UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ISABELA TAQUES VITEK

DESENVOLVIMENTO DA ÁREA DESTINADA À CRIAÇÃO DE OBJETOS DE APRENDIZAGEM PARA FERRAMENTA DE AUTORIA FARMA

GUARAPUAVA

2023

ISABELA TAQUES VITEK

DESENVOLVIMENTO DA ÁREA DESTINADA À CRIAÇÃO DE OBJETOS DE APRENDIZAGEM PARA FERRAMENTA DE AUTORIA FARMA

Development of the Learning Object Creation Area for the FARMA Authoring Tool

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Diego Marczal

Coorientadores: Prof. Me. Alex Sandro De Castilho e Prof. Me. Denis Lucas Silva

GUARAPUAVA

2023

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

RESUMO

A Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem (FARMA) foi idealizada como um meio que oferecesse funcionalidades para construir Objetos de Aprendizagem (OAs). Seu objetivo é trazer aos professores leigos em programação a possibilidade de criar e gerenciar OAs de forma dinâmica e descomplicada, permitindo que os alunos e professores possam visualizar os erros que ocorreram até obterem êxito na resolução do exercício, e a cada submissão errada do discente, mostra conteúdos que auxiliam a encontrar a solução. Nesta ferramenta, os docentes podem configurar seu exercício com uma explicação, incorporando recursos como vídeos e *links* para contribuir com a compreensão do tema. Com base nisso, o desenvolvimento deste trabalho terá como propósito a implementação *Frontend* e *Backend* da área de professor da FARMA, unificando as versões já existentes em um único projeto, permitindo a criação de Objetos de Aprendizagem, que pode conter vídeos, gráficos, imagens, dentre outros. Durante a criação, o professor poderá configurar também as dicas que aparecerão em casos de erros, bem como a sequência em que serão exibidas, juntamente com a possibilidade de gerenciar esses OAs (edição, visualização e exclusão). Além disso, ambiciona-se um sistema que possua uma interface atrativa com facilidade na criação de Objetos de Aprendizagem.

Palavras-chave: objeto; aprendizagem; exercício; ferramenta; sistema.

ABSTRACT

The Authoring Tool for Remediation of Errors with Mobility in Learning (FARMA) was conceived as a means of offering functionalities for the construction of Learning Objects (LOs). Its objective is to bring lay teachers in programming the possibility of creating and managing LOs of dynamic and uncomplicated activities, allowing students and teachers to visualize the errors that occurred until they managed to solve the exercise, and with each wrong submission by the student, it shows contents that help to find the solution. In this tool, the teacher can configure his activity with an explanation, incorporating resources such as videos and *links* to contribute to the understanding of the topic. Based on this, the development of this work will have as its purpose the implementation of the *frontend* and *backend* of the FARMA professor area, unifying the already existing versions in a single project, allowing the creation of Learning Objects, which can or not contain videos, graphics, images, among others. During creation, the teacher will also be able to configure the tips that will appear in cases of errors, as well as the sequence in which they will be displayed, in addition to the possibility of managing these LOs (edit, view and delete). In addition, a system is envisaged that has an attractive interface and without complexity in the creation of Learning Objects.

Keywords: object; learning; exercise; tool; system.

LISTA DE FIGURAS

Figura 1 – Caso de uso do professor que será desenvolvido 9
Figura 2 – Home da Farma 10
Figura 3 – Criar Introdução na Farma 11
Figura 4 – Criar Exercício na Farma 12
Figura 5 – Criar/Editar OA na Farma 12
Figura 6 – Home Farma Reborn 13
Figura 7 – Criar/Editar OA's Farma Reborn 13
Figura 8 – Criar Introdução na Farma Reborn 14
Figura 9 – Criar Exercício Farma Reborn 14
Figura 10 – Planejamento 16
Figura 11 – Fluxo de Desenvolvimento 17
Figura 12 – Git Flow 19
Figura 13 – CI/CD 19
Figura 14 – Projects FARMA API 25
Figura 15 – Rota Index da API 25
Figura 16 – Branch de modificação 26
Figura 17 – Rota Index da API com a modificação 26
Figura 18 – Commit das modificações 27
Figura 19 – Teste Rubocop e exemplo de uma correção 28
Figura 20 – Teste Ruby Critic 28
Figura 21 – Teste Brakeman 28
Figura 22 – Rails teste 29
Figura 23 – Criação da PR 29
Figura 24 – Visualização da PR 30
Figura 25 – Execução do CI 31
Figura 26 – Aprovação da PR 32
Figura 27 – Comentários na PR 32
Figura 28 – PR aprovada 33
Figura 29 – Relação dos usuários 37
Figura 30 – Caso de uso visitante 37

Figura 31 – Caso de uso aluno **38**
Figura 32 – Caso de uso professor **38**

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface*
CD	*Continuous Delivery*
CI	*Continuous Integration*
CI/CD	*Continuous Integration/Continuous Delivery*
FARMA	Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem
OA	Objeto de Aprendizagem
PR	*Pull Request*
SI	Sistemas para Internet
UTFPR-GP	Universidade Tecnológica Federal do Paraná, câmpus Guarapuava

SUMÁRIO

1	**INTRODUÇÃO**	**7**
1.1	**Objetivos**	**9**
1.1.1	Objetivo geral	9
1.1.2	Objetivos específicos	9
2	**FARMA**	**10**
3	**MATERIAIS E MÉTODOS**	**16**
3.1	**Materiais**	**16**
3.2	**Métodos**	**16**
3.2.1	Processo de Desenvolvimento	16
4	**RESULTADOS PARCIAIS**	**20**
4.1	**Requisitos**	**20**
4.1.1	Requisitos Funcionais	20
4.1.2	Requisitos Não Funcionais	21
4.2	**Escolha do Framework CSS**	**21**
4.3	**Ambiente de desenvolvimento**	**22**
4.3.1	Configuração da API	22
4.3.2	Exemplo de funcionamento da API	25
5	**CONCLUSÃO**	**34**
	REFERÊNCIAS	**35**
	APÊNDICE A CASOS DE USO DOS USUÁRIOS DO SISTEMA FARMA	**37**
	A.1 Relação dos usuários.	**37**
	A.2 Caso de Uso Visitante.	**37**
	A.3 Caso de Uso Aluno.	**38**
	A.4 Caso de Uso Professor.	**38**

1 INTRODUÇÃO

A educação é o ato de educar alguém ou a si próprio em questões culturais, cientificas, religiosas ou em bons modos de interação e comunicação com outras pessoas (VICENTE; ROCHA, 2022). Para isso, é preciso adotar métodos e processos de forma a assegurar a formação e o aprendizado.

> Aprendizagem é o processo pelo qual as competências, habilidades, conhecimentos, comportamento ou valores são adquiridos, ou modificados, como resultado de estudo, experiência, formação, raciocínio e observação[1].

Para ensinar geografia, além dos textos e discursos, os professores utilizam mapas, fragmentos de rochas e vídeos para ilustrar os temas de estudo em sala de aula. Isto é um exemplo, no qual um professor recorre a outros objetos, além do quadro negro e giz, para propiciar ao aluno o entendimento e, consequente, aprendizado sobre algo. Através da experimentação e vivência rotineira, ambas lúdicas ou não, dúvidas e confusões de entendimento são trabalhados de forma a saná-las. Com o avanço da tecnologia, outros objetos de apoio ao ensino, como, por exemplo, televisões, projetores, computadores, dentre outros, começam a compor o ambiente escolar, permitindo aos alunos visualizarem de uma maneira melhor o que se quer transmitir (KENSKI, 2003).

A importância de garantir a educação a todos começa nos primeiros anos de vida, onde a criança é alfabetizada, com curiosidade, aprende a escrever seu nome e a ler uma placa na rua. São conhecimentos básicos, mas essenciais durante toda vida. Contudo, muitos fatores contribuíram para a precariedade da educação, tendo como resultado os analfabetos funcionais (LORENZO, 2007). Um destes fatores foi o surgimento de outras coisas “mais interessantes” que o aprender, como os computadores, jogos e, atualmente, os smartphones.

O acesso cada vez maior a músicas, filmes, jogos e redes sociais propiciado pelos aparelhos já citados, além da “*ilusão do conhecimento*”[2], alimentada pelos diversos vídeos disponibilizados na Internet, teve lugar na vida das pessoas, colocando a educação e aprendizado extraclasse de lado.

