

UNIVERSIDADE FEDERAL DO PARANÁ

JÚLIO DO LAGO MÜLLER

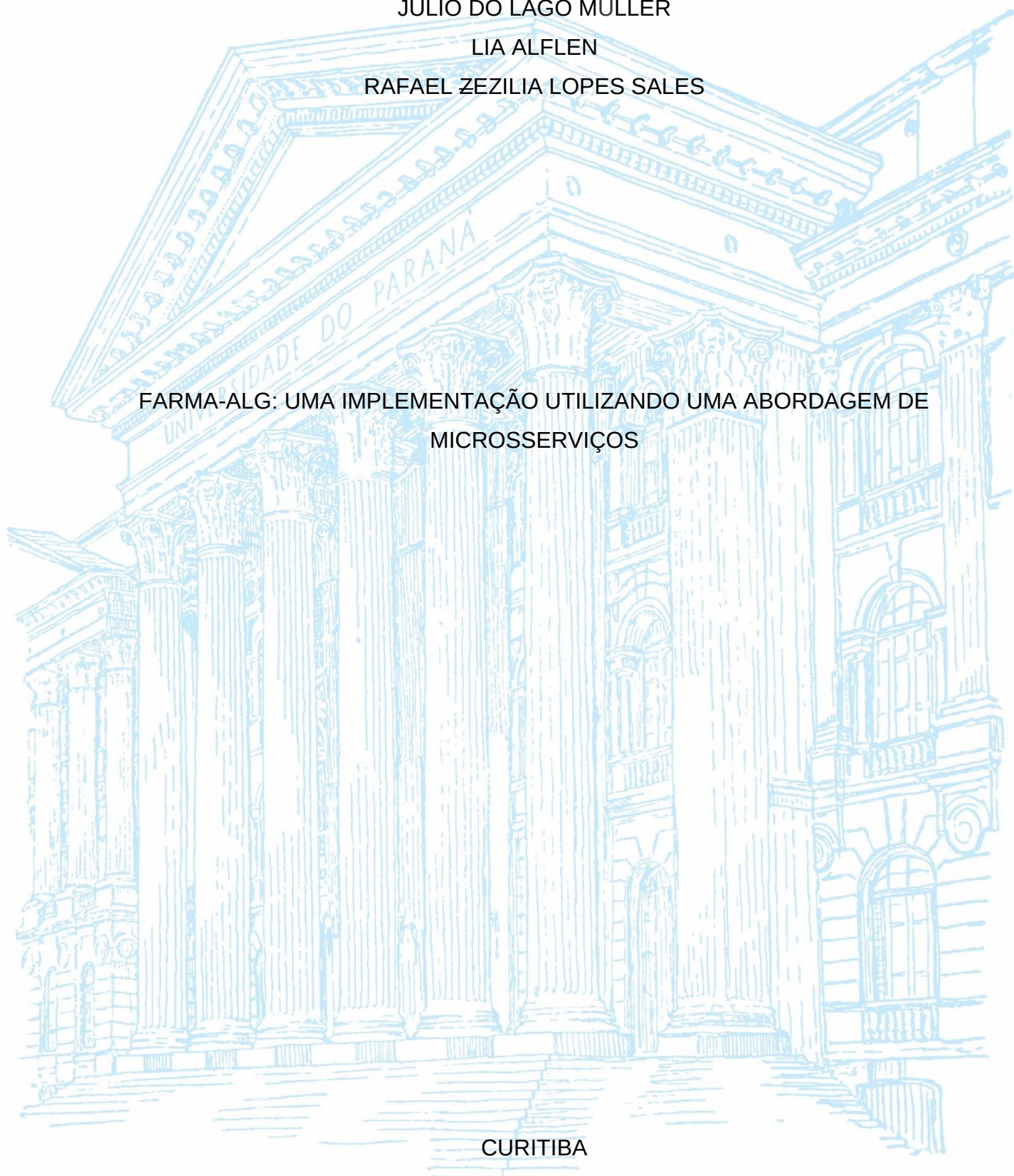
LIA ALFLEN

RAFAEL ZEZILIA LOPES SALES

FARMA-ALG: UMA IMPLEMENTAÇÃO UTILIZANDO UMA ABORDAGEM DE
MICROSSERVIÇOS

CURITIBA

2021



JÚLIO DO LAGO MÜLLER
LIA ALFLEN
RAFAEL ZEZILIA LOPES SALES

FARMA-ALG: UMA IMPLEMENTAÇÃO UTILIZANDO UMA ABORDAGEM DE
MICROSSERVIÇOS

Trabalho de conclusão de curso apresentado ao curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Alexander Robert Kutzke

CURITIBA
2021



UNIVERSIDADE FEDERAL DO PARANÁ

ATA DE REUNIÃO

TERMO DE APROVAÇÃO

JÚLIO DO LAGO MÜLLER

LIA ALFLEN

RAFAEL ZEZILIA LOPES SALES

FARMA-ALG: UMA IMPLEMENTAÇÃO UTILIZANDO UMA ABORDAGEM DE MICROSERVIÇOS

Monografia aprovada como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná.

Prof. Dr. Alexander Robert Kutzke
Orientador – SEPT/UFPR

Prof. Dr. João Eugênio Marynowski
SEPT/UFPR

Prof. Dr. Mario de Paula Soares Filho
SEPT/UFPR

Curitiba, 21 de dezembro de 2021.



Documento assinado eletronicamente por **MARIO DE PAULA SOARES FILHO, PROFESSOR ENSINO BAS/TEC/TECNOL**, em 21/12/2021, às 19:43, conforme art. 1º, III, "b", da Lei 11.419/2006.



Documento assinado eletronicamente por **JOAO EUGENIO MARYNOWSKI, PROFESSOR DO MAGISTERIO SUPERIOR**, em 21/12/2021, às 21:14, conforme art. 1º, III, "b", da Lei 11.419/2006.



Documento assinado eletronicamente por **ALEXANDER ROBERT KUTZKE, PROFESSOR DO MAGISTERIO SUPERIOR**, em 22/12/2021, às 13:27, conforme art. 1º, III, "b", da Lei 11.419/2006.



A autenticidade do documento pode ser conferida [aqui](#) informando o código verificador **4145411** e o código CRC **538AE933**.

AGRADECIMENTOS

Ao nosso sempre presente orientador, Prof. Dr. Alexander Robert Kutzke, que para além de nos guiar, também não mediu esforços para nos auxiliar nos diversos desafios encontrados, onde atuou conosco e pacientemente acreditou em nós.

Não deixando de agradecer a todos os nossos familiares, que nos acompanharam junto às dificuldades e aos curtos cronogramas e além disso, nos motivaram a seguir em frente todos os dias. O orgulho de ver este projeto pronto é delas também.

Aos nossos amigos, par romântico e pessoas especiais para nós, que nunca deixaram de nos cuidar ou de nos animar durante as diversas madrugadas para a realização deste projeto.

Ao Rafael Lammel Marinheiro, que acreditou neste projeto desde o primeiro instante e mostrou que todos os problemas têm saída.

RESUMO

O amadurecimento dos cursos técnicos e tecnológicos de desenvolvimento de software fez emergir necessidades tanto para docentes quanto para estudantes. Para os docentes, há a demanda de conhecer metodologias de ensino e de como compartilhar conhecimento e avaliar a efetividade das informações fornecidas de modo imediato, preciso e, cada vez mais, online. Do lado dos estudantes, demandam-se conteúdos alinhados com o mercado e de qualidade para auxiliar no seu desenvolvimento educacional e profissional. Diante destes principais pontos, a aplicação Farma-Alg propõe um ambiente para interação aluno-professor perante o ensino de lógica de programação e algoritmos, entregando ao aluno uma área de estudos e ao professor uma plataforma para acompanhar e mensurar a assimilação destes temas pelas suas turmas. Entretanto, a atual versão do Farma-Alg (2ª versão) possui interfaces complexas, realiza a execução de códigos de terceiros sem muita tratativa de segurança e tem um alto acoplamento como um todo, o que evidencia o déficit no quesito de usabilidade, segurança, padronização de desenvolvimento e manutenibilidade do sistema. Com estes principais pontos em mente, o principal objetivo deste trabalho de conclusão de curso é realizar a análise e reescrita do código-fonte do *software* para melhor adoção pela comunidade acadêmica da área e escalabilidade de recursos úteis no ensino e aprendizagem de programação.

Palavras-chave: Software Educacional. Lógica de Programação. Algoritmo. Desenvolvimento Web. Contêiner. Microsserviços.

ABSTRACT

The maturation of technical and technological courses on software development has created demands for both teachers and students. For teachers, there is a demand for teaching methodologies and how to share knowledge and assess the effectiveness of the information provided in an immediate, accurate and online way. On the students' side, quality content aligned with the market is demanded to assist in their educational and professional development. In face of these key points, the application Farma-Alg proposes an interaction environment for both teacher and student focused on programming logic and algorithms, providing the student with an area of studies and the teacher with a panel to monitor and measure the assimilation of these themes by their classes. However, Farma-Alg's current version (2nd version) has complicated interfaces, executes insecure codes without great security treatments and has high coupling, which does not reach adequate levels of satisfaction in aspects such as usability, security, maintenance and compliance with market practices. Regarding these main issues, this undergraduate thesis' main objective is recreating Farma-Alg, rewriting it's code focusing on the academic community's use and a better software scalability using useful resources for the teaching and learning of programming.

Keywords: Educational Software. Programming logic. Algorithm. Web Development. Container. Microservices.

LISTA DE FIGURAS

FIGURA 1 – INTERFACE DE INDEXAÇÃO DE CÓDIGO - FARMA-ALG 2ª VERSÃO.	19
FIGURA 2 – INTERFACE INICIAL - FARMA-ALG 1ª VERSÃO.	20
FIGURA 3 – INTERFACE DE COMPARAÇÃO DE CÓDIGO FONTE - FARMA-ALG 1ª VERSÃO.	21
FIGURA 4 – INTERFACE DE RANKING DOS EXERCÍCIOS - FARMA-ALG 2ª VERSÃO.	22
FIGURA 5 - ETAPAS DA METODOLOGIA CASCATA DE DESENVOLVIMENTO	26
FIGURA 6 – INTERFACE INICIAL DO CODEWARS.....	31
FIGURA 7 – INTERFACE DE SUBMISSÃO DOS EXERCÍCIOS DO CODERBYTE.	32
FIGURA 8 – INTERFACE DE INICIAL DO EXERCISM.	33
FIGURA 9 - ARQUITETURA GERAL DOS MÓDULOS DA APLICAÇÃO FARMA-ALG 3ª VERSÃO ..	35
FIGURA 10 – EXECUÇÃO ENCERRADA POR TIMEOUT	37
FIGURA 11 – PLANEJAMENTO KANBAN ATRAVÉS DO TRELLO.	39
FIGURA 12 - DIAGRAMA RELACIONAL.....	45
FIGURA 13 - DIAGRAMA DE CASOS DE USO.....	46
FIGURA 14 - DIAGRAMA DE CLASSES.....	47
FIGURA 15 — INTERFACE INICIAL DO SISTEMA - FARMA-ALG PROPOSTO.	50
FIGURA 16 – INTERFACE DE LOGIN DO SISTEMA - FARMA-ALG PROPOSTO.....	50
FIGURA 17 – INTERFACE DE LOGIN DO ESTUDANTE - FARMA-ALG PROPOSTO.	51
FIGURA 18 – INTERFACE DE LOGIN DO DOCENTE - FARMA-ALG PROPOSTO.	52
FIGURA 19 – INTERFACE DE USUÁRIOS- FARMA-ALG PROPOSTO.	52
FIGURA 20 – INTERFACE DE CRIAÇÃO DE TURMA - FARMA-ALG PROPOSTO.....	53
FIGURA 21 – INTERFACE DE DETALHES DE TURMA - FARMA-ALG PROPOSTO.	54
FIGURA 22 – INTERFACE DE CRIAÇÃO DE EXERCÍCIOS - FARMA-ALG PROPOSTO.	55
FIGURA 23 – INTERFACE DE SUBMISSÃO DE RESPOSTAS - FARMA-ALG PROPOSTO.	55
FIGURA 24 – DS001: CADASTRO DE USUÁRIO	104
FIGURA 25 – DS002: LOGIN DE USUÁRIO	105
FIGURA 26 – DS003: ATUALIZAR USUÁRIO	106
FIGURA 27 – DS004: TROCAR SENHA DE USUÁRIO	107
FIGURA 28 – DS005: REMOVER USUÁRIO	108
FIGURA 29 – DS006: GERENCIAR TURMAS (LISTAR)	109
FIGURA 30 – DS007: GERENCIAR TURMAS (CADASTRAR).....	110
FIGURA 31 – DS008: GERENCIAR TURMAS (ATUALIZAR)	111
FIGURA 32 – DS009: GERENCIAR TURMAS (REMOVER).....	112
FIGURA 33 – DS010: GERENCIAR EXERCÍCIOS (LISTAR PÚBLICOS)	113

FIGURA 34 – DS011: GERENCIAR EXERCÍCIOS (BUSCAR)	114
FIGURA 35 – DS012: GERENCIAR EXERCÍCIOS (CADASTRAR).....	115
FIGURA 36 – DS013: GERENCIAR EXERCÍCIOS (ATUALIZAR)	116
FIGURA 37 – DS014: GERENCIAR EXERCÍCIOS (REMOVER).....	117
FIGURA 38 – DS015: GERENCIAR EXERCÍCIOS (REPLICAR)	118
FIGURA 39 – DS016: CADASTRO DO ALUNO NA TURMA.....	119
FIGURA 40 – DS017: CADASTRAR RESPOSTA.....	120
FIGURA 41 – DS018: EXECUTAR RESPOSTA	121
FIGURA 42 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 1	122
FIGURA 43 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 2	123
FIGURA 44 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 3	124
FIGURA 45 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 4	125
FIGURA 46 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 5	126
FIGURA 47 – DOCUMENTAÇÃO DOS ENDPOINTS: PÁGINA 6	127

LISTA DE TABELAS

TABELA 1 - CRONOGRAMA DE DESENVOLVIMENTO DO PROJETO FARMA-ALG.....	49
TABELA 2 - UC001: SE CADASTRAR	63
TABELA 3 - UC002: LOGAR	65
TABELA 4 - UC003: TROCAR SENHA	66
TABELA 5 – UC004: ATUALIZAR CADASTRO	67
TABELA 6 – UC005: REMOVER CADASTRO	69
TABELA 7 - UC006: GERENCIAR TURMA.....	69
TABELA 8 - UC007: GERENCIAR EXERCÍCIO.....	74
TABELA 9 - UC008: GERENCIAR TAG	78
TABELA 10 - UC009: GERENCIAR CASO DE TESTE	79
TABELA 11 - UC010: EXECUTAR CASO DE TESTE	82
TABELA 12 - UC011: VISUALIZAR EXERCÍCIOS PÚBLICOS	85
TABELA 13 - UC012: COPIAR EXERCÍCIO PÚBLICO PARA UMA TURMA	85
TABELA 14 - UC013: SE INSCREVER TURMA.....	86
TABELA 15 - UC014: CRIAR RESPOSTA	88
TABELA 16 - UC015: EXECUTAR RESPOSTA.....	90
TABELA 17 - UC016: VALIDA CASO TESTE.....	91
TABELA 18 - UC017: CANCELAR INSCRIÇÃO TURMA	91
TABELA 19 - UC018: VISUALIZAR EXERCÍCIOS TURMA	92
TABELA 20 - UC019: VISUALIZAR TURMAS INSCRITAS	93
TABELA 21 - UC020: GERENCIAR INSTITUIÇÃO	93
TABELA 22 - UC021: GERENCIAR CONTATOS INSTITUIÇÃO	96
TABELA 23 - UC022: CADASTRAR USUÁRIO	99
TABELA 24 - UC023: TROCAR SENHA DE USUÁRIO.....	100
TABELA 25 - UC024: GERENCIAR LINGUAGEM	101

LISTA DE ABREVIATURAS

API	- <i>Application Programming Interface</i> (interface de programação de aplicação)
DR	- Diagrama Relacional
HTTP	- <i>Hypertext Transfer Protocol</i> (protocolo de transferência de hipertexto)
IDE	- <i>Integrated Development Environment</i> (Ambiente de Desenvolvimento Integrado)
JVM	- <i>Java Virtual Machine</i> (máquina virtual Java)
MVC	- <i>Model-View-Controller</i> (Modelo-Visualização-Controlador)
NoSQL	- <i>Not-only SQL</i> (não apenas SQL)
ORM	- <i>Object-Relational Mapping</i> (mapeamento objeto-relacionamento)
PNAD	- Pesquisa Nacional por Amostra de Domicílios
REST	- <i>Representational State Transfer</i> (Transferência de Estado Representacional)
SEO	- <i>Search Engine Optimization</i> (otimização para motores de busca)
SPA	- <i>Single Page Application</i> (aplicação de uma única página)
SQL	- <i>Structured Query Language</i> (linguagem estruturada de consulta)
SSG	- <i>Static Site Generation</i> (geração de páginas estáticas)
SSR	- <i>Server-Side Rendering</i> (renderização do lado servidor)
TADS	- Tecnologia em Análise e Desenvolvimento de Sistemas
UFPR	- Universidade Federal do Paraná
UML	- <i>Unified Modeling Language</i> (Linguagem de Modelagem Unificada)
URI	- <i>Uniform Resource Identifier</i> (identificador uniforme de recurso)

SUMÁRIO

1 INTRODUÇÃO	12
1.1 PROBLEMA	13
1.2 OBJETIVOS	14
1.2.1 Geral	14
1.2.2 Específicos	14
1.3 JUSTIFICATIVA	15
1.4 ORGANIZAÇÃO DO TRABALHO	15
2 FUNDAMENTAÇÃO TEÓRICA	17
2.1 ENSINO DE PROGRAMAÇÃO	17
2.2 FARMA-ALG	18
2.2.1 Implementação do Farma-Alg 2ª	22
2.2.2 Alto Acoplamento	23
2.3 PLANEJAMENTO E DESENVOLVIMENTO DE SOFTWARE	24
2.3.1 Modelagem	24
2.3.2 Metodologia	24
2.3.3 Microserviços	27
2.3.4 Containerização	28
2.4 APLICAÇÕES WEB	29
2.5 APLICAÇÕES SEMELHANTES	30
2.5.1 CodeWars	31
2.5.2 Coderbyte	32
2.5.3 Exercism	32
3 MATERIAIS E MÉTODOS	35
3.1 FUNDAMENTAÇÃO	35
3.1.1 Virtualização de Ambiente	36
3.2 METODOLOGIA DE DESENVOLVIMENTO	38
3.3 FERRAMENTAS DE DESENVOLVIMENTO	39
3.3.1 Docker	39
3.3.2 Java	40
3.3.3 Git	40
3.3.4 GitHub	41
3.3.5 Node.js	41
3.3.6 PostgreSQL	42

3.3.7 React.js	42
3.3.8 REST Web Service	43
3.3.9 Spring	43
3.3.10 TypeScript	43
3.3.11 Swagger	44
3.4 MODELAGEM DE PROJETO	44
3.4.1 Modelagem de Dados	44
3.4.2 Modelagem de Sistema.....	45
3.5 PLANEJAMENTO DE DESENVOLVIMENTO	47
3.6 DESENVOLVIMENTO DO PROJETO	48
4 APRESENTAÇÃO DO SISTEMA	49
5 CONSIDERAÇÕES FINAIS	56
REFERÊNCIAS	58
APÊNDICE A – ENTREVISTA COM O MANTENEDOR	61
APÊNDICE B – ESPECIFICAÇÃO DE CASOS DE USO	63
APÊNDICE C – DIAGRAMA DE SEQUÊNCIA	104
APÊNDICE D – DOCUMENTAÇÃO DOS ENDPOINTS	122

1 INTRODUÇÃO

Em um mundo cada vez mais conectado, as relações humanas se diversificam, extrapolando a dimensão física para existir também no virtual. Isso extingue muitas dificuldades e, ao mesmo tempo, introduz novas.

Uma das barreiras extintas é a do alcance de indivíduos à educação e às informações das mais diversas áreas do conhecimento. O que antes era restrito a livros e salas de aulas, passou a poder ser consumido através das telas de diferentes dispositivos tecnológicos. Entretanto, mesmo essas novas possibilidades trazem consigo desafios já conhecidos, como a constante troca de informações entre educador e aluno, que muitas vezes se limita a uma comunicação unidirecional e vazia de pessoalidade.

Com base no relatório de 2019 emitido pela Pesquisa Nacional por Amostra de Domicílios (PNAD), cerca de 20% dos 50 milhões de brasileiros entre 14 a 29 anos não completaram alguma das etapas da Educação Básica. Dentre estes, a alarmante taxa de 29,2% é representada pelas pessoas que deixaram de frequentar o ambiente escolar por “falta de interesse”, destacando-se como uma das principais causas da evasão escolar.

Este enredo, muito comum no cenário brasileiro, é abordado na obra de Patto (1996), declarando-o como fracasso escolar. O autor diz que o aluno após cometer um erro permanece submerso no mesmo, submetido a um efeito “bola de neve”, em que a dificuldade, com o passar do tempo, gera tanto um desconforto quanto uma reclusão do indivíduo diante do desenvolvimento das aulas e da incapacidade do discente de acompanhá-las.

Este fator simboliza não apenas o erro originário que acometeu o aluno às dificuldades, mas também a falta de atenção dada ao seu erro, considerando que o fator do erro é inerente ao ser humano e ao processo de aprendizagem. Com o auxílio do professor, todavia, a ocorrência de erros tenderia a reduzir, se bem destacadas e acompanhadas. Porém tal situação, como esclarecida por Patto (1996) ainda permeia a realidade atual, infligindo a concepção do ambiente escolar, para a infeliz realidade do fracasso na escola.

Assim, combinar métodos de ensino e as tecnologias para a superação destas dificuldades se torna primordial para que a qualidade do ensino não seja prejudicada e com isso, agrave ainda mais a situação educacional no país.

O sistema educacional Farma-Alg elaborado por Kutzke (2015) surgiu com o intuito de suprir essa demanda para o ensino de lógica de programação e algoritmos na UFPR. Além de ser um canal *online*¹ que permite que estudantes escrevam e executem códigos-fonte, ele é uma plataforma de análise individual, que gera estatística dos resultados para os docentes, permitindo-lhes identificar tendências no aprendizado de programação e acompanhar a evolução e dificuldades dos discentes durante as realizações das atividades.

1.1 PROBLEMA

O projeto Farma-Alg foi concebido e proposto por Kutzke, (2015) como tese de doutorado acerca da informática na educação e formas de medição de erros no ensino de lógica de programação e algoritmos. Esta primeira implementação do Farma-Alg é referenciada no presente trabalho como a primeira versão do sistema. Em 2018, a ferramenta foi objeto de uma nova dissertação de mestrado (EHRESMANN, 2018) com foco em aspectos de gameificação. Esta contribuição teve impactos voltados ao campo da pesquisa, mas também acabou sendo incorporada na versão que atualmente está sendo utilizada pelos professores e estudantes da UFPR. Essa implementação incremental realizada pelo Ehresmann será tratada como Farma-Alg 2ª versão, neste trabalho, a fim de facilitar a compreensão, além de obedecer a ordem cronológica de desenvolvimento.

A ferramenta atendeu inicialmente às finalidades de pesquisa às quais foi submetida. Porém, seu projeto não foi desenvolvido levando-se em conta práticas de mercado e de melhoria contínua. Assim, com o passar do tempo, a versão implantada do Farma-Alg deixou de ser submetida a atualizações, tanto de correções de mal funcionamentos como de incrementos de funcionalidades para se adequar às necessidades de seu público-alvo ou de novas tecnologias. Isso ficou especialmente demonstrado após entrevista com o mantenedor do projeto (KUTZKE, 2021), conforme evidenciado no apêndice A.

¹ Online: Algo ou alguém que esteja conectado à internet.

A principal consequência dessa falta de planejamento do *software*² se refere principalmente à complexidade excessiva de seus componentes e, portanto, às dificuldades de manutenção e refatoração do código-fonte, a fim de adequá-lo a uma arquitetura ou a conceitos de engenharia de *software* mais modernos.

E, por ser um sistema de interesse pedagógico e acadêmico, o Farma-Alg precisa se adaptar às necessidades de seus usuários, sejam eles professores ou estudantes. A proposta desta iniciativa é revisitar a ideia inicial do projeto e reimplementá-lo fazendo uso de práticas de desenvolvimento modernas e sustentáveis a fim de garantir longevidade e escalabilidade do *software*.

1.2 OBJETIVOS

O objetivo geral e os objetivos específicos estão descritos a seguir.

1.2.1 Geral

Reimplementar parcialmente o projeto Farma-Alg com técnicas e tecnologias atualizadas de desenvolvimento de *software*, enfocando no uso prático de microsserviços, a fim de garantir a longevidade, manutenibilidade e escalabilidade do sistema, e de permitir que este continue útil ao ensino de lógica de programação e algoritmos tanto para a comunidade da UFPR, quanto para diferentes instituições que possuam interesse no uso.

1.2.2 Específicos

- a) Analisar as funcionalidades e características positivas do sistema existente e reproduzi-las na nova versão;
- b) Analisar as falhas e dificuldades do sistema existente;
- c) Propor uma nova modularização do sistema em partes menores (microsserviços) para favorecer sua manutenibilidade;

² Software: Programa de computador.

- d) Fazer uso de ferramentas e tecnologias recentes para aumentar a vida útil da aplicação, incrementar a produtividade no desenvolvimento e facilitar a manutenção do código-fonte;
- e) Garantir a independência e a segurança na execução de códigos-fonte não-confiáveis;
- f) Aperfeiçoar a interface do usuário para garantir uma melhor experiência com os recursos do sistema;

1.3 JUSTIFICATIVA

O Farma-Alg é uma ferramenta com recursos promissores (AINSWORTH, 2006). Os conceitos pelos quais foi concebido e o alinhamento com as tendências no ensino de lógica de programação e algoritmos a jovens são evidências de uma fundamentação sólida e contemporânea.

Todavia, a arquitetura do sistema e as metodologias de engenharia de *software* foram deixadas em segundo plano por não serem o enfoque da proposta inicial, trazendo complexidade e dificuldades na manutenção do código-fonte.

Desta maneira, a principal motivação deste trabalho se dá na forma e na implementação do projeto original, a fim de melhorar a vida útil do *software* e permitir que outros times de pesquisa e de desenvolvimento possam colaborativamente contribuir na manutenção do código-fonte, na correção de falhas e no aperfeiçoamento de usabilidade e de funcionalidades conforme os usuários tenham suas necessidades modificadas, seja por demanda do mercado, seja por métodos de ensino de programação mais atualizados ou simplesmente pela praticidade do ensino da linguagem.

1.4 ORGANIZAÇÃO DO TRABALHO

Durante a decorrência desta dissertação, será abordada a fundamentação teórica e os seus respectivos subtópicos, tratando de expressar estrutura e a base principal para o desenvolvimento do projeto Farma-Alg em sua 3ª versão. A pertinência neste tópico denominado fundamentação teórica, se resume em abordar a motivação do seu desenvolvimento, conceitos bases e fundamentais para um projeto deste nível de atuação educacional, relatar o conceito de modularização em

microsserviços e containerização pertinentes a segurança, observando a futura manutenção, praticidade para atualização e respeitando a qualidade de desenvolvimento do software.

Além de relatar conceitos voltados a aplicações *web*, sua estruturação e pertinência no mundo vigente, realiza a apresentação de suas versões anteriores, bem como os projetos semelhantes, tendo como questão principal expressar as qualidades e características mais relevantes dos *softwares*, relatando os pontos de melhorias necessárias para a adequação com as exigências do projeto Farma-Alg em sua 3ª versão.

Em sua continuidade é tratado do tópico referente aos materiais e métodos, onde se acentua a arquitetura elaborada na 3ª versão do Farma-Alg, especificando os conceitos de virtualização através da containerização em respeito aos microsserviços e a segurança, enquanto apresenta a metodologia de desenvolvimento de *software* aplicada neste projeto, o seu planejamento e os conceitos das tecnologias principais utilizadas neste sistema. Além de relatar a modelagem do projeto, seguindo as notações de desenvolvimento da Linguagem de Modelagem Unificada (UML).

Por continuidade, é retratado no tópico de apresentação do sistema, as principais interface do Farma-Alg em sua 3ª versão, contemplando os principais casos de uso, durante uma abordagem rotineira de criação de exercícios ou de turma e de seu respectivo acesso para execução, além de apresentar os casos de teste para o exercício e o seu retorno.

Diante do quinto tópico, é retratado as considerações finais em que se apresenta a conclusão do trabalho, indicações de atividades futuras que possam ser implementadas e comentários da equipe sobre a atividade de desenvolvimento como um todo. Logo a mais, é descrito as referências de artigos, sites, livros e projetos semelhantes que nos auxiliaram no planejamento e desenvolvimento deste trabalho, deixando por fim, os apêndices que são por suma, figuras, tabelas e diagramas, como os diagramas de sequência, tabelas especificando os casos de uso, apresentação da entrevista com o mantenedor do Farma-Alg em sua 2ª versão e a documentação dos pontos para requisições.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda questões fundamentais que guiarão a orientação e decisões ao longo deste trabalho. Primeiramente, é exposta a principal motivação de se investir tempo e esforços na criação de um *software* educacional e a literatura pertinente. Logo em seguida, o próprio sistema Farma-Alg é apresentado de forma geral, levando em conta principalmente a tese de doutorado que o concebeu.

Na sequência, uma breve revisão literária acerca de arquitetura e aplicações *web*³ embasará as escolhas feitas ao longo da pesquisa, seguido de uma breve seção ressaltando *softwares* de funcionalidade ou de objetivos similares pertinentes ao objeto de estudo deste trabalho.

2.1 ENSINO DE PROGRAMAÇÃO

A formação lógica para o desenvolvimento profissional e cognitivo das pessoas que optam por um curso na área de computação é fundamental para a criação de bases sólidas que irão guiar sua maneira de pensar e de solucionar problemas.

