica ampla adotando a turma como objeto de estudo.

Na Análise Qualitativa Individual, se inicia com a seleção dos alunos, sendo esta efetuada pelo próprio professor. Esta análise tem como objetivo observar o comportamento dos alunos junto a plataforma Cosmo. Os critérios da análise focam em aspectos como precisão ao responder as atividades, refinamento de código-fonte, esforço, erros e aproveitamento de uso.

Na Análise da Turma, a partir das características observadas nos alunos da Análise Qualitativa Individual, tenta-se buscar padrões de comportamento para assim agrupar os voluntários do experimentos em conjuntos os mais similares.

Detalhes completos sobre esta análise são vistos na seção 4.3.

Resultados gerais sobre o experimento é possível destacar as seguintes informações gerais:

- 444 atividades foram submetidas com a resposta correta;
- 117 atividades foram submetidas com a resposta incorreta.

Buscando uma outra visão dos dados, têm-se a relação entre os agrupamentos temáticos e as questões respondidas. A Tabela 11, apresenta a relação citada.

Tabela 11 – Agrupamentos Temáticos x Questões Respondidas

Temática	Acertou	Errou	Total Geral
Comando IF	48	3	51
Comando Caso	30	4	34
Comandos print, read, atribuição	366	110	476
Total Geral	444	117	561

4.3 Resultados Obtidos com o Experimento

Nesta seção são apresentados os resultados obtidos com a inspeção detalhada do histórico dos códigos fonte de alguns alunos selecionados (seção 4.3.1) e em seguida tomando com referência dados globais da participação da turma (seção 4.3.2) na plataforma Cosmo. Duas variáveis de destaque serão tópicos de análise: a primeira variável diz respeito à quantidade de tentativas submetidas para obtenção de um acerto; e a outra é o código fonte gerado durante as submissões das tentativas.

4.3.1 Análise Qualitativa Individualizada

Nesta seção é feita uma análise detalhada sobre o uso da plataforma Cosmo a partir de alguns alunos específicos.

A seleção dos alunos para avaliação qualitativa individualizada foi feita diretamente pelo professor da disciplina, tomando como base tanto o desempenho dos discentes na avaliação escrita da disciplina, quanto a percepção do professor em sala de aula. A ideia da avaliação individualizada é analisar padrões de perfis de uso na plataforma Cosmo. A Tabela 12 apresenta os alunos selecionados e os critérios de escolha. Para este trabalho foram dados nomes fictícios e sugestivos aos alunos selecionados para análise.

A análise se inicia validando a presença dos discentes junto ao experimento do Cosmo. Como resultado, dos alunos selecionados somente um dos discentes não participou do experimento. A segunda parte da análise foi identificar perfis de uso desses alunos junto à ferramenta Cosmo, com base nos *Logs* ofertados.

Tabela 12 – Alunos Selecionados para Avaliação Qualitativa

Alunos (Nome Fictício)	Critérios de escolha definido pelo Professor
Exemplar	Fez a melhor prova da turma e o seu desempenho é destaque na plataforma Cosmo
Participativo	Foi aquele com o maior tempo no Cosmo
Ocasional	Considerado pelo professor o aluno com mais dificuldade nas atividades de laboratório
Ausente	Considerado pelo professor um aluno que continua frequentando a disciplina mas não costuma realizar as atividades da mesma

A quantidade de dados obtidos com a plataforma Cosmo sobre cada aluno individualmente é bem ampla. Relembrando o que foi visto na seção 4.1, o Cosmo armazena diversas informações do aluno. De maneira geral, há uma quantidade significativa de dados que precisam ser estudados para avaliar o desempenho individual de cada aluno.

A análise se sustenta em dados como o código-fonte submetido, quantidade de questões, o número de tentativas e refinamento de código-fonte. Todas conclusões são visualizadas nas subseções seguintes, onde é possível observar os *logs* de como cada Aluno se comporta junto ao Cosmo durante o experimento.

4.3.1.1 Aluno Exemplar

O Aluno Exemplar é um aluno com características bem assertivas. Em seu histórico foi possível observar qualidades positivas, como um alto número no sucesso ao responder as atividades associado ao baixo número de tentativas. Em mais de 90% das atividades respondidas foi possível observar submissões positivas com apenas uma tentativa. Ainda tomando o número de tentativas como métrica, seu pior caso de submissão foi de 8 tentativas para 1 acerto.

Avaliando ainda outros aspectos que puderam ser minerados do *Log* da aplicação Cosmo, o aluno apresentou um código sucinto, limpo e um baixo índice de erros de compilação. Este último item só teve 5 incidências registradas em *Log*.

A Figura 32, apresenta um exemplo do código-fonte criado pelo Aluno Exemplar e o Aluno Ausente. Neste exemplo o Aluno precisava entrar com 2 números e um caractere representando um operador matemático, o resultado de retorno seria a operação aritmética entre os dois números.

Avaliando o código fonte deste aluno se percebe o uso de recursos avançados da linguagem Lua, como curto-circuito e meta-funções, mostrando mais maturidade que a média no aprendizado inclusive de tópicos avançados. O padrão identificado neste aluno é de muita precisão e bastante rapidez para resolver os problemas quando comparado a outros alunos.

<pre>a=io.read() n=string.find(a, " ") a,b=string.sub(a,1,n-1),string.sub(a,n+1) n=string.find(b, " ") b,c=string.sub(b,1,n-1),string.sub(b,n+1) print((c=="+" and a+b) or (c=="-" and a-b) or (c=="*" and a*b) or a/b)</pre>	<pre>n1 = io.read('*number') n2 = io.read('*number') operacao = io.read() if operacao=="+" then print(n1+n2) elseif operacao=="-" then print(n1-n2) elseif operacao=="*" then print(n1*n2) elseif operacao=="/" then print(n1/n2) else print("Operação Inválida") end</pre>
---	--

Figura 32 – Exemplo do código-fonte criado pelo Aluno Exemplar (à esquerda) e exemplo de código fonte criado pelo Aluno Ausente (à direita)

4.3.1.2 Aluno Participativo

O Aluno Participativo, é um aluno com características que se assemelham ao Aluno Exemplar. Ele possui um bom número de acertos em atividades, mas a contrapartida está no esforço efetuado para responder uma dessas atividades. Foram 71 tentativas para concluir as 22 atividades disponibilizadas pelo Cosmo. Em seu pior caso foram necessárias 21 tentativas para um acerto. Outro fato interessante é que houve um aumento no número de erros de compilação, totalizando 28 erros.

Fonte	Retorno Interpretador
s1, s2, s3 = io.read(),io.read(),io.read()\nprint(s1 .. s2 .. s3)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_200QQH:2: attempt to concatenate global 's2'
a,b,c = io.read(),io.read(),io.read()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_Y1Uoxc:3: attempt to concatenate global 'b' (a r
a,b,c = io.write(),io.write(),io.write()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_RjKdZW:3: attempt to concatenate global 'b' (a r
a,b,c = io.read(),io.read(),io.read()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_hsZC5i:2: attempt to concatenate global 'b' (a ni
a,b,c = io.write(),io.write(),io.write()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_LkGzD:2: attempt to concatenate global 'b' (a t
a,b,c = io.read(),io.read(),io.read()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_4291R5:2: attempt to concatenate global 'b' (a r
a, b, c = io.read(),io.read(),io.read()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_0M33z1:2: attempt to concatenate global 'b' (a
a, b, c = io.write(),io.write(),io.write()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_LWAPj9:2: attempt to concatenate global 'b' (a t
a, b, c = io.read(),io.read(),io.read()\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_izwzlf:2: attempt to concatenate global 'b' (a nil
a, b, c = io.read('*l'),io.read('*l'),io.read('*l')\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_BmBcfe:2: attempt to concatenate global 'b' (a r
a,b,c = io.read(),io.read(),io.read()\nio.write(a)\nio.write(b)\nio.write(c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_s1hUZ3:3: bad argument #1 to 'write' (string ext
a,b,c = io.read(),io.read(),io.read()\nio.write(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_7qs8H1:2: attempt to concatenate global 'b' (a r
a,b,c = io.read(),io.read(),io.read()\nio.write(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_OQZkVA:2: attempt to concatenate global 'b' (a r
a,b,c = io.read('*n'),io.read('*n'),io.read('*n')\nio.write(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_n72MLb:2: attempt to concatenate global 'b' (a
a,b,c = io.read('*l'),io.read('*l'),io.read('*l')\nio.write(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_fTAhVH:2: attempt to concatenate global 'b' (a r
a,b,c = io.read('*l'),io.read('*l'),io.read('*l')\nreturn a .. b .. c	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_Mk0vA5:2: attempt to concatenate global 'b' (a
a,b,c = io.read('*l'),io.read('*l'),io.read('*l')\nprint(a .. b .. c)	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_JPT3Wb:2: attempt to concatenate global 'b' (a
a,b,c = io.read('*l'),io.read('*l'),io.read('*l')\nprint(a,b,c)	pikachu pokemon joana\tnil\tnil\n
a,b,c = io.read('*l'),io.read('*l'),io.read('*l')\nprint(a..b..c)\n	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_5Gokoa:2: attempt to concatenate global 'b' (a r
a,b,c = io.write('*l'),io.write('*l'),io.write('*l')\nprint(a..b..c)\n	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_h5rZxj:2: attempt to concatenate global 'b' (a bc
a,b,c = io.write('*l'),io.write('*l'),io.write('*l')\ntostring(a)\ntostring(b)\ntostring(c)\nprint(tostring(a..b..c))\n	/usr/bin/lua: /tmp/TMP_COSMO_PROBLEMS_kvvk7i:5: attempt to concatenate global 'b' (a bc
text = io.read()\nspace = string.find(text, " ")\ntext1, text2 = string.sub(text, 1, space - 1), string.sub(text, space + 1)\nnspace = string.find(text2, " ")\ntext2, text3 = string.sub(text2, 1, space - 1), string.sub(text2, space + 1)\nprint(text1 .. text2 .. text3)	pikachupokeballjoana\n