Neste novo mundo, a educação por meio de seus gestores, professores e pesquisadores, precisou ser revista. Uma destas revisões trouxe a necessidade de se incluir mais objetos para ensinar. Os softwares educacionais são uma opção viável no processo de ensino-aprendizagem, de forma a complementar a transferência de conhecimentos, auxiliando os professores em atingir o aprendizado de seus alunos.

Na matemática, por exemplo, os professores dispõem do GeoGebra[3], um aplicativo que pode ser usado na Internet, ou instalado em computadores e celulares, o qual combina concei-

[1] https://pt.wikipedia.org/wiki/Aprendizagem
[2] https://www.bbc.com/portuguese/vert-cap-62985145
[3] https://www.geogebra.org/calculator

tos de geometria e álgebra, permitindo aos professores demostrarem de forma visual figuras e gráficos, sendo usado para ajudar a resolver vários modelos de equações, principalmente no ensino médio. Também pode-se citar o Google Earth[4], com ele é possível explorar imagens de satélite do mundo todo, terrenos e construções em 3D, podendo ser utilizado tanto nas aulas de geografia sobre o estudo da cartografia, como também ao se estudar pontos de relevância como as 7 maravilhas do mundo. Contudo, mesmo sabendo da existência destes, dentre tantos outros, ainda existe a carência por ferramentas computacionais para uso prático em sala de aula ou extra-classe.

> Isso significa que trazer as tecnologias para o ambiente educativo pode tornar o processo de ensino e aprendizagem mais prazeroso, mais chamativo e significativo para aquele que aprende e mais dinâmico para aquele que educa (SILVA; CORREA, 2014).

Dentre os mais variados assuntos, um que se destaca, quando se trata de dificuldade de aprendizagem, é a matemática, ou conteúdos de exatas em geral. Segundo Santos, França e SANTOS (2007) “A Matemática não é uma ciência cristalizada e imóvel; ela está afetada por uma contínua expansão e revisão dos seus próprios conceitos”, com isso entende-se que a dificuldade ocorre pela forma com que a matemática foi construída. A matemática é um dos conteúdos primordiais que devem ser aprendidos, assim como o português é essencial para o desenvolvimento humano, desde uma conta básica de troco após uma compra até assuntos mais complexos como finanças.

Com este foco, no período de 2015 a 2018, um grupo de alunos do Curso de Graduação em Sistemas para Internet (SI) da Universidade Tecnológica Federal do Paraná, câmpus Guarapuava (UTFPR-GP), participaram do desenvolvimento de uma ferramenta para auxiliar no processo de ensino-aprendizagem da matemática, idealizado pelo professor Diego Marczal em sua tese de doutorado no ano de 2014 (MARCZAL *et al.*, 2014). O objetivo é dispor de um ambiente (**F**erramenta) onde cada professor pode criar atividades (**A**utoria) para testar, avaliar e corrigir o processo de resolução adotado pelos alunos (**R**emediação de Erros). Um sistema Web, acessível a partir de um computador localizado em qualquer parte do mundo, proporcionando aos alunos (**M**obilidade na **A**prendizagem) e professores mobilidade no seu uso.

Na Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem (FARMA), um Objeto de Aprendizagem (OA) é apresentado em formato de atividades que são formados por uma explicação do conteúdo seguido de exercícios referente ao tema, permitindo o professor montar essa estrutura de uma forma mais fácil e o aluno testar o entendimento do assunto abordado, logo em seguida verificando seus resultados com auditoria de seus erros. Também permite tanto o aluno quanto o professor visualizar as submissões dos exercícios até a resposta final, com isso podendo entender melhor as dificuldades existentes no conteúdo abordado.

[4] https://www.google.com.br/intl/pt-BR/earth/

Propõem-se, neste trabalho, o desenvolvimento *Frontend* e *Backend* da área de professores da FARMA, disponibilizando recursos para a criação e gerenciamento de OA's, conforme a Figura 1. Este módulo disponibilizará aos docentes, principalmente aqueles sem conhecimentos em programação de computadores, a possibilidade de criar OA's interativos, caracterizados como pequenos softwares educacionais de forma mais assertiva e descomplicada.

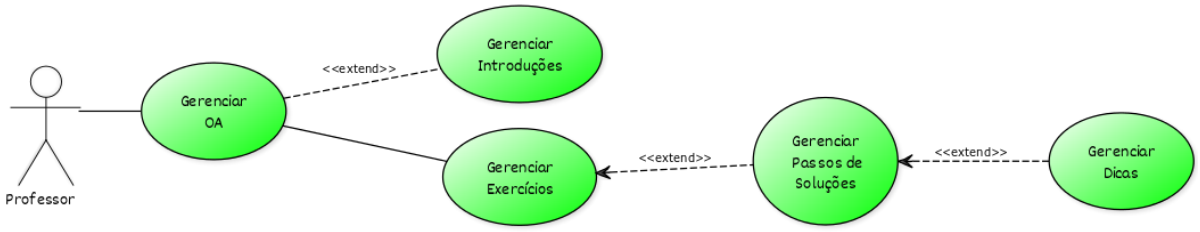


Figura 1 – Caso de uso do professor que será desenvolvido
Fonte: Autoria própria (2023).

O desenvolvimento desta área motivou-se principalmente por existirem várias contribuições anteriores que necessitam se adequar todas em uma única versão estável da FARMA, corrigindo também as carências que ainda permanecem ao criar um OA. Assim, inicia-se uma nova versão da FARMA que terá sua criação de modo incremental, no qual primeiramente será desenvolvido a criação de OA's na área do professor.

1.1 Objetivos

Almeja-se neste trabalho o desenvolvimento desta área para criação de um OA contendo todos os seus elementos (introduções, materiais de apoio e exercícios), bem como como a possibilidade de configurar as dicas dos exercícios (ordem e quantidade) e realizar as operações de gerenciamento (visualização, edição e listagem) conforme descrito anteriormente.

1.1.1 Objetivo geral

Implementação do módulo da área dos professores, destinado à criação e gerenciamento de OA's da FARMA.

1.1.2 Objetivos específicos

- Entender as necessidades do ponto de vista de professor(es) de matemática na criação de Objetos de Aprendizagem na FARMA;
- Prototipar as telas que atendam as necessidades levantadas;
- Entender o funcionamento da FARMA;
- Desenvolver o módulo de criação de objetos da FARMA.

2 FARMA

A Ferramenta de Autoria para a Remediação de Erros com Mobilidade na Aprendizagem teve sua concepção, projeto e desenvolvimento com a tese de doutorado de Marczal *et al.* (2014). Estimulado pela ausência de meios de averiguar o processo de ensino e aprendizagem, por meio de tentativas de resolução e erros, das ferramentas educativas, como pode-se averiguar no trabalho de Marczal e Direne (2011) que foca na demanda da implementação de um gerenciador de OA's atrativo que utilize os erros do educando ao resolver questões de forma que auxilie seu entendimento. E no trabalho de doutorado de Leite (2013), onde foi desenvolvida uma pesquisa relacionada ao uso e aplicação de remediação de erros.

O objetivo da ferramenta é disponibilizar um ambiente para a aprendizagem de conceitos matemáticos. Para isso, a ferramenta tem embutido em seu escopo Objetos de Aprendizagem, com o intuito de dar flexibilidade na forma de abordar os temas matemáticos. Desta forma, monitores e professores de matemática podem trabalhar conceitos adotando viés e objetivos personalizados à cada aluno.

A primeira versão da ferramenta, elaborada por Marczal *et al.* (2014), está disponível na FARMA educacional[1], conforme a Figura 2.



Figura 2 – Home da Farma
Fonte: http://farma.educacional.mat.br.

Objetos de Aprendizagem são qualquer meio digital ou não, para transmitir conhecimento, como animações e simulações (AGUIAR; FLÔRES, 2014; TAROUCO *et al.*, 2014).

[1] http://farma.educacional.mat.br/

Exemplos clássicos de OA's muito utilizados nos anos iniciais são jogos, cruzadinhas, caça-palavras, dentre outros. Entretanto vem ganhando destaque formas de misturar a tecnologia ao ensino, trazendo mais interatividade e curiosidade ao aprender.