Pesquisadores como Pereira (2013) apoiam e defendem o ensino de programação desde a infância no ambiente escolar, já iniciando no Ensino Fundamental, junto com outras matérias como Português, Matemática e Ciências. O tema é tão relevante para a estruturação tecnológica do país que para isso foi criado o Workshop⁴ de Educação em Computação (WEI) e o Simpósio Brasileiro de Informática Educativa (SBIE), que abordam, entre outros temas, metodologias, resultados e ideias que geram e fomentam o ensino de programação no Brasil.

No ambiente em que vivemos, é cada vez maior a demanda por profissionais capacitados para atuar na área de programação e afins. Por isso, a área de ensino de programação não é trivial, e tampouco pode ser estática. E, mesmo pertencendo à área de conhecimento das exatas, as metodologias, paradigmas e abordagens de se desenvolver *softwares* e soluções só crescem, e aumentam também sua complexidade como apresentado por Terceiro (2012).

³ Web: Relativo à rede de internet.

⁴ Workshop: Seminário ou curso intensivo.

Por consequência disso, a criatividade e a resiliência dos educadores destas áreas são colocados à prova diante da necessidade de manter-se em constante evolução. Cada tema ou disciplina possui suas próprias especificidades e podem exigir uma diferenciação nos métodos de ensino.

A complexidade no que diz respeito ao ensino da programação, tanto para o desenvolvimento da didática, quanto para o aprendizado do aluno, traz consigo um grande número de evasão dos cursos como Ciência da Computação e Análise de Sistemas, fato este que tem uma ligação direta com as dificuldades de aprendizagem (CASTRO, et al., 2003) (HOED, 2017).

2.2 FARMA-ALG

O Farma-Alg nasceu com o pressuposto de desenvolver a qualidade de ensino de programação, através de uma abordagem intrínseca entre as dificuldades encontradas na aprendizagem e o retorno das experiências obtidas nesse processo. A sua origem se dá inspirada no estudo e pesquisas de uma tese de doutorado, denominada “FARMA: Uma Ferramenta de Autoria para Objetos de Aprendizagem de Conceitos Matemáticos”, (MARCZAL, 2015) que abordava o estudo do erro no âmbito matemático escolar e uma abordagem para lidar com tal dificuldade, onde se principia o conceito de análise do aprendizado por parte do professor.

Devido a semelhança dentre a matemática e a programação, através de uma tese de doutorado (KUTZKE, 2015), foi reformulado a tese anterior e ajustada para o ambiente de lógica de programação, tendo uma visão mais direcionada para a análise do aprendizado através da mediação do erro dos estudantes.

Tal análise se deriva da afinidade entre o estigma do erro e o aprendizado, apresentando um novo ponto de vista para a abordagem dos erros levantados pelo aluno, onde cada emissão de atividade encaminhada pelo aluno, será tratada pelo sistema Farma-Alg, que irá executar o código encaminhado pelo aluno, tentando observar se através de entradas pré-determinadas, o resultado será condizente com o valor previamente cadastrado pelo professor.

O sistema Farma-Alg em sua primeira versão atua de forma *online*, recebendo e gerindo os códigos de alunos em diferentes linguagens, sendo comumente utilizada a linguagem C e a linguagem Pascal, respeitando a área de estudo de cada disciplina.

Na Figura 1, demonstra-se uma tela do *software*, em que pode-se notar através das guias de retorno, os conceitos de Turma, Questão, Listas de Exercícios e Respostas.

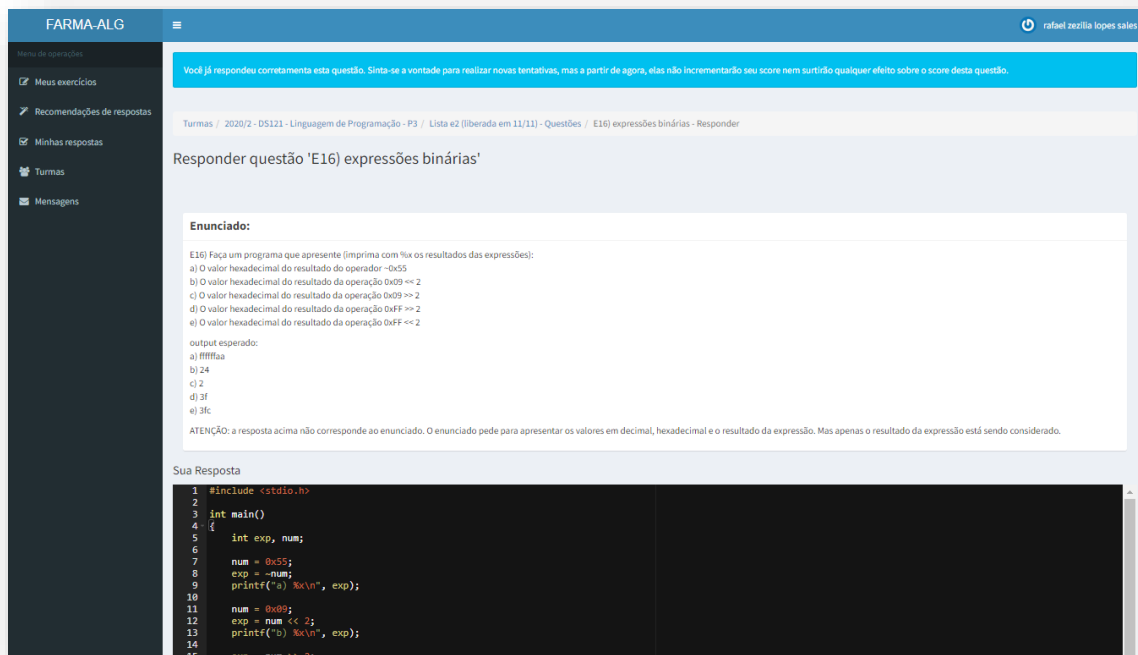


Figura 1 – Interface de indexação de código - Farma-Alg 2ª Versão.

O Farma-Alg promove um ambiente de desenvolvimento integrado (IDE) de forma *online*, organizada e segura, aos alunos e aos docentes. Através desta plataforma com o perfil de Professor, é possível não apenas cadastrar turmas e exercícios, mas acessar todas as tentativas de respostas entregues, inclusive as que geraram erros, facilitando a análise das maiores dificuldades das turmas.

Já as interfaces voltadas ao discente contém a opção de acesso às turmas inscritas, descrição da turma, “exercício exemplo” e de exercícios inseridos pelo professor, com as dicas que irão ser apresentadas, conforme o encaminhamento do código fonte para a realização da tarefa.

O Farma-Alg foi desenvolvido como uma ferramenta para o ensino de lógica de programação, e enquanto atua exerce um papel fundamental para o desenvolvimento do aluno e do professor, apresentando para ambos de forma mensurável: relatórios, pontuação e até mesmo grafos (entretanto com a ressalva de que grafos não são propostos neste trabalho) relativos ao aprendizado.

O sistema apresenta além disso, de forma concisa, as dificuldades do aluno que por vezes, podem ser reprimidas, esquecidas ou até mesmo subjugadas, criando um círculo vicioso que em ambientes futuros pode acarretar prejuízos para o aprendizado. Ademais para o docente, pode-se verificar se a sua metodologia é válida e qual o efeito dela no desenvolvimento acadêmico do aluno.

Estas foram as necessidades tratadas na primeira versão do Farma-Alg (KUTZKE, 2015), expressa pela Figura 2, que possui o foco principal na medição de erro através de estudo das dificuldades do aluno e do enfoque na tratativa de uma solução.

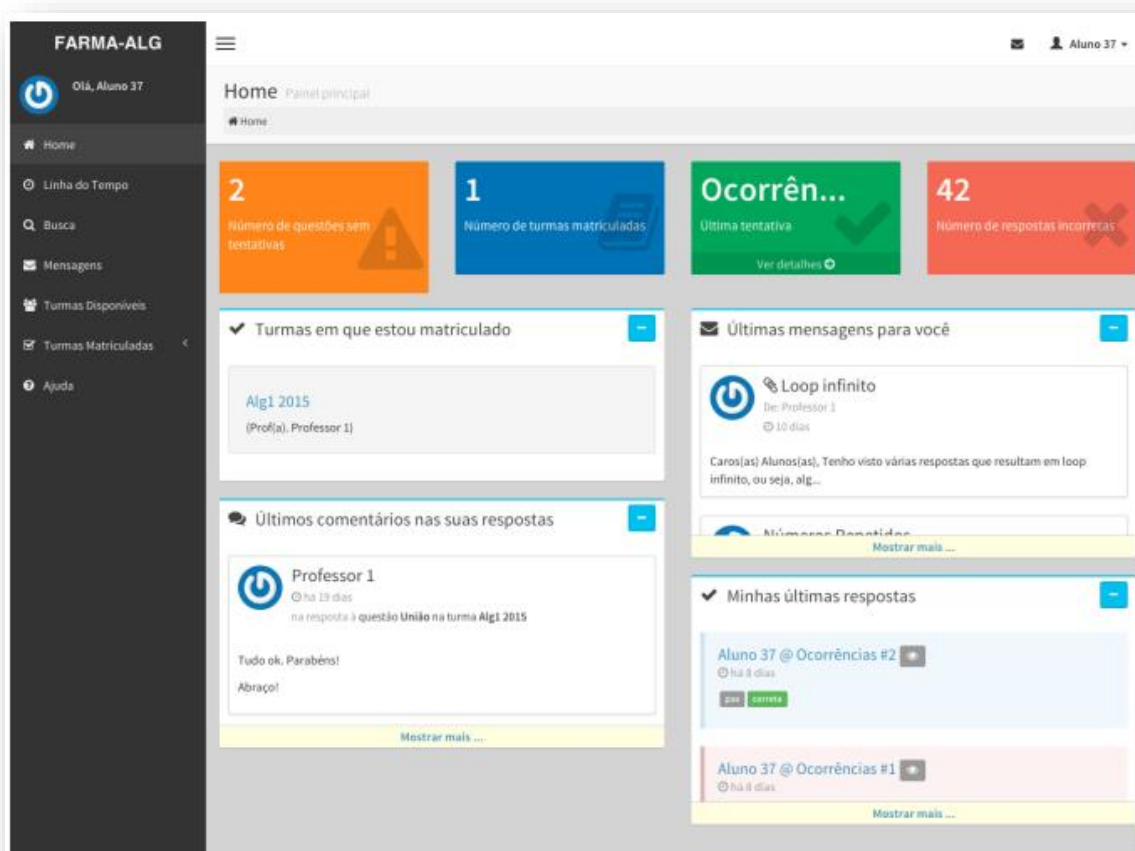


Figura 2 – Interface Inicial - Farma-Alg 1ª Versão.

Um dos diferenciais desta 1ª versão é expresso através de sua capacidade de diferenciação de respostas semelhantes: por meio de comparação tanto de respostas quanto de resultados pode-se visualizar códigos lado a lado, exaltando suas diferenças, como mostrado na Figura 3.

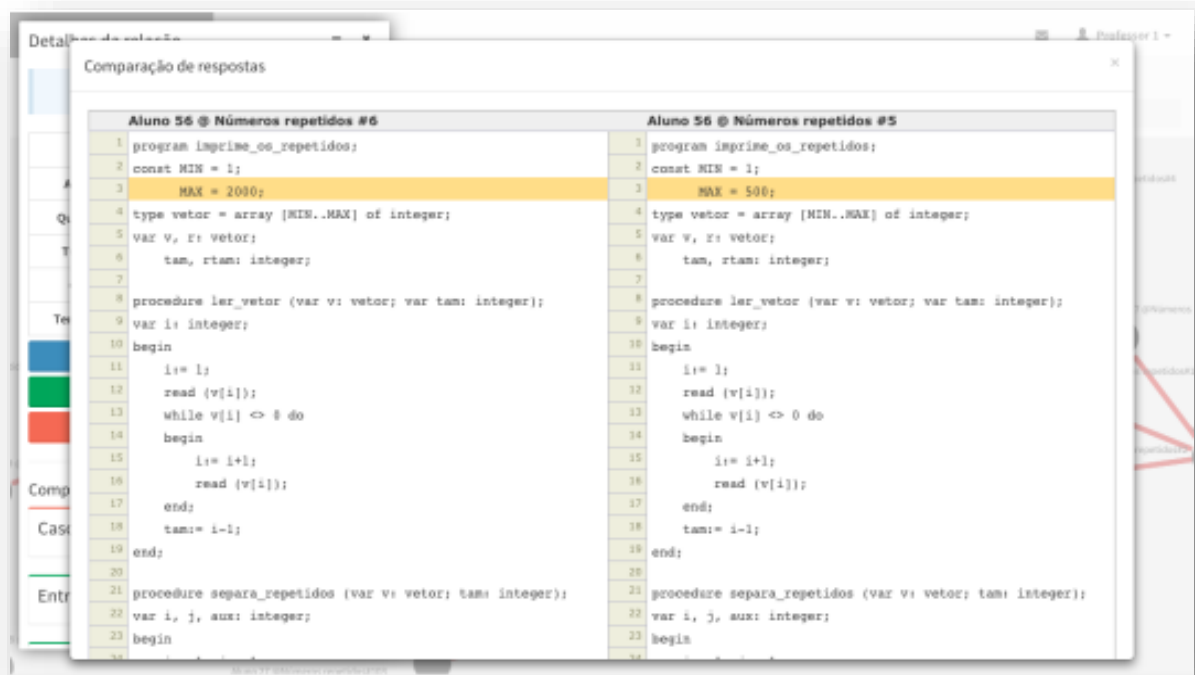


Figura 3 – Interface de comparação de código fonte - Farma-Alg 1ª Versão.

Já na 2ª versão do Farma-Alg houve a proposta e implementação de um sistema de gamificação, com o intuito de incentivar os estudos dos alunos e torna-lo mais instigante. Foi desenvolvido um sistema de placar, que expõe os alunos que obtiveram a maior quantidade de acertos nos exercícios, apresentando o valor através de uma interface de podium, como apresentado na Figura 4.

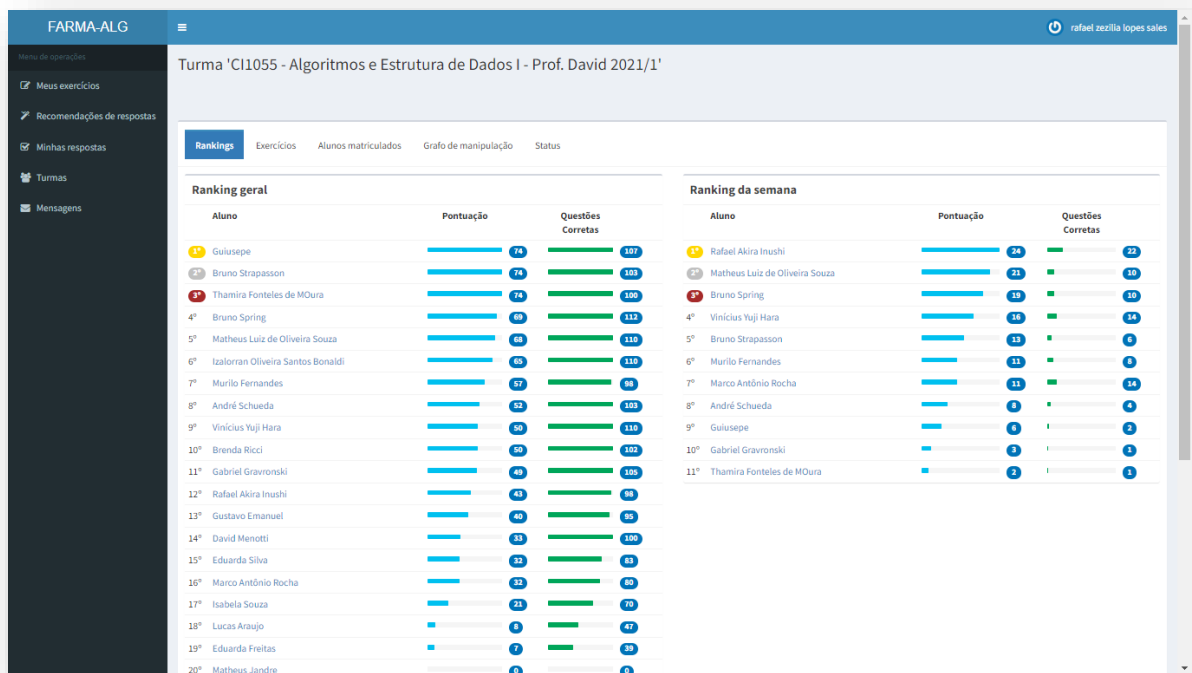


Figura 4 – Interface de ranking dos exercícios - Farma-Alg 2ª versão.

2.2.1 Implementação do Farma-Alg 2ª

A versão vigente do Farma-Alg (2ª versão), em execução nos servidores da UFPR e utilizada pelos professores do curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS), foi desenvolvida inteiramente na versão 2 da linguagem de programação *Ruby*⁵, que é uma linguagem interpretada multi-paradigma, com tipagem forte e dinâmica para desenvolvimento principalmente de aplicações *web* (ruby-lang.org, visitado). Também foi usado o *framework*⁶ *Ruby on Rails*, em sua versão 3.2, que é o principal *framework* de criação de *softwares* da linguagem, proporcionando aos desenvolvedores extenso ferramental para implementação de recursos comuns em sistemas *web*, como módulos de segurança, validação, conexão com o banco de dados, roteamento de chamadas HTTP ao servidor e confecção de páginas dinâmicas.

Como o Rails orienta seus projetos a adoção da arquitetura MVC (*Model-View-Controller*), este foi o padrão utilizado na implementação do projeto. Todas as regras

⁵ Ruby: Uma linguagem de programação.

⁶ Framework: Conjunto de códigos.

de negócio e lógicas da aplicação, desde as telas de usuário, passando pelas interações com banco de dados e a análise das informações geradas, até à execução dos códigos dos alunos, foram implementadas como um único projeto, ou seja, sendo parte de um único repositório e dependentes de poucas tecnologias.

Essa abordagem pode ser interessante ao curto prazo do ponto de vista de produtividade e integração das partes, mas no longo prazo, tal decisão pode ampliar as dificuldades de manter o projeto, seja pela correção de falhas, pela complexidade nas atualizações de segurança ou de dependências ou podendo ser devido a adição de novos recursos.

Uma vez que todas as partes do projeto dependem umas das outras diretamente e, paralelamente, dependem de um único *framework*, fazer alterações em qualquer camada do software pode arriscar o funcionamento de qualquer parte do sistema.

Para armazenar os dados do sistema, optou-se pelo MongoDB, um banco de dados NoSQL (*not-only SQL*) baseado em documentos. Sua escolha foi proposital devido à quantidade de leituras dos dados que eram previstas na funcionalidade de análise do aprendizado pelo professor. E, ao mesmo tempo que ela apresenta um bom desempenho nesse tipo de operação, esta ferramenta torna os relacionamentos entre dados dificultados, devido a vasta complexidade de manutenção e de interpretação do escopo para resolução de problemas, sendo que pequenas mudanças acarretariam em grandes alterações na utilização do software, deixando inviável a sua utilização, tanto para o caso de uso de análise pelo professor, como em outros casos de uso que afetam diferentes domínios da aplicação.

A ferramenta *Elasticsearch* também faz parte da aplicação na atual versão 2 do Farma-Alg, como recurso avançado de busca e leitura de dados no módulo de análise do aprendizado de alunos e turmas. Este é um recurso bem específico da aplicação e adiciona uma complexidade considerável ao projeto.

2.2.2 Alto Acoplamento

O principal fator identificado que torna a manutenção do projeto dificultada é o da grande interdependência das partes do projeto e da grande dependência de poucas tecnologias. Esta característica pode ser denominada “acoplamento”, que é um padrão de projeto pouco desejável e que mede o nível de dependência de

diferentes partes do sistema umas com as outras. Quanto maior o acoplamento, mais difícil é isolar as partes do software e garantir coesão, manutenção e teste.

Por isso, geralmente se busca o menor grau de acoplamento possível, permitindo que o software seja dividido em pedaços menores (módulos) e que estes pedaços possam ser executados e testados fora do contexto da aplicação, permitindo, inclusive a reutilização do código em diferentes módulos e até projetos (MARTIN, 2014).

2.3 PLANEJAMENTO E DESENVOLVIMENTO DE SOFTWARE

2.3.1 Modelagem

A modelagem de um *software* pode ser entendida como a representação abstrata do sistema e envolve a construção de modelos para mostrar a estrutura e o comportamento exercidos pelo programa. Através dos modelos, é possível obter diferentes visões sobre o sistema, dividir a complexidade presente nele e auxiliar na sua compreensão, que é um ponto chave para obter uma implementação assertiva do que se espera do software.

Em vista disso, é imprescindível que a modelagem de software siga padrões que sejam universais e simples de serem compreendidos. Um dos mais conhecidos padrões para modelagem de software é a Unified Modeling Language, ou Linguagem de Modelagem Unificada (UML).

Essa linguagem de notação foi criada por Grady Booch, James Rumbaugh e Ivar Jacobson em 1996 e é atualmente uma das mais aceitas e utilizadas para a modelagem de softwares, uma vez que os diagramas construídos com a UML contêm elementos padronizados que deixam o entendimento dos envolvidos mais claro e fácil.

2.3.2 Metodologia

O termo metodologia de desenvolvimento de software se refere a uma série de atividades que auxiliam na criação de *softwares*, em busca de um produto final que representa o próprio *software*. A não adoção de metodologias de desenvolvimento pode ocasionar em baixa qualidade do *software* produzido, assim como atrasos nos prazos e, conseqüentemente, insatisfação e maior custo monetário (KOSCIANSKI E SOARES, 2007).

Uma das mudanças mais significativas propostas à está 3ª versão do Farma-Alg é a de dividir o projeto em subprojetos com escopos menores e finalidades bem definidas. Uma vez implementados os recursos de modo independente, as partes serão integradas através das interfaces (API) que cada uma delas dispõem.

Além da trabalhosa, porém objetiva e prática arquitetura estrutural do sistema em sua 3ª versão, tal atualização agrega funcionalidades tanto na parte dedicada ao servidor e a sua manutenção, quanto na interface para o usuário, agregando recursos como interrupções de códigos de execução infinita, como durante algum comando de repetição de ciclo mal desenvolvido ou execuções muito longas e custosas em termos de processamento.

Além da inclusão do *markdown* que é uma funcionalidade para destaque de texto, muito utilizada durante a explicação de termos de código ou a apresentação do código como um todo, possuindo também outras funcionalidades como a possibilidade de realizar uma duplicata de exercícios dos acervos públicos elaborados por outros docentes ou a prática interface de inserção de uma nova linguagem de programação.

Isso claramente adiciona certa complexidade a arquitetura do sistema como um todo, mas permite que os componentes do sistema sejam separados por módulos, de forma independentes. Está é uma abordagem fortemente baseada no padrão de projeto *bridge* (GAMMA, HELM, JOHNSON, VLISSIDES, 1995).

De acordo com Sommerville (2003), as atividades essenciais presentes em todas as metodologias consistem em:

- a) Especificação: etapa em que são definidas as funcionalidades do software.
- b) Projeto e implementação: etapa em que o software for desenvolvido.
- c) Validação: etapa em que são realizados testes e revisões para verificar se os requisitos foram atendidos.
- d) Evolução: etapa em que são realizadas manutenções no software.

As principais metodologias de desenvolvimento de software se dividem em duas categorias: metodologias tradicionais e ágeis. As tradicionais também são conhecidas como orientadas ao planejamento ou pesadas, e se caracterizam por documentar e planejar muito antes de realizar a implementação de um software. Já as metodologias ágeis prezam por uma maior flexibilidade tanto na documentação quanto na implementação, e buscam realizar entregas contínuas.

Entretanto, as metodologias ágeis em sua maioria não possuem nada de novo no desenvolvimento em si, e sim enfoques e valores diferenciados (COCKBURN et al., 2001). Enquanto o tradicional valoriza o processo e a predição, o ágil foca nas pessoas e na adaptação. Contudo, ambas as metodologias são válidas para diferentes situações e a escolha de uso deve depender do ambiente organizacional e o contexto do software.

Esta metodologia foi o primeiro processo publicado de desenvolvimento de software (PRESSMAN, 2001) e propõe o desenvolvimento em sequência de etapas, em que uma só pode ser iniciada após o término da anterior, expresso na Figura 5.

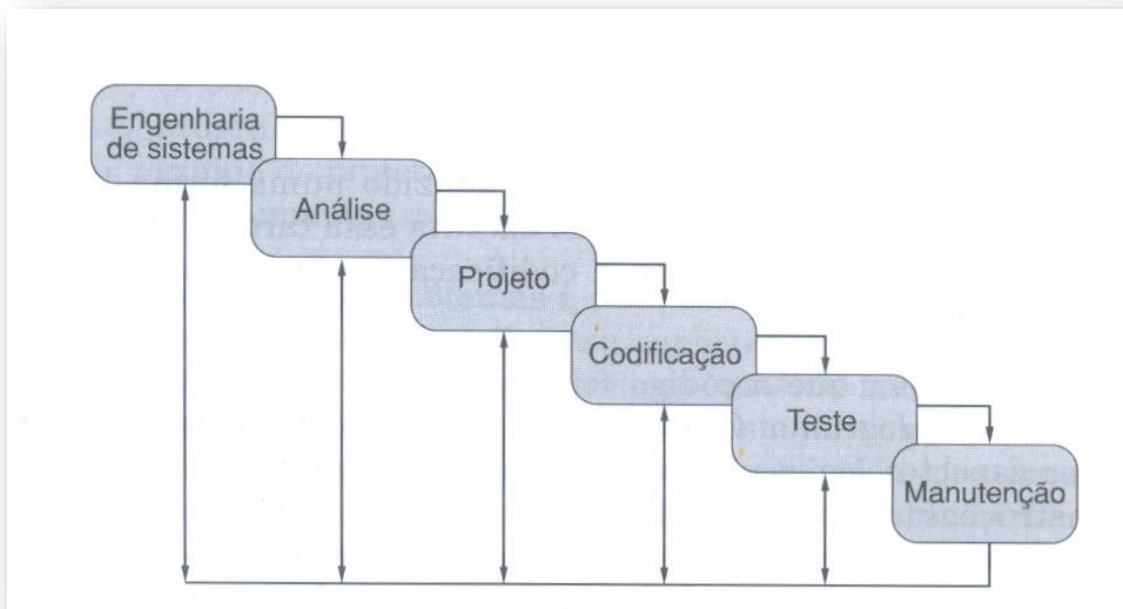


Figura 5 - Etapas da metodologia cascata de desenvolvimento

FONTE: PRESSMAN, 2001

Na Figura 5, estão representadas as principais etapas da metodologia cascata. Pode-se observar que as etapas mais conceituais e de planejamento (como engenharia de sistemas, análise e projeto) ocupam grande parte do processo, enquanto a etapa de codificação ocupa consideravelmente menos.

Isto põe em evidência a grande importância que a documentação ocupa neste modelo. É necessário primeiramente compreender completamente o problema a ser resolvido com o software, seus requisitos e restrições e documentá-los para em

seguida projetar soluções que atendam a todo o escopo. Até esta parte do processo nada é codificado, e isso não é coincidência.

A importância da definição, entendimento e alinhamento dos requisitos no modelo cascata ocorre principalmente porque a ocorrência de um erro em etapas mais avançadas do projeto custa diversas vezes mais do que identificar os erros nas etapas de documentação (PRESSMAN, 2001). Nesse sentido, quanto melhor revisado e documentado o escopo de um *software* é, menor o risco de falhas e menor o custo gasto para consertá-las.

2.3.3 Microserviços

O conceito de microserviços se origina da utilização de uma abordagem em que funcionalidades são particionadas em pequenas aplicações, onde cada particionamento é desenvolvido e disponibilizado de forma independente, tratando cada funcionalidade como um serviço, a fim de permitir uma maior modularização e com isso, uma maior praticidade para incrementação e manutenção destes serviços. Tais funcionalidades por vezes, são denominadas como interface de programação de aplicações (API), onde após o recebimento de requisições, retornam os dados tratados ou informações pertinentes.

A utilização de microserviços foi sugerida e abordada devido a percepção da dificuldade em atualizar o projeto Farma-Alg na 2ª versão, sendo uma clara ocorrência do alto acoplamento, onde um sistema inteiro ou grande parte deste sistema, utiliza de uma mesma linguagem para a sua estruturação, sem que ocorra algum tipo de segmentação do código. Dificultando drasticamente a sua interpretação e manutenção.

Para tanto, utilizando linguagens e abordagens metodológicas mais eficazes e que ainda possuam grande demanda por parte do mercado de trabalho, tanto como ferramentas e bibliotecas de desenvolvimento atuais e conceituadas, se fez possível o desenvolvimento do Farma-Alg em sua 3ª versão. Onde se ocorre a abordagem de microserviços, segmentando o sistema em quatro partes principais, sendo elas o *frontend* (interface), *backend* (servidor controlador), *sandbox* (contêiner de execução de códigos) e o banco de dados, permitindo assim, ajustes, manutenções, incrementações e atualizações. E devido a sua atualização para ferramentas e linguagens mais novas, podemos esperar um grande período de tempo até indícios de defasagem.

2.3.4 Containerização

Já o embasamento da virtualização é definido como a execução virtualizada de um sistema operacional em uma mesma máquina, focado na execução de determinado conjunto de tarefa. Entretanto, o conceito de containerização vai além, propondo uma abordagem granular de atuação, se dedicando apenas no fracionamento de um sistema operacional, com o conceito de portabilidade e distinção dentre o próprio sistema operacional de execução.

A destinação da containerização neste projeto se dedica ao setor de execução dos algoritmos fornecido pelos discentes, sendo retratado nesta documentação como *sandbox*. A escolha por esta abordagem está intimamente ligada ao fator de segurança do projeto, tendo em vista que a submissão e execução dos algoritmos fornecido pelos discentes poderá conter códigos maliciosos que visam acessar informações indevidas, alterar dados, escalonar permissões de administrador ou simplesmente, inviabilizar a utilização do sistema através de inserções de códigos que permanecem com execuções infinitas.