Figura 33 – Quadro de submissão do aluno Participativo

Na Figura 33 é possível observar o comportamento do Aluno participativo e a confirmação de suas características. Neste exemplo o aluno precisava responder uma atividade que recebia como entrada de dados 3 strings e em seu retorno deveria ser apresentado a concatenação das strings. Para obter o êxito em responder a atividade, o Aluno participativo teve que efetuar 22 tentativas.

Este aluno é caracterizado pela atenção em treinar e aprender a programar, com muita persistência para tentar resolver os problemas, mesmo quando ocorrem erros em

alguns deles. O Aluno Participativo não é tão preciso quanto o Exemplar mas conseguiu a mesma quantidade bruta de acertos, apesar de precisar de mais tempo para isso. Ao analisar os códigos fonte deste aluno não se vê o emprego de recursos tão avançados quanto o Exemplar.

4.3.1.3 Aluno Ocasional

O Aluno Ocasional foi escolhido pelo professor da disciplina por mostrar interesse em aprender o conteúdo, porém com muita dificuldade principalmente de base matemática e lógica para fazer isso de forma mais rápida e como a média da turma. Avaliando os *logs* do Cosmo, ele conseguiu atingir o objetivo do experimento e concluiu todas as 22 atividades disponíveis. Mas esse êxito só foi possível através de 175 tentativas. É uma média de 8 tentativas por acerto de atividade. Na atividade com mais dificuldades, o aluno fez 60 tentativas para um acerto.

O número de tentativas se reflete na quantidade de erros de compilação que para este discente chegou ao valor de 41 erros catalogados.

O Aluno Ocasional está bem animado em aprender mas sua curva de aprendizagem é mais lenta que os demais. Ele precisa que os Problemas sejam mais numerosos e com uma acentuação de dificuldade progressiva pouco íngreme para ele crescer em aprendizado e não desistir facilmente.

4.3.1.4 Aluno Ausente

Por fim, o Aluno Ausente não conseguiu fornecer informações o suficiente para o experimento de tal forma a permitir que se tire muitas conclusões. Ele tentou apenas um problema, com um código ingênuo e longe de obter sucesso.

Provavelmente o Aluno Ausente exige o emprego de técnicas não-convencionais que o estimulem a praticar programação. Ele precisa de mais motivação ou sentimento maior de segurança que o permita progredir no aprendizado quando comparado aos outros alunos estudados.

4.3.1.5 Perfis dos Alunos no Cosmo

A partir das características encontradas na análise qualitativa individual em alunos selecionados no Cosmo, foi possível observar algumas características comuns na turma. Assim, para uso na plataforma Cosmo, os alunos foram tabulados conforme a Tabela 13. A Tabela 13 apresentada é a composição entre as características comuns encontradas na turma e o critério de aceitação para enquadrar o aluno. Todas as informações definidas no quadro, foram definidas a partir de conclusões obtidas através do *Log* do Cosmo.

Tabela 13 – Perfis Alunos na Plataforma Cosmo

Perfil	Características	Critério de aceitação
Exemplar	Alto número de assertiva com um baixo número de tentativas, código sucinto, baixo índices de erros de compilação e boas notas na prova.	Nota 10 e Quantidade de tentativas abaixo de 50
Participativo	Alto número de assertiva, mas com um esforço de tentativa elevado e um número de erros de compilação superior ao visto no Perfil Exemplar. Normalmente tem notas medianas.	Características entre Exemplar e Ocasional
Ocasional	Alto número de assertivas, mas com um esforço de tentativas e números de erros de compilação elevadíssimos. Normalmente apresentam nota baixa na prova.	Nota abaixo de 5 e/ou Quantidade de tentativas a partir de 50
Ausente	Baixo número de assertivas ou até nenhuma. Normalmente apresentam baixa nota na prova ou não a fazem.	Não participou do experimento ou obteve um aproveitamento abaixo do Ocasional.

A intenção com esse agrupamento dos alunos em quatro perfis não é meramente a de tentar fazer um *clustering* da turma ou encontrar uma estratificação de aprendizes que seja genérica. A presente pesquisa não é quantitativa e deve ser reforçado que a intenção aqui não é criar esteriótipos para aprendizes de programação. O foco, na verdade, é estudar a informação coletada por esta turma específica de 45 alunos e o padrão de aprendizado que alguns dos perfis de alunos de tal turma apresenta.

4.3.2 Análise da Turma

Esta subseção tem como objetivo apresentar uma análise dos dados obtidos a partir *Log* do Cosmo e tomando como referencia a Turma.

4.3.2.1 Como foi feita a Análise?

A análise foi construída a partir de uma lista nominal dos alunos, o número de acertos, número de tentativas e a nota da prova.

O primeiro passo foi categorizar os alunos, com base nos perfis definidos na seção 4.3.1.5. A partir desse agrupamento pode-se obter a Tabela 14.

Outro fato importante foi a definição de aproveitamento de uso na plataforma Cosmo. O cálculo para obtenção do aproveitamento é a relação entre a quantidade de acertos pelo valor total de atividades disponíveis no Cosmo. A partir desta categorização é

Tabela 14 – Quantidade de alunos agrupados por perfis

Perfil	Quantidade
Exemplar	10
Participativo	18
Ocasional	10
Ausente	7
Total Geral	45

possível concluir fatos que são destacados na seção 4.3.2.2.

4.3.2.2 Conclusões da análise da turma

Com a obtenção dos valores de aproveitamento, foi possível observar um padrão intrigante com a turma em questão. Os alunos Exemplares tiveram um aproveitamento médio de 41% de uso da ferramenta, o que em contrapartida os alunos com o perfil Ocasional tiveram um percentual médio de 62% de uso da ferramenta Cosmo. Na Figura 34 é apresentado um gráfico com o percentual de aproveitamento dos alunos em suas respectivas categorias.

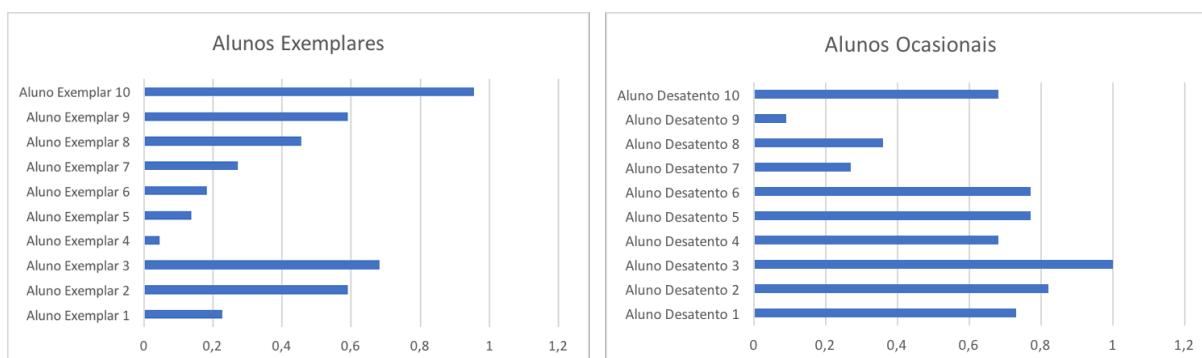


Figura 34 – Gráficos de aproveitamento de uso: Alunos Exemplares e Desatentos

A partir disto é possível observar que mesmo o aluno sendo categorizado como "Ocasional", este mostra interesse sobre a disciplina, possui uma desenvoltura promissora junto a plataforma Cosmo, mas infelizmente não obteve um êxito em obter uma boa nota da avaliação da disciplina.