Na FARMA o objeto de aprendizagem é composto por:

- Textos descritivos e explicativos sobre o tema de estudo (introduções), que podem conter figuras, links de vídeos e gráficos;
- É possível criar exercícios (enunciado), que abrange uma ou mais questões;
- As questões devem incluir dicas para auxiliar os alunos quando ocorrer um erro. Cada tentativa de resolução das questões geram um histórico para análise de erros para entender o processo até a resposta final.

Para criar um OA na FARMA seguimos o fluxo conforme a Figura 3, onde está criando uma nova introdução para o OA, adicionando um título e conteúdo. Na Figura 4 observa-se a tela para criar um novo exercício, adicionando um título e conteúdo. E na Figura 5 visualiza-se a página que permite acessar as páginas de criação de novas introduções e exercícios para um OA, adicionar questões para os exercícios, ou editar os já existentes.

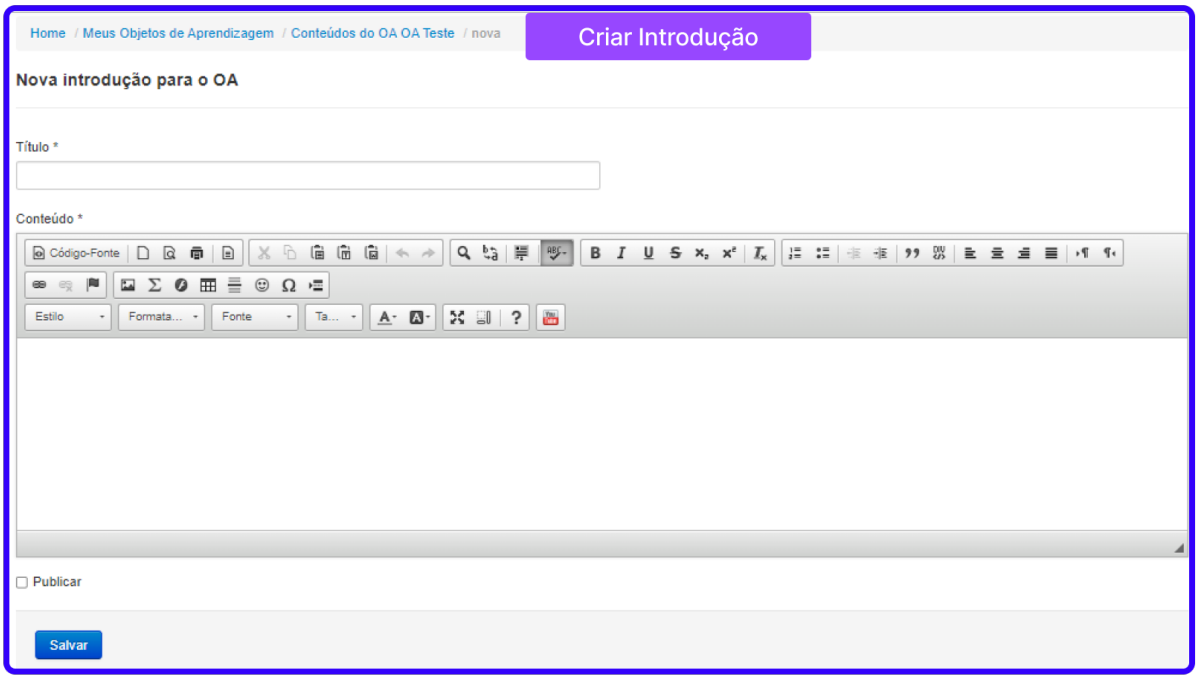


Figura 3 – Criar Introdução na Farma
Fonte: http://farma.educacional.mat.br.

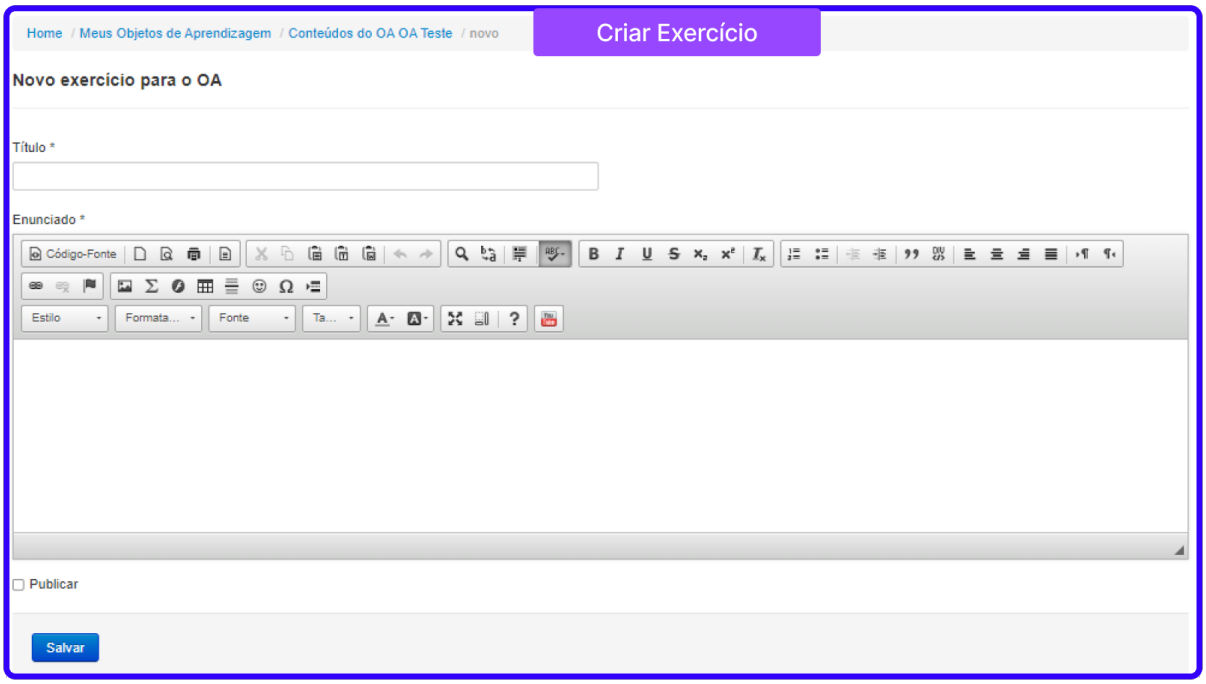


Figura 4 – Criar Exercício na Farma
Fonte: http://farma.educacional.mat.br.

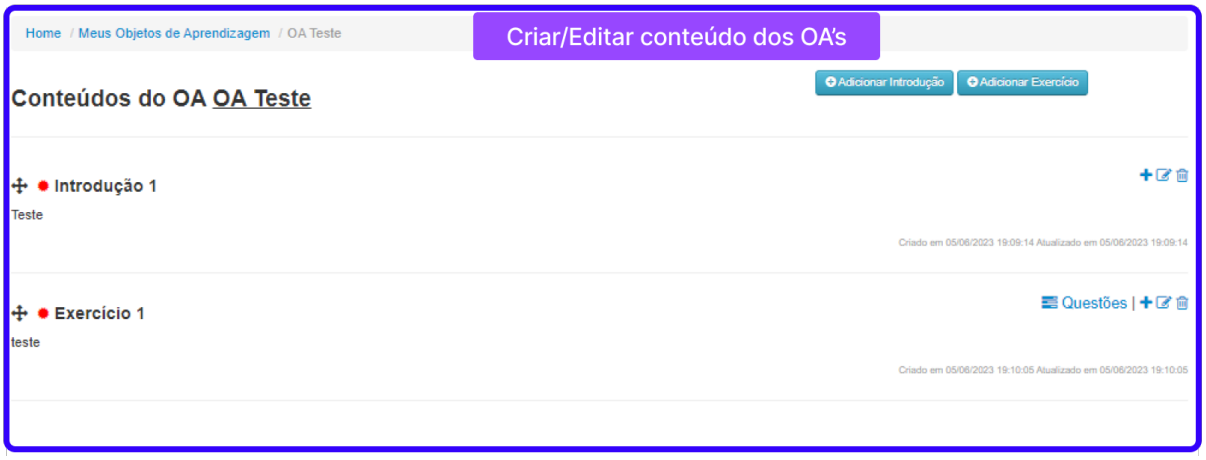


Figura 5 – Criar/Editar OA na Farma
Fonte: http://farma.educacional.mat.br.