Para tanto, decidiu-se pelo uso da virtualização com contêineres, pois além de ser uma possível solução, é uma das mais flexíveis, de fácil manutenção, que ainda agrega no desacoplamento do projeto e com isso, a segurança do *software*, garantindo a separação dos processos junto do sistema operacional, sem que ocorra interrupções devido a execução de algum processo criado por algum aluno e que ainda não o encerrou.

Um dos principais componentes do Farma-Alg em sua 3ª versão, é o módulo de execução de códigos de alunos, denominado de *sandbox*. Mas por se tratar de uma inserção de algoritmos e de sua compilação/execução, o torna um processo extremamente sensível em termos de segurança e se não for bem tratado, pode gerar riscos ao sistema, como a ocorrência de vazamento de dados, gasto excessivo de processamento com apenas uma atividade, execução de comandos externos, acesso indevido aos diversos setores do programa, entre outras possibilidades.

Todavia, isto não diz respeito apenas a códigos intencionalmente maliciosos que buscam vandalizar ou obter vantagens das áreas sensíveis da aplicação, mas se refere a qualquer código que comprometa o funcionamento normal da aplicação ou exponha informações delicadas da infraestrutura, especialmente por se tratar de um software de ensino de programação e, portanto, estar suscetível a execução de códigos inseguros.

Como pontuado pelo mantenedor do sistema durante a entrevista (KUTZKE, 2021), já houveram situações em que o desempenho e disponibilidade do servidor foram comprometidos porque foram submetidos códigos com ciclos de repetição infinitos, ou seja, o processo permaneceu em execução até ser interrompido manualmente.

2.4 APLICAÇÕES WEB

A partir do momento que a web surgiu na década de 90, ganhou imensa popularidade devido a sua disponibilidade e às possibilidades que oferecia. A decisão de Tim Berners-Lee, criador e idealizador da *World Wide Web*, de mantê-la livre e aberta, em muito contribuiu para sua disseminação rápida e extensa (FERRAZ, 2018).

Lado a lado à criação da *web*, estão as aplicações *web*. Definidas como *softwares* disponíveis e acessados através de um navegador, que acompanham diretamente a evolução e o desenvolvimento da internet como um todo.

Em busca de se adaptar a um mundo em constante mudança e cada vez mais digital, fez-se necessário evoluir as primordiais páginas estáticas chamadas de *websites*. O dinamismo e a possibilidade dos usuários interagirem de modos cada vez mais complexos abriu vez a novas tecnologias que emergiram para acompanhar este progresso (BARESI, et al. 2000).

Hoje as aplicações *web* continuam a influenciar e ter um papel importante no cotidiano da população. Elas permitem resolver problemas e fazer operações que inicialmente eram possíveis apenas fisicamente, e de modo muito acessível devido a evolução da responsividade das aplicações, que garantem o acesso em muitos tipos de dispositivos.

Se torna cada vez mais evidente que as aplicações *web* estão em constante evolução durante seu ciclo de vida. Mesmo após serem implementadas e começarem a servir usuários, o tempo pode exigir mudanças nas funcionalidades e estrutura (GINIGE e MURUGESAN, 2001), e como esta necessidade de garantir fácil suporte e desenvolvimento contínuo se vê cada vez mais presente, salienta-se a importância de uma maior manutenibilidade da aplicação.

Devido a este requisito nas aplicações modernas, a escolha de tecnologias para atuar na aplicação é um ponto crítico. A arquitetura e os padrões de desenvolvimento também ganham destaque e devem ser considerados mais a fundo.

Um dos pontos chave da proposta de reimplementação da aplicação Farma-Alg neste trabalho é a consideração por tecnologias mais modernas e uma arquitetura mais adequada que garanta facilidade na possível continuidade do desenvolvimento do Farma-Alg por outros discentes.

Levando em conta a entrevista elaborada com o cliente (KUTZKE, 2021), onde se cita a dificuldade de manutenção da aplicação e de algumas falhas estruturais, em que os códigos encaminhados pelos discentes poderiam gerar alguns problemas durante a execução, podendo realizar consumo excessivo de processamento devido a processos que não se encerram, necessitando finalizar o mesmo na mão ou resetando a máquina hospedeira do sistema.

Devido a tais empecilhos levantados e também devido a importância de uma boa estruturação do projeto, a fim de juntar a praticidade de manutenção e a funcionalidade de modularização da aplicação para tratar de códigos externos, podendo ser encaminhado por diversas origens, sem que isso possa comprometer o sistema. Desta forma, a utilização de microsserviços rodando em contêineres, provê tanto a segurança, por rodar outros processos e com outras permissões, quanto também, a modularização almejada para poder executar futuras manutenções e implementações ao projeto.

No Capítulo 3 deste trabalho serão abordadas com mais detalhes cada uma das tecnologias e metodologias escolhidas e o porquê de sua escolha.

2.5 APLICAÇÕES SEMELHANTES

A ferramenta Farma-Alg tem forte inspiração na ferramenta FARMA (MARCZAL, 2015), que se concentra no ensino e aprendizagem de matemática, em vez de algoritmos. Ela propõe uma alternativa às pesquisas com enfoque no conceito de mobilidade tecnológico-educacional de sua época, partindo da premissa de que o erro cometido pelo aprendiz é parte primordial da construção do conhecimento e do seu desenvolvimento cognitivo (MARCZAL, 2014). Seu grande diferencial ante às demais ferramentas, é o foco nos erros dos alunos e o ferramental analítico para analisar a curva de aprendizagem de cada indivíduo ou turma por parte do professor. E isso foi incorporado também pelo projeto Farma-Alg 1ª versão, em 2015.

Outros exemplos de aplicações com foco no desenvolvimento de algoritmos e no aprendizado de programação não são difíceis de serem encontrados na *web*.

Todavia, durante a elaboração deste trabalho não foi encontrada nenhuma plataforma de ampla distribuição e utilização no ambiente acadêmico, onde aprendizes possam ser acompanhados durante o aprendizado de programação e professores possam acompanhar a curva de aprendizado de seus alunos. Ainda assim, são abordados a seguir sistemas que possuem maiores similaridades em relação à submissão e execução de códigos-fontes e resolução de exercícios de programação.

2.5.1 CodeWars

A plataforma CodeWars é uma plataforma *online* de competição e treinamento de programação, tendo uma vasta quantidade de linguagens cadastrada em sua galeria. O CodeWars permite que o usuário defina as suas linguagens de treinamento (Figura 6) e logo então, lhe é apresentado exercícios que vão tendo a sua dificuldade gradativamente aumentada em decorrência do seu avanço e desempenho durante a realização das atividades, possui ainda uma interface de fórum que permite realizar a comunicação com outras pessoas que tem conhecimentos e interesses nesta mesma linguagem.

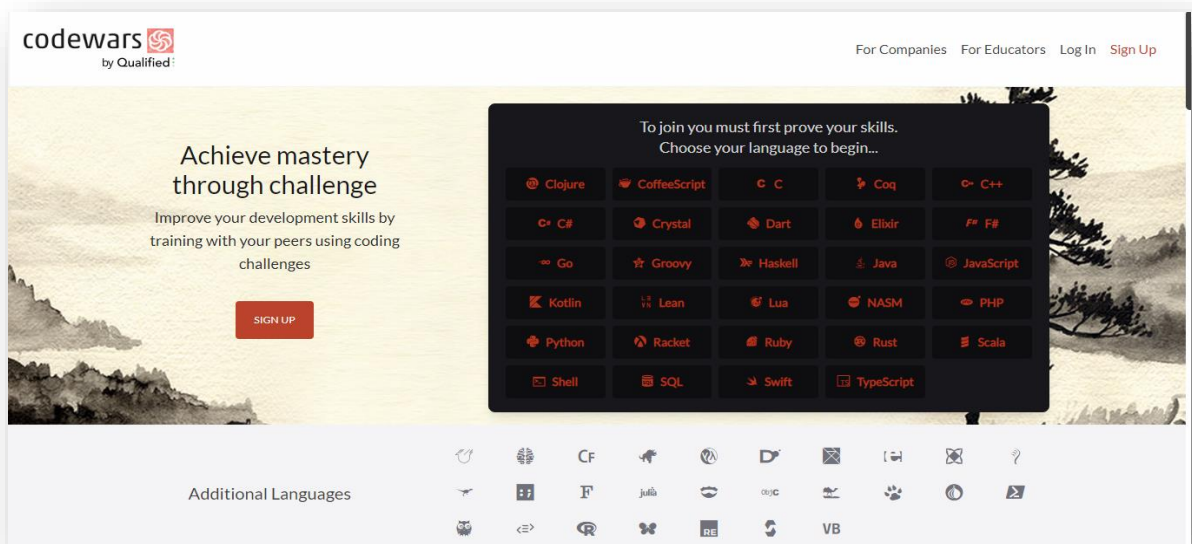


Figura 6 – Interface inicial do Codewars

2.5.2 Coderbyte

O Coderbyte é outra plataforma interessante, devido a sua interface intuitiva e facilitada apresentando todas as informações necessárias para a execução da atividade em uma mesma tela (Figura 7). Com exemplos de entrada e saída dos dados esperados, contém além disto, uma guia com as resoluções dos problemas criada e alimentada pelos próprios usuários, tendo ainda a possibilidade de importar as respostas dos outros e executar em seu perfil, além de possuir uma classificação de exercícios, de fácil, médio e difícil.

Conta ainda com uma opção paga onde possui um acervo maior de desafios e soluções, tutoriais em vídeo, entrevistas falsas de emprego para treinamento, compiladores de código melhorado, entre outras opções.

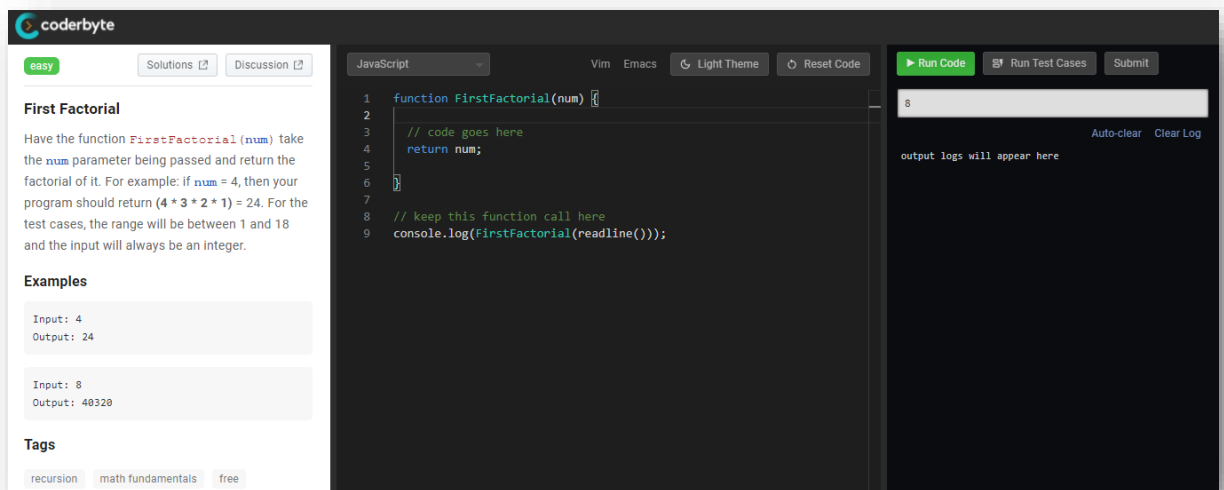


Figura 7 – Interface de submissão dos exercícios do Coderbyte.

2.5.3 Exercism

Esta plataforma contém um vasto acervo com cerca de 55 linguagens de programação, incluindo linguagens mais incomuns, como x86-64 assembly, vim script e MIPS assembly. Em sua documentação informa que é e sempre será livre de taxas ou cobranças (Figura 8), possui uma interface bem colorida e chamativa, trazendo como abordagem o encaminhamento e envolvimento de crianças e adolescentes ao mundo de desenvolvimento.

Apesar de suas ofertas e a sua explicação na hora do cadastramento de exercícios, é deixado um pouco a desejar na interface do editor, em que é realizada a escrita e a construção do código propriamente dito.

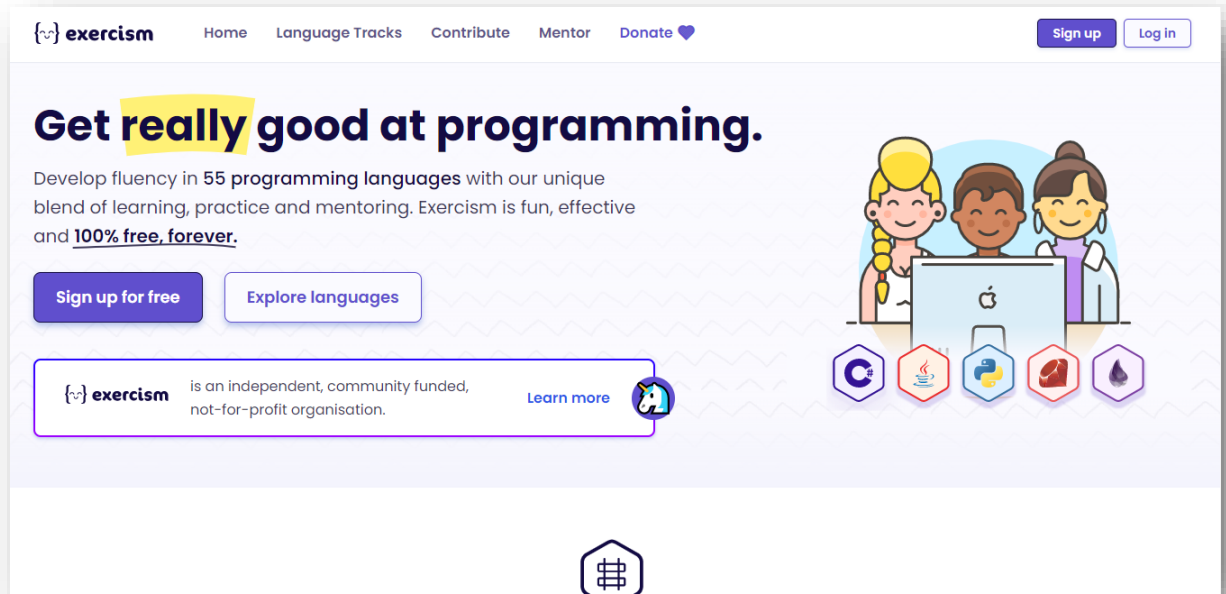


Figura 8 – Interface de inicial do Exercism.

A grande gama de plataformas encontradas atualmente, de maneira pública, geralmente requerem o preenchimento de um cadastro simples e costumam possuir alguns recursos pago, buscam se diferenciar umas das outras adicionando elementos gráficos, incrementação de novas linguagens ou através de abordagens como na criação de comunidades para o compartilhamento de informações e a realização de trocas de conhecimentos, além de algumas plataformas possuírem um *ranking* de desempenho de atividades, exposição de perfil a empresas de tecnologia, falsas entrevistas de empregos para teste e assim por diante.

Todavia, apesar da grande facilidade de utilização de ferramentas como estas para o crescimento pessoal e profissional, por se tratar de um evento externo ao próprio ambiente educacional, não permite a flexibilidade do ensino, seguindo diferentes padrões de atuação, podendo divergir das abordagens ensinadas e praticadas pelas instituições, além de não permitir o controle e verificação dos exercícios por parte dos professores, durante o momento de desenvolvimento.

A fim de suprir tal necessidade, a plataforma Farma-Alg é desenvolvida sob as concepções e análises do próprio docente mantenedor do sistema Farma-Alg em sua

2ª versão. Sendo uma plataforma gratuita, funcional, bem estruturada em termos de desenvolvimento e de engenharia de software, seguindo padrões e documentações de modelagem de dados e da própria estrutura dos dados, com foco na containerização e na execução de códigos externos de forma segura, maleável, rápida e de fácil manutenção.

Além de ser possível utilizar, gerir, exportar e implementar novas funcionalidades à ferramenta Farma-Alg, notando que o mesmo será instalado em alguma instância da UFPR, sem ter nenhuma taxa ou custo com licença, além de possuir uma grande praticidade para atuar frente as manutenções e atualizações do sistema, visto a abordagem metodológica e a modularização do *software*.

3 MATERIAIS E MÉTODOS

Neste tópico são descritas com mais detalhes as ferramentas para o desenvolvimento do sistema, a arquitetura escolhida para a solução e o diferencial da mesma para destacar o motivo de sua utilização. É também abordado a metodologia, contendo diagramas e explicações, além de ser apresentado um detalhado cronograma dos passos tratados durante a realização deste trabalho.

3.1 FUNDAMENTAÇÃO

A arquitetura do sistema Farma-Alg é um dos principais fatores para a reimplementação da ideia, levando à total reestruturação e reescrita do código-fonte do projeto.

Um dos principais pontos levantados com o mantenedor do projeto, conforme pode ser visto no apêndice A, foi a dificuldade de manutenção do projeto. Isso, conforme foi exposto no Capítulo 2, ocorre também devido ao alto acoplamento do software. Em vista de solucionar este problema, foi elaborada uma arquitetura em módulos para o novo Farma-Alg, dividindo-o em quatro setores de menor acoplamento. Esta arquitetura está representada na Figura 9:

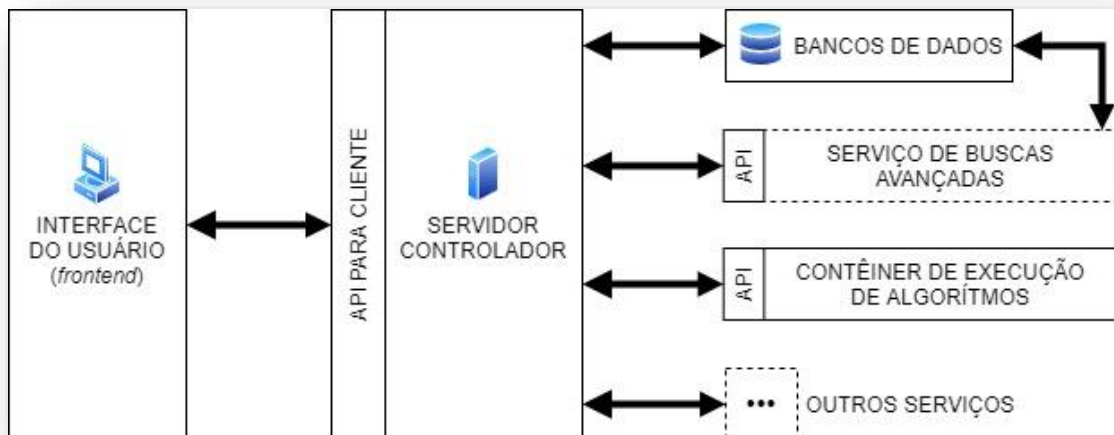


Figura 9 - Arquitetura Geral dos Módulos da Aplicação Farma-Alg 3ª versão

- a) Interface do usuário: neste módulo estão presentes a lógica de exibição das telas para o usuário e todo código que executa no lado cliente (*frontend*). Ele está completamente separado do código-fonte que executam no lado servidor

(*backend*). Todos os dados relevantes aos casos de uso e ações executadas são obtidos e persistidos no servidor através de chamadas HTTP explicitando os recursos que se deseja ter acesso.

- b) Servidor controlador: este módulo atua para integrar todos os outros e coordenar o funcionamento da aplicação como um todo. Atuando do lado servidor(*backend*), ele é responsável pelas conexões e transações em bancos de dados, assim como servir os recursos necessários ao *frontend*. Ele se comunica com o contêiner de execução de algoritmos e, posteriormente, com o sistema de buscas avançadas, que funcionam como serviços prestados a este módulo.
- c) Buscas avançadas para análise de dados: este módulo lida com o ferramental analítico quanto ao aprendizado dos alunos e das turmas. Apesar de se encontrar fora do escopo de implementação da nova versão neste momento, recomenda-se que seja um módulo separado. Isso vai permitir que a implementação seja ocultada dos demais módulos e possibilitará em uma maior liberdade na escolha das tecnologias que atenderão os casos de uso correspondentes.
- d) Contêiner de execução de algoritmos: o processo que compila e executa códigos-fonte não-confiáveis, no caso os códigos dos alunos, executa em um ambiente isolado e completamente controlado, para que a execução dos códigos não comprometa o desempenho e nem a disponibilidade dos demais módulos ou recursos do sistema.

Desta forma, a complexidade da aplicação deixa de estar na dependência de tecnologias e de profissionais que possam prover algum suporte a elas, e passa a estar nos protocolos de comunicação entre os módulos.

É importante citar que para cada um dos módulos, pretende-se estabelecer um ciclo de desenvolvimento próprio, inclusive repositórios de projeto separados e ciclos de implantações completamente individuais.

3.1.1 Virtualização de Ambiente

Outro ponto frisado durante a entrevista com o mantenedor, presente no Apêndice A, foi a falta de segurança e resiliência na execução de códigos.

Para evitar que problemas como vazamento de dados, escalonamento de privilégio e consumo excessivo de recursos do servidor, conforme exposto no Capítulo 2, Seção 2.3.4, a tecnologia de virtualização em contêineres foi escolhida para se obter maior controle do fluxo de compilação e execução dos códigos-fonte, assim como isolar do resto do sistema o módulo em que é realizada esta execução.

Este isolamento é necessário e inovador pois na corrente versão do Farma-Alg (2ª versão) a compilação e execução dos códigos é feita no mesmo servidor da aplicação, expondo o sistema operacional inteiro a falhas e vulnerabilidades imprevisíveis. Busca-se a resolução destes problemas ao ser realizada a modularização e containerização deste módulo principal e mais sensível do Farma-Alg: o módulo de execução de algoritmos.

Além disso, como citado na entrevista realizada com o mantenedor do Farma-Alg versão 2, presente no Apêndice A, um dos problemas existentes também era a submissão de códigos que possuíam ciclos de repetição infinitos. Para atuar neste problema, foi desenvolvido na versão 3 do Farma-Alg um método que realiza o encerramento do contêiner e assim, o encerramento do processo onde o código em ciclo infinito se encontra, apresentado na Figura 10.

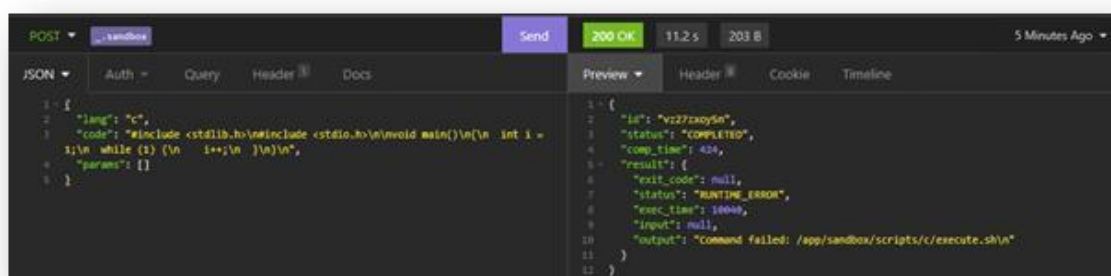


Figura 10 – Execução encerrada por Timeout

Como pode-se observar na Figura 10, após determinado período de execução (atualmente definido em 10000 milissegundos), se o processo não tiver sido finalizado é realizado encerramento do mesmo, para economizar recursos do sistema e também garantir seu contínuo funcionamento. Deixa-se enfatizado que o temporizador que realiza o encerramento dos processos pode ser facilmente ajustado através do código fonte do projeto Farma-Alg 3ª versão.

Não somente isso, mas os contêineres proporcionam uma enorme facilidade na escalabilidade horizontal, ou seja, casos de picos de acesso ou se deseja um

ambiente completamente individualizado – cada requisição faz uso de um novo contêiner, por exemplo –, a replicação deste se torna um processo simples.

Contudo, a utilização de contêineres vai muito além do isolamento de processos não-confiáveis ou inseguros. Por ser uma prática crescente de mercado e, cada mais utilizada em diversos portes de software e níveis do desenvolvimento, fica proposto também, nesta dissertação, a utilização do Docker como ferramenta de execução de contêineres, permitindo, assim, uma maior padronização do ambiente de desenvolvimento entre os membros da equipe, e uma portabilidade facilitada no momento da implantação de quaisquer módulos do sistema.

3.2 METODOLOGIA DE DESENVOLVIMENTO

Após assimilação dos conteúdos abordados no curso e levantamentos com o orientador, foi levado em conta a escolha de tecnologias que facilitassem o desenvolvimento para a web, como frameworks modernos que auxiliasse tanto na hora para gerar a documentação, quanto na praticidade de ser tratado. Novamente a escolha se deu levando em conta as vantagens de soluções mais assertivas, novas e que em nossa opinião e devido a boa estruturação e usabilidade, deduzíamos que irá perpetuar por um longo tempo.

Devido a demanda apresentada e a conclusão de abordagem, se tornou primordial a inclusão de uma tecnologia que permitisse a separação entre cliente e servidor da aplicação, com o objetivo de garantir a divisão entre o *frontend*⁷ e *backend*⁸, garantindo assim, uma complexidade menor para o desenvolvimento de futuras funcionalidades, além de priorizar a segurança de execução.

A metodologia proposta para uso neste trabalho é a clássica ou cascata, a principal presente nas metodologias tradicionais de desenvolvimento.

Tendo em vias as dificuldades que a má comunicação e a divergência de abordagem de execução podem vir a causar, foram realizadas reuniões semanais com a equipe deste trabalho, juntamente com o professor orientador, onde foi desenvolvido uma forma análoga do método Kanban, através da ferramenta Trello. Nesta ferramenta foram anexadas informações de atividades realizadas, em execução

⁷ Frontend: Relativo ao código da interface do sistema.

⁸ Backend: Relativo ao código central do sistema.

e de atividades futuras, sendo gerado 2 quadros, um quadro para atuação semanal das atividades, para ciência de todos, e outro quadro para apresentação das maiores necessidades do projeto que faltam, estão sendo ou ainda serão implementadas, apresentado na Figura 11.

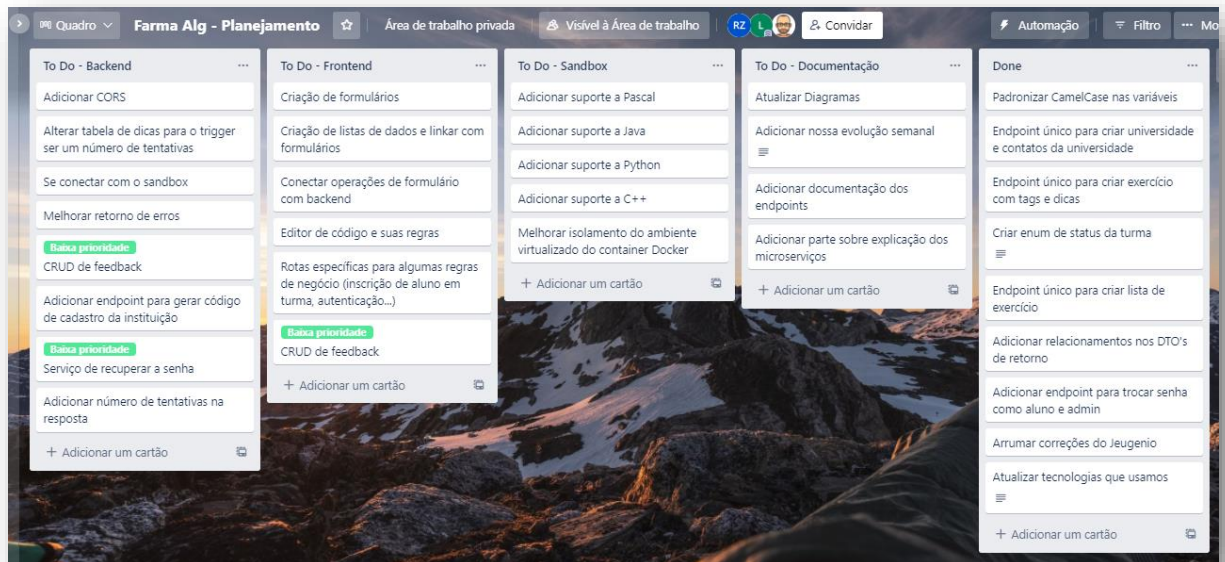


Figura 11 – Planejamento Kanban através do Trello.

3.3 FERRAMENTAS DE DESENVOLVIMENTO

Nesta sessão serão apresentadas as principais tecnologias de interesse na implementação do software Farma-Alg em sua nova versão, assim como suas principais vantagens e fatores pelos quais se destacam nesta análise.

3.3.1 Docker

O Docker é uma ferramenta gratuita de containerização para o desenvolvimento e o uso de containers Linux, com uma ativa comunidade voluntária que trabalha atuando no desenvolvimento desta tecnologia, consertando falhas e implementando novas soluções a ferramenta, podendo através da contratação de um plano, ter suporte da empresa Docker Inc; o *docker* teve o seu início através da segregação de processos do sistema operacional *Linux*, a fim de obter uma maior economia de infraestrutura gastando recursos energéticos apenas com os processos

essenciais, maior independência entre processos e com isso, maior segurança – como explanado no ponto 3.1.3 –, julga-se imprescindível a utilização desta tecnologia, principalmente no setor de compilação e execução dos códigos fontes (*sandbox*).

Além disto, a ferramenta *docker* é desenvolvida através de um modelo de implantação baseado em imagens, onde é possível realizar o compartilhamento da imagem, a fácil manutenção e ainda, executar prontamente a imagem em outro ambiente, o fator principal de sua escolha está definido através da sua relevância no setor de containerização, sendo a principal ferramenta do seu ramo na atualidade e influenciando diversas práticas no mercado.

3.3.2 Java

Para o módulo principal do Farma-Alg, o servidor controlador, estima-se que a linguagem de programação *Java* pode trazer grande produtividade no desenvolvimento – uma vez que a equipe deste trabalho tem domínio da tecnologia – e escalabilidade para aplicação.

Além da linguagem de programação, *Java* diz respeito a um conjunto de *softwares* e especificações desenvolvidos originalmente pela *Sun Microsystems*, na década de 1990, e mantida pela *Oracle* desde 2008. Um dos principais diferenciais do *Java*, quando comparado com outras linguagens de programação clássicas, é o produto resultante do processo de compilação ser executado por uma máquina virtual, a JVM (*Java Virtual Machine*), em vez de diretamente pelo sistema operacional.