Os alunos classificados como Participativos e Desistente, tiveram dados dentro do critério de aceitação estipulado, mas com características bem diversificadas o que não possibilitou a definição de um padrão que agregasse algo relevante a pesquisa.

Ainda tomando como referência os valores de aproveitamento de uso da ferramenta, foi possível observar que os alunos Participativos e Ocasionais, são os alunos que mais utilizaram a ferramenta tendo o aproveitamento entre 70% a 100% de uso.

Todos essas métricas coletadas foram apresentadas ao professor que a consideraram de cunho relevante para a melhoria de ensino em sala de aula.

Conclusão

Este trabalho apresenta o Cosmo como centro de um experimento controlado em que o conjunto de informações coletadas pela plataforma é usado para análise de dois fatores. Primeiro, tais dados ajudam a detectar aspectos que auxiliem na detecção do nível ou curva de aprendizado de alunos de programação. Segundo, os dados podem atuar como entrada potencial para ferramentas de suporte ao processo de aprendizado de alunos em uma disciplina de Introdução a Algoritmos.

A partir dos *Logs*, foi possível fazer uma análise individualizada dos alunos da turma. O artigo descreve os resultados obtidos com quatro desses alunos selecionados pelo professor de tal forma a ilustrar quatro diferentes perfis identificados por ele. Os dados foram extrapolados para a turma, de tal forma a agrupar os voluntários do experimento em conjuntos os mais similares possíveis aos alunos avaliados individualmente.

Apesar do trabalho ter usado uma análise qualitativa inspecionada por avaliadores humanos, um resultado relevante do trabalho é a demonstração que é possível analisar os dados coletados de forma automática ou semi-automática. Isso cria o potencial para o desenvolvimento futuro de uma ferramenta de tomada de decisão que apoie professores ou mesmo a plataforma Cosmo a selecionar melhor atividades mais adequadas à curva de aprendizagem individual dos alunos.

É importante ser ressaltado que o foco do trabalho não foi avaliar o processo metodológico empregado pelo professor em sala de aula, mas de aprender com os dados e enriquecer a plataforma Cosmo. Um aspecto interessante que está sendo tomado como requisito para aplicação é o uso de *Machine Learning* e uso de *Dashboards Realtime*, que permitam à ferramenta oferecer ao professor subsídios que possam lhe auxiliar em sala de aula de uma forma mais adequada.

Um exemplo de possível trabalho futuro reside no uso de técnicas de *Machine Learning* para um tutor inteligente que possa recomendar e atuar junto aos alunos, ou também atuar de forma diferenciada para os alunos Opcionais e Desistentes. Já os *Dashboards Realtime* podem ser empregados para auxiliarem o professor a observar o desempenho da turma, pontos fortes e fracos dos alunos e outras métricas em tempo real.

O Cosmo ainda não está finalizado, outros recursos podem ser incorporados a plataforma. Por ser extensível a *plugins*, como trabalhos futuros poderiam ser implementados novas atividades para a plataforma, algumas até já levantadas na pesquisa de expectativa de uso. Outra funcionalidade interessante citada como requisito para a plataforma é a gamificação do Cosmo, a utilização de ranqueamento, medalhas, conquistas e outros elementos que compõem a gamificação.

Esta dissertação teve um trabalho aceito para apresentação como artigo completo no 26º WEI, *Workshop* sobre Educação em Computação¹. O trabalho é intitulado "Cosmo: Um ambiente virtual de aprendizado com foco no Ensino de Algoritmos" e apresenta a plataforma Cosmo como ambiente virtual de aprendizado.

Outro trabalho foi submetido como artigo completo para o 29º SBIE, Simpósio Brasileiro de Informática na Educação². O trabalho tem o título "Aprendendo com o Código Fonte de Aprendizes de Programação: Um Experimento com a Plataforma Cosmo" e discute como podemos aprender com código-fonte dos alunos de programação. Este trabalho está aguardando aprovação do simpósio.

¹ http://natal.uern.br/eventos/csb2018/?page_id=197

² <http://cbie2018.virtual.ufc.br/index.php/sbie/>


Referências

- ALMEIDA, M. E. B. de. Educação a distância na internet: abordagens e contribuições dos ambientes digitais de aprendizagem. *Educação e pesquisa*, SciELO Brasil, v. 29, n. 2, p. 327–340, 2003. Citado na página 15.
- BECK, K.; FOWLER, M. *Planning extreme programming*. [S.l.]: Addison-Wesley Professional, 2001. Citado na página 28.
- CARVALHO, M. d.; TAFNER, P. Ensino superior brasileiro: a evasão dos alunos e a relação entre formação e profissão. *Anais do 30º Encontro anual da ANPOCS*, 2006. Citado na página 15.
- DALL’OGLIO, P. *PHP Programando com Orientação a Objetos 3ª Edição*. [S.l.]: Novatec Editora, 2015. Citado na página 45.
- DANN, W. et al. Mediated transfer: Alice 3 to java. In: ACM. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. [S.l.], 2012. p. 141–146. Citado na página 21.
- DUARTE, E. V.; PEARCE, J. L. A cross-cultural review of lightbot for introducing functions and code reuse. *Journal of Computing Sciences in Colleges*, Consortium for Computing Sciences in Colleges, v. 33, n. 2, p. 100–105, 2017. Citado na página 19.
- ECMASCRIPT, E.; ASSOCIATION, E. C. M. et al. *Ecmascript language specification*. 2016. <<https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>>. Acessado em: 15/04/2018. Citado na página 42.
- GOMES, R. *Docker para desenvolvedores*. [S.l.]: Leanpub, 2017. Citado na página 49.
- KIRAN, E. L.; MOUDGALYA, K. M. Evaluation of programming competency using student error patterns. In: IEEE. *Learning and Teaching in Computing and Engineering (LaTiCE), 2015 International Conference on*. [S.l.], 2015. p. 34–41. Citado na página 15.
- MEHLECKE, Q. T. C.; TAROUÇO, L. M. R. Ambientes de suporte para a educação a distância: a mediação para aprendizagem cooperativa. *RENOTE: revista novas tecnologias na educação [recurso eletrônico]*. Porto Alegre, RS, 2003. Citado na página 15.
- MONGODB. *What is MongoDB?* 2018. Disponível em: <<https://www.mongodb.com/what-is-mongodb>>. Citado na página 42.
- PALMEIRA, L. B.; SANTOS, M. P. Evasão no bacharelado em ciência da computação da universidade de Brasília: análise e mineração de dados. *Monografia do curso de Ciência da Computação da UnB–Universidade de Brasília. Brasília: UnB*, 2014. Citado na página 15.
- PONTES, H. Game development in learning process of algorithms and computer programming. 10 2013. Citado na página 18.
- RAJARAVIVARMA, R. A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, ACM, v. 37, n. 4, p. 98–102, 2005. Citado na página 18.