Neste sistema é possível encontrar 3 (três) tipos de usuários que interagem de formas diferentes com a ferramenta, sendo eles: o professor, o aluno e o visitante conforme apresentados nos diagramas de caso de uso no Capítulo A de apêndices.

No período entre 2015 e 2018 ocorreu o “Redesign e Refatoração da Ferramenta de Autoria para a Remedição de Erros com Mobilidade na Aprendizagem – FARMA”, envolvendo acadêmicos do curso de Graduação em SI. Com o objetivo de reestruturar o sistema, organizando suas estruturas internas e telas. Resultando na segunda versão da ferramenta, que está em fase de desenvolvimento, apresentada na Figura 6.



Figura 6 – Home Farma Reborn
Fonte: http://farma-reborn.educacional.mat.br.

Nas figuras abaixo é possível visualizar o processo para criar OA's na FARMA Reborn, onde, na Figura 7 identifica-se a tela que trás as informações dos conteúdos já criados, e concede acesso para criar novos ou editar, na Figura 8 visualiza-se a tela para criar introdução e na Figura 9 a tela para criar exercício. Todavia nas imagens abaixo é possível notar um visual mais atrativo, com os elementos distribuídos na tela de uma maneira mais interessante e agradável.



Figura 7 – Criar/Editar OA's Farma Reborn
Fonte: http://farma-reborn.educacional.mat.br.

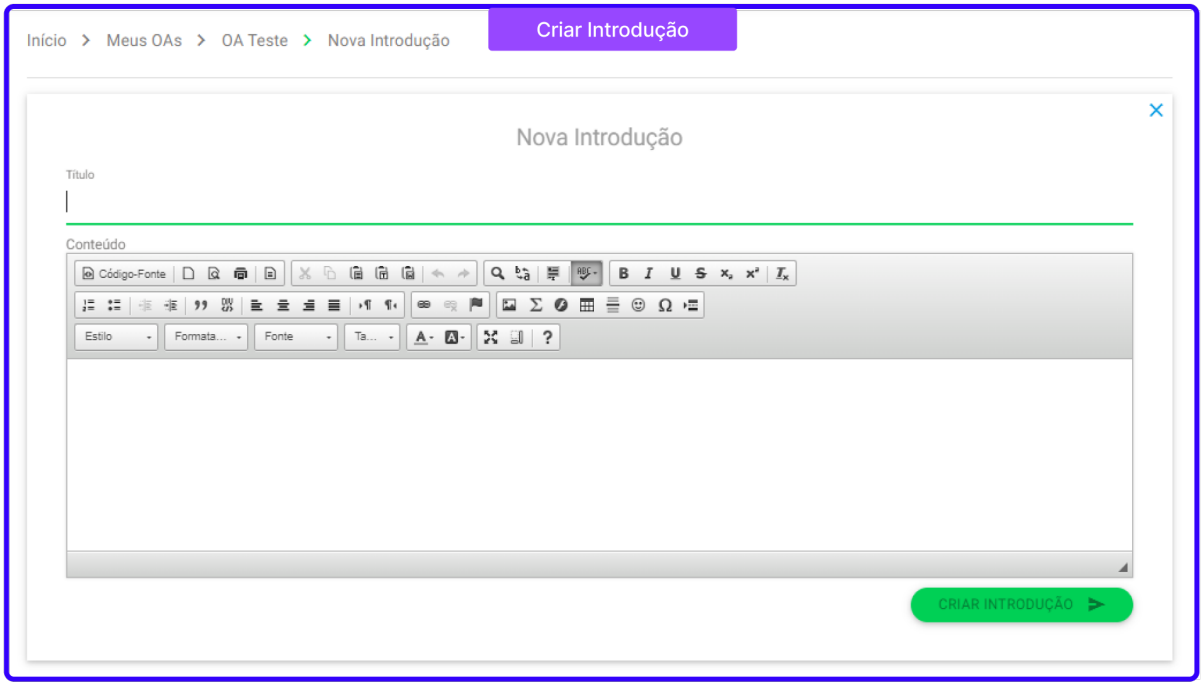


Figura 8 – Criar Introdução na Farma Reborn
Fonte: http://farma-reborn.educacional.mat.br.

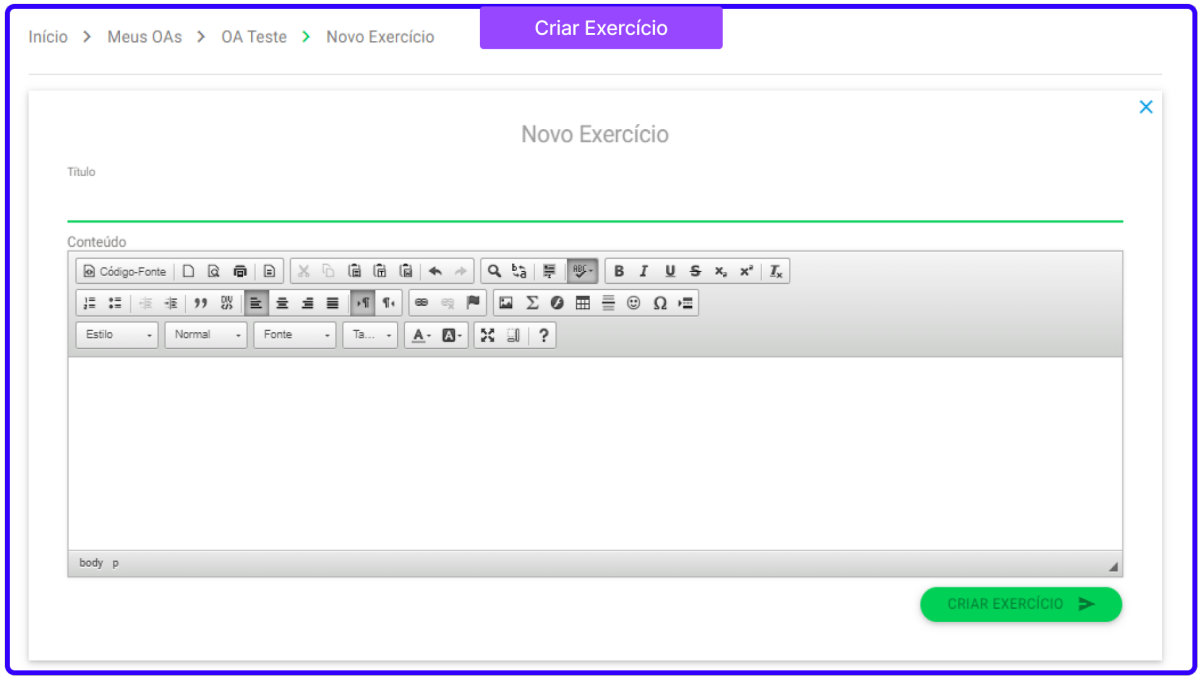


Figura 9 – Criar Exercício Farma Reborn
Fonte: http://farma-reborn.educacional.mat.br.

A FARMA já foi foco de estudo para alguns trabalhos. Em seu doutorado, Silva e Direne (2015) focou na adaptação sequencial de exercícios relacionando a sua dificuldade e o desemprenho dos estudantes, ocasionado a implementação de modificações na FARMA nomeado como ADAPTFARMA. No trabalho de Kutzke e Direne (2016) foi foco a possibilidade

de utilização dos erros do aprendiz como forma de aprimorar seu conhecimento. Em trabalhos de mestrado, a FARMA foi foco de Moura e Peres (2017) avaliando o impacto da retroação na aprendizagem com auxilio da FARMA. E Santos (2022), em seu trabalho de conclusão de curso, desenvolveu o módulo de estatísticas para a FARMA.

O professor Alex S. de Castilho está desenvolvendo seu doutorado utilizando a FARMA Calc. Esta versão da FARMA visa suprir as necessidades encontradas nas versões anteriores mantendo o *design* como o da Figura 2 e voltada apenas para uso em relação a cálculos. Devido a sua temática e relevância, a FARMA é citada em diversos artigos que podem ser encontrados na farma-reborn[2].

[2] http://farma-reborn.educacional.mat.br/

3 MATERIAIS E MÉTODOS

Neste capítulo serão descritos os materiais utilizados para o desenvolvimento deste trabalho, bem como os métodos adotados para alcançar os resultados propostos.