Sua utilização se baseará em toda a estrutura de servidor da aplicação (*backend*), atuando diretamente nas *Controllers*, *Model's* e na *Service*, onde poderá conterá as regras de negócios, vinculação com o banco de dados e as tratativas para o recebimento das *request*⁹ fornecidas pelo *frontend*.

3.3.3 Git

A ferramenta *Git* é definida como um sistema de controle de versões, com seu uso principal no setor de desenvolvimento de *softwares* para a realização do controle de versionamento, porém, pode ser utilizado para registrar o histórico de edições de diversos tipos de arquivos, tais como, imagens, textos como livros digitais, entre outros

⁹ Request: requisições.

registros. Sua utilização é amplamente divulgada, devido a sua utilização em grandes ambientes corporativos ou educacional, já que é um *software* livre, podendo utilizar sem custo algum.

O *Git* é utilizado nas três ramificações deste trabalho, no ambiente de servidor (*backend*), durante as interfaces (*frontend*) e no setor do microsserviços e execução de códigos (*sandbox*), sendo o seu uso como peça fundamental, junto ao *GitHub*, para tratativa em conjunto, na hora do desenvolvimento deste trabalho, guardando os registros de edições do código, alterações de arquivos, horários de mudanças e o registro do usuário que executou tal ação.

3.3.4 GitHub

A ferramenta *GitHub* é uma plataforma online para realizar a hospedagem de código-fonte e de arquivos, possui ainda uma sólida integração com o *Git*, onde através de sua utilização, permite através do repositório, verificar o versionamento de cada atualização do sistema. Por se tratar de um repositório *online* e que permite obter-se um histórico tanto das mudanças, quanto de observações levantadas, depoimentos apresentando erros, guia para sugestões, entre outras funcionalidades. Se torna de suma importância para a realização deste trabalho em equipe.

Tendo em vista que a execução deste trabalho possui três setores principais como, *backend*, *frontend* e o *sandbox*, além de possuir uma equipe diversificada, a utilização do *GitHub* como repositório para todos os setores citados, se torna crucial, tanto para validação de compatibilidade entre os setores, apresentação e armazenamento do código-fonte mais atualizado a todos os integrantes, permitindo assim, saber o andamento do projeto e realizar uma comunicação mais objetiva entre os três setores, principalmente quanto a demandas, dúvidas e solicitações de execução.

3.3.5 Node.js

Node é interpretador *JavaScript* que tira proveito da arquitetura da linguagem na execução de códigos fora do ambiente do navegador, onde o *JavaScript* foi originalmente implantado. Atualmente, ele tem um espaço significativo do mercado de desenvolvimento de aplicações *web*, devido a sua característica de lidar com grande volume de requisições com assincronismo e popularidade da linguagem *JavaScript*.

A utilização do *Node* se destina ao módulo Contêiner de Execução de Algoritmos, que foi denominado de *sandbox*.

3.3.6 PostgreSQL

O PostgreSQL é um dos Sistemas Gerenciadores de Bancos de Dados (SGBD) mais avançados e utilizados da atualidade. O fato de ser um projeto de código aberto contribui para que se forme toda uma comunidade comprometida a torná-lo mais seguro e rápido, devido a sua comunidade e a sua forte estruturação no mercado atual, acarreta uma grande integração com novos *frameworks* que possibilitam novas interações e assim, a revitalização da própria ferramenta. Por esses motivos, e o fato de ser um banco do tipo relacional, que contribui para a organização dos dados do Farma-Alg, é que é uma tecnologia promissora para o curto e longo prazo.

3.3.7 React.js

React é uma biblioteca *JavaScript* voltada para a construção de interfaces de usuário para páginas *web*. Inicialmente desenvolvida para a *web*, a popularidade da ferramenta fomentou projetos paralelos para levar o *React* às mais diversas plataformas, como dispositivos móveis (com *React Native*), televisões (com *React TV*), *desktop*¹⁰ (com *React Desktop*) e até mesmo interfaces ricas para terminais (com *React Blessed*). Essa popularidade é resultado da competência dos desenvolvedores de manter a biblioteca relevante para utilização em aplicações dos mais diversos portes e pela baixa curva de aprendizagem para assimilá-la.

E por conta dessas características marcantes e versáteis, julga-se adequado adotar o *React* como tecnologia na construção das interfaces de usuário (*frontend*) do Farma-Alg, devido tanto a sua grande utilização e com isso, a disposição de recursos, integrações e ferramentas novas, além de possuir uma maior probabilidade de encontrar profissionais que possam manter o código. Outro fator importante, é devido ao excelente suporte ao *TypeScript* oferecido pela biblioteca.

¹⁰ Desktop: Computador.

3.3.8 REST Web Service

REST é uma especificação arquitetural que se refere a um conjunto de princípios e conceitos para a comunicação e desenvolvimento de aplicações web. Ela se faz uso extensivo do protocolo *HTTP* e se baseia fortemente em recursos que um serviço pode prover. Isso quer dizer que URIs (*Uniform Resource Identifier*) e os verbos *HTTP* (*get*, *post*, *put*, *patch*, *delete* e *outros*) tem importância semântica na definição dos contratos de comunicação. Sistemas que implementam os princípios *REST* são comumente chamados de *RESTful*.

Devido a facilidade de assimilação e popularidade deste padrão, julga-se apropriada a utilização do *REST* nas comunicações entre os serviços e módulos do Farma-Alg, especialmente entre o *frontend* e *backend*, que vão requerer uma quantidade de recursos importantes para atendimento dos casos de uso.

3.3.9 Spring

Para melhorar a produtividade e orientar o projeto em um padrão arquitetural mais bem definido, o *framework Spring*, é a escolha para desenvolvimento do módulo em *Java* (*backend*), por se tratar de uma tecnologia madura e amplamente difundida entre os desenvolvedores dessa linguagem. Ele é um dos mais utilizados no ambiente de mercado de trabalho, porque ajuda enormemente na produtividade, entregando soluções para os problemas mais comuns no desenvolvimento de aplicações web, além de implementar o que há de mais moderno em termos de padrões de projeto, como inversão de controle e injeção de dependências.

O uso da inversão de controle significa que a sequência e o controle das chamadas presentes no programa não são definidos pelo programador, e sim pelo *framework*. Na prática existe uma infraestrutura presente no *Spring* chamada de contêiner que é responsável por instanciar classes e definir as dependências entre elas, promovendo um baixo acoplamento entre os componentes. Já a injeção de dependência é um padrão que delega para o container a responsabilidade por injetar em cada componente suas dependências declaradas e gerenciar o ciclo de vida dessas dependências.

3.3.10 TypeScript

O TypeScript é uma linguagem de programação e também um superconjunto de código aberto baseado inteiramente na linguagem de programação denominada

de JavaScript. Ela foi criada em 2012 pela Microsoft a fim de adicionar tipagem estática ao JavaScript, permitindo verificações em tempo de compilação e tornando erros em tempo de execução mais previsíveis.

A sua utilização no Farma-Alg permeia o *frontend*, devido a sua tipagem estática, o que facilita o entendimento de erros e a correção do mesmo, desta forma, por consequência, impacta positivamente e previsivelmente no comportamento do código, sua outra indicação para utilização, foi devido a facilidade e praticidade de comunicação com a linguagem *React.js*, possibilitando integrações mais simples e dinâmicas. Por isso, quaisquer partes do projeto que demandem do *JavaScript*, será preferível a adoção do *TypeScript* no lugar.

3.3.11 Swagger

O Swagger é um *framework* de código-livre formado de muitas ferramentas que auxiliam na descrição, consumo e visualização de serviços de uma API REST. Ele foi criado e continua sendo mantido pela *SmartBear Software*.

A sua utilização no Farma-Alg está ligada ao *backend*, permitindo através de uma URL acessar todos os *endpoints* que o módulo possui.

3.4 MODELAGEM DE PROJETO

Neste tópico são abordados os diagramas seguindo o padrão UML, dos diversos setores do programa, originando uma visão macro das interligações, execuções e funcionalidades do *software*.

3.4.1 Modelagem de Dados

A Figura 12 apresenta o Diagrama Relacional ou DR repensado e criado para realizar a modelagem dos dados do Farma-Alg 3ª versão e mostrar como eles serão armazenados no banco de dados. Este diagrama mostra as relações das tabelas, os atributos de cada tabela, os identificadores únicos de cada instância da tabela e os relacionamentos entre cada instância.

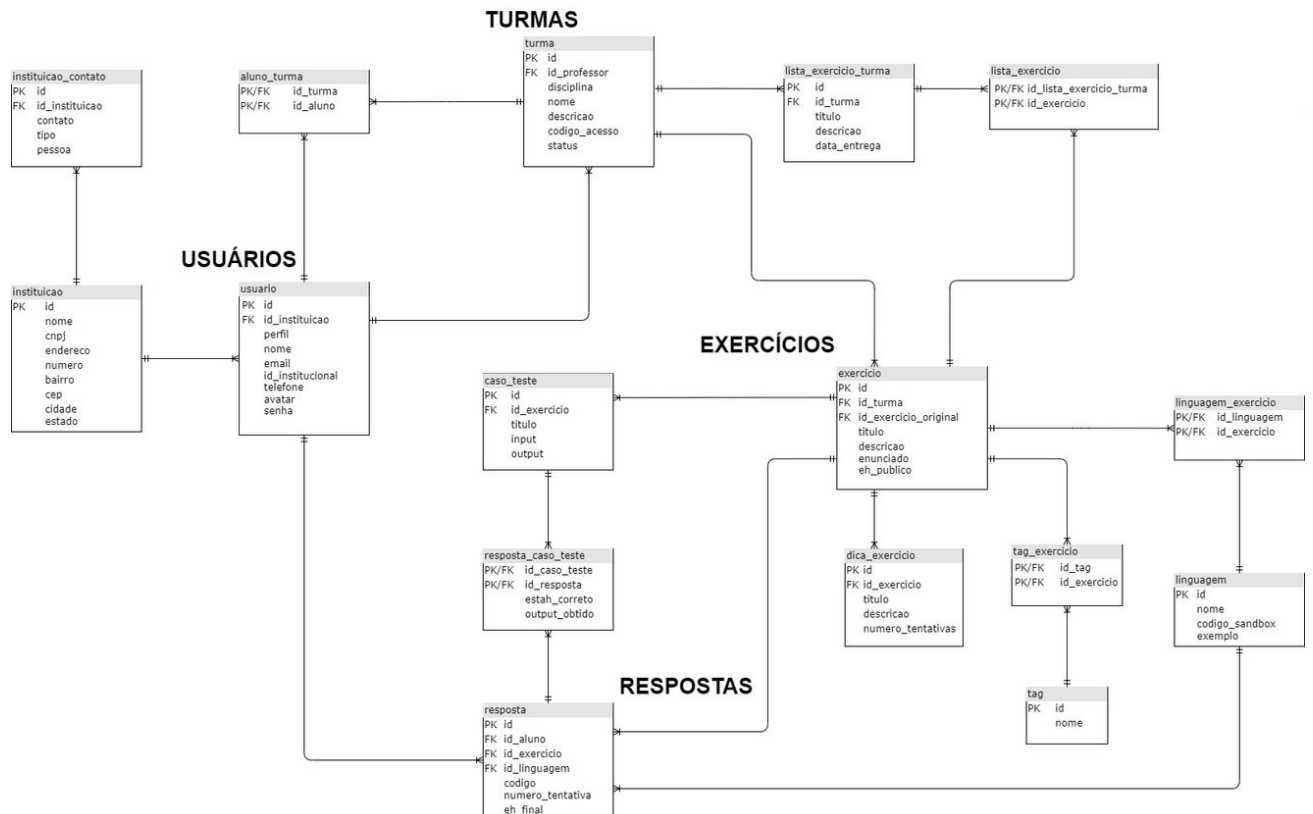


Figura 12 - Diagrama Relacional

Diante do diagrama relacional apresentado acima, pode-se observar que nele existem quatro domínios (grupos) principais de tabelas:

1. Usuários: tabelas referentes ao aluno, professor e suas instituições de ensino;
2. Turmas: tabelas referentes a turma;
3. Exercícios: tabelas referentes ao agrupamento de exercícios e suas relações com dicas, tags, linguagens e casos de teste.
4. Respostas: tabelas referentes às respostas submetidas pelos alunos para um exercício.

3.4.2 Modelagem de Sistema

Para a modelagem do comportamento do sistema e das funcionalidades esperadas, foi mapeado o Diagrama de Casos de Uso e o Diagrama de Classes, ambos seguindo a notação da UML.

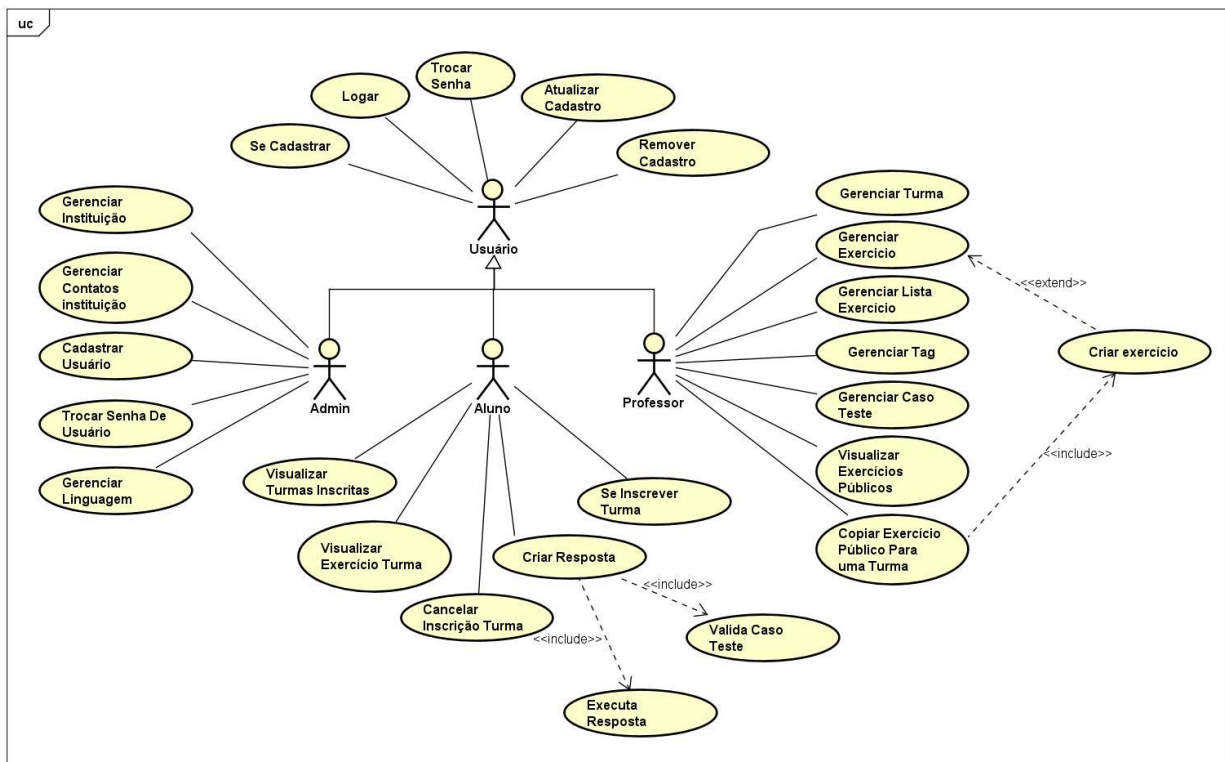


Figura 13 - Diagrama de Casos de Uso

O Diagrama de Casos de Uso apresentado na Figura 13, representa o escopo do *software* e de que modo cada usuário pode interagir com o sistema. De modo geral, observa-se que existem três atores principais que interagem no Farma-Alg: o aluno, o professor e o Admin, pois o ator Usuário é apenas a representação macro das opções básicas de qualquer outro ator. Para melhor detalhar o que cada um deles pode fazer, está presente no apêndice B a especificação dos casos de uso, contendo os fluxos de ação e resposta esperados. Além disso, no apêndice C estão presentes os Diagramas de Sequência, que também fazem referência a alguns casos de uso presentes no sistema.

O Diagrama de Classes apresentado abaixo na Figura 14 representa visualmente como é a estrutura do sistema, mostrando as principais classes modeladas, seus atributos e relacionamentos.

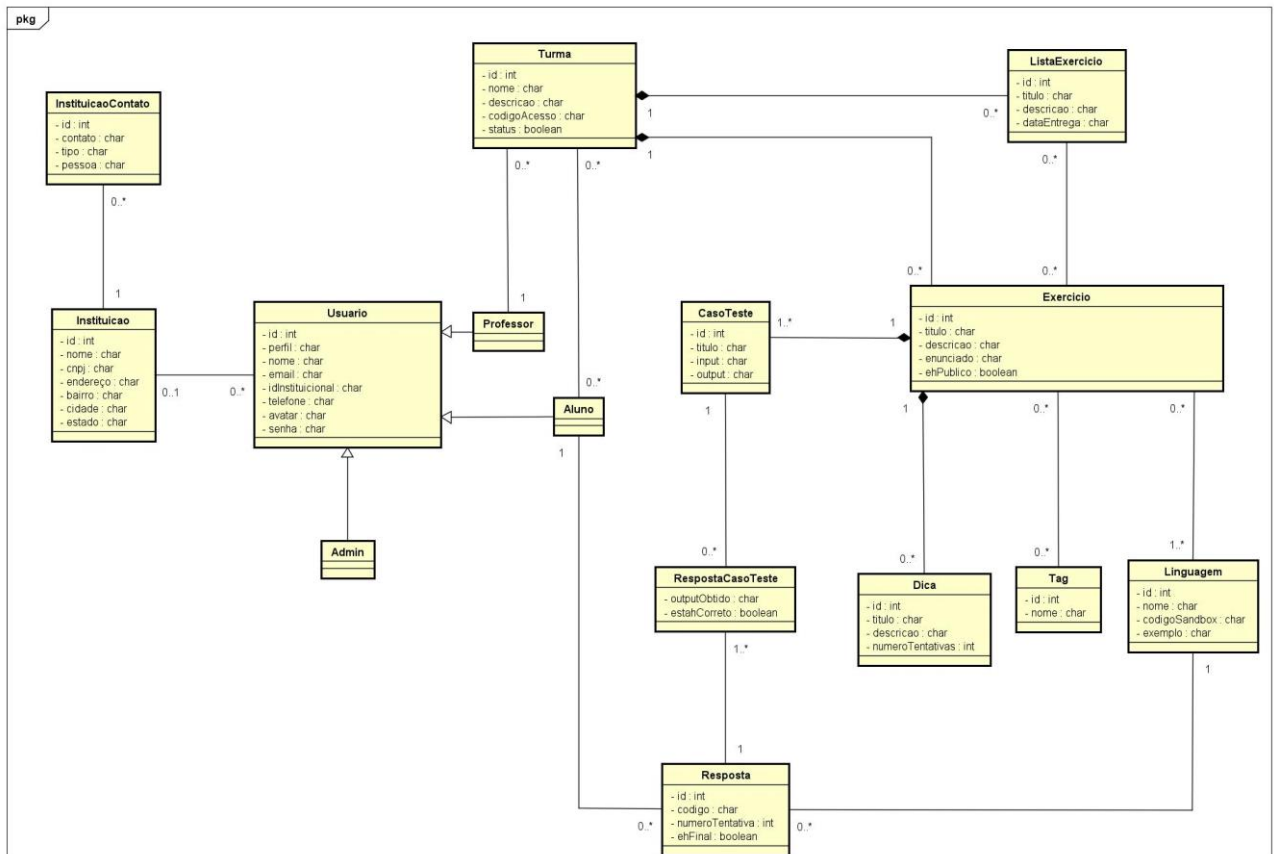


Figura 14 - Diagrama de Classes

3.5 PLANEJAMENTO DE DESENVOLVIMENTO

Na primeira parte da idealização deste projeto, foi mensurado e entendido o escopo e as funcionalidades mais importantes para atender as necessidades do Farma-Alg.

A criação dos diagramas Relacional e de Caso de Uso auxiliou muito neste período, e com eles foi possível absorver melhor qual seria o projeto em si. Foi possível também expressado as funcionalidades do sistema de uma forma clara e objetiva, podendo assim, usar estes diagramas como um guia para observar suas funcionalidades e a metodologia de ação no caso de uma reestruturação ou alteração do sistema.

Com base no escopo e nas necessidades, foi discutido e deliberado parcialmente quais eram as melhores tecnologias e métodos para a implementação do sistema. A ideia de desenvolver o projeto isolando seus módulos em uma

abordagem de microsserviços já estava presente desde este começo, mas a escolha de algumas tecnologias para o desenvolvimento se modificou com o passar do tempo.

O embasamento teórico e documentação dos motivos para o trabalho foram escritos, dando início à teste do projeto. Enquanto isso, um protótipo do Contêiner de Execução de Algoritmos foi desenvolvido, para mostrar a viabilidade da abordagem proposta.

A partir daí, imaginava-se que seria necessário um mês para cada parte do sistema: um para o Banco de Dados, outro para o *backend* e mais um para o *frontend*. No fim, não foi o que acabou acontecendo.

Dando continuidade, seria desenvolvido e formalizados outros diagramas, como de sequência e revisados os anteriormente criados.

As funcionalidades mais complexas como a geração de relatórios para o acompanhamento do aluno, grafos, e um processo de gameificação não foram abordadas neste trabalho, ficando dispostas como sugestão para futuras implementações no sistema.

O âmagô deste trabalho será a realização da transportação dentre o sistema já existente para a sua versão mais profissional e containerizada, utilizando outras linguagens para a sua programação, diversos frameworks e superconjuntos para auxiliar na sua eficiência, tais ferramentas foram escolhidas devido ao seu grande uso atualmente e sua facilidade para a execução deste trabalho, sem contar que permanecerão atualizadas e eficientes por diversos anos diante de sua demanda, todavia, devida a sua distinção dentre o *backend* e o *frontend*, ambos poderão ser substituídos ou alterados da melhor forma possível, assim como desejado.

3.6 DESENVOLVIMENTO DO PROJETO

A seguir é apresentado de forma resumida e gráfica o cronograma efetivo realizado ao longo do desenvolvimento do projeto, estendendo-se em ambos os períodos das disciplinas de TCC 1 e 2.

Tarefa	mai-21	jun-21	jul-21	ago-21	set-21	out-21	nov-21	dez-21
Busca por professor orientador								
Busca por projeto naparea da linha de pesquisa								
Entrevistas e levantamento de requisitos								
Revisão literária e fundamentação teórica								
Elaboração da proposta e da arquitetura								
Elaboração de Diagrama Relacional								
Elaboração de Diagrama de Casos de Uso								
Desenvolvimento de prova de conceito de <i>sandbox</i>								
Desenvolvimento do protótipo de interface								
Desenvolvimento do projeto <i>sandbox</i>								
Desenvolvimento do projeto <i>frontend</i>								
Desenvolvimento do projeto <i>backend</i>								
Criação de <i>endpoints</i> da aplicação (REST)								
Implementação de protocolos de autenticação								
Testes manuais e correções de erros								
Revisão do trabalho desenvolvido em TCC 1								
Preparativos finais								

Tabela 1 - Cronograma de desenvolvimeto do projeto Farma-Alg

Como durante a primeira parte do planejamento deste projeto já foi criado um protótipo do Contêiner de Execução de Algoritmos, este só foi lapidado durante a segunda parte do desenvolvimento. O Banco de Dados do Farma-Alg foi o primeiro a ser criado depois disso.

Em seguida o desenvolvimento do *frontend* desvinculado do *backend* já estava começando e não demorou muito para a criação do *backend* começar também.

Como pode ser observado, principalmente durante a segunda parte do desenvolvimento e por questões de tempo, foram realizados a implementação de módulos em paralelo. Por exemplo, é possível notar que o *frontend* e o *backend* caminharam em desenvolvimento quase que lado a lado.

4 APRESENTAÇÃO DO SISTEMA

Este tópico apresenta de forma clara e objetiva as principais funcionalidades do sistema, sendo abordado no formato de tutorial descritivo.

Ao realizar acesso ao sistema Farma-Alg na versão proposta por este documento, será apresentado a interface de entrada do sistema (Figura 15), juntamente com dois botões, dedicado ao login e ao cadastro de usuário.



Figura 15 — Interface inicial do sistema - Farma-Alg proposto.

Acessando a opção de login, é feito o redirecionamento direto a interface apresentada na figura abaixo (Figura 16), onde através da inserção de um email e de senhas válidas, se torna possível acessar o *Dashboard* do sistema.

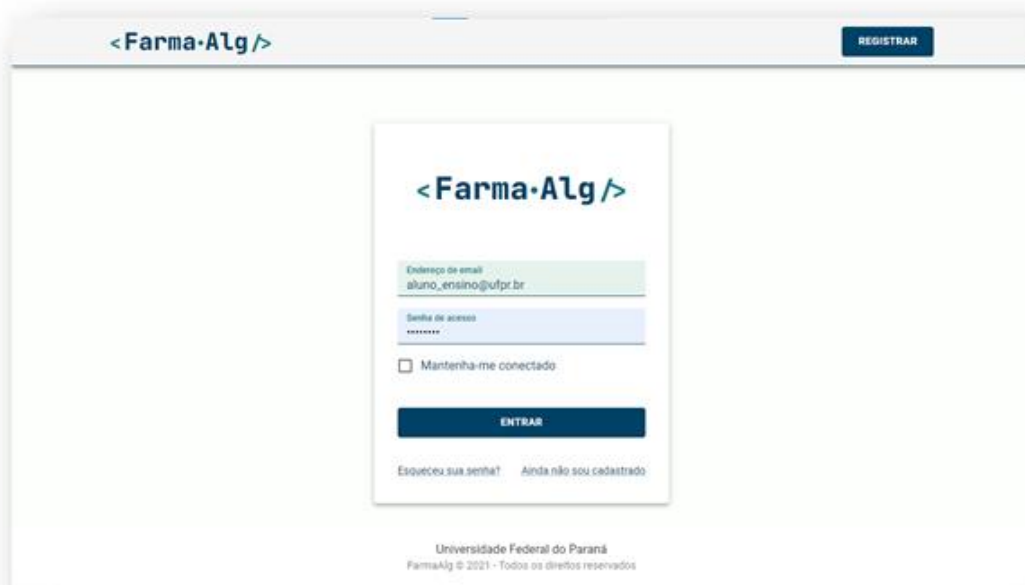


Figura 16 – Interface de login do sistema - Farma-Alg proposto.

Caso o usuário não possua acesso, pode selecionar tanto o botão “Registrar”, quanto pelo botão “Ainda não sou cadastrado” para ser redirecionado para a tela de criação de usuário.

Na interface de cadastro, é possível realizar o cadastro tanto do aluno (Figura 17), quanto do docente (Figura 18), tendo em vista que o preenchimento de ambos os formulários é similar, apresentando apenas alguns pontos específicos em que ocorre alguma diferença.



A interface de login do estudante, intitulada "Cadastre-se na Plataforma", apresenta uma barra de navegação superior com o logo "<Farma-Alg/>" à esquerda e o link "ENTRAR" à direita. Abaixo do título, há uma barra de seleção com duas opções: "SOU ESTUDANTE" (destacada em verde) e "SOU DOCENTE". O formulário de cadastro para o estudante contém os seguintes campos: "Nome completo", "Email de contato", "Instituição de ensino" (com uma seta para baixo indicando uma lista suspensa), "GRR", "Senha de acesso" e "Confirmar senha de acesso". Um botão "CADASTRAR" em azul escuro está posicionado abaixo dos campos. À direita do formulário, há uma ilustração de uma pessoa com cabelo escuro sentada à uma mesa, escrevendo em um caderno, com uma planta em um vaso ao lado.

Figura 17 – Interface de login do estudante - Farma-Alg proposto.

Cadastre-se na Plataforma

SOU ESTUDANTE
SOU DOCENTE

CADASTRAR

Figura 18 – Interface de login do docente - Farma-Alg proposto.

Após a realização do login, adentrando no sistema com o usuário administrador, pode se notar a capacidade de visualização da guia denominada “Usuários”, onde é apresentado todos os usuários cadastrados no banco de dados e onde pode ser feito a criação, edição ou busca de um usuário específico (Figura 19).

<
<Farma-Alg/>
Desenvolvedor

- INÍCIO
- INSTITUIÇÕES
- USUÁRIOS
- TURMAS
- EXERCÍCIOS
- LINGUAGENS
- MENSAGENS
- CONFIGURAÇÕES

Usuários

+ NOVO USUARIO

ID	Nome	GRR	Perfil	
2	Ian Osvaldo das Neves	GRR20180000	ALUNO	
3	Alex Kutzke	MATRICULA	ADMIN	
4	Desenvolvedor		ADMIN	
5	Nathaniel Stoker	GRR20165485	ALUNO	
1	Lia Aiffen	GRR20170000	PROFESSOR	
6	Estudante 1	201521	ALUNO	
7	Estudante 2	123456	ALUNO	
8	Professor 1	1232020	PROFESSOR	

Figura 19 – Interface de Usuários- Farma-Alg proposto.

Já através do perfil de professor, é possível realizar a criação de turmas (Figura 20), onde futuramente o professor poderá criar exercícios. Cada turma possui uma situação, conforme escolhido pelo professor dentre as possibilidades: “Ativa, Encerrada, Arquivada e Pausada”.



Figura 20 – Interface de criação de Turma - Farma-Alg proposto.