- RAPKIEWICZ, C. E. et al. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE: revista novas tecnologias na educação [recurso eletrônico]*. Porto Alegre, RS, 2007. Citado na página 15.
- RESNICK, M. et al. Scratch: programming for all. *Communications of the ACM*, ACM, v. 52, n. 11, p. 60–67, 2009. Citado 3 vezes nas páginas 18, 26 e 27.
- RICE, D.; FOEMMEL, M. *Plugin*. 2018. Acessado em: 25/06/2018. Disponível em: <<https://martinfowler.com/eaCatalog/plugin.html>>. Citado na página 43.
- RIES, E. *A startup enxuta*. [S.l.]: Leya, 2012. Citado na página 33.
- SABBAGH, R. *Scrum: Gestão ágil para projetos de sucesso*. [S.l.]: Editora Casa do Código, 2014. Citado na página 28.
- SANTOS, R. P. dos; COSTA, H. A. X. Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *INFOCOMP Journal of Computer Science*, v. 5, n. 1, p. 41–50, 2006. Citado na página 15.
- SOMMERVILLE, I. *Engenharia de software*. [S.l.]: PEARSON BRASIL, 2011. ISBN 9788579361081. Citado na página 29.
- SPENDOLINI, M. *The benchmarking book*. [S.l.], 1992. Citado na página 29.
- TRELLO. *Trello*. 2018. Acessado em: 30/06/2018. Disponível em: <<https://trello.com/home>>. Citado na página 28.
- W3C. *Introduction to CSS3*. 2001. <<https://www.w3.org/TR/css3-selectors/>>. Acessado em: 15/04/2018. Citado na página 42.
- W3C. *HTML5 Specification*. 2014. <<https://www.w3.org/TR/html5/>>. Acessado em: 31/05/2016. Citado na página 42.
- ZANCHETT, G. A.; VAHLIDICK, A.; RAABE, A. Games for programming as an approach for first programming experiences. *International Journal on Computational Thinking (IJCThink)*, v. 1, n. 1, p. 39, 2017. Citado na página 20.

Anexos

ANEXO A – Termo de Consentimento Livre e Esclarecido



1ª via participante

Termo de Consentimento Livre e Esclarecido

Eu, _____

autorizo o uso, com fins estritamente acadêmicos, das informações fornecidas durante a entrevista sobre o uso de tecnologias na educação, realizada no dia_/2017, a partir das_h.

Estou ciente que:


- (1) minha participação é voluntária;
- (2) a sessão será registrada com anotações e captura de áudio;
- (3) esta sessão visa prover informações a uma pesquisa acadêmica relacionada ao uso de tecnologias educacionais, desenvolvido por Dilson José Lins Rabelo Júnior, sob orientação do professor (a) Carlos de Salles Soares Neto, na Universidade Federal do Maranhão;
- (4) será garantido o anonimato no uso das informações capturadas nesta sessão;
- (5) todos os dados brutos serão acessados somente pelos pesquisadores envolvidos nesta pesquisa.
- (6) a qualquer momento, até dois ano após o término da pesquisa, poderei solicitar mais informações sobre o estudo ou cópias dos materiais divulgados, entrando em contato com o pesquisador através do e-mail **dilsonjrjr@gmail.com**

() Autorizo o uso dos dados coletados conforme as condições supracitadas.

() Não autorizo o uso dos dados coletados.

Pesquisador	Participante
Dilson José Lins Rabelo Junior	_____
<i>(nome)</i>	<i>(nome)</i>
_____	_____
<i>(assinatura)</i>	<i>(assinatura)</i>

Figura 35 – Termo de Consentimento Livre e Esclarecimento - Página 1



2ª via pesquisador

Termo de Consentimento Livre e Esclarecido

Eu, _____,

autorizo o uso, com fins estritamente acadêmicos, das informações fornecidas durante a entrevista sobre o uso de tecnologias na educação, realizada no dia_/ /2017, a partir das_h.

Estou ciente que:

- (1) minha participação é voluntária;
- (2) a sessão será registrada com anotações e captura de áudio;
- (3) esta sessão visa prover informações a uma pesquisa acadêmica relacionada ao uso de tecnologias educacionais, desenvolvido por Dilson José Lins Rabelo Júnior, sob orientação da professor (a) Carlos de Salles Soares Neto, na Universidade Federal do Maranhão;
- (4) será garantido o anonimato no uso das informações capturadas nesta sessão;
- (5) todos os dados brutos serão acessados somente pelos pesquisadores envolvidos nesta pesquisa.
- (6) a qualquer momento, até dois ano após o término da pesquisa, poderei solicitar mais informações sobre o estudo ou cópias dos materiais divulgados, entrando em contato com o pesquisador através do e-mail **dilsonjrjr@gmail.com**.

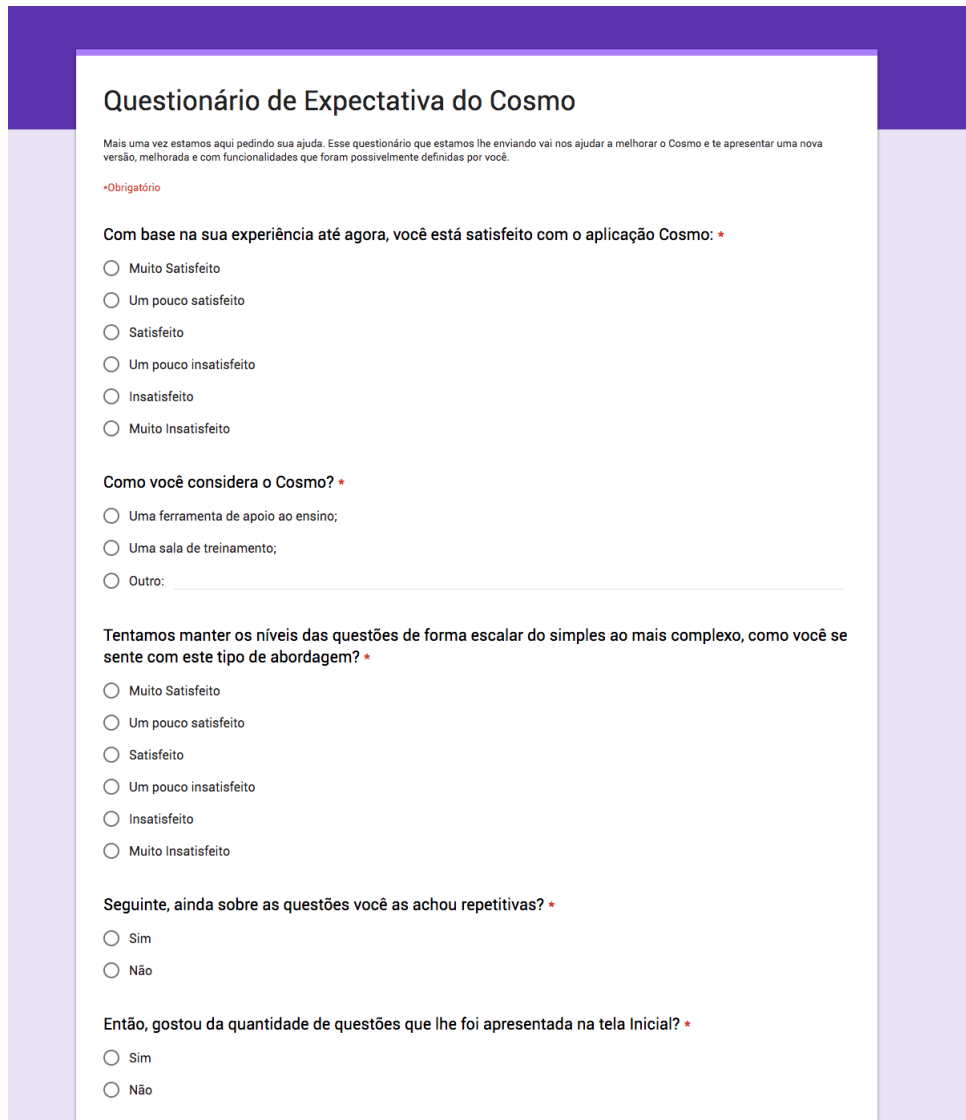
() Autorizo o uso dos dados coletados conforme as condições supracitadas.

() Não autorizo o uso dos dados coletados.

Pesquisador	Participante
_____ Dilson José Lins Rabelo Junior (nome)	_____ (nome)
_____ (assinatura)	_____ (assinatura)

Figura 36 – Termo de Consentimento Livre e Esclarecimento - Página 2

ANEXO B – Questionário de Expectativa do Cosmo



Questionário de Expectativa do Cosmo

Mais uma vez estamos aqui pedindo sua ajuda. Esse questionário que estamos lhe enviando vai nos ajudar a melhorar o Cosmo e te apresentar uma nova versão, melhorada e com funcionalidades que foram possivelmente definidas por você.