3.1 Materiais

Para o desenvolvimento desse trabalho será utilizado o React Next no *Frontend* e Ruby Rails no *Backend*, ambos escolhidos pelo fato de já serem as linguagens utilizadas nas versões anteriores da FARMA, e para Framework CSS será utilizado o Bootstrap que foi escolhido por meio de reunião conforme descrito na Seção 4.2. Para gerenciar o banco de dados será utilizado o PostgreSQL.

Para a organização e gestão do projeto será usado o Git WorkFlow, o Git Lab Flow para o versionamento e a prática CI/CD. Será utilizada a ferramenta Projets do próprio Github para a organização das tarefas. O software de desenvolvimento de código adotado será o Visual Studio Code.

3.2 Métodos

3.2.1 Processo de Desenvolvimento

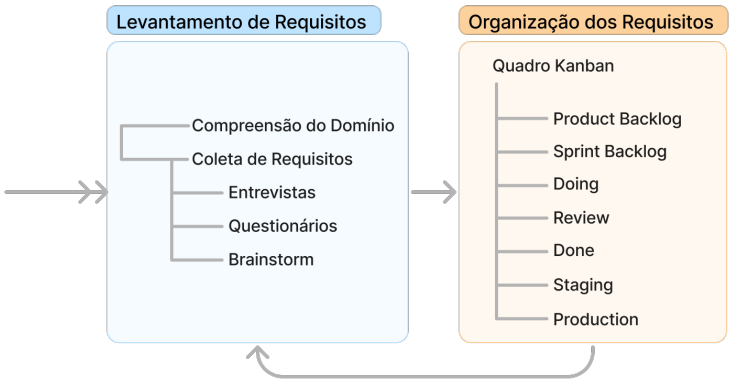


Figura 10 – Planejamento
Fonte: Autoria própria (2023).

Inicialmente é preciso o entendimento do sistema FARMA frente as necessidades dos usuários. Com estas, os requisitos para o sistema podem ser melhor definidos junto aos usuários, por meio de conversas, reuniões, questionários e *brainstorming*, dentre outras técnicas. A organização dos requisitos vai ser feita conforme a Figura 10, com o uso do quadro Kanban, divi-

dindo as etapas do desenvolvimento. Os requisitos levantados vão compor o Product Backlog[1] e, conforme são desenvolvidos, avançam no quadro Kanban (Spint Backlog, Doing, Review, DOne, Staging e Production). Na sequência os requisitos serão particionados em tarefas de implementação, ou seja, funcionalidades menores e independentes para serem desenvolvidas.

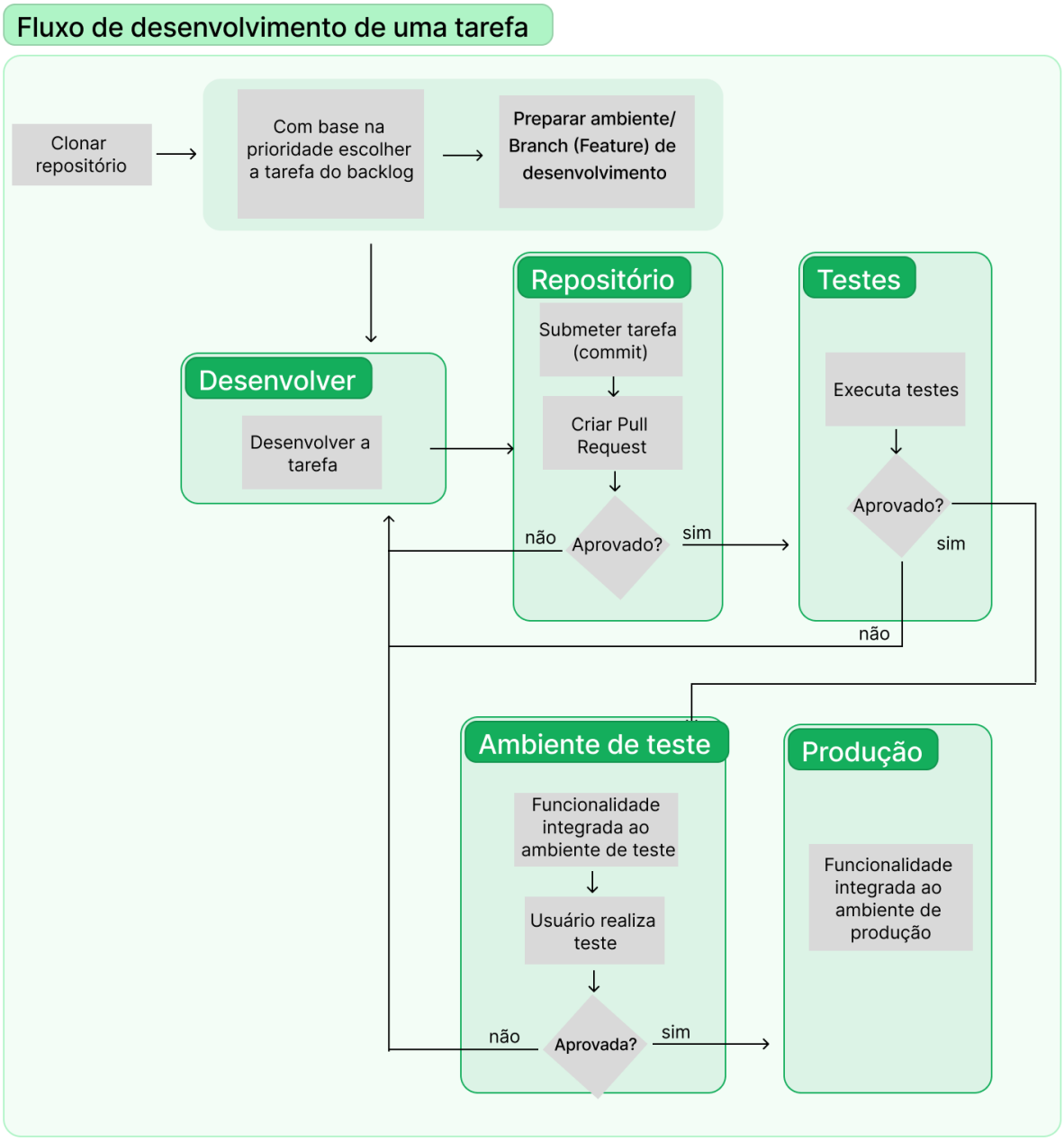


Figura 11 – Fluxo de Desenvolvimento
Fonte: Autoria própria (2023).

Para o fluxo de desenvolvimento das tarefas, o procedimento seguirá conforme a Figura 11. Primeiramente, clona-se o projeto e realiza *checkout* para a *branch* principal (*main* ou *master*), sendo necessário executar o comando “git pull” para manter o projeto local atualizado.

[1] Lista contendo os requisitos que precisam estar no produto

Escolhe-se a tarefa para desenvolver com base na sua prioridade e cria-se uma *branch* para iniciar sua implementação. Após finalizada a implementação, será realizado um *commit* e envio das modificações para o repositório. Na sequencia, abre-se uma *pull request* para revisão do código, onde em caso de aprovação será enviada para o ambiente de testes. Se não for aprovada, ela retornará para o desenvolvimento a fim de que sejam feitas as devidas correções.

No ambiente de testes, se o *feedback* do usuário for positivo, a tarefa irá para o ambiente de produção, e em caso de *feedback* negativo, retornará para a fase de desenvolvimento para correção das inconsistências encontradas durante os testes. Após integrada em ambiente de produção em caso de *feedback* de usuários, será criada uma tarefa com demanda urgente para a correção do problema encontrado em produção.

Para o desenvolvimento das tarefas será utilizado o Git Workflow, “essa técnica” agrega diversos benefícios a codificação, tais como, organização, controle e versionamento de código, proporcionando a rastreabilidade, integração (*Continuous Integration* (CI)) e entrega (*Continuous Delivery* (CD)) de código. Existem alguns tipos de WorkFlow, como por exemplo, Feature Branch, Forking, Git Lab Flow, entre outros. Neste projeto será utilizado o Git Lab Flow, como pode ser verificado na Figura 12, o código é divido em *branchs*, permitindo rastrear possíveis problemas. Suas *branchs* são divididas em:

Principais

- Branch Master/Main será a de desenvolvimento;

- Branch Production que é responsável por armazenar os códigos do sistema que está em produção;

- Branch Staging armazenará os códigos disponíveis no ambiente de teste, e ao finalizada é incrementada a *branch* de Produção.