Após criar uma turma, ela pode ser visualizada e juntamente a ela os alunos cadastrados e os exercícios relacionados. Isso é mostrado na Figura 21:

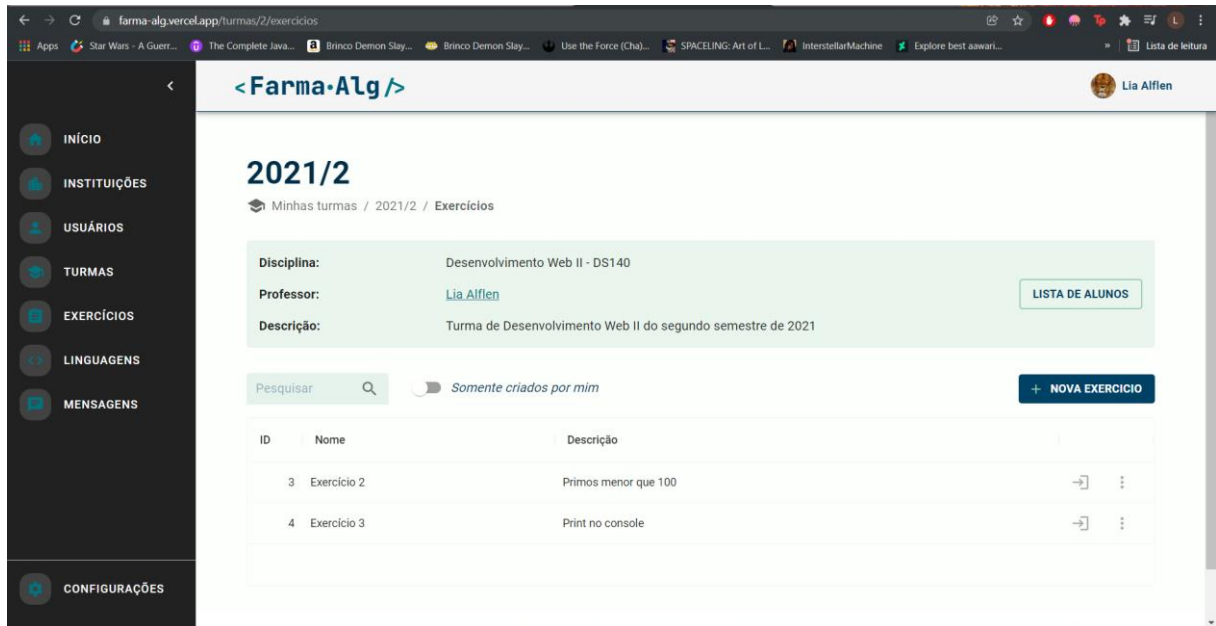


Figura 21 – Interface de Detalhes de Turma - Farma-Alg proposto.

Ainda através do perfil de professor, é possível realizar o cadastramento de exercícios (Figura 22), em que se faz necessário realizar o preenchimento do formulário com a turma desejada, nome do exercício, descrição, enunciado, além de ser necessário definir a linguagem que será abordada neste exercício e uma tag com alguma palavra chave.

O professor tem ainda a possibilidade de cadastrar dicas que são apresentados ao aluno conforme suas tentativas de resposta do exercício. Estas dicas podem, por exemplo, conter links para vídeos ou assuntos que tratem de questões semelhantes. O docente define em qual número de tentativa a dica se tornará visível ao aluno, tornando esta opção indicada para a realização de atividades mais complexas ou até mesmo para auxiliar no desenvolvimento do aluno.

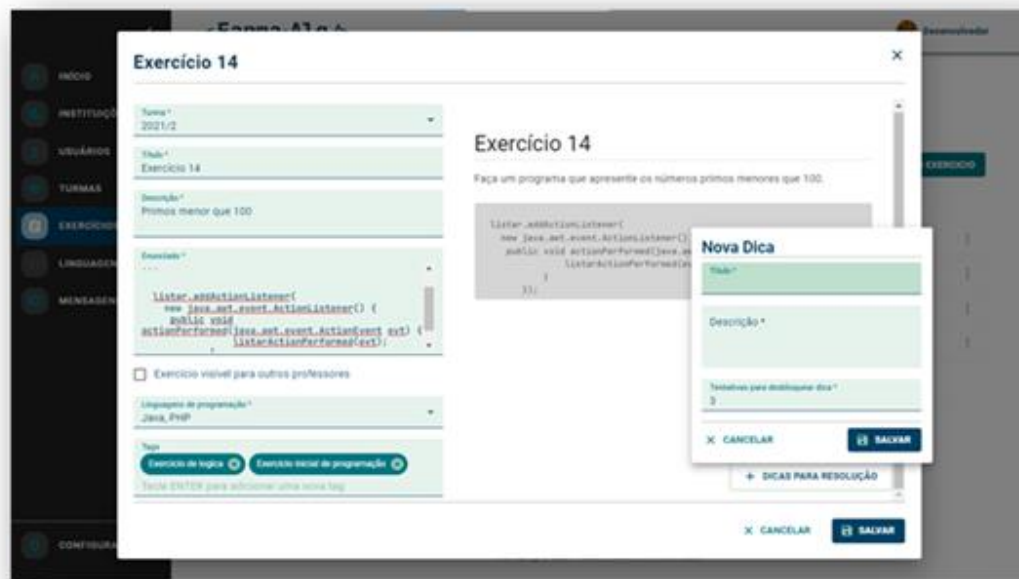


Figura 22 – Interface de criação de exercícios - Farma-Alg proposto.

Com o perfil de Aluno, os estudantes podem se matricular nas turmas para ter acesso a todos os seus exercícios. Esta matrícula é feita através de um código de acesso único para a turma, que o professor registrou ao criar a turma. Quando o aluno entra com este código de acesso, ele consegue se inscrever na turma.

Ainda com o perfil de Aluno é possível submeter respostas a exercícios cadastrados. Isso é feito através da interface Editor, como mostrada na figura 23.

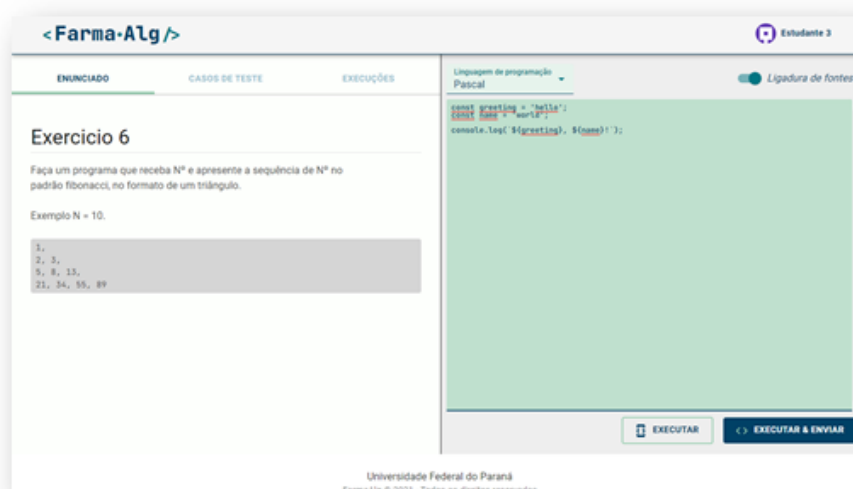


Figura 23 – Interface de submissão de respostas - Farma-Alg proposto.

O aluno deve preencher o código-fonte de resposta do exercício, selecionar a linguagem que está presente o código-fonte e clicar em Executar e Enviar para submeter a resposta. Para maiores detalhes de fluxo, estão presentes no Apêndice C os Diagramas de Sequência do *software*, que mostram em maiores detalhes o que ocorre por trás das interfaces.

Além disso, a documentação dos *endpoints* do *backend* se encontra no Apêndice D. Para uma visualização mais completa e com todos os detalhes de cada *endpoint*, pode-se acessar este link: [`https://farmaalg-back.herokuapp.com/swagger-ui/index.html#`](https://farmaalg-back.herokuapp.com/swagger-ui/index.html#/)

5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi a reimplementação buscando aperfeiçoar a plataforma Farma-Alg existente, atualizando as tecnologias presentes e segmentando o sistema. Uma das maiores preocupações presentes foi o cuidado para que futuramente ao serem realizados incrementos, atualizações, modificações ou manutenções, o Farma-Alg não se torne inutilizável e possa, assim, ter um longo período de uso.

Para a formalização de tais atividades, a implementação se deu através da separação dos serviços por meio da utilização de microsserviços, os quais serão os responsáveis por diferentes ações no sistema. Utilizou-se também contêineres para possibilitar a separação dos códigos externos e possivelmente maliciosos do código fonte do sistema, visando o não comprometimento do *software* e gerando um ambiente mais seguro, confiável e disponível.

Fica evidenciado durante toda a pesquisa e reimplementação que o Farma-Alg é uma ferramenta que, se bem utilizada e mantida, pode ser uma grande aliada no ensino e aprendizagem de programação. Existe muito espaço para novas funcionalidades e potencial para melhoria. Espera-se que, apesar do curto tempo para a realização desta reimplementação, a abordagem escolhida se revele de muito proveito e possibilite um avanço ainda maior do uso do Farma-Alg na UFPR e futuramente até fora dela.

As funcionalidades do escopo proposto foram bem implementadas, mas muito ainda pode ser aproveitado das versões passadas do sistema, e elas são as principais

sugestões para futuras implementações. Como, por exemplo, a implantação de um sistema de grafos, que permitiria ao professor uma melhor apresentação visual da sua evolução dos alunos nos exercícios.

Outra proposta de acréscimo para o futuro é um sistema de gamificação dentro do software, para fomentar os alunos e instruí-los a progredir nos estudos, enquanto competem de maneira saudável por uma melhor classificação.

Além disso, ficam propostas funcionalidade simples que gerariam mais facilidade aos usuários, como um serviço de recuperar senha que utilize o envio de e-mails – o que atualmente no sistema é feito através de uma solicitação informal ao admin, que trocava a senha do usuário; uma funcionalidade de trocas de mensagens ou *feedbacks* também seria interessante para ampliar a comunicação entre aluno e professor em relação a dúvidas;

REFERÊNCIAS

AINSWORTH, S. Deft: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, v. 16, n. 3, p. 183 – 198, 2006. ISSN 0959-4752. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0959475206000259>> Acesso em: 14 dez. 2021.

BARESI L., GARZOTTO F., PAOLINI P. (2000) From Web Sites to Web Applications: New Issues for Conceptual Modeling. *Rev. Lecture Notes in Computer Science*, v. 1921, Dec. 2001. Disponível em: https://link.springer.com/chapter/10.1007%2F3-540-45394-6_9. Acesso em: 01 ago. 2021. https://doi.org/10.1007/3-540-45394-6_9

CASTRO, C. T., CASTRO JUNIOR, A., MENESES, C., B., M. e RAUBER, M. (2003) Utilizando Programação Funcional em Disciplinas Introdutórias de Computação, In: XI Workshop de Educação em Computação – WEI, Campinas/SP.

CHACON, S. About Git, Disponível em: <https://git-scm.com/about/free-and-open-source>. Acesso em: 12 dez. 2021.

CODERBYTE. Disponível em <https://www.coderbyte.com/>. Acesso em: 13 set. 2021.

CODEWARS. Disponível em <https://www.codewars.com/>. Acesso em: 13 set. 2021.

COMUNNITY TYPESCRIPT; TypeScript. Disponível em: <https://www.typescriptlang.org/> Acesso em: 06 ago. 2021.

EHRESMANN, R. W. Gamificação significativa na educação: uma proposta de ferramenta com enfoque no aspecto motivacional do aluno, Curitiba. 2018. 103 f. Dissertação de Mestrado – Setor de Ciências Exatas, Universidade Federal do Paraná Curitiba, 2018. Disponível em: <https://acervodigital.ufpr.br/handle/1884/54903>. Acesso em: 02 dez. 2021.

EXERCISM. Disponível em <https://exercism.org/>. Acesso em: 13 set. 2021.

FACEBOOK OPEN SOURCE, ReactJS. Disponível em: <https://pt-br.reactjs.org/community/support.html> Acesso em: 06 ago. 2021.

FERNANDES, A. O que é Containerização de aplicação? Disponível em: <https://vertigo.com.br/o-que-e-containerizacao-de-aplicacao/>. Acessado em: 12 dez. 2021.

FERRAZ, R. Tendências da Web. Editora Senac São Paulo - São Paulo - 2018. Disponível em: https://books.google.com/books/about/Tend%C3%A2ncias_da_web.html?hl=pt-BR&id=bppWDwAAQBAJ. Acesso em: 02 ago. 2021.

GAMMA, E, HELM, R, JOHNSON, R, VLISSIDES, J. Design Patterns. (page 151 - Addison-Wesley, 1995) Disponível em: <http://www.javier8a.com/itc/bd1/articulo.pdf> Acesso em: 06 ago. 2021.

GINIGE, A.; MURUGESAN, S. Web engineering: an introduction. Rev. IEEE MultiMedia, vol. 8, no. 1, pág. 14-18, Mar. 2001.
<https://ieeexplore.ieee.org/document/923949>. Acesso em: 04 ago. 2021.
 10.1109/93.923949

GLOBAL DEVELOPMENT GROUP; PostgreSQL. Disponível em:
<https://www.postgresql.org/about/> Acesso em: 06 ago. 2021.

HIGHSMITH, J.; COCKBURN, A. Agile software development: the business of innovation. Rev. IEEE, vol. 34, no. 9, pág. 120-127, Set. 2001. Disponível em:
<https://ieeexplore.ieee.org/document/947100>. Acesso em: 04 ago. 2021.
 10.1109/2.947100

IBM Cloud Education, Microserviços, Disponível em: <https://www.ibm.com/br-pt/cloud/learn/microservices>. Acessado em: 13 dez. 2021.

KOSCIANSKI, A.; SOARES, S. M. Qualidade de Software - 2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software (Novatec Editora, 13 de mar. de 2007 - 395 páginas). E-book. Disponível em:
<https://books.google.com.br/books?id=O9aWoUq6L88C&dq>. Acesso em: 04 ago. 2021.

KUTZKE, R. A. Informática educacional e a mediação do erro na educação: um estudo teórico-crítico e uma proposta de instrumento computacional, Curitiba. 2015. 197 f. Tese (Doutorado em Informática) – Setor de Ciências Exatas, Universidade Federal do Paraná Curitiba, 2015. Disponível em:
<https://acervodigital.ufpr.br/handle/1884/40907>. Acesso em: 31 jul. 2021.

MARCZAL, D.; DIRENE, A.; PIMENTEL, A.; KRYNSKI, M. E. FARMA: Uma Ferramenta de Autoria para Objetos de Aprendizagem de Conceitos Matemáticos. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2015, Curitiba. Anais... Curitiba: UFPR – Programa de Pós-Graduação em Informática Universidade Federal do Paraná (UFPR), 2015.

MARTIN, C. R.; JAVA. Disponível em: <https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleResponsibilityPrinciple.html> Acesso em: 06 ago. 2021.

MATSUMOTO, M. Y; Sobre o Ruby Disponível em: <https://www.ruby-lang.org/pt/about/> Acesso em: 06 ago. 2021.

MESH, J. Método Kanban: Guia detalhado e 5 modelos prontos para usar, Disponível em: <https://blog.trello.com/br/metodo-kanban>. Acesso em: 07 dez. 2021.

OPENJS FOUNDATION, About Node.js. Disponível em:
<https://nodejs.org/en/about/community/> Acesso em: 06 ago. 2021.

ORACLE CORPORATION; JAVA. Disponível em: https://www.java.com/pt-BR/download/help/whatis_java.html Acesso em: 06 ago. 2021.

PATTO, M. H. S. A produção do fracasso escolar: histórias de submissão e rebeldia. 1. ed. São Paulo: T. A. Quiroz, 1996.

IBGE ESTATÍSTICAS SOCIAIS, PNAD Educação 2019: Mais da metade das pessoas de 25 anos ou mais não completaram o ensino médio. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/28285-pnad-educacao-2019-mais-da-metade-das-pessoas-de-25-anos-ou-mais-nao-completaram-o-ensino-medio>. Acesso em: 04 jan. 2022.

PRESSMAN, S. R. Engenharia de Software Uma Abordagem Profissional (AMGH Editora LTDA; 2011 - 780 páginas). Disponível em: https://www.academia.edu/41207478/Engenharia_de_Software_Uma_Abordagem_Profissional. Acesso em: 04 ago. 2021.

RED HAT, O que é Docker? Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-docker>. Acesso em: 07 dez. 2021

SPRING FRAMEWORK. Disponível em: <https://spring.io/projects/spring-framework> Acesso em: 06 ago. 2021.

ROBINS, A., ROUNTREE, J., & ROUNTREE, Learning and teaching programming: A review and discussion. Computer Science Education, Taylor & Francis, v. 13, n. 2, p. 137 - 172, 2003. Disponível em: <http://dx.doi.org/10.1076/csed.13.2.137.14200>. Acesso em: 02 ago. 2021.

TERCEIRO, A. S. de. Caracterização da Complexidade Estrutural em Sistemas de Software. 2012. Disponível em: < <https://repositorio.ufba.br/ri/handle/ri/8149> > Acesso em: 13 dez 2021.

VERCEL; NextJS. Disponível em: <https://nextjs.org/> Acesso em: 06 ago. 2021.

APÊNDICE A – ENTREVISTA COM O MANTENEDOR

Entrevista realizada com o mantenedor atual e desenvolvedor do Farma-Alg versão 1, Kutzke em 15 de julho de 2021.

1. Quais os principais objetivos do software?

Facilitar a vida do professor com a publicação, coleta e análise dos exercícios

2. Qual é a principal necessidade que o sistema pretende suprir? O que é essencial e qual é a forma esperada desta ideia?

O professor tenha acesso a todas e quaisquer tentativas do aluno de resolver o exercício. Professor tenha acesso mais geral às tentativas dos alunos nos exercícios. Além disso também tem a correção automatizada, mas não é o principal. Melhorar a coleta de dados e confiabilidade geral da aplicação. Pensar também em como seria a visualização das tentativas.

3. Quem vai utilizar este software? Há diferentes perfis de acesso (aluno, professor, gestor...)?

Alunos, professores e 1 admin (Alex).

4. Que tipo de ações (casos de uso) se espera de cada perfil de usuário?

- Auto-cadastro de aluno;
- [pensar em cadastro atrelado à instituição (matrícula, código da ins)];
- Aluno se inscreve em turmas;
- Aluno acessa exercícios;
- Aluno responde exercícios;
- Auto-cadastro de professor;

5. Em que tipo de ambiente se espera que o sistema execute (browser, smartphone, desktop...)?

Web (browsers modernos) e possivelmente smartphones, mas simplificado.

6. Quais os principais problemas identificados no sistema hoje que requerem atenção na nova versão?

- Interface bagunçada, não intuitiva e faltando;
- Segurança e resiliência na execução de códigos;
- LIMITAÇÃO: falta de relacionamento dos enunciados e suas alterações

7. E os recursos incrementais que se deseja adicionar?

Serviço de comparação dos códigos-fonte;

8. O professor criará e posteriormente excluirá as “turmas”?

Não, talvez apenas inativasse;

9. O professor criará exercícios e código e o programa verá semelhança nos resultados para validar?

Sim, através do teste de caixa preta, onde apenas o resultado importa.

10. Existirá uma quantidade mínima de resultado de saída cadastrado pelo professor? Quais diretrizes da empresa?

Não precisa existir, mas um valor como 2 resultados de saída, me parece interessante.

APÊNDICE B – ESPECIFICAÇÃO DE CASOS DE USO

Tabela 2 - UC001: SE CADASTRAR

Caso de uso	UC001 – Se Cadastrar
Ator	Usuário
Descrição	Permite que o usuário entre com suas informações pessoais e se cadastre no sistema.
Pré condições	Não se aplica.
Fluxo principal: Aluno se Cadastrando	1. O usuário acessa a página de cadastro. 2. O usuário seleciona que é um estudante. 3. O usuário preenche o campo nome. Exemplo de preenchimento: José da Silva 4. O usuário preenche o campo e-mail. Exemplo de preenchimento: jose.silva@ufpr.br 5. O usuário seleciona sua instituição de ensino. Exemplo de preenchimento: Universidade Federal do Paraná 6. O usuário preenche o campo código de matrícula da instituição. Exemplo de preenchimento: GRR20210068 7. O usuário preenche o campo senha. Exemplo de preenchimento: Senha854 8. O usuário preenche o campo confirmar senha. Exemplo de preenchimento: Senha854 9. O usuário clica no botão de cadastrar. 10. O sistema verifica que o email fornecido pelo usuário está disponível.

	11. O sistema registra as informações inseridas pelo usuário. 12. O sistema redireciona o usuário para a página de dashboard já o tendo logado.
Fluxo alternativo: Professor se Cadastrando	1. O usuário acessa a página de cadastro. 2. O usuário seleciona que é um docente. 3. O usuário preenche o campo nome. Exemplo de preenchimento: José da Silva 4. O usuário preenche o campo e-mail. Exemplo de preenchimento: jose.silva@ufpr.br 5. O usuário seleciona sua instituição de ensino. Exemplo de preenchimento: Universidade Federal do Paraná 6. O usuário preenche o campo código de matrícula da instituição. Exemplo de preenchimento: 20215485 7. O usuário preenche o campo telefone para contato. Exemplo de preenchimento: 41999748596 8. O usuário preenche o campo senha. Exemplo de preenchimento: Senha854 9. O usuário preenche o campo confirmar senha. Exemplo de preenchimento: Senha854 10. O usuário clica no botão de cadastrar. 11. O sistema verifica que o email fornecido pelo usuário está disponível. 12. O sistema registra as informações inseridas pelo usuário.

	13. O sistema redireciona o usuário para a página de dashboard já o tendo logado.
Fluxo alternativo: E-mail não disponível	1. O usuário acessa a página de cadastro. 2. O usuário preenche seus dados, preenchendo o campo email com um email que não está disponível. 3. O usuário clica no botão de cadastrar. 4. O sistema verifica que o email fornecido pelo usuário não está disponível. 5. O sistema retorna uma mensagem de erro para o usuário e pede que forneça um novo email.

Tabela 3 - UC002: LOGAR

Caso de uso	UC002 – Logar
Ator	Usuário
Descrição	Permite que o usuário acesse o sistema usando suas credenciais.
Pré condições	O usuário deve possuir um cadastro no sistema.
Fluxo principal: Logar	1. O usuário acessa a página de login. 2. O usuário preenche o campo e-mail. Exemplo de preenchimento: usuario@ufpr.br 3. O usuário preenche o campo senha. Exemplo de preenchimento: Senha842 4. O usuário clica no botão de entrar. 5. O sistema verifica que as informações inseridas correspondem às informações cadastradas para o usuário.

	6. O sistema cria uma sessão para o usuário e o redireciona a página de dashboard.
Fluxo alternativo: Logar – Usuário e/ou senha inválido	1. O usuário acessa a página de login. 2. O usuário preenche o campo e-mail. Exemplo de preenchimento: usuario@ufpr.br 3. O usuário preenche o campo senha. Exemplo de preenchimento: Senha841 4. O usuário clica no botão de entrar. 5. O sistema verifica que as informações inseridas não correspondem às informações cadastradas para o usuário. 6. O sistema retorna uma mensagem de erro para o usuário.

Tabela 4 - UC003: TROCAR SENHA

Caso de uso	UC003 – Trocar Senha
Ator	Usuário
Descrição	Permite que o usuário troque a senha de acesso a sua conta.
Pré condições	O usuário deve ter cadastro no sistema.
Fluxo principal: Trocar Senha	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. Em atualizar senha, o usuário preenche o campo senha atual. Exemplo de preenchimento: Senha854 4. O usuário preenche o campo nova senha. Exemplo de preenchimento: Senha221

	5. O usuário preenche o campo de confirmação da nova senha. Exemplo de preenchimento: Senha221 6. O usuário clica no botão de salvar. 7. O sistema mostra uma mensagem de sucesso, indicando que a senha foi alterada.
Fluxo alternativo: Trocar Senha – Usuário não fornece sua senha atual corretamente.	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. Em atualizar senha, o usuário preenche o campo senha atual de forma incorreta. Exemplo de preenchimento: Senha855 4. O usuário preenche o campo nova senha. Exemplo de preenchimento: Senha221 5. O usuário preenche o campo de confirmação da nova senha. Exemplo de preenchimento: Senha221 6. O usuário clica no botão de salvar. 7. O sistema identifica que a senha atual do usuário não está correta. 8. O sistema retorna uma mensagem de erro para o usuário.

Tabela 5 – UC004: ATUALIZAR CADASTRO

Caso de uso	UC004 – Atualizar Cadastro
Ator	Usuário
Descrição	Permite que o usuário atualize suas informações cadastrais.
Pré condições	O usuário deve ter cadastro no sistema.

Fluxo principal: Atualizar Cadastro	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. Em Dados Gerais, o usuário altera o campo nome. Exemplo de preenchimento: José da Silva Pereira 4. O usuário altera o campo email. Exemplo de preenchimento: jose.pereira@ufpr.br 13. O usuário altera o campo instituição de ensino. Exemplo de preenchimento: Universidade Tecnológica Federal do Paraná 5. O usuário altera o campo matrícula. Exemplo de preenchimento: GRR20210068 6. O usuário altera o campo avatar. Exemplo de preenchimento: https://avatars.dicebear.com/api/identicon/farmaalg-09819508109861852.svg?b=%23ffffff 7. Se for professor, o usuário altera o campo telefone. Exemplo de preenchimento: 47999857485 8. O usuário clica no botão de salvar. 9. O sistema mostra uma mensagem de sucesso, indicando que os dados cadastrais foram alterados.
Fluxo alternativo: Atualizar Cadastro – E-mail não disponível	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. Em Dados Gerais, o usuário preenche seus dados, preenchendo o campo email com um email que não é seu email atual e não está disponível. Exemplo de preenchimento: jose.fonceca@ufpr.br 4. O usuário clica no botão de salvar. 5. O sistema verifica que o email fornecido pelo usuário não está disponível.

	6. O sistema retorna uma mensagem de erro para o usuário e pede que forneça um novo email.
--	--

Tabela 6 – UC005: REMOVER CADASTRO

Caso de uso	UC004 – Remover Cadastro
Ator	Usuário
Descrição	Permite que o usuário remova sua conta do sistema.
Pré condições	O usuário deve ter cadastro no sistema.
Fluxo principal: Remover Cadastro	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. O usuário clica no botão Remover conta. 4. O sistema pede a confirmação do usuário. 5. O usuário confirma que deseja remover sua conta. 6. O sistema remove o usuário e todos os dados relacionados a ele. 7. O sistema encerra a sessão e redireciona o usuário para a página de login.
Fluxo alternativo: Remover Cadastro – Usuário opta por não remover	1. O usuário acessa a página de configurações e seleciona “Gerenciar Conta” 2. O sistema mostra todos os dados cadastrais do usuário. 3. O usuário clica no botão Remover conta. 4. O sistema pede a confirmação do usuário. 5. O usuário nega que deseja remover sua conta. 6. O caso de uso se encerra.

Tabela 7 - UC006: GERENCIAR TURMA

Caso de uso	UC006 – Gerenciar Turma
Ator	Professor

Descrição	Permite que o professor liste, acesse, crie, edite e remova suas turmas.
Pré condições	O professor deve ter cadastro no sistema.
Fluxo principal: Listar Turmas	1. O professor acessa a página Turmas. 2. O sistema busca todas as turmas pertencentes ao professor. 3. O sistema exibe a listagem de turmas.
Fluxo alternativo: Acessar Turma	1. O professor acessa a página Turmas. 2. O professor seleciona uma turma na listagem das suas turmas. 3. O professor clica em acessar. 4. O sistema redireciona o professor a página de detalhes da turma.
Fluxo alternativo: Criar Turma	1. O professor acessa a página Turmas. 2. O professor clica em Nova Turma. 3. O professor preenche o campo nome da disciplina. Exemplo de preenchimento: Desenvolvimento Web I - DS122 4. O professor preenche o campo nome da turma. Exemplo de preenchimento: 2021/2 5. O professor preenche opcionalmente o campo descrição. Exemplo de preenchimento: Turma de Desenvolvimento Web I do segundo semestre de 2021 6. O professor preenche o campo código de acesso da turma. Exemplo de preenchimento: ds122 7. O sistema valida que o código preenchido está disponível.

	8. O professor seleciona o campo situação da turma. Exemplo de preenchimento: ATIVA 9. O professor clica em Salvar. 10. O sistema registra as informações inseridas pelo professor e cria a turma. 11. O sistema exibe uma mensagem de sucesso e atualiza a página de turmas com a nova turma.
Fluxo alternativo: Editar Turma	1. O professor acessa a página Turmas. 2. O professor seleciona uma turma na listagem das suas turmas. 3. O professor clica em Editar. 4. O professor altera o campo nome da disciplina. Exemplo de preenchimento: Desenvolvimento Web I - DS121 5. O professor altera o campo nome da turma. Exemplo de preenchimento: 2021/2 - Tarde 6. O professor altera opcionalmente o campo descrição. Exemplo de preenchimento: Turma de Desenvolvimento Web I do segundo semestre de 2021 da Tarde. 7. O professor altera o campo código de acesso da turma. Exemplo de preenchimento: ds121 8. O sistema valida que o código preenchido está disponível. 9. O professor seleciona o campo situação da turma.

	Exemplo de preenchimento: PAUSADA 10. O professor clica em Salvar. 11. O sistema altera a turma de acordo com as informações inseridas pelo professor. 1. O sistema exibe uma mensagem de sucesso e atualiza a página de turma.
Fluxo alternativo: Criar ou Editar turma - Código de acesso da turma não disponível	2. O professor acessa a página Turmas. 3. O professor clica em Nova Turma. 4. O professor preenche as informações referentes a turma, preenchendo o campo código de acesso da turma com um código que já está sendo utilizado em outra turma. Exemplo de preenchimento: ds140 5. O sistema valida que o código preenchido não está disponível. 6. O sistema exibe uma mensagem de erro, pedindo que o professor altere o código de acesso. 12. O sistema inativa o botão de salvar.
Fluxo alternativo: Remover Turma	1. O professor acessa a página Turmas. 2. O professor seleciona uma turma na listagem das suas turmas. 3. O professor clica em Excluir. 4. O sistema pede que o professor confirme a remoção. 5. O professor confirma que deseja remover a turma. 6. O sistema remove a turma e todas as suas informações relacionadas. 7. O sistema exibe uma mensagem de sucesso e atualiza a página de turma para retirar a turma excluída.