***Obrigatório**

Com base na sua experiência até agora, você está satisfeito com o aplicação Cosmo: *

- Muito Satisfeito
- Um pouco satisfeito
- Satisfeito
- Um pouco insatisfeito
- Insatisfeito
- Muito Insatisfeito

Como você considera o Cosmo? *

- Uma ferramenta de apoio ao ensino;
- Uma sala de treinamento;
- Outro: _____

Tentamos manter os níveis das questões de forma escalar do simples ao mais complexo, como você se sente com este tipo de abordagem? *

- Muito Satisfeito
- Um pouco satisfeito
- Satisfeito
- Um pouco insatisfeito
- Insatisfeito
- Muito Insatisfeito

Seguinte, ainda sobre as questões você as achou repetitivas? *

- Sim
- Não

Então, gostou da quantidade de questões que lhe foi apresentada na tela Inicial? *

- Sim
- Não

Figura 37 – Questionário de Expectativa do Cosmo - Página 1

Caso você tenha dito não, de boas, diga-nos quantas questões você acredita que deveríamos exibir na sua área de aprendizagem:

Sua resposta

E a interface você a considera intuitiva? Gostaríamos de ouvir seu ponto de vista. *

Sua resposta

Estamos trabalhando em outras atividades, dentre elas um Caça Bugs e um RPG. Agora eu queria saber de você, que tipo de atividade gostaria de visualizar no Cosmo? Você tem total liberdade para indicar o que quiser. *

Sua resposta

Prometo que estamos perto do fim, abaixo vou te apresentar alguma coisas em que estamos trabalhando e pedimos que você defina a ordem de desenvolvimento delas, isso mesmo, você é que define como iremos trabalhar :D. 1. Organização por Turma 2. Caça Bugs 3. RPG 4. Gamificação do cosmo *

Sua resposta

Agora é algo sendo bem abrangente, esteja livre para nos indicar novos requisitos que você deseja ver no cosmo e assim chegaremos ao fim do questionário. *

Sua resposta

ENVIAR

Nunca envie senhas pelo Formulários Google.

Este conteúdo não foi criado nem aprovado pelo Google. Denunciar abuso - Termos de Serviço - Termos Adicionais

Google Formulários

Figura 38 – Questionário de Expectativa do Cosmo - Página 2

ANEXO C – Resultado do Questionário de Expectativa do Cosmo

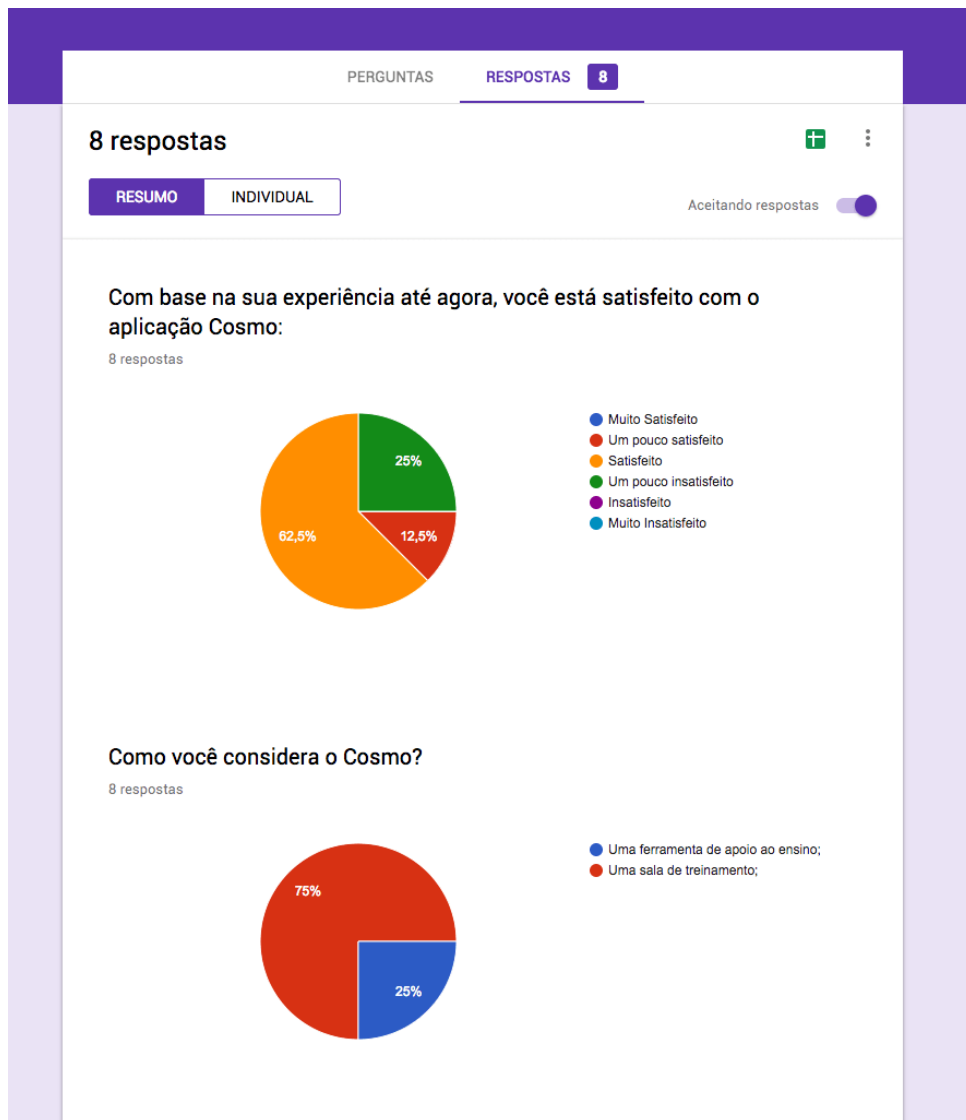


Figura 39 – Resultado do Questionário de Expectativa do Cosmo - Página 1



Figura 40 – Resultado do Questionário de Expectativa do Cosmo - Página 2

Caso você tenha dito não, de boas, diga-nos quantas questões você acredita que deveríamos exibir na sua área de aprendizagem:

0 resposta

Ainda não há respostas para esta pergunta.

E a interface você a considera intuitiva? Gostaríamos de ouvir seu ponto de vista.

8 respostas

Ela é um pouco intuitiva

Eu gosto da interface, no entanto eu nunca consegui refazer as atividades já completas, queria poder refazer elas.

A interface está de acordo com as necessidades do aluno e muito bem elaboradas, ótima organização e estrutura.

Bem clara, bem intuitiva! Gostei.

É um pouco complicada no início, deveria haver um pequeno tutorial ou exemplo demonstrando a forma correta de responder. Em alguns exercícios o código fazia o que era pedido mas a plataforma não validava.

Sim, considero. Apesar que pode ser melhorado. Principalmente em relação as questões que podem ser mais claras.

Gosto muito da ideia do cosmo. Considero sua interface bem intuitiva e de fácil entendimento, mas precisa concertar uns bug quanto requisito para fazer a questão, em alguns casos que não pedia if passava caso o usuário fizesse o uso do if para terminar a questão.

Sim, acredito que precisa melhorar um pouco em alguns aspectos mas no geral considero sim a interface intuitiva

Estamos trabalhando em outras atividades, dentre elas um Caça Bugs e um RPG. Agora eu queria saber de você, que tipo de atividade gostaria de visualizar no Cosmo? Você tem total liberdade para indicar o que quiser.

8 respostas

as mesmas já aplicadas e as novas que foram mencionadas

Caça bugs e rpg parece bom pra mim <3

Para o primeiro período, seria recomendado questões básicas e as com um pouco mais de dificuldades seriam para o segundo período.

RPG

Após responder todos os exercícios deveria haver um "extra" cujo o aluno deveria fazer um programa abordando todas as atividades que foram realizadas na plataforma, uma espécie de revisão.

Caça Bugs é bom para aprendermos a visualizar códigos.

Um sistema de rankeada. 1 determinado usuário logado no sistema desafia outro usuário, o cosmo gera uma mesma lista de desafios para ambos os jogadores e quem terminasse primeiro era o vencedor, podia ser adicionado um rank e um moeda virtual para os desafios e quanto mais alto a aposta baseado na moeda mais o desafio se tornava difícil e mais a pontuação do jogador crescia em caso de vitória ou decrescia em caso de derrota com o rank sendo atualizado diariamente.

Acharia interessante que tivesse uma opção "Desisto" para ver a resposta caso passasse muito tempo sem conseguir resolver a questão. Talvez pequenos "tutoriais" sobre o conteúdo abordado nas questões, bem resumido, apenas para ajudar aqueles usuários que tem certa dificuldade a fixar e lembrar mais o assunto e a sistema tutorial independente das questões, sempre que o usuário clicar em "Desisto" ou "Pausa" o sistema apresentaria um pequeno tutorial sobre o conteúdo da questão.