Temporárias

- Branch Features criada para desenvolver novas funcionalidades, que serão incrementadas a Main;

- Branch Hotflix é gerada da Production quando existem problemas urgentes para corrigir é incrementada na Production e Staging após finalizada;

- Branch Bugfix é gerada da Staging quando existem necessidades de correções e também é incrementada na Main.

Serão aplicadas integrações e entregas contínuas com uso da prática *Continuous Integration/Continuous Delivery* (CI/CD), que conforme ilustrado na Figura 13 , consiste em desenvolver, testar, integrar e implantar códigos de maneira contínua. Essa prática permite desenvolver de maneira ágil e de qualidade, pois no ciclo de desenvolvimento seus testes são automatizados, ou seja, utiliza testes pré definidos. Quando aprovados, passam por *deploy* de forma automática para o ambiente de teste. Haverá dois ambientes funcionando, o ambiente de *Staging* que é restrito apenas para os usuários que forem realizar os testes, e o ambiente de Produção, que fica disponível para acesso ao público em geral.

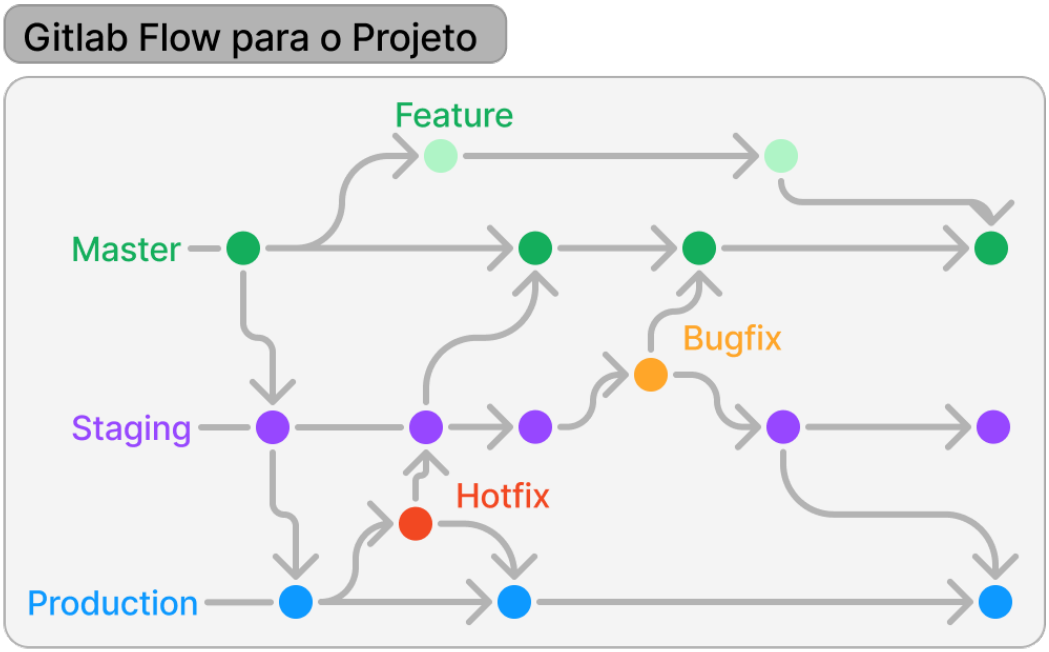


Figura 12 – Git Flow

Fonte: Autoria própria (2023), com base no GitWorkflow.

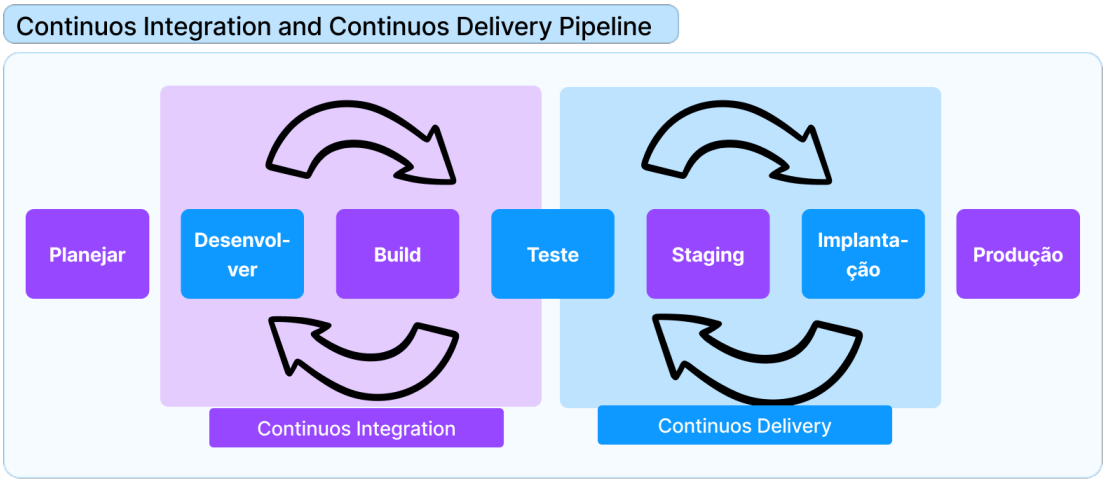


Figura 13 – CI/CD

Fonte: Autoria própria (2023).

4 RESULTADOS PARCIAIS

Neste capítulo serão apresentados os resultados que foram obtidos em cada etapa do processo de desenvolvimento deste trabalho.

4.1 Requisitos

A primeira ação realizada no desenvolvimento deste trabalho foi o levantamento dos requisitos necessários para o sistema conforme descritos abaixo.

4.1.1 Requisitos Funcionais

a. O sistema deve permitir a criação de OA's;

b. Os OA's devem possuir título (nome), descrição e opcionalmente podem conter uma imagem;

c. O sistema deve permitir escolher se os OA's serão ou não compartilhados;

d. O sistema deve permitir criar uma ou mais introduções, em texto formatado, para cada OA, com suporte para adicionar vídeos, imagens e gráficos;

e. As introduções devem possuir título e conteúdo;

f. O sistema deve permitir escolher se as introduções serão ou não disponíveis para a visualização;

g. O sistema deve permitir criar um ou mais exercícios para cada OA. Permite-se adicionar em cada exercício textos, imagens e gráficos;

h. Os exercícios devem possuir título e conteúdo;

i. O sistema deve permitir escolher se os exercícios serão ou não disponíveis para a visualização;

j. O sistema deve permitir criar um ou mais passos de solução (questões) para cada exercício, em formato de textos, imagens ou gráficos;

k. Os passos de solução devem possuir título e conteúdo a resposta correta e quantas casas decimais são necessárias para avaliar a resposta;

l. O sistema deve permitir configurar a ordem de exibição dos passos, como por exemplo, todos de um vez, de dois em dois, um por vez, dentre outras maneiras;

m. O sistema deve permitir escolher se os passos serão ou não disponíveis para a visualização;

n. O sistema deve permitir criar uma ou mais dicas para cada questão;

o. As dicas devem possuir conteúdo;

p. O sistema deve permitir configurar a ordem de exibição das dicas, como por exemplo, em sequência, de duas em duas, aleatório, todas de uma vez, dentre outras maneiras;

q. O sistema deve permitir editar e excluir OA's de forma precisa, garantindo que todas as alterações sejam aplicadas corretamente;

r. O sistema deve permitir duplicar os OA's completamente de forma eficiente e precisa, garantindo que todas as informações sejam duplicadas corretamente;

s. O sistema deve permitir duplicar as introduções dos OA's, de forma eficiente e precisa, garantindo que todas as informações sejam duplicadas corretamente;

t. O sistema deve permitir duplicar os exercícios dos OA's, de forma eficiente e precisa, garantindo que todas as informações sejam duplicadas corretamente;

u. O sistema deve permitir duplicar os passos de solução dos exercícios, de forma eficiente e precisa, garantindo que todas as informações sejam duplicadas corretamente;

v. O sistema deve permitir duplicar as dicas, de forma eficiente e precisa, garantindo que todas as informações sejam duplicadas corretamente.