Fluxo alternativo: Remover Turma – Professor opta por não remover	1. O professor acessa a página Turmas. 2. O professor seleciona uma turma na listagem das suas turmas. 3. O professor clica em Excluir. 4. O sistema pede que o professor confirme a remoção. 5. O professor nega que deseja remover a turma. 6. O caso de uso se encerra.
--	--

Tabela 8 - UC007: GERENCIAR EXERCÍCIO

Caso de uso	UC007 – Gerenciar Exercício
Ator	Professor
Descrição	Permite que o professor liste, acesse, crie, edite e remova seus exercícios.
Pré condições	O professor deve ter cadastro no sistema e deve possuir pelo menos uma turma cadastrada.
Fluxo principal: Listar Exercícios	1. O professor acessa a página Detalhes de uma Turma. 2. O sistema busca todos os exercícios existentes para aquela turma. 3. O sistema exibe a listagem de exercícios.
Fluxo alternativo: Acessar Exercício	1. O professor acessa a página Detalhes de uma Turma. 2. O professor seleciona um exercício na listagem dos exercícios da turma. 3. O professor clica em acessar. 4. O sistema redireciona o professor a página de detalhes do exercício.
Fluxo alternativo: Criar Exercício	1. O professor acessa a página Detalhes de uma Turma. 2. O professor clica em Novo Exercício. 3. O professor seleciona a turma do exercício. Exemplo de preenchimento: 2021/2 4. O professor preenche o campo título. Exemplo de preenchimento: Exercício 1. 5. O professor preenche o campo descrição. Exemplo de preenchimento: Primos menor que 100

	6. O professor preenche o campo enunciado. Exemplo de preenchimento: Faça um programa que apresente os números primos menores que 100. 7. O professor seleciona se o exercício estará visível para outros professores ou não. Exemplo de preenchimento: Verdadeiro. 8. O professor seleciona as linguagens permitidas para resolver o exercício. Exemplo de preenchimento: Javascript, C. 9. O professor descreve as tags pertinentes para o exercício. Exemplo de preenchimento: Exercício de lógica. 10. O sistema sugere tags já criadas conforme o professor vai digitando. 11. O professor pode clicar em criar Dica para resolução. 12. O professor preenche o campo título da dica. Exemplo de preenchimento: Fórmula 13. O professor preenche o campo descrição da dica. Exemplo de preenchimento: Busque como categorizar números em primos e não primos. 14. O professor preenche o número de tentativas de respostas submetidas até a dica aparecer. Exemplo de preenchimento: 2
--	--

	15. O professor clica em Salvar dica. 16. O professor clica em Salvar o exercício. 17. O sistema registra um novo exercício e cria seus relacionamentos com linguagem, tag e dica. 18. O sistema exibe uma mensagem de sucesso e atualiza a página da turma com o novo exercício.
Fluxo alternativo: Editar Exercício	1. O professor acessa a página Detalhes de uma Turma. 2. O professor seleciona um exercício na listagem exercício da turma. 3. O professor clica em Editar. 4. O professor altera a turma do exercício. Exemplo de preenchimento: 2021/1 5. O professor altera o campo título. Exemplo de preenchimento: Exercício 2. 6. O professor altera o campo descrição. Exemplo de preenchimento: Primos menor que 50 7. O professor altera o campo enunciado. Exemplo de preenchimento: Faça um programa que apresente os números primos menores que 50. 8. O professor altera se o exercício estará visível para outros professores ou não. Exemplo de preenchimento: Falso. 9. O professor seleciona as linguagens permitidas para resolver o exercício. Exemplo de preenchimento: Javascript. 10. O professor altera as tags pertinentes para o exercício.

	Exemplo de preenchimento: Exercício inicial de programação. 11. O sistema sugere tags já criadas conforme o professor vai digitando. 12. O professor clica em uma Dica para resolução já criada. 13. O professor altera o campo título da dica. Exemplo de preenchimento: Primo 14. O professor altera o campo descrição da dica. Exemplo de preenchimento: Entenda como é definido um número primo. 15. O professor altera o número de tentativas de respostas submetidas até a dica aparecer. Exemplo de preenchimento: 1 16. O professor clica em Salvar dica. 17. O professor clica em Salvar o exercício. 18. O sistema atualiza o exercício e recria seus relacionamentos com linguagem, tag e dica. 19. O sistema exibe uma mensagem de sucesso e atualiza os exercícios listados na página da turma.
Fluxo alternativo: Remover Exercício	1. O professor acessa a página Detalhes de uma Turma. 2. O professor seleciona um exercício na listagem exercício da turma. 3. O professor clica em Excluir. 4. O sistema pede que o professor confirme a remoção.

	5. O professor confirma que deseja remover o exercício. 6. O sistema remove o exercício e todas as suas informações relacionadas. 7. O sistema exibe uma mensagem de sucesso e atualiza a página da turma para retirar o exercício excluída.
Fluxo alternativo: Remover Exercício – Professor opta por não remover	1. O professor acessa a página Detalhes de uma Turma. 2. O professor seleciona um exercício na listagem exercício da turma. 3. O professor clica em Excluir. 4. O sistema pede que o professor confirme a remoção. 5. O professor nega que deseja remover o exercício. 6. O caso de uso se encerra.

Tabela 9 - UC008: GERENCIAR TAG

Caso de uso	UC008 – Gerenciar Tag
Ator	Professor
Descrição	Permite que o professor liste, crie e remova tags de seus exercícios.
Pré condições	O professor deve ter cadastro no sistema, possuir pelo menos uma turma e um exercício cadastrado nesta turma.
Fluxo principal: Listar tags de um exercício	1. O professor acessa a página Detalhes de um Exercício. 2. O sistema busca todas as informações referentes àquele exercício, incluindo suas tags. 3. O sistema exibe o exercício e as tags que ele possui.

Fluxo alternativo: Criar tags em um exercício	1. O professor acessa a página Criar Exercício ou a página Editar Exercício. 2. O professor descreve as tags pertinentes para o exercício no campo tags. Exemplo de preenchimento: Exercício de lógica. 3. O sistema sugere tags já criadas conforme o professor vai digitando. 4. O professor clica em Salvar o exercício. 5. O sistema cria ou atualiza o exercício com as tags preenchidas. 6. O sistema exibe uma mensagem de sucesso.
Fluxo alternativo: Remover Tag	1. O professor acessa a página Criar Exercício ou a página Editar Exercício. 2. O professor clica no X para remover uma tag no campo tags. 3. O professor clica em Salvar o exercício. 4. O sistema cria ou atualiza o exercício apenas com as tags que não foram excluídas (se houver alguma). 5. O sistema exibe uma mensagem de sucesso.

Tabela 10 - UC009: GERENCIAR CASO DE TESTE

Caso de uso	UC009 – Gerenciar Caso de Teste
Ator	Professor
Descrição	Permite que o professor liste, crie, edite e remova casos de teste de seus exercícios.

Pré condições	O professor deve ter cadastro no sistema, possuir pelo menos uma turma e um exercício cadastrado nesta turma.
Fluxo principal: Listar Casos de Teste	1. O professor acessa a página Detalhes de um exercício. 2. O sistema busca todas as informações referentes àquele exercício, incluindo seus casos de teste. 3. O sistema exibe o exercício e os casos de teste que ele possui.
Fluxo alternativo: Criar Caso de Teste	1. O professor acessa a página Detalhes de um exercício. 2. O professor clica em Novo Caso de Teste. 3. O professor seleciona o exercício do caso de teste. Exemplo de preenchimento: Exercício 1 4. O professor preenche o campo título. Exemplo de preenchimento: Caso de teste 1. 5. O professor preenche o campo input com a entrada que será usada na execução para validar o caso de teste. Exemplo de preenchimento: 2 6. O professor preenche o campo output com a saída esperada após a execução do código. Exemplo de preenchimento: 5 7. O professor clica em Salvar. 8. O sistema registra um novo caso de teste.

	9. O sistema exibe uma mensagem de sucesso e atualiza a página do exercício para mostrar o novo caso de teste.
Fluxo alternativo: Editar Caso de Teste	1. O professor acessa a página Detalhes de um exercício. 2. O professor seleciona um caso de teste na listagem do exercício. 3. O professor clica em Editar. 4. O professor altera o exercício do caso de teste. Exemplo de preenchimento: Exercício 2 5. O professor altera o campo título. Exemplo de preenchimento: Caso de teste 2. 6. O professor altera o campo input com a entrada que será usada na execução para validar o caso de teste. Exemplo de preenchimento: 3 7. O professor altera o campo output com a saída esperada após a execução do código. Exemplo de preenchimento: 6 8. O professor clica em Salvar. 9. O sistema atualiza o caso de teste com as informações fornecidas. 10. O sistema exibe uma mensagem de sucesso e atualiza a página do exercício.
Fluxo alternativo: Remover Caso de Teste	1. O professor acessa a página Detalhes de um exercício. 2. O professor seleciona um caso de teste na listagem do exercício. 3. O professor clica em Excluir.

	- O sistema pede que o professor confirme a remoção. - O professor confirma que deseja remover o caso de teste. - O sistema remove o caso de teste e todas as suas informações relacionadas. - O sistema exibe uma mensagem de sucesso e atualiza a página do exercício para retirar o caso de teste excluído.
Fluxo alternativo: Remover Caso de Teste – Professor opta por não remover	- O professor acessa a página Detalhes de um exercício. - O professor seleciona um caso de teste na listagem do exercício. - O professor clica em Excluir. - O sistema pede que o professor confirme a remoção. - O professor nega que deseja remover o caso de teste. - O caso de uso se encerra.

Tabela 11 - UC010: EXECUTAR CASO DE TESTE

Caso de uso	UC010– Executar Caso de Teste
Ator	Professor
Descrição	Permite que o professor execute casos de teste de seus exercícios para fins de teste e validação.
Pré condições	O professor deve ter cadastro no sistema, possuir pelo menos uma turma, um exercício cadastrado nesta turma e um caso de teste deste exercício.
Fluxo principal: Executar Caso de Teste	O professor acessa a página Detalhes de uma Turma.

	2. O professor seleciona um exercício na listagem exercício da turma. 3. O professor clica em Executar. 4. O sistema redireciona o professor para a página de Editor. 5. O professor seleciona o caso de teste que deseja validar. Exemplo de preenchimento: Caso de Teste 1 6. O professor insere o código-fonte que deve ser executado para validar o caso de teste. 7. O professor seleciona a linguagem do código-fonte inserido. 8. O professor clica em executar. 9. O sistema executa o código-fonte fornecido utilizando com entrada o valor do campo input do caso de teste. 10. O sistema verifica que o output obtido após a execução do código-fonte fornecido é exatamente igual ao output cadastrado no caso de teste. 11. O sistema retorna uma mensagem pro professor mostrando o output obtido e que ele está correto de acordo com o output do caso de teste.
Fluxo alternativo: Executar Caso de Teste – Output obtido pela execução não confere	1. O professor acessa a página Detalhes de um exercício. 2. O professor seleciona um caso de teste na listagem do exercício. 3. O professor clica em Executar. 4. O professor seleciona o caso de teste que deseja validar.

	Exemplo de preenchimento: Caso de Teste 1 5. O professor insere o código-fonte que deve ser executado para validar o caso de teste. 6. O professor seleciona a linguagem do código-fonte inserido. 7. O professor clica em executar. 8. O sistema executa o código-fonte fornecido utilizando com entrada o valor do campo input do caso de teste. 9. O sistema verifica que o output obtido após a execução do código-fonte fornecido é não confere com o output cadastrado no caso de teste. 10. O sistema retorna uma mensagem pro professor indicando o output obtido e que ele está incorreto de acordo com o output do caso de teste.
Fluxo alternativo: Executar Caso de Teste – Erro de compilação/execução do código	1. O professor acessa a página Detalhes de um exercício. 2. O professor seleciona um caso de teste na listagem do exercício. 3. O professor clica em Executar. 4. O professor seleciona o caso de teste que deseja validar. Exemplo de preenchimento: Caso de Teste 1 5. O professor insere o código-fonte que deve ser executado para validar o caso de teste, com erros de sintaxe. 6. O professor seleciona a linguagem do código-fonte inserido.

	7. O professor clica em executar. 8. O sistema executa o código-fonte fornecido utilizando com entrada o valor do campo input do caso de teste. 9. O sistema verifica houve um erro durante a execução do código. 11. O sistema retorna uma mensagem pro professor indicando o erro obtido na execução.
--	---

Tabela 12 - UC011: VISUALIZAR EXERCÍCIOS PÚBLICOS

Caso de uso	UC011– Visualizar Exercícios Públicos
Ator	Professor
Descrição	Permite que o professor liste exercícios públicos.
Pré condições	O professor deve ter cadastro no sistema.
Fluxo principal: Visualizar Exercícios Públicos	1. O professor acessa a página Exercícios. 2. O sistema busca todos os exercícios públicos que existem na universidade do professor, juntamente com os exercícios privados ou públicos que pertencem ao professor. 3. O sistema exibe uma listagem dos exercícios.

Tabela 13 - UC012: COPIAR EXERCÍCIO PÚBLICO PARA UMA TURMA

Caso de uso	UC012– Copiar Exercício Público Para Uma Turma
Ator	Professor
Descrição	Permite que o professor replique um exercício público, copiando-o para uma das turmas pertencentes a ele.

Pré condições	O professor deve ter cadastro no sistema e possuir pelo menos uma turma.
Fluxo principal: Copiar Exercício Público Para Uma Turma	1. O professor acessa a página Exercícios. 2. O professor seleciona um exercício da listagem. 3. O professor clica em Replicar. 4. O professor seleciona a turma na qual gostaria de replicar aquele exercício. 5. O professor clica em Confirmar. 6. O sistema copia todas as informações do exercício, incluindo suas tags, linguagens e dicas para um novo exercício. 7. O sistema coloca no campo <code>exercicioOriginal</code> deste novo exercício o id do exercício sendo copiado. 8. O sistema registra esse novo exercício idêntico ao outro, mas pertencendo à turma selecionada pelo professor. 9. O sistema exibe uma mensagem de sucesso para o professor.

Tabela 14 - UC013: SE INSCREVER TURMA

Caso de uso	UC013 – Se Inscrever Turma
Ator	Aluno
Descrição	Permite que, fornecendo o código de acesso, o aluno se matricule em uma turma.
Pré condições	O aluno deve ter cadastro no sistema. A turma deve estar cadastrada no sistema. A turma deve pertencer a mesma instituição que o aluno.
Fluxo principal: Se Inscrever Turma	1. O aluno acessa a página Turmas. 2. O aluno clica em Se Inscrever.

	3. O aluno preenche o campo código de acesso da turma. Exemplo de preenchimento: ds122 4. O aluno clica em Salvar. 5. O sistema verifica que existe uma turma de status “ativa” ou “pausada” e da mesma instituição que o aluno que tem esse código de acesso. 6. O sistema cadastra o aluno na turma. 7. O sistema retorna uma mensagem de sucesso.
Fluxo alternativo: Se Inscrever Turma – Código de acesso não corresponde a nenhuma turma	1. O aluno acessa a página Turmas. 2. O aluno clica em Se Inscrever. 3. O aluno preenche o campo código de acesso da turma. Exemplo de preenchimento: ds122a 4. O aluno clica em Salvar. 5. O sistema verifica que não existe nenhuma turma de status “ativa” ou “pausada” e da mesma instituição que o aluno que tem esse código de acesso. 6. O sistema retorna uma mensagem de erro.

Tabela 15 - UC014: CRIAR RESPOSTA

Caso de uso	UC014 – Criar Resposta
Ator	Aluno
Descrição	Permite que o aluno cadastre tentativas de resposta à um exercício.
Pré condições	O aluno deve ter cadastro no sistema. O aluno deve estar cadastrado em uma turma. A turma deve ter pelo menos um exercício.
Fluxo principal: Criar Resposta	1. O aluno acessa a página Detalhes do Exercício. 2. O aluno seleciona a linguagem de programação da resposta. Exemplo de preenchimento: Javascript. 3. O aluno preenche o código-fonte do exercício. Exemplo de preenchimento: <code>console.log(3+2);</code> 4. O aluno seleciona que a resposta que ele submeterá é final. Exemplo de preenchimento: Verdadeiro. 5. O aluno clica em Salvar. 6. O sistema registra a resposta do aluno. 7. O sistema verifica que a resposta é final e não permite mais envios do aluno para aquele exercício. 8. O sistema roda o UC015 – Executar Resposta. 9. O sistema roda o UC016 – Validar Resposta. 10. O sistema retorna o resultado da validação.

Fluxo alternativo: Criar Resposta – Tentativa de resposta não é final	1. O aluno acessa a página Detalhes do Exercício. 2. O aluno seleciona a linguagem de programação da resposta. Exemplo de preenchimento: Javascript. 3. O aluno preenche o código-fonte do exercício. 4. O aluno seleciona que a resposta que ele submeterá é final. Exemplo de preenchimento: Falso. 5. O aluno clica em Salvar. 6. O sistema registra a resposta do aluno. 7. O sistema verifica que a resposta não é final e continua permitindo mais envios do aluno para aquele exercício. 8. O sistema busca os casos de teste cadastrados para o exercício. 9. O sistema roda o UC015 – Executar Resposta. 10. O sistema roda o UC016 – Validar Resposta. O sistema retorna o resultado da validação.
Fluxo alternativo: Criar Resposta – Linguagem não permitida no exercício	1. O aluno acessa a página Detalhes do Exercício. 2. O aluno seleciona uma linguagem de programação da resposta que o exercício não permite. Exemplo de preenchimento: C. 3. O aluno preenche o código-fonte do exercício. 4. O aluno seleciona que a resposta que ele submeterá é final.

	Exemplo de preenchimento: Falso. - O aluno clica em Salvar. - O sistema verifica que o exercício não permite respostas naquela linguagem de programação. - O sistema retorna um erro para o aluno.
Fluxo alternativo: Criar Resposta – Aluno submete resposta em turma que não está ativa	- O aluno acessa a página Detalhes do Exercício. - O aluno preenche os campos da Resposta. - O aluno clica em Salvar. - O sistema verifica que o status da Turma não é ativo. - O sistema retorna um erro para o aluno.

Tabela 16 - UC015: EXECUTAR RESPOSTA

Caso de uso	UC015 – Executar Resposta
Ator	Sistema
Descrição	Permite que o sistema execute o código-fonte fornecido pelo aluno ou professor.
Pré condições	O sistema deve ter acesso ao sandbox. O sistema deve possuir um caso de teste que está sendo coberto. O sistema deve possuir um código-fonte que será executado.
Fluxo principal: Executar Resposta	- O sistema busca o caso de teste que vai cobrir a execução do código. - O sistema faz uma requisição ao sandbox para executar o código fonte na linguagem pedida, e usando o parâmetro de entrada “input” registrado no caso de teste.

	3. O sistema recebe a resposta do sandbox e retorna o output da execução do código.
--	---

Tabela 17 - UC016: VALIDA CASO TESTE

Caso de uso	UC016 – Valida Caso Teste
Ator	Sistema
Descrição	Permite que o sistema valide o output da execução de um código-fonte de acordo com o output esperado em um caso de teste.
Pré condições	O sistema deve possuir um caso de teste sendo coberto. O sistema deve possuir o output da execução de um código-fonte.
Fluxo principal: Valida Caso Teste	1. O sistema compara o output da execução do código-fonte com o output esperado pelo caso de teste. 2. O sistema verifica que eles são idênticos e registra o output recebido e o status correto relacionado à resposta do aluno naquele caso de teste.
Fluxo alternativo: Valida Caso Teste – Output não coincide com o output esperado pelo caso de teste	1. O sistema compara o output da execução do código-fonte com o output esperado pelo caso de teste. 2. O sistema verifica que eles não são idênticos e registra o output recebido e o status incorreto relacionado à resposta do aluno naquele caso de teste.

Tabela 18 - UC017: CANCELAR INSCRIÇÃO TURMA

Caso de uso	UC017 – Cancelar Inscrição Turma
Ator	Aluno

Descrição	Permite que o aluno cancele sua matrícula em uma turma.
Pré condições	O aluno deve ter cadastro no sistema. O aluno deve estar cadastrado em uma turma.
Fluxo principal: Cancelar Inscrição Turma	1. O aluno acessa a página Detalhes de uma Turma. 2. O aluno clica em Cancelar Inscrição. 3. O sistema pede que o aluno confirme o cancelamento. 4. O aluno confirma que deseja cancelar a inscrição na turma. 5. O sistema remove o aluno da turma. 6. O sistema retorna uma mensagem de sucesso e redireciona o aluno para a página Turmas.
Fluxo alternativo: Cancelar Inscrição Turma – O aluno não confirma o cancelamento	1. O aluno acessa a página Detalhes de uma Turma. 2. O aluno clica em Cancelar Inscrição. 3. O sistema pede que o aluno confirme o cancelamento. 4. O aluno nega que deseja cancelar a inscrição na turma. 5. O caso de uso se encerra.

Tabela 19 - UC018: VISUALIZAR EXERCÍCIOS TURMA

Caso de uso	UC018 – Visualizar Exercícios Turma
Ator	Aluno
Descrição	Permite que o aluno liste os exercícios de uma turma.
Pré condições	O aluno deve ter cadastro no sistema. O aluno deve estar cadastrado em uma turma. A turma deve possuir pelo menos um exercício.

Fluxo principal: Visualizar Exercícios Turma	1. O aluno acessa a página Detalhes de uma Turma. 2. O sistema busca todos os exercícios existentes para aquela turma. 3. O sistema exibe a listagem de exercícios da turma.
--	--

Tabela 20 - UC019: VISUALIZAR TURMAS INSCRITAS

Caso de uso	UC019 – Visualizar Turmas Inscritas
Ator	Aluno
Descrição	Permite que o aluno liste as turmas em que está inscrito.
Pré condições	O aluno deve ter cadastro no sistema. O aluno deve estar cadastrado em pelo menos uma turma.
Fluxo principal: Visualizar Turmas Inscritas	1. O aluno acessa a página Turmas. 2. O sistema busca todas as turmas em que o aluno está inscrito. 3. O sistema exibe a listagem de turmas.

Tabela 21 - UC020: GERENCIAR INSTITUIÇÃO

Caso de uso	UC020 – Gerenciar Instituição
Ator	Admin
Descrição	Permite que o admin liste, acesse, crie, edite e remova instituições.
Pré condições	O admin deve ter cadastro no sistema.
Fluxo principal: Listar instituições	1. O admin acessa a página Instituições. 2. O sistema busca todas as instituições cadastradas. 3. O sistema exibe a listagem de instituições.
Fluxo alternativo: Acessar instituição	1. O admin acessa a página Instituições.

	2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. O sistema mostra os dados referentes à instituição selecionada.
Fluxo alternativo: Criar instituição	1. O admin acessa a página Instituições. 2. O admin clica em Nova Instituição. 3. O admin preenche o campo nome da instituição. Exemplo de preenchimento: Universidade Federal do Paraná 4. O admin preenche o campo CNPJ. Exemplo de preenchimento: 69075252000186 5. O admin preenche o campo CEP. Exemplo de preenchimento: 80060000 6. O admin preenche o campo endereço. Exemplo de preenchimento: Rua XV de Novembro 7. O admin preenche o campo número. Exemplo de preenchimento: 1299 8. O admin preenche o campo UF. Exemplo de preenchimento: PR 9. O admin preenche o campo cidade. Exemplo de preenchimento: Curitiba 10. O admin preenche o campo bairro. Exemplo de preenchimento: Centro 11. O admin clica em Salvar. 12. O sistema registra as informações inseridas pelo admin e cria a instituição. 13. O sistema exibe uma mensagem de sucesso e atualiza a página de instituições com a nova instituição.

Fluxo alternativo: Editar instituição	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Editar. 4. O admin atualiza o campo nome da instituição. Exemplo de preenchimento: Universidade Tecnológica Federal do Paraná 5. O admin atualiza o campo CNPJ. Exemplo de preenchimento: 73417735000199 6. O admin atualiza o campo CEP. Exemplo de preenchimento: 80230901 7. O admin atualiza o campo endereço. Exemplo de preenchimento: Av. Sete de Setembro 8. O admin atualiza o campo número. Exemplo de preenchimento: 3165 9. O admin atualiza o campo UF. Exemplo de preenchimento: PR 10. O admin atualiza o campo cidade. Exemplo de preenchimento: Curitiba 11. O admin atualiza o campo bairro. Exemplo de preenchimento: Rebouças 12. O admin clica em Salvar. 13. O sistema atualiza a instituição de acordo com os dados fornecidos. 14. O sistema exibe uma mensagem de sucesso e atualiza a página de instituições.
Fluxo alternativo: Remover instituição	1. O admin acessa a página Instituições.

	2. O admin seleciona uma instituição na listagem. 3. O admin clica em Excluir. 4. O sistema pede que o admin confirme a remoção. 5. O admin confirma que deseja remover a instituição. 6. O sistema remove a instituição e todas as suas informações relacionadas. 7. O sistema exibe uma mensagem de sucesso e atualiza a página de instituições para retirar a instituição excluída.
Fluxo alternativo: Remover instituição – Admin opta por não remover	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Excluir. 4. O sistema pede que o admin confirme a remoção. 5. O admin nega que deseja remover a instituição. 6. O caso de uso se encerra.

Tabela 22 - UC021: GERENCIAR CONTATOS INSTITUIÇÃO

Caso de uso	UC021 – Gerenciar Contatos Instituição
Ator	Admin
Descrição	Permite que o admin liste, crie, edite e remova contatos de uma instituição.
Pré condições	O admin deve ter cadastro no sistema. O sistema deve ter pelo menos uma instituição cadastrada.

Fluxo principal: Listar Contatos de uma Instituição	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. O sistema mostra os dados referentes à instituição selecionada, incluindo todos os seus contatos.
Fluxo alternativo: Criar Contato de uma Instituição	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. O admin clica em Novo Contato. 5. O admin preenche o campo tipo. Exemplo de preenchimento: Telefone. 6. O admin preenche o campo contato. Exemplo de preenchimento: 4133605000 7. O admin preenche o campo pessoa. Exemplo de preenchimento: Secretaria. 8. O admin clica em Salvar. 9. O sistema registra as informações inseridas pelo admin e cria o contato da instituição. 10. O sistema exibe uma mensagem de sucesso e atualiza a página de Detalhes da instituição com o novo contato.
Fluxo alternativo: Editar Contato de uma Instituição	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. Na seção Contatos, o admin seleciona um contato. 5. O admin clica em Editar.

	6. O admin atualiza o campo tipo. Exemplo de preenchimento: Email. 7. O admin atualiza o campo contato. Exemplo de preenchimento: secretaria@ufpr.br 8. O admin atualiza o campo pessoa. Exemplo de preenchimento: Secretaria. 9. O admin clica em Salvar. 10. O sistema atualiza o contato da instituição com as novas informações inseridas. 11. O sistema exibe uma mensagem de sucesso e atualiza a página de Detalhes da instituição.
Fluxo alternativo: Remover Contato de uma Instituição	1. O admin acessa a página Instituições. 2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. Na seção Contatos, o admin seleciona um contato. 5. O admin clica em Excluir. 6. O sistema pede que o admin confirme a remoção. 7. O admin confirma que deseja remover o contato da instituição. 8. O sistema remove o contato da instituição. 9. O sistema exibe uma mensagem de sucesso e atualiza a página de Detalhes da instituição para retirar o contato excluído.
Fluxo alternativo: Remover Contato de uma Instituição	1. O admin acessa a página Instituições.

– Admin opta por não remover	2. O admin seleciona uma instituição na listagem. 3. O admin clica em Visualizar. 4. Na seção Contatos, o admin seleciona um contato. 5. O admin clica em Excluir. 6. O sistema pede que o admin confirme a remoção. 7. O admin nega que deseja remover o contato da instituição. 8. O caso de uso se encerra.
------------------------------	--

Tabela 23 - UC022: CADASTRAR USUÁRIO

Caso de uso	UC022 – Cadastrar Usuário
Ator	Admin
Descrição	Permite que o admin crie usuários com o perfil admin.
Pré condições	O admin deve ter cadastro no sistema.
Fluxo principal: Cadastrar Usuário	1. O admin acessa a página Usuários. 2. O admin clica em Novo Usuário. 3. O admin seleciona o perfil de acesso do usuário que será criado. Exemplo de preenchimento: Admin 4. O admin preenche o campo nome. Exemplo de preenchimento: José Ferreira. 5. O admin preenche o campo e-mail. Exemplo de preenchimento: jose.ferreira@gmail.com 6. O admin preenche o telefone. Exemplo de preenchimento: 41992525252 7. O admin preenche o campo avatar. Exemplo de preenchimento: https://avatars.dicebear.com/api/identicon/farmaalg-1799694199098878.svg?b=%23ffffff

	8. O admin preenche o campo senha. Exemplo de preenchimento: Senha854 9. O admin preenche o campo confirmar senha. Exemplo de preenchimento: Senha854 10. O admin clica em Salvar. 11. O sistema registra as informações do novo usuário. 12. O sistema retorna uma mensagem de sucesso e atualiza a página Usuários com o novo usuário.
Fluxo alternativo: Cadastrar Usuário – E-mail não disponível	1. O admin acessa a página Usuários. 2. O admin clica em Novo Usuário. 3. O admin seleciona o perfil de acesso do usuário que será criado. Exemplo de preenchimento: Admin 4. O admin preenche o resto dos dados, preenchendo o campo email com um email que não está disponível. 5. O admin clica no botão de cadastrar. 6. O sistema verifica que o email fornecido pelo usuário não está disponível. 7. O sistema retorna uma mensagem de erro para o admin e pede que forneça um novo email.

Tabela 24 - UC023: TROCAR SENHA DE USUÁRIO

Caso de uso	UC023 – Trocar Senha De Usuário
Ator	Admin
Descrição	Permite que o admin troque a senha dos usuários para eventuais casos de perda de senha.
Pré condições	O admin deve ter cadastro no sistema.
Fluxo principal: Trocar Senha De Usuário	1. O admin acessa a página Usuários. 2. O admin seleciona um usuário na listagem. 3. O admin clica em Editar. 4. Na seção Atualizar Senha, o admin preenche o campo nova senha.