Figura 41 – Resultado do Questionário de Expectativa do Cosmo - Página 3

Prometo que estamos perto do fim, abaixo vou te apresentar algumas coisas em que estamos trabalhando e pedimos que você defina a ordem de desenvolvimento delas, isso mesmo, você é que define como iremos trabalhar :D. 1. Organização por Turma 2. Caça Bugs 3. RPG 4. Gamificação do cosmo

8 respostas

2 (2)

3412

2 3 1 4

3. RPG seria mais dinâmico

1,3,2,4

1>2>4>3

2,1,4,3

Agora é algo sendo bem abrangente, esteja livre para nos indicar novos requisitos que você deseja ver no cosmo e assim chegaremos ao fim do questionário.

8 respostas

Gostaria que ele fosse mais interativo e desse dicas

Quero só poder fazer as atividades passadas mesmo <3

Se fosse possível colocar dicas de comandos e exemplos práticos desses comandos.

Abertura para lançamentos de programas pelo usuário.

Um espaço para tirar dúvidas

Indicação onde está o erro no código.

Se gostarem da ideia do rankeado quero meus credits. r...nm...i

Opção para reportar bugs sobre determinada questão; breves tutoriais sobre a linguagem que podem ser usados para consulta antes ou depois que responder as questões; possibilidade de ver a resolução da questão, isso depois de muitas tentativas; talvez poderia colocar uma pontuação no Cosmo, que leva em consideração o tempo para realizar cada questão e a quantidade de questões feitas (excluindo aquelas que clicaram para ver a resposta)... Com isso, poderia fazer um ranking onde teria os nomes das pessoas que tiveram maior pontuação na turma slá, acredito que isso poderia ser bom já que vai ser algo mais competitivo e as pessoas irão querer ser melhores.