4.1.2 Requisitos Não Funcionais

a. O sistema deve possuir uma interface atrativa, possuindo mais cores e dispondo também de *layouts* mais harmônicos, pois a versão atual do sistema é monocromática e com *layout* muito quadrado;

b. O sistema deve ter uma navegação intuitiva, onde não precise necessariamente seguir uma sequência de etapas para chegar ao destino desejado;

c. As informações do sistema devem ser apresentadas de maneira agradável, evitando repetições e trazendo as mesmas de modo intuitivo.

4.2 Escolha do Framework CSS

Para a escolha de Framework CSS que será aplicado foram realizadas reuniões com o Orientador e Coorientador deste trabalho, levando em consideração alguns itens: como po-

pularidade do Framework entre os desenvolvedores, qual Framework se adequaria melhor a FARMA, recursos visuais mais atrativos, dentre outros, conforme a Tabela 1.

Ao tofo, 5 (cinco) Frameworks foram escolhidos com intuito de debate e realização da escolha, sendo eles o Bootstrap[1], o Tailwind[2] , o Materialize[3], o Semantic[4] e o Tabler IO[5].

O Bootstrap é um Framework muito conhecido e utilizado em diversos sites e sistemas, como por exemplo no Twitter, dentre os escolhidos para comparação é o que mais possui estrelas no Github e dispõem de diversos recursos personalizáveis. O Tailwind por sua vez é um Framework de código aberto que ao contrário do Bootstrap e demais, que tem classes pré definidas, cria um arquivo com classes base que serão utilizadas mescladas para obter a personalização desejada e também possui muitas estrelas no Github. O Materialize já é utilizado na FARMA e conta com diversos recursos assim como os demais e segue o padrão Google's material *Design*. O Semantic tem como diferencial a possibilidade de criar layouts sofisticados e criar animações 3D, é um Framework com semântica intuitiva e contem recurso para depuração. E por fim o Tabler IO, que é um template com base no Bootstrap, utilizado na FARMA, é totalmente responsivo, conta com modo escuro e outros recursos.

Tabela 1 – Tabela de Frameworks CSS

	Bootstrap	**Tailwind**	**Materialize**	**Semantic**	**Tabler IO**
Conhecimento Prévio	5	4	3	1	1
Popularidade	5	4	3	1	1
Stars no GitHub	5	4	1	3	1
Usado na FARMA	1	1	5	1	5
Visual Atrativo	5	4	2	3	1
Média	4,2	3,4	2,8	1,8	1,8

Fonte: Autoria própria (2023).

Analisando todas as ponderações, é possível chegar a conclusão final de utilizar o Bootstrap, pois o mesmo contém os recursos esperados, adequa-se melhor as necessidades e como pode-se observar na Tabela 1 obteve melhores resultados e sua média foi a mais elevada na comparação com os demais.

4.3 Ambiente de desenvolvimento

4.3.1 Configuração da API

Para a aplicação configurou-se uma imagem no Docker com o uso do Dockerfile, que será utilizada para levantar um container, incluindo as dependências necessárias para executa-

[1] https://getbootstrap.com/
[2] https://tailwindcss.com/
[3] https://materializecss.com/
[4] https://semantic-ui.com/
[5] https://tabler.io/

lá. Contém a imagem do Ruby em sua última versão (3.2.2) e algumas configurações adicionais responsáveis por criar a aplicação em Rails a partir dessa imagem. O Docker Compose foi configurado para orquestrar os containers, que inicialmente são, o container da API, container do banco de dados e o container de testes da API.

Foi configurado o PostgreSQL como gerenciador de banco de dados. Com isso é necessário configurar o usuário e senha do PostgreSQL e sua localização para que o Rails consiga acessar, isso será feito no arquivo credentials do Rails, editando-o tanto no ambiente de desenvolvimento quanto no ambiente de testes. O *credentials* é criptografado e não fica visível já que vai para o repositório, permitindo visualizar as configurações apenas por meio da chave, visto que contém informações que não podem ficar públicas, como por exemplo senha do banco de dados, chave da Amazon, *Rails Secret Key*, dentre outras.

Com o intuito de realizar um teste, foi criada uma rota na API que lista alguns membros da FARMA, que será retornada em formato *json* apenas com finalidade de exemplificar o funcionamento da API neste primeiro momento.

Posteriormente é feita a configuração dos testes, que serão executados pelo CI. Para isso será realizado o *build* do container de testes que já foi configurado no Docker Compose e a imagem que foi configurada no Dockerfile, com isso será instalado as dependências dos testes e em seguida executa-os, neste primeiro momento todos serão bem sucedidos, pois o teste é para exemplo então só vai tentar acessar a rota e verificará o status.

Em seguida foi configurado no Github actions o *build* da aplicação, a execução dos testes e a execução dos verificadores de código, para garantir qualidade e segurança. Para isso foi criado uma organização visto que existirá mais de um repositório relacionado a FARMA, sendo o primeiro o da API.

O CI executará quando chegar uma *Pull Request* (PR) para a main, quando a PR for atualizada ou quando ela for reaberta. No CI exite os jobs, que são uma tarefa específica responsável por executar atividades configuradas previamente, essas configurações são definidas para cada job com suas particularidades. Nesta aplicação existirá dois jobs:

O primeiro é o job de teste, configurado em uma maquina Ubuntu na última versão e dentro dele tera um container com a imagem do Ruby 3.2.2 (última versão). Será necessário também um serviço de banco de dados, que é um container de banco que é igual o que foi feito no Docker Compose, todavia com algumas configurações diferentes por causa do Github Actions. Em seguida são definidos os passos (steps de execução) que são basicamente baixar a aplicação já mergeada com a main. e instalar as dependências, fazendo um *cache* para que nas próximas execuções do CI não necessite baixar novamente e seja mais ágil. Só então é executado os testes, sendo necessário acessar as configurações de banco de dados através das *credentials* do *rails*, para isso não pode ser colocado direto no projeto as *credentials*, então é utilizado o Rails Master Key que é a chave para acessar essas *credentials*, sendo configurada no Github como uma variável secreta, colocando nessa configuração a chave que abre essas credenciais para acessar as configurações. Em seguida prepara o banco de testes e executa-os.

O segundo job é utilizado para verificar a qualidade do código, ele é bem semelhante ao anterior, também executará em uma maquina Ubuntu na última versão onde irá baixar a aplicação, configurar o Ruby, instalar as dependências necessárias. Neste job sera executado 3 verificadores de código que serão:

- RuboCop, um analisador de código estático Ruby e formatador de código. Além de relatar os problemas descobertos em seu código, o RuboCop também corrigi automaticamente alguns erros.
- Brakeman, uma ferramenta de análise estática que verifica os aplicativos Ruby on Rails em busca de vulnerabilidades de segurança.
- Rubycritic, uma gem que envolve gems de análise estática como Reek, Flay e Flog para fornecer um relatório de qualidade do seu código Ruby.

Para o CD é necessário configurar um Servidor Virtual Privado, do inglês Virtual Private Server (VPS). Nesse foi instalado o Docker para gerenciar os containers e Docker Compose para orquestrar os diversos containers necessários, como, por exemplo, o container de aplicação e de banco de dados. Além disso, foi necessário instalar o servidor de aplicações ngnix na VPS para ser utilizado como proxy reverso, o qual recebe as requisições e repassa para o container da aplicação. Para garantir acesso seguro a VPS, foi configurado o Secure Shell Protocol (SSH) por meio das chaves públicas e privadas.

Com a VPS configurada, foi definido os passos a serem realizado pelo Github Actions para a entrega contínua. Sendo que um *deploy* é realizado a cada nova atualização recebida na branch main, para não ter inconsistências, as a adição de novas funcionalidades e correções somente são aceitas por meio de PR. Quando uma PR é aceita na branch main, o o processo de *deploy* se incia, seguindo o seguinte processo:

1. Baixar versão atual da branch main;
2. Realizar o build da aplicação;
3. Executar os testes;
4. Executar os linters;

Se tudo passar com sucesso realiza o *deploy* por meio dos seguintes passos:

1. Cria uma imagem nova contendo a versão mais atual da aplicação;
2. Registra essa imagem no Container Registry do Github;
3. Loga no servidor via SSH usando a chave privada;
4. Baixa a imagem mais atual do Container Registry.

5. Atualiza a imagem em execução para mais atual.

Todo esse processo é automatizado usando Github Actions o que permite agilidade na entrega das novas funcionalidades para teste em um ambiente de Staging.