	Exemplo de preenchimento: Senha123 5. O admin preenche o confirmar nova senha. Exemplo de preenchimento: Senha123 6. O admin clica em Salvar. 7. O sistema atualiza a senha do usuário selecionado. 8. O sistema retorna uma mensagem de sucesso.
Fluxo alternativo: Trocar Senha De Usuário – Confirmação de senha incorreta	1. O admin acessa a página Usuários. 2. O admin seleciona um usuário na listagem. 3. O admin clica em Editar. 4. Na seção Atualizar Senha, o admin preenche o campo nova senha. Exemplo de preenchimento: Senha123 5. O admin preenche o confirmar nova senha. Exemplo de preenchimento: Senha123 6. O admin clica em Salvar. 7. O sistema identifica que a senha informada não está igual a confirmação da senha. 8. O sistema retorna uma mensagem de erro para o usuário.

Tabela 25 - UC024: GERENCIAR LINGUAGEM

Caso de uso	UC024 – Gerenciar Linguagem
Ator	Admin
Descrição	Permite que o admin liste, crie, edite e remova linguagens de programação.
Pré condições	O admin deve ter cadastro no sistema.
Fluxo principal: Listar Linguagens	1. O admin acessa a página Linguagens. 2. O sistema busca todas as linguagens cadastradas. 3. O sistema exibe a listagem de linguagens.
Fluxo alternativo: Criar Linguagem	1. O admin acessa a página Linguagens. 2. O admin clica em Nova Linguagem.

	3. O admin preenche o campo nome da linguagem. Exemplo de preenchimento: Javascript. 4. O admin preenche o campo código Sandbox. Exemplo de preenchimento: js 5. O admin preenche o campo exemplo. Exemplo de preenchimento: console.log(`hello, world!`); 6. O admin clica em Salvar. 7. O sistema registra as informações inseridas pelo admin e cria a linguagem. 8. O sistema exibe uma mensagem de sucesso e atualiza a página de linguagens com o novo contato.
Fluxo alternativo: Editar Linguagem	1. O admin acessa a página Linguagens. 2. O admin seleciona uma linguagem na listagem. 3. O admin clica em Editar. 4. O admin altera o campo nome da linguagem. Exemplo de preenchimento: PHP. 5. O admin altera o campo código Sandbox. Exemplo de preenchimento: php 6. O admin altera o campo exemplo. Exemplo de preenchimento: <?php echo "hello world!\n"; 7. O admin clica em Salvar. 8. O admin clica em Salvar. 9. O sistema atualiza a linguagem com as novas informações inseridas.

	10. O sistema exibe uma mensagem de sucesso e atualiza a página de linguagens.
Fluxo alternativo: Remover Linguagem	1. O admin acessa a página Linguagens. 2. O admin seleciona uma linguagem na listagem. 3. O admin clica em Excluir. 4. O sistema pede que o admin confirme a remoção. 5. O admin confirma que deseja remover a linguagem. 6. O sistema remove a linguagem. 1. O sistema exibe uma mensagem de sucesso e atualiza a página de linguagens para retirar a linguagem excluída.
Fluxo alternativo: Remover Linguagem – Admin opta por não remover	1. O admin acessa a página Linguagens. 2. O admin seleciona uma linguagem na listagem. 3. O admin clica em Excluir. 4. O sistema pede que o admin confirme a remoção. 5. O admin nega que deseja remover a linguagem. 6. O caso de uso se encerra.

APÊNDICE C – DIAGRAMA DE SEQUÊNCIA

Figura 24 – DS001: CADASTRO DE USUÁRIO

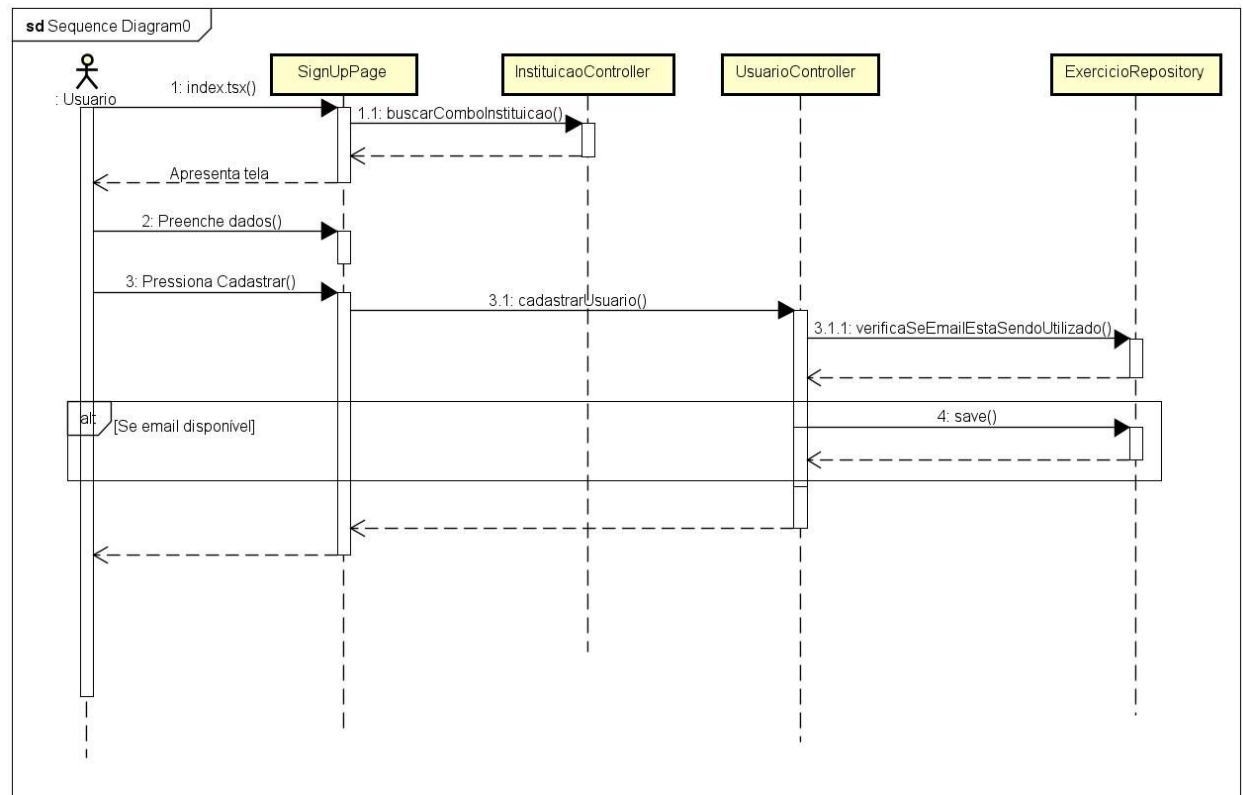


Figura 25 – DS002: LOGIN DE USUÁRIO

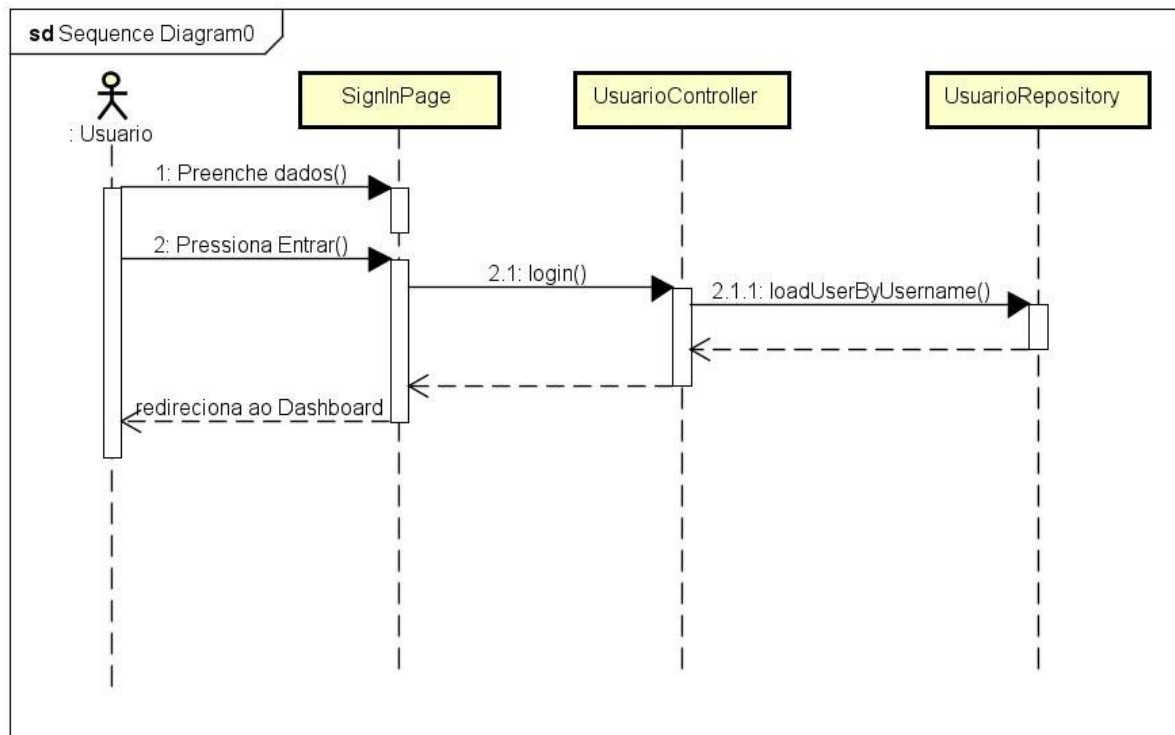


Figura 26 – DS003: ATUALIZAR USUÁRIO

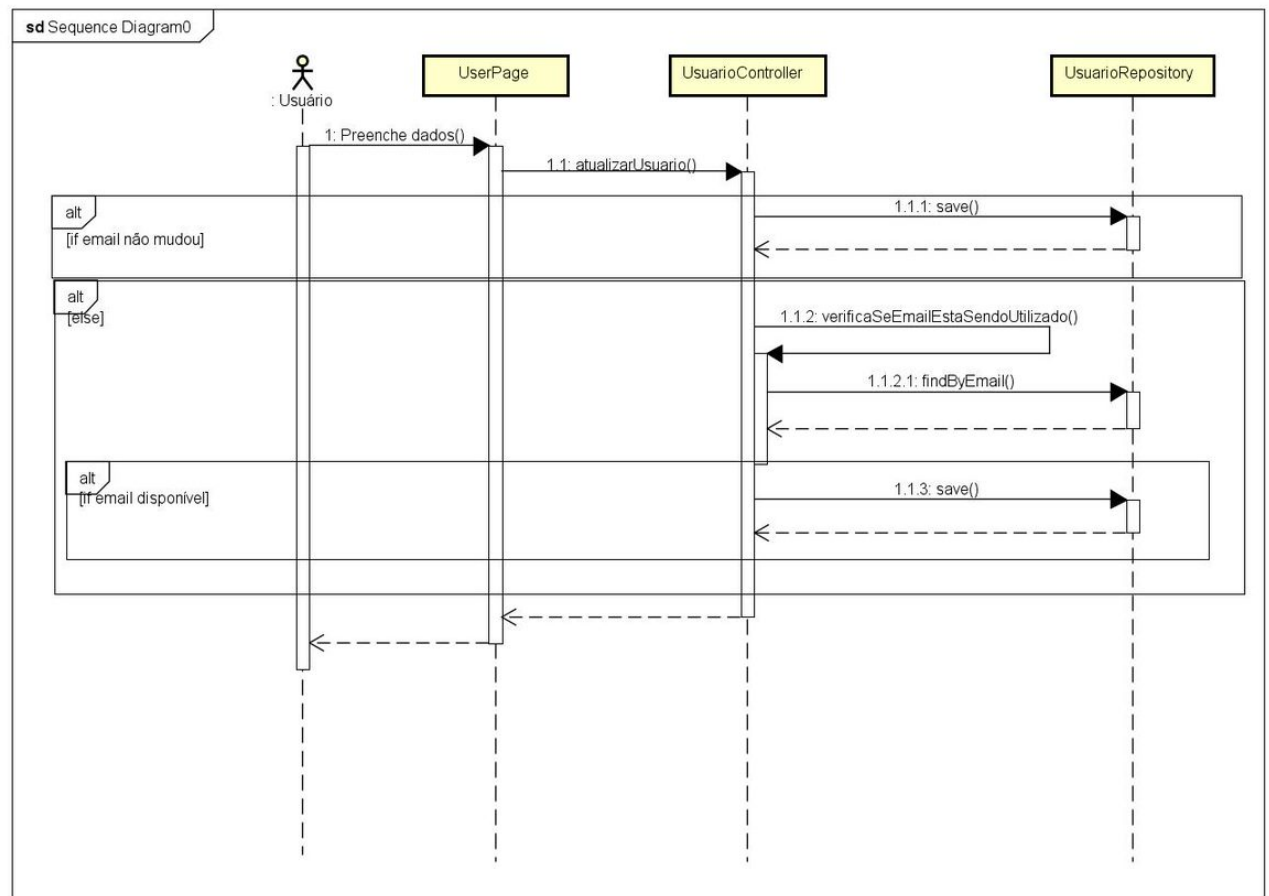


Figura 27 – DS004: TROCAR SENHA DE USUÁRIO

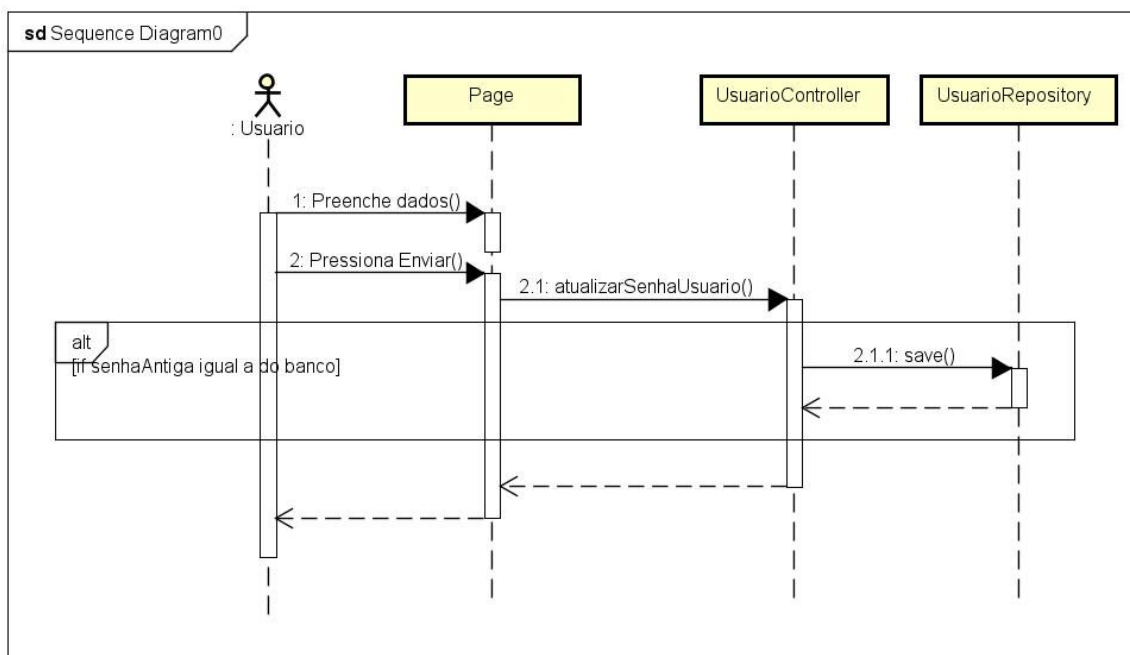


Figura 28 – DS005: REMOVER USUÁRIO

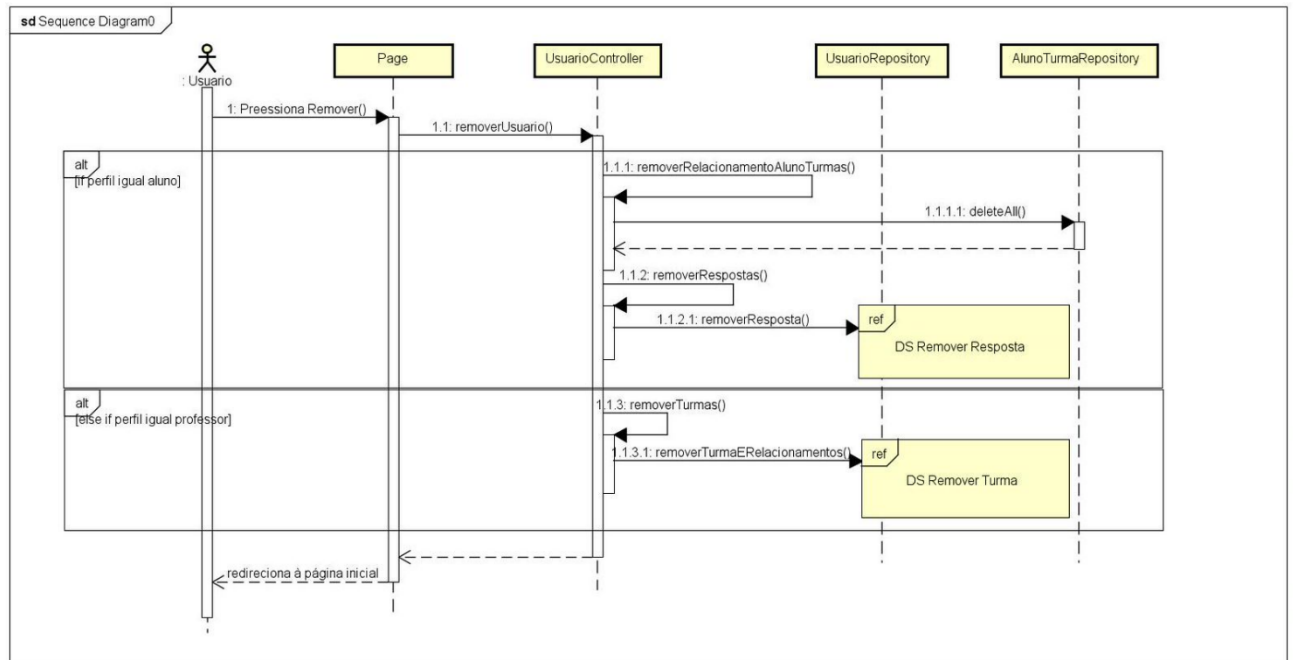


Figura 29 – DS006: GERENCIAR TURMAS (LISTAR)

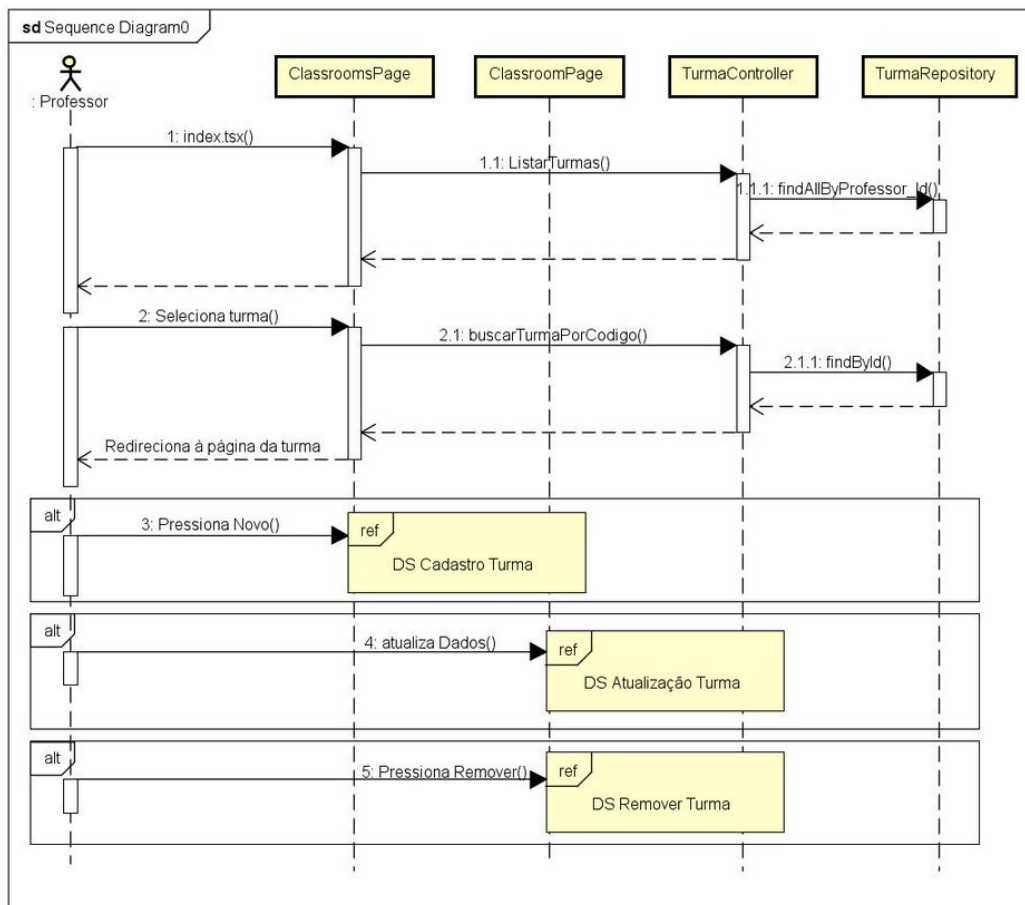


Figura 30 – DS007: GERENCIAR TURMAS (CADASTRAR)

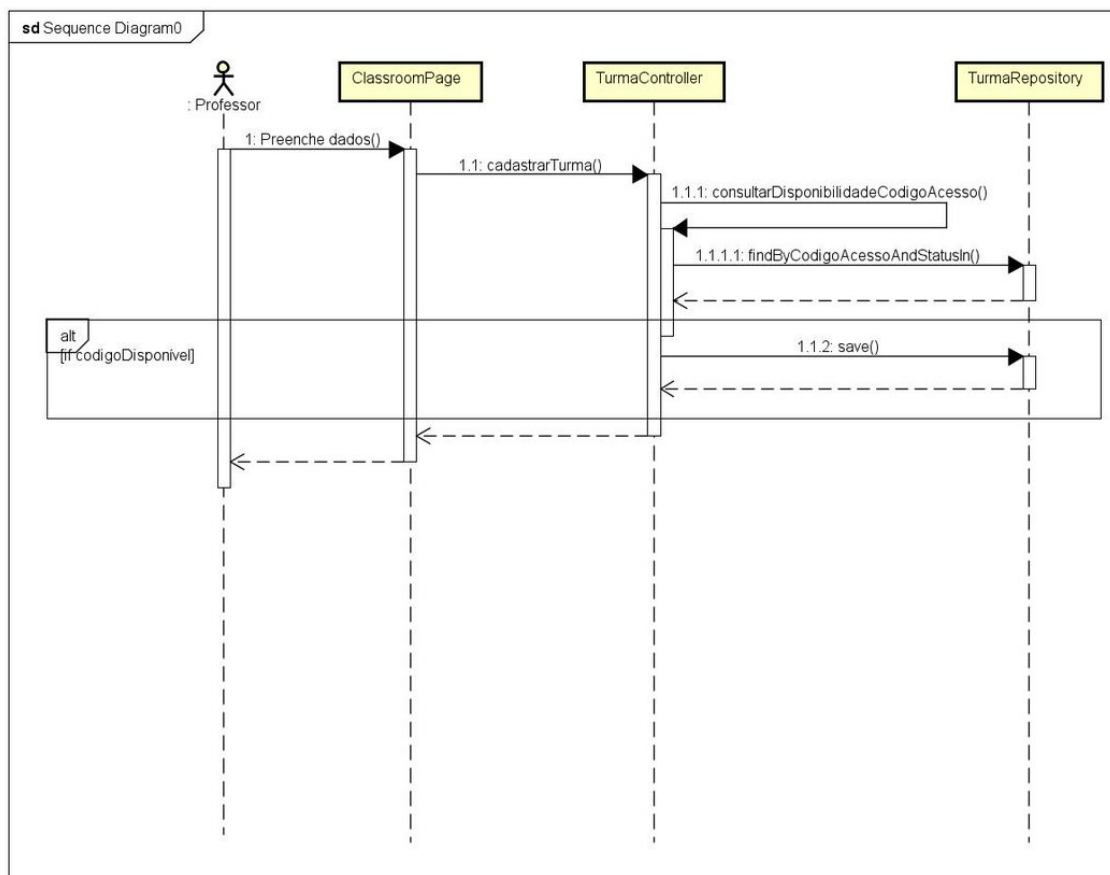


Figura 31 – DS008: GERENCIAR TURMAS (ATUALIZAR)

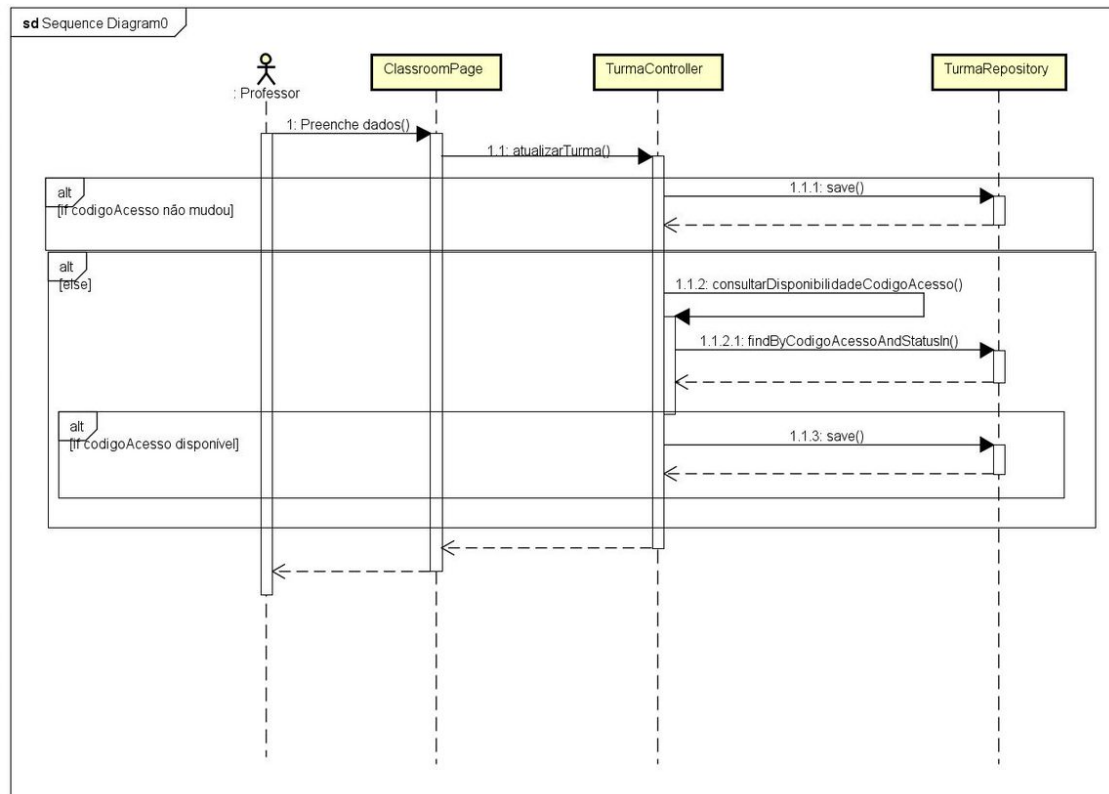


Figura 32 – DS009: GERENCIAR TURMAS (REMOVER)

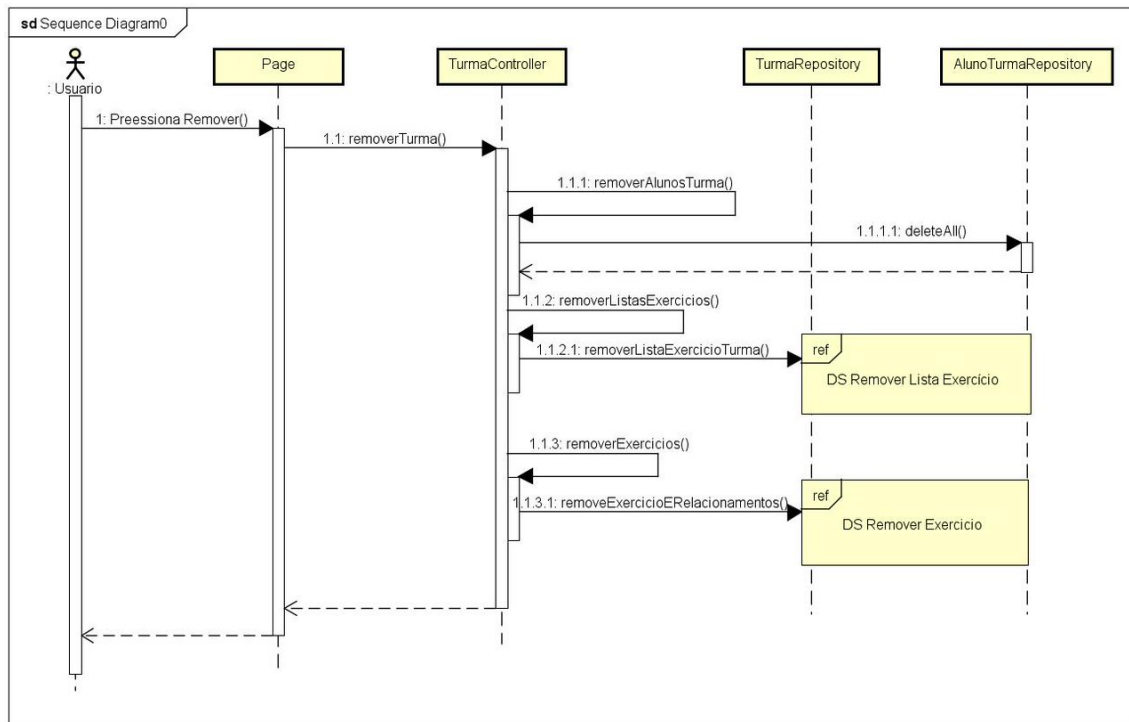


Figura 33 – DS010: GERENCIAR EXERCÍCIOS (LISTAR PÚBLICOS)