Figura 42 – Resultado do Questionário de Expectativa do Cosmo - Página 4

ANEXO D – Wireframe versão atual Cosmo

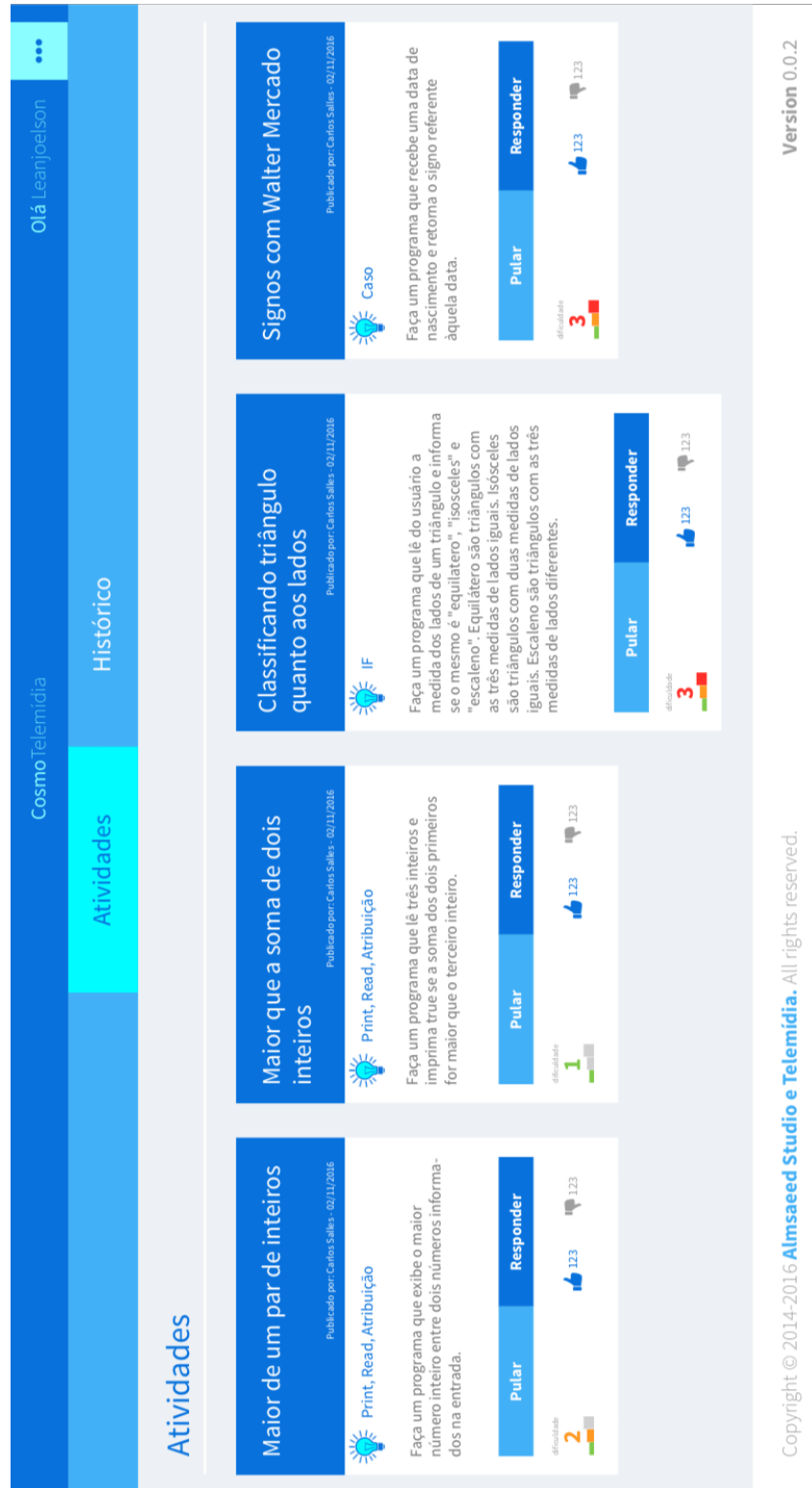


Figura 43 – Wireframe versão atual Cosmo - Página 1

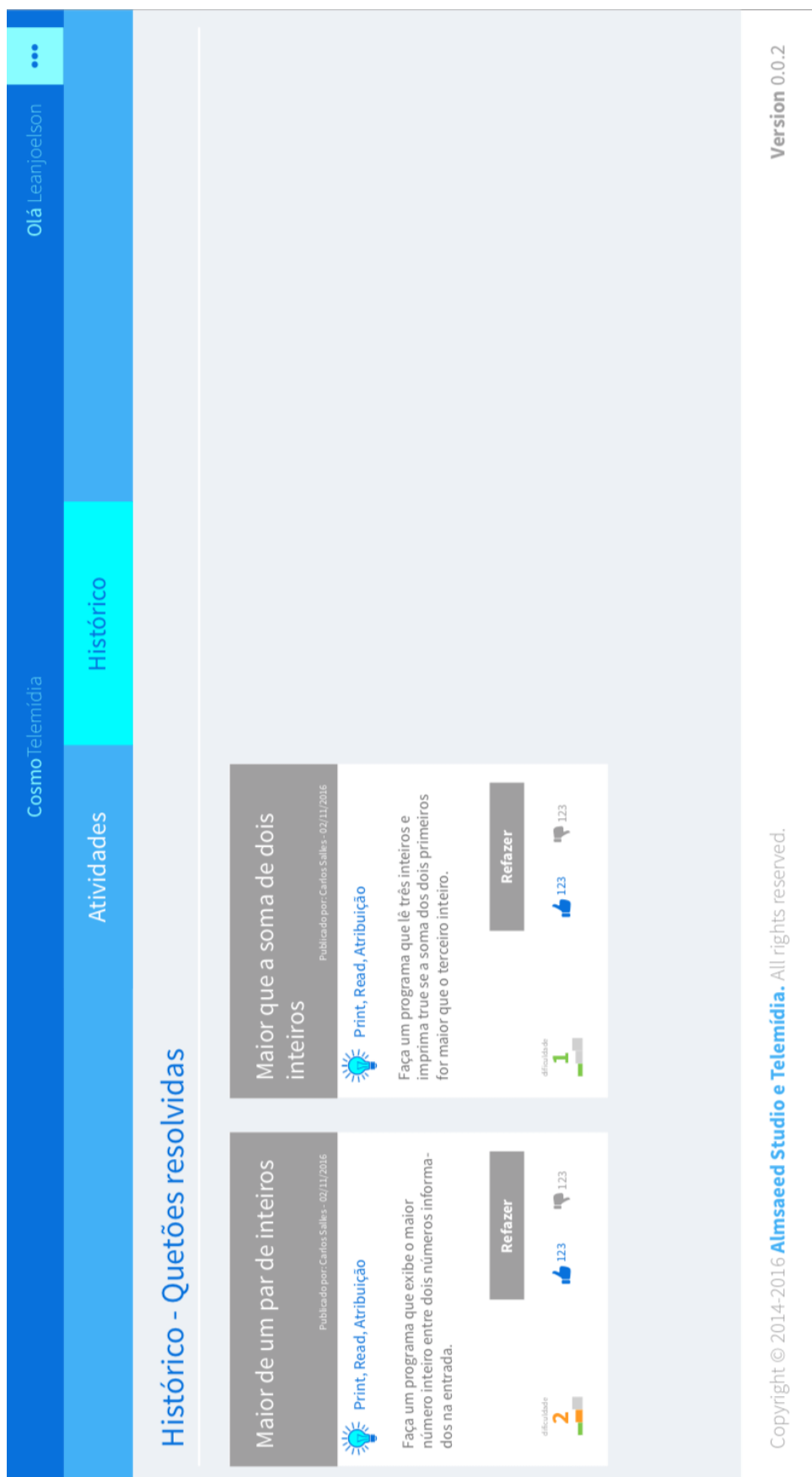


Figura 44 – Wireframe versão atual Cosmo - Página 2

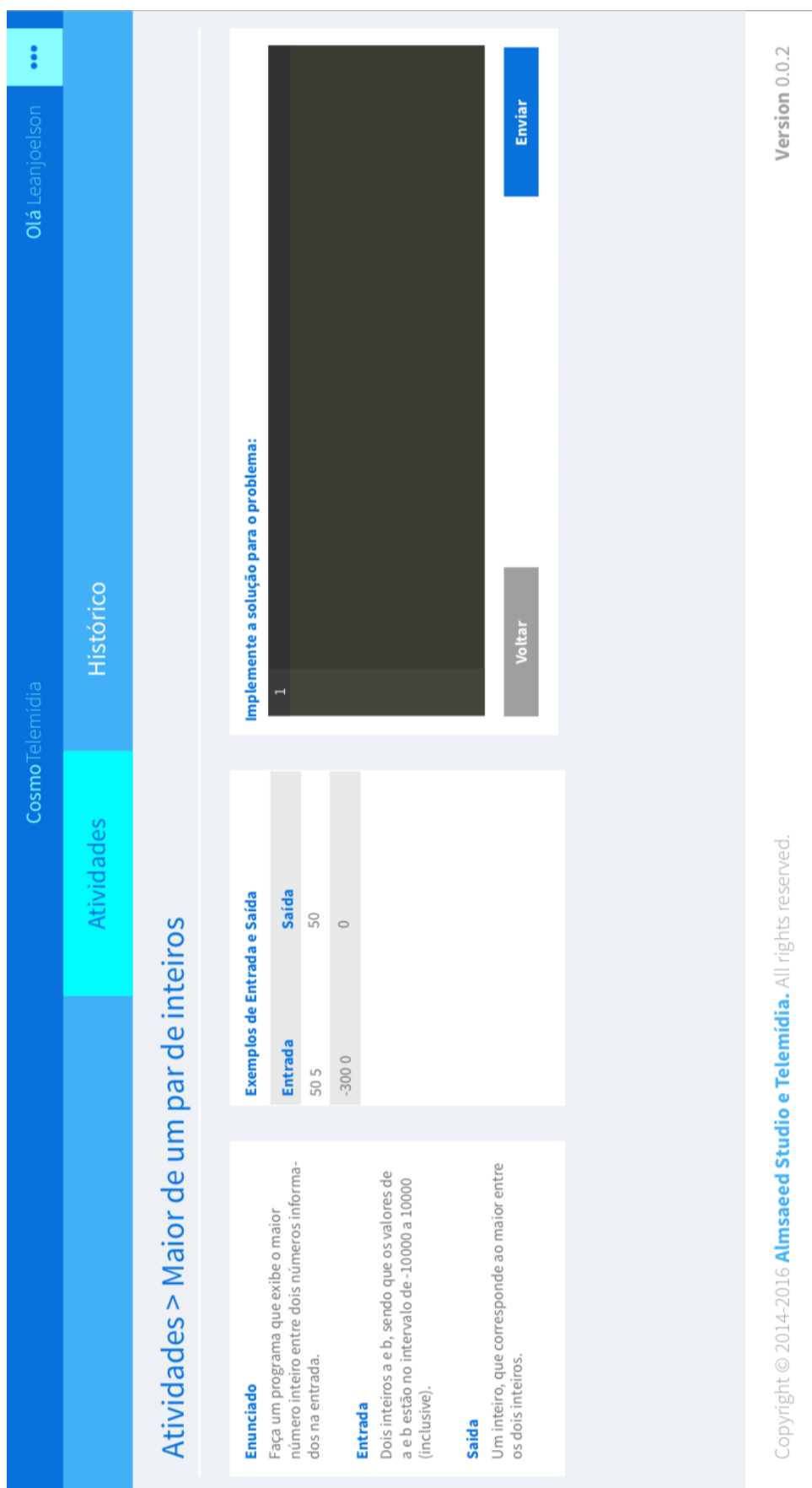