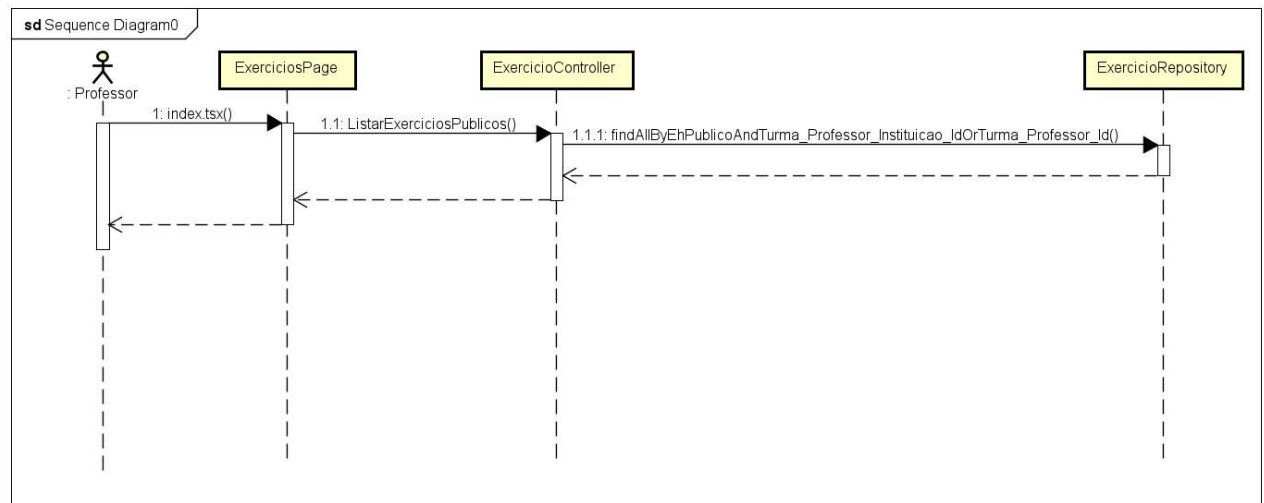


Figura 34 – DS011: GERENCIAR EXERCÍCIOS (BUSCAR)

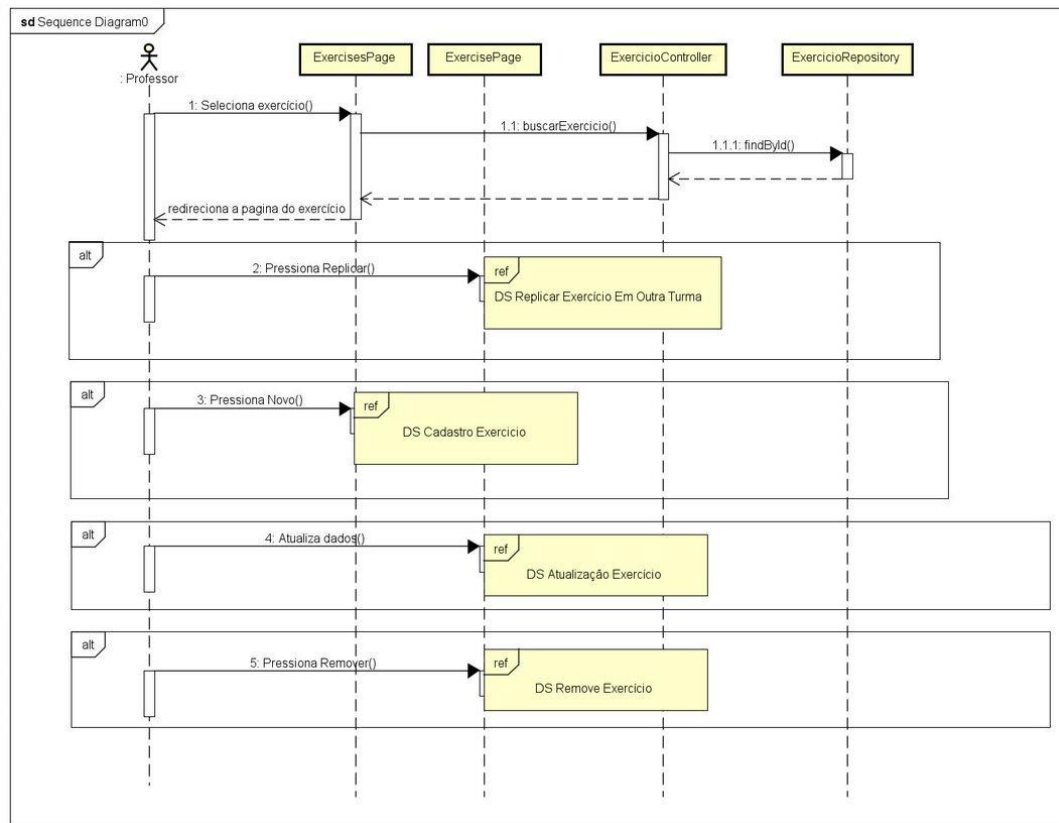


Figura 35 – DS012: GERENCIAR EXERCÍCIOS (CADASTRAR)

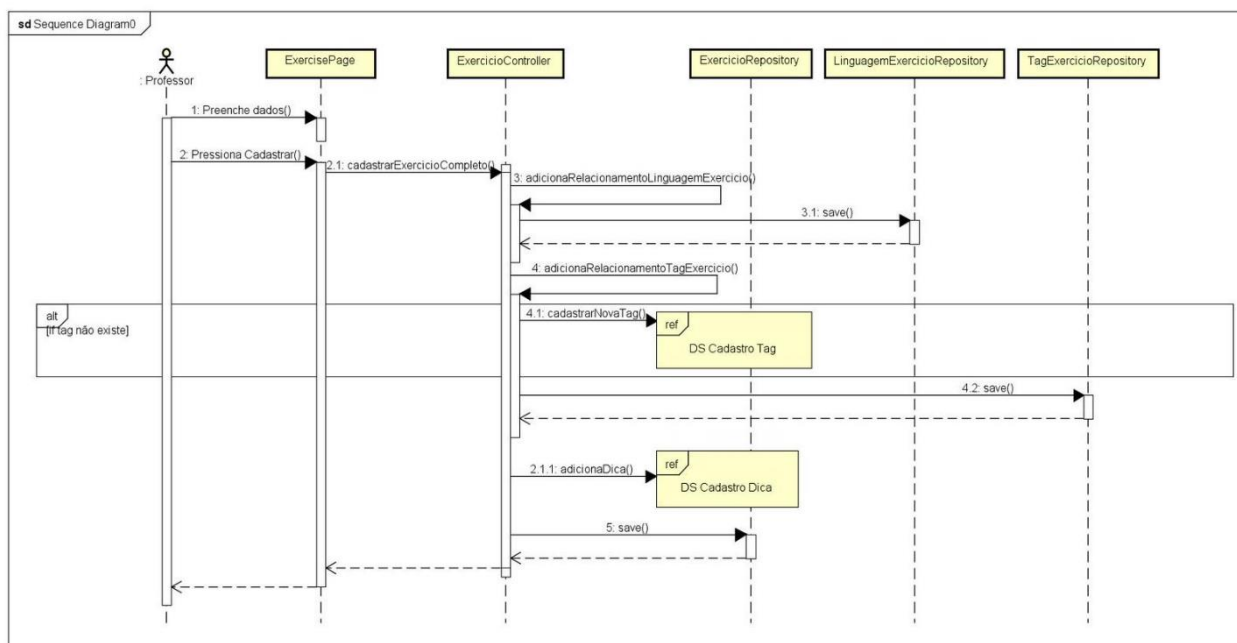


Figura 36 – DS013: GERENCIAR EXERCÍCIOS (ATUALIZAR)

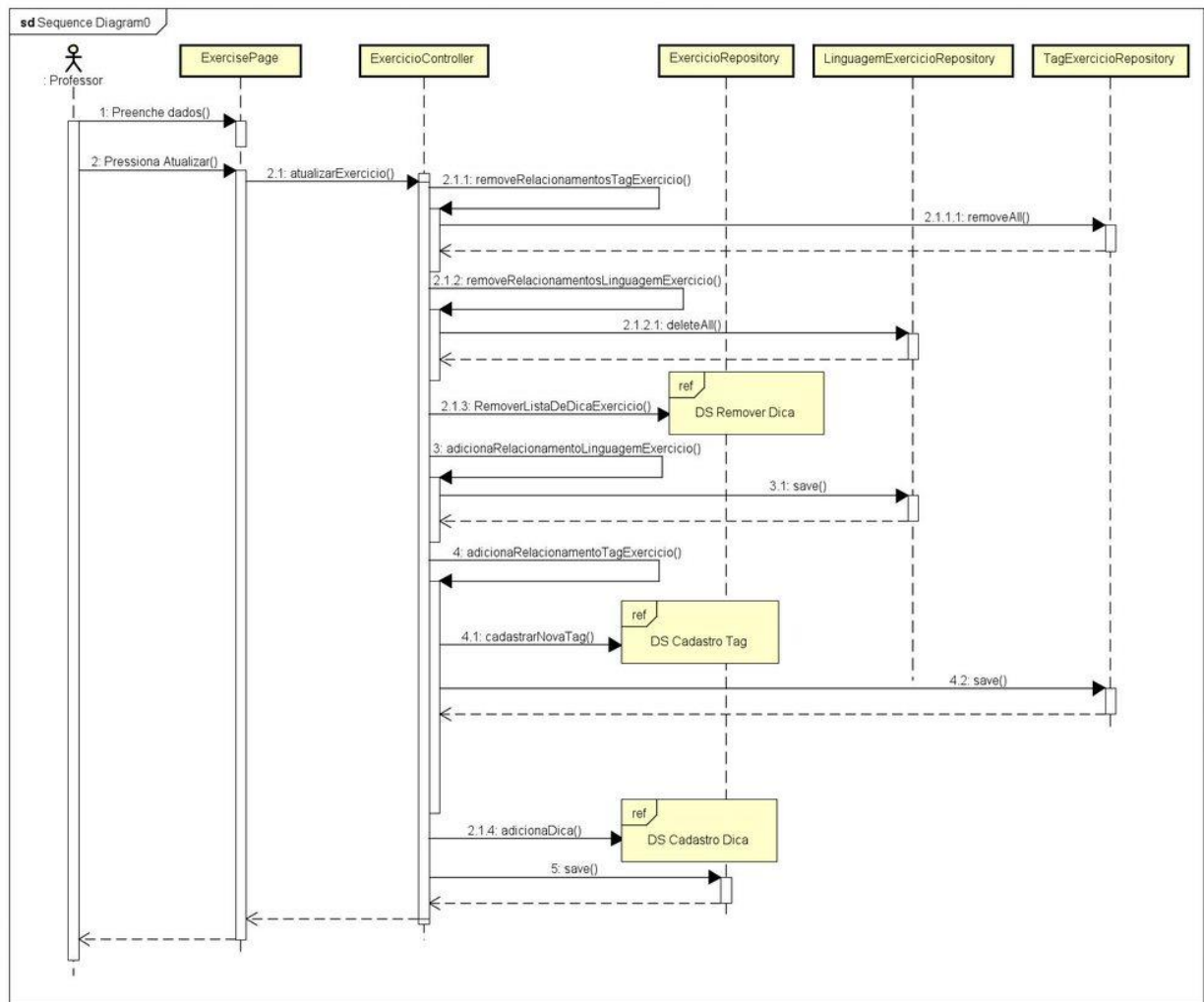


Figura 37 – DS014: GERENCIAR EXERCÍCIOS (REMOVER)

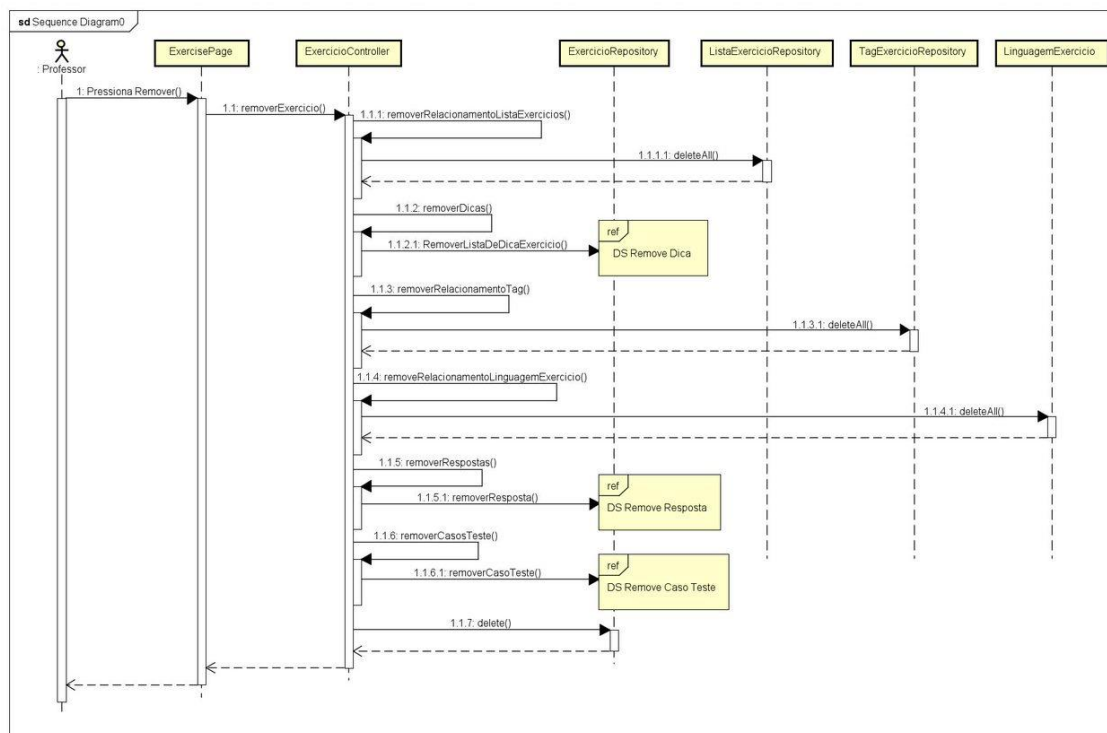


Figura 38 – DS015: GERENCIAR EXERCÍCIOS (REPLICAR)

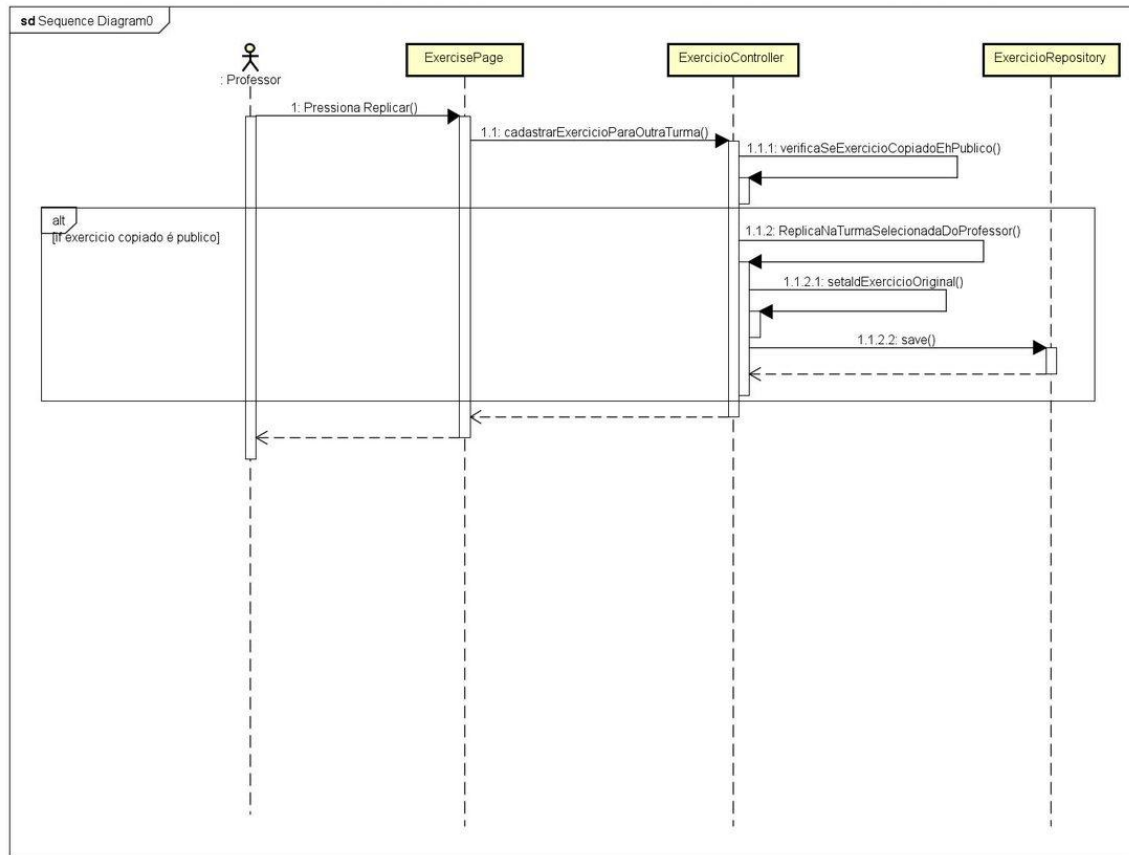


Figura 39 – DS016: CADASTRO DO ALUNO NA TURMA

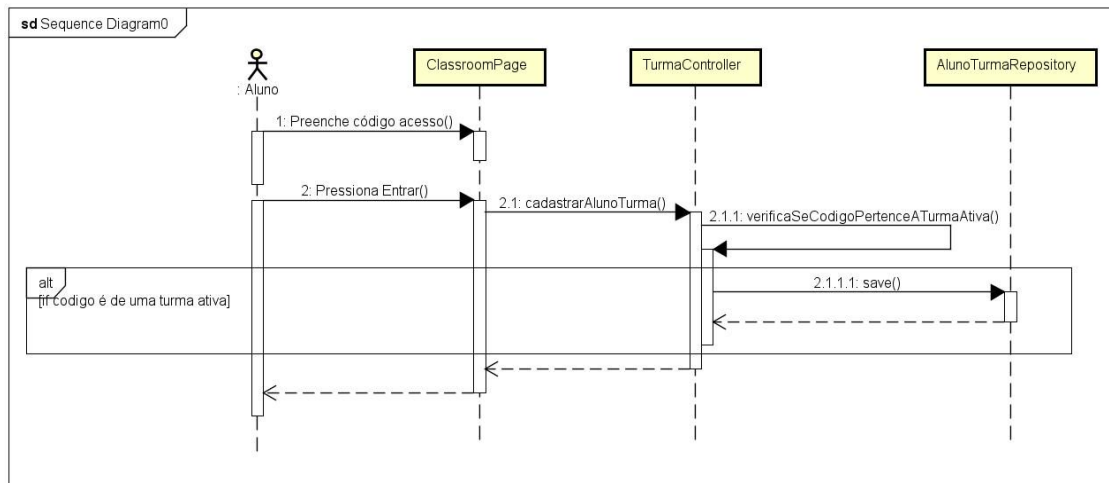


Figura 40 – DS017: CADASTRAR RESPOSTA

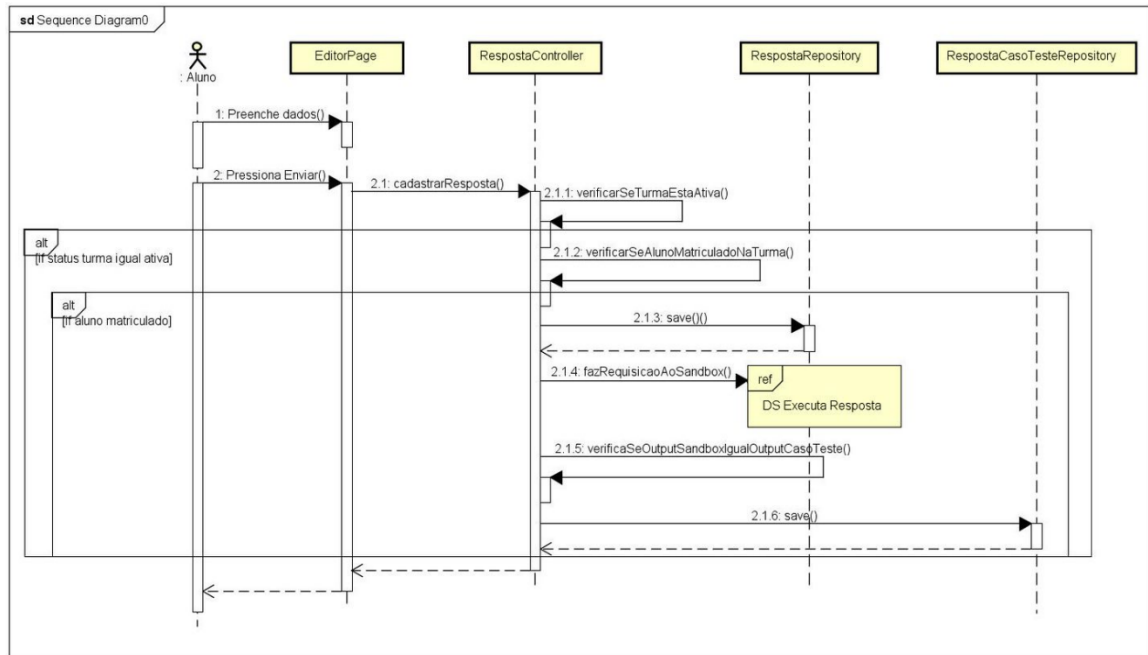
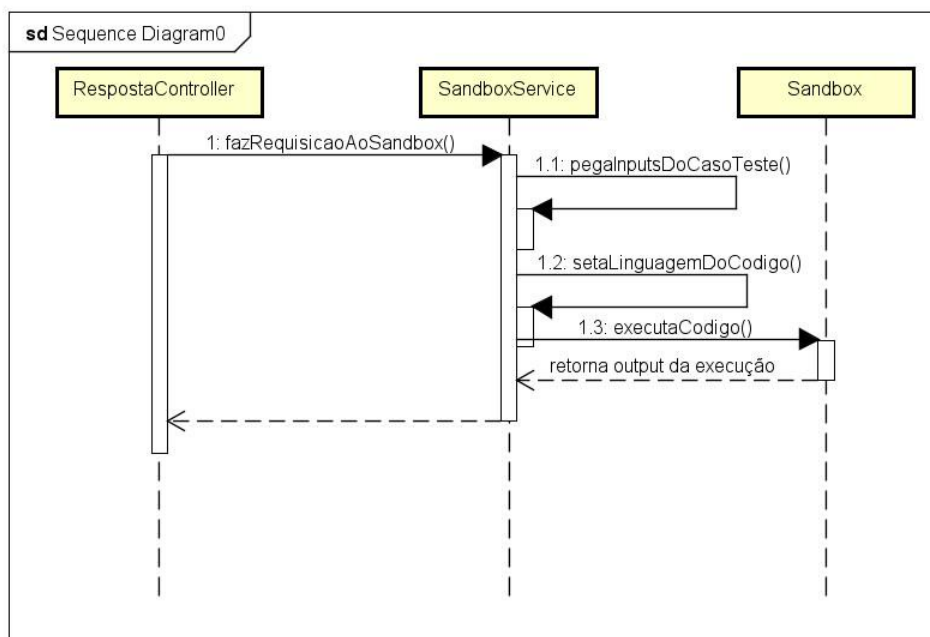


Figura 41 – DS018: EXECUTAR RESPOSTA



APÊNDICE D – DOCUMENTAÇÃO DOS ENDPOINTS

Figura 42 – Documentação dos Endpoints: Página 1

 Select a definition default

Api Documentation ^{1.0}

[Base URL: farmaalg-back.herokuapp.com/]
<https://farmaalg-back.herokuapp.com/v2/api-docs>

[Api Documentation](#)
[Terms of service](#)
[Apache 2.0](#)

basic-error-controller Basic Error Controller

GET /error errorHtml

HEAD /error errorHtml

POST /error errorHtml

PUT /error errorHtml

DELETE /error errorHtml

OPTIONS /error errorHtml

PATCH /error errorHtml

caso-teste-controller Caso Teste Controller

POST /caso/teste cadastrarCasoTeste

GET /caso/teste/{codigo} buscarCasoTeste

Figura 43 – Documentação dos Endpoints: Página 2

POST	/caso/teste/{codigo}	testarCasoTeste
PUT	/caso/teste/{codigo}	atualizarCasoTeste
DELETE	/caso/teste/{codigo}	removerCasoTeste
GET	/caso/teste/exercicio/{codigoExercicio}	ListarCasosTesteDeUmExercicio

dica-exercicio-controller
Dica Exercicio Controller
▼

POST	/exercicio/dica	cadastrarDicaExercicio
GET	/exercicio/dica/{codigo}	buscarDicaExercicio
PUT	/exercicio/dica/{codigo}	atualizarDicaExercicio
DELETE	/exercicio/dica/{codigo}	removerDicaExercicio
GET	/exercicio/dica/exercicio/{codigoExercicio}	ListarDicasExercicio

exercicio-controller
Exercicio Controller
▼

GET	/exercicio	ListarExerciciosPublicos
POST	/exercicio	cadastrarExercicio
GET	/exercicio/{codigo}	buscarExercicio
PUT	/exercicio/{codigo}	atualizarExercicio
DELETE	/exercicio/{codigo}	removerExercicio
POST	/exercicio/{codigoExercicio}/linguagem/{codigoLinguagem}	cadastrarLinguagemEmUmExercicio
DELETE	/exercicio/{codigoExercicio}/linguagem/{codigoLinguagem}	removerLinguagemDeUmExercicio

Figura 44 – Documentação dos Endpoints: Página 3

POST	/exercicio/{codigoExercicio}/lista/{codigoLista}	cadastrarExercicioEmUmaLista
DELETE	/exercicio/{codigoExercicio}/lista/{codigoLista}	removerExercicioDeUmaLista
DELETE	/exercicio/{codigoExercicio}/tag/{codigoTag}	removerTagDeUmExercicio
POST	/exercicio/{codigoExercicio}/tag/{nomeTag}	cadastrarTagEmUmExercicio
POST	/exercicio/{codigoExercicio}/turma/{codigoTurma}	cadastrarExercicioParaOutraTurma
POST	/exercicio/completo	cadastrarExercicioCompleto
instituicao-contato-controller Instituicao Contato Controller		
GET	/instituicao/contato	ListarInstituicoesContatos
POST	/instituicao/contato	cadastrarInstituicaoContato
GET	/instituicao/contato/{codigo}	buscarInstituicaoContato
PUT	/instituicao/contato/{codigo}	atualizarInstituicaoContato
DELETE	/instituicao/contato/{codigo}	removerInstituicaoContato
instituicao-controller Instituicao Controller		
GET	/instituicao	ListarInstituicoes
POST	/instituicao	cadastrarInstituicao
GET	/instituicao/{codigo}	buscarInstituicao
PUT	/instituicao/{codigo}	atualizarInstituicao

Figura 45 – Documentação dos Endpoints: Página 4

DELETE	/instituicao/{codigo}	removerInstituicao
GET	/instituicao/combo	buscarComboInstituicao
POST	/instituicao/completa	cadastarInstituicaoCompleta

linguagem-controller	Linguagem Controller	▼
-----------------------------	----------------------	---

GET	/linguagem	ListarLinguagem
POST	/linguagem	cadastarLinguagem
GET	/linguagem/{codigo}	buscarLinguagem
PUT	/linguagem/{codigo}	atualizarLinguagem
DELETE	/linguagem/{codigo}	removerLinguagem

lista-exercicio-turma-controller	Lista Exercio Turma Controller	▼
---	--------------------------------	---

POST	/lista/exercicio/	cadastarListaExercicioTurma
PUT	/lista/exercicio/{codigo}	atualizarListaExercicioTurma
DELETE	/lista/exercicio/{codigo}	removerListaExercicioTurma
GET	/lista/exercicio/{codigoLista}	buscarListaExercicioTurma
GET	/lista/exercicio/turma/{codigoTurma}	ListarListaExerciciosTurma

resposta-controller	Resposta Controller	▼
----------------------------	---------------------	---

POST	/resposta	cadastarResposta
------	-----------	------------------

Figura 46 – Documentação dos Endpoints: Página 5

GET	/resposta/{codigo}	buscarResposta
PUT	/resposta/{codigo}	atualizarResposta
DELETE	/resposta/{codigo}	removerResposta
GET	/resposta/aluno/{codigoAluno}	ListarRespostasDeUmAluno
GET	/resposta/aluno/{codigoAluno}/exercicio/{codigoExercicio}	ListarRespostasDeUmAlunoAUmExercicio
GET	/resposta/exercicio/{codigoExercicio}	ListarRespostasDeUmExercicio

tag-controller Tag Controller			▼
GET	/tag	ListarTags	
POST	/tag	cadastrarTag	
GET	/tag/{codigo}	buscarTag	
PUT	/tag/{codigo}	atualizarTag	
DELETE	/tag/{codigo}	removerTag	

turma-controller Turma Controller			▼
GET	/turma	ListarTurmas	
POST	/turma	cadastrarTurma	
GET	/turma/{codigo}	buscarTurmaPorCodigo	
PUT	/turma/{codigo}	atualizarTurma	

Figura 47 – Documentação dos Endpoints: Página 6

DELETE	/turma/{codigo} removerTurma
DELETE	/turma/{codigoTurma}/aluno/{codigoAluno} removerAlunoTurma
POST	/turma/aluno/{codigoAluno} cadastrarAlunoTurma
GET	/turma/codigo/acao/{codigoAcesso} consultarDisponibilidadeCodigoAcesso
GET	/turma/mim buscarTurmasDoAluno

usuario-controller Usuario Controller 

GET	/usuario ListarUsuarios
POST	/usuario cadastrarUsuario
GET	/usuario/{codigo} buscarUsuario
PUT	/usuario/{codigo} atualizarUsuario
DELETE	/usuario/{codigo} removerUsuario
PUT	/usuario/admin/senha adminAtualizarSenhaUsuario
GET	/usuario/mim buscarUsuarioPorToken
PUT	/usuario/senha atualizarSenhaUsuario

Models 