Figura 45 – Wireframe versão atual Cosmo - Página 3

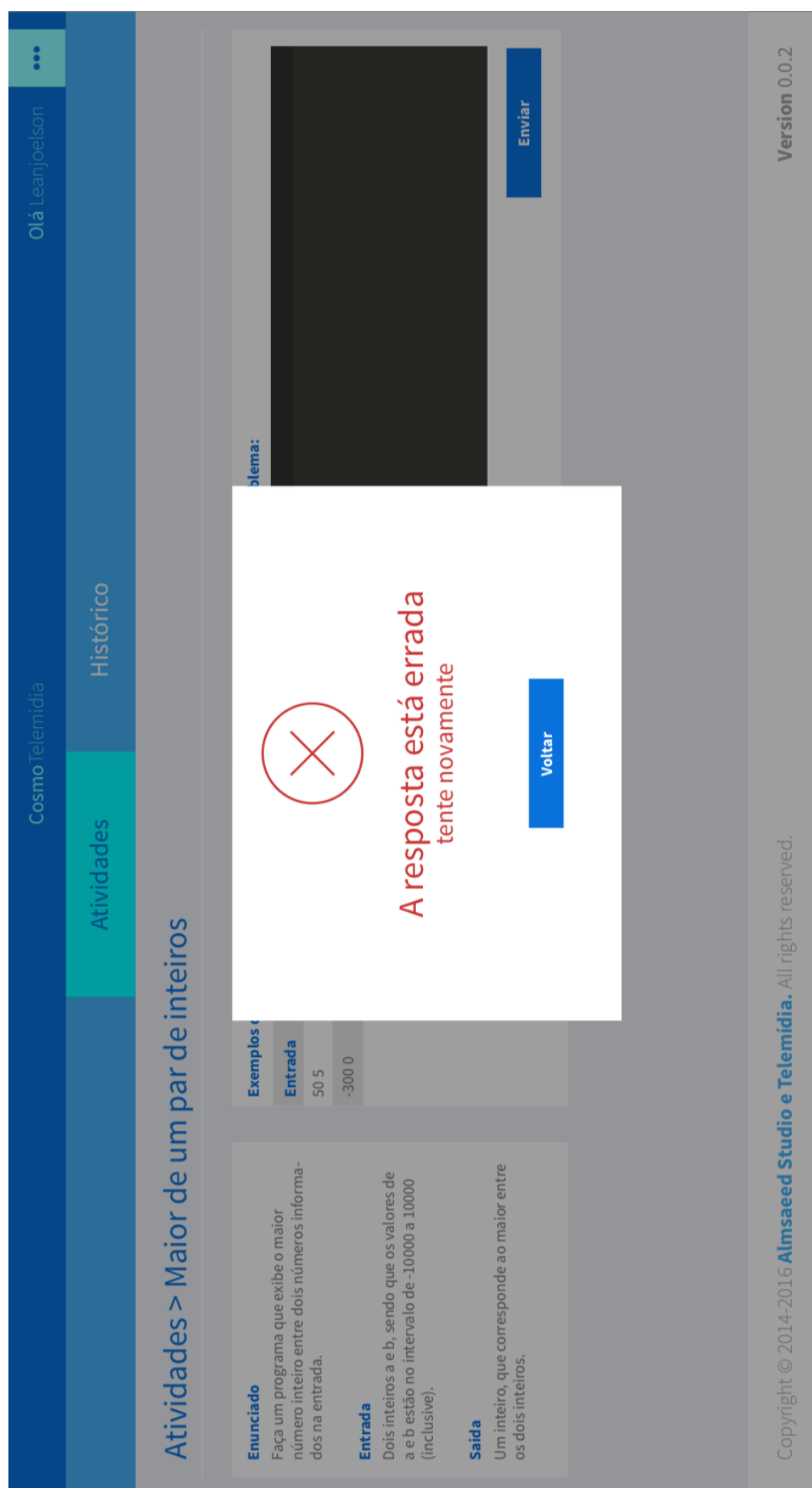


Figura 46 – Wireframe versão atual Cosmo - Página 4

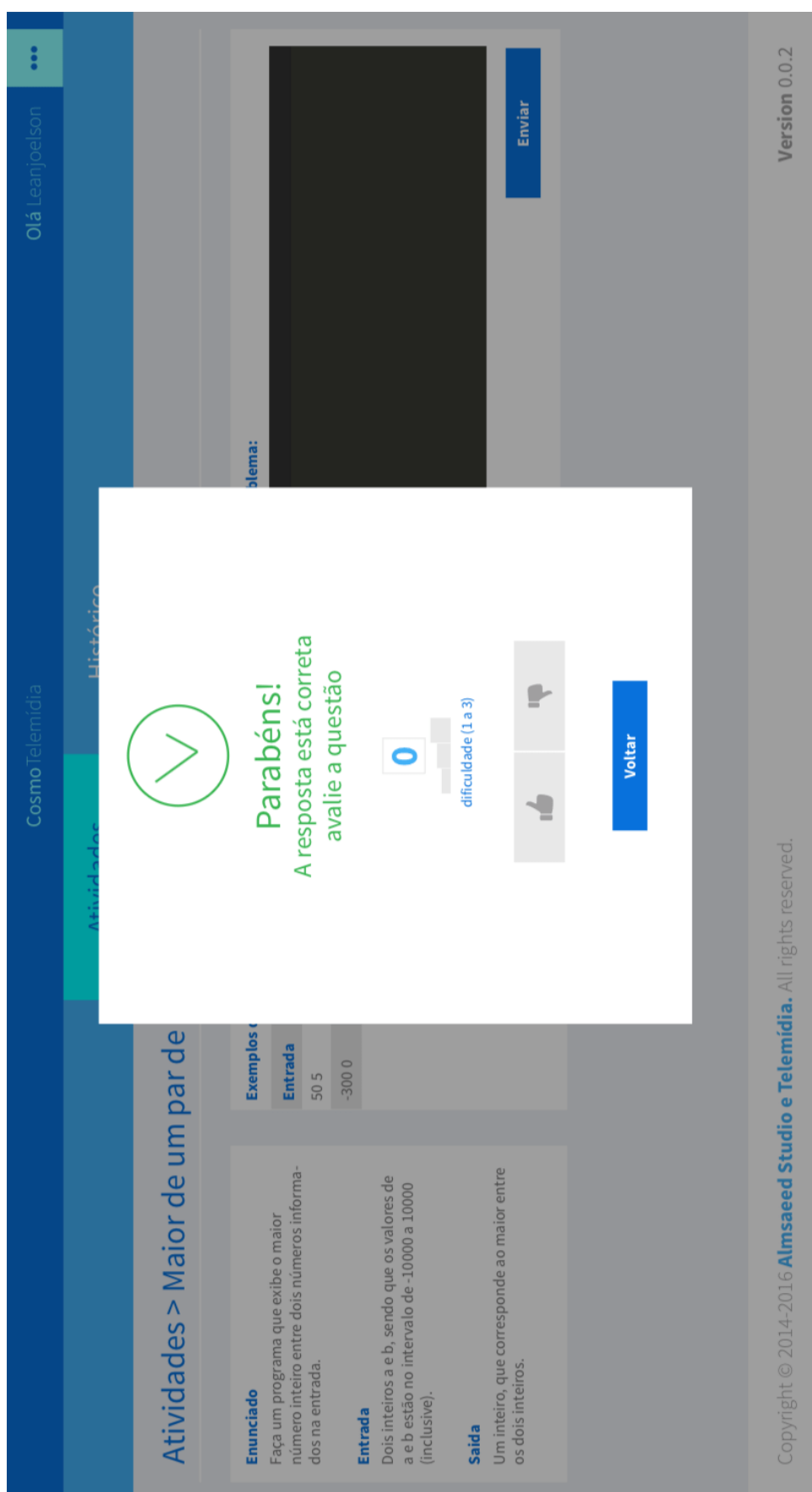


Figura 47 – Wireframe versão atual Cosmo - Página 5

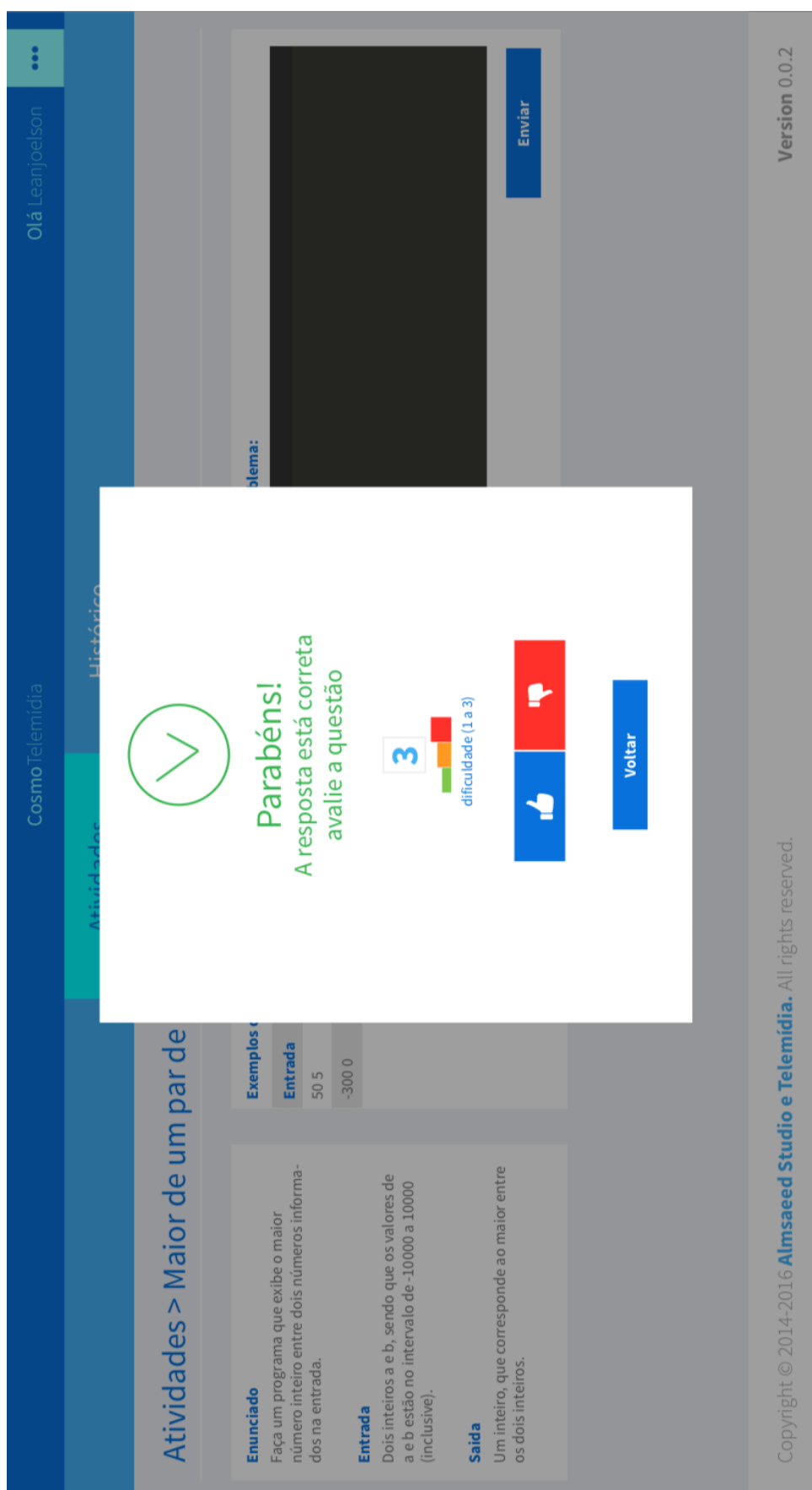


Figura 48 – Wireframe versão atual Cosmo - Página 6