Com isso a API foi configurada com sucesso. Na Figura 14 visualizamos o Projects do Github, que será utilizado para gerenciar e descrever todas as tarefas que deverão serem executadas neste projeto.

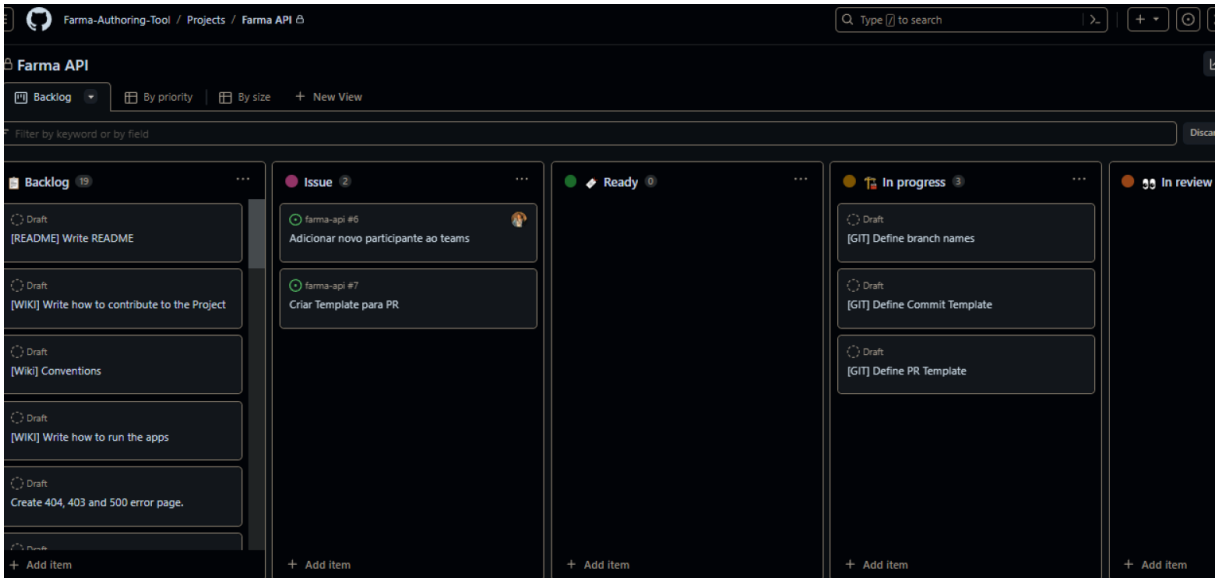


Figura 14 – Projects FARMA API
Fonte: Autoria própria (2023).

4.3.2 Exemplo de funcionamento da API

Segue abaixo um exemplo do fluxo da aplicação que acabamos de configurar. Na Figura 15 é possível visualizar como a API estava funcionando após ter sido configurada.

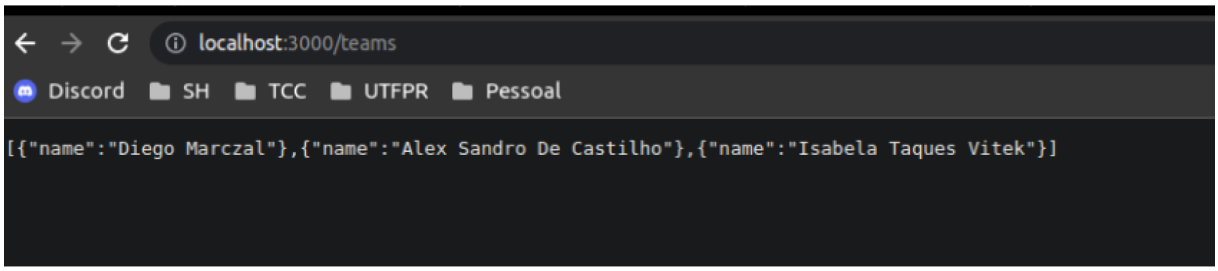


Figura 15 – Rota Index da API
Fonte: Autoria própria (2023).

Na Figura 16 vizualiza-se os comandos utilizados para criar a *branch* para realizar as modificações, nomeada como 'exemple-modification-api'. Nesta mesma figura observa-se quais foram as modificações. Em seguida na Figura 17 exibe no site as modificações já aplicadas.

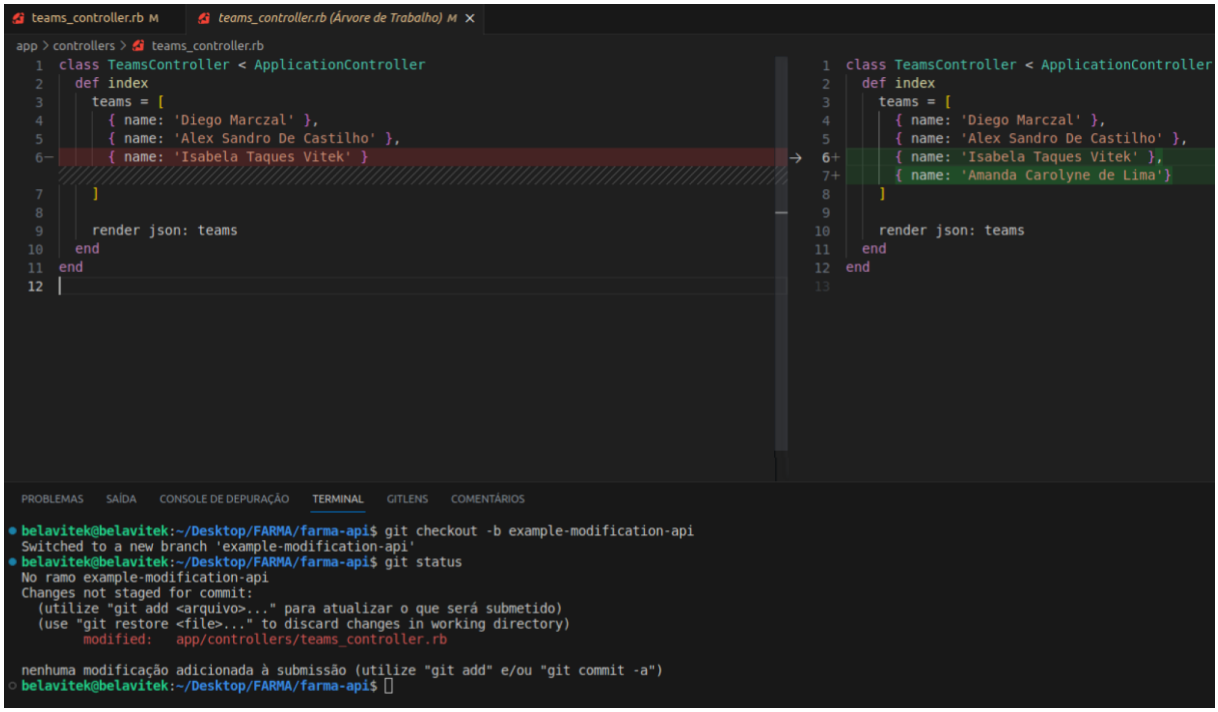


Figura 16 – Branch de modificação
Fonte: Autoria própria (2023).

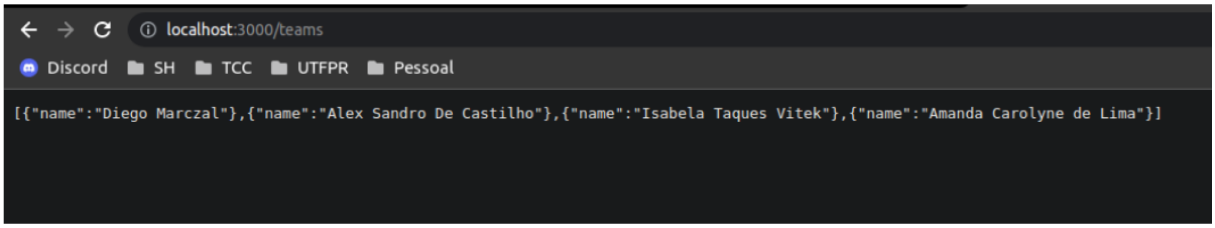


Figura 17 – Rota Index da API com a modificação
Fonte: Autoria própria (2023).

Com as modificações funcionando da maneira esperada é realizado o commit de todos os arquivos que foram modificados, conforme a Figura 18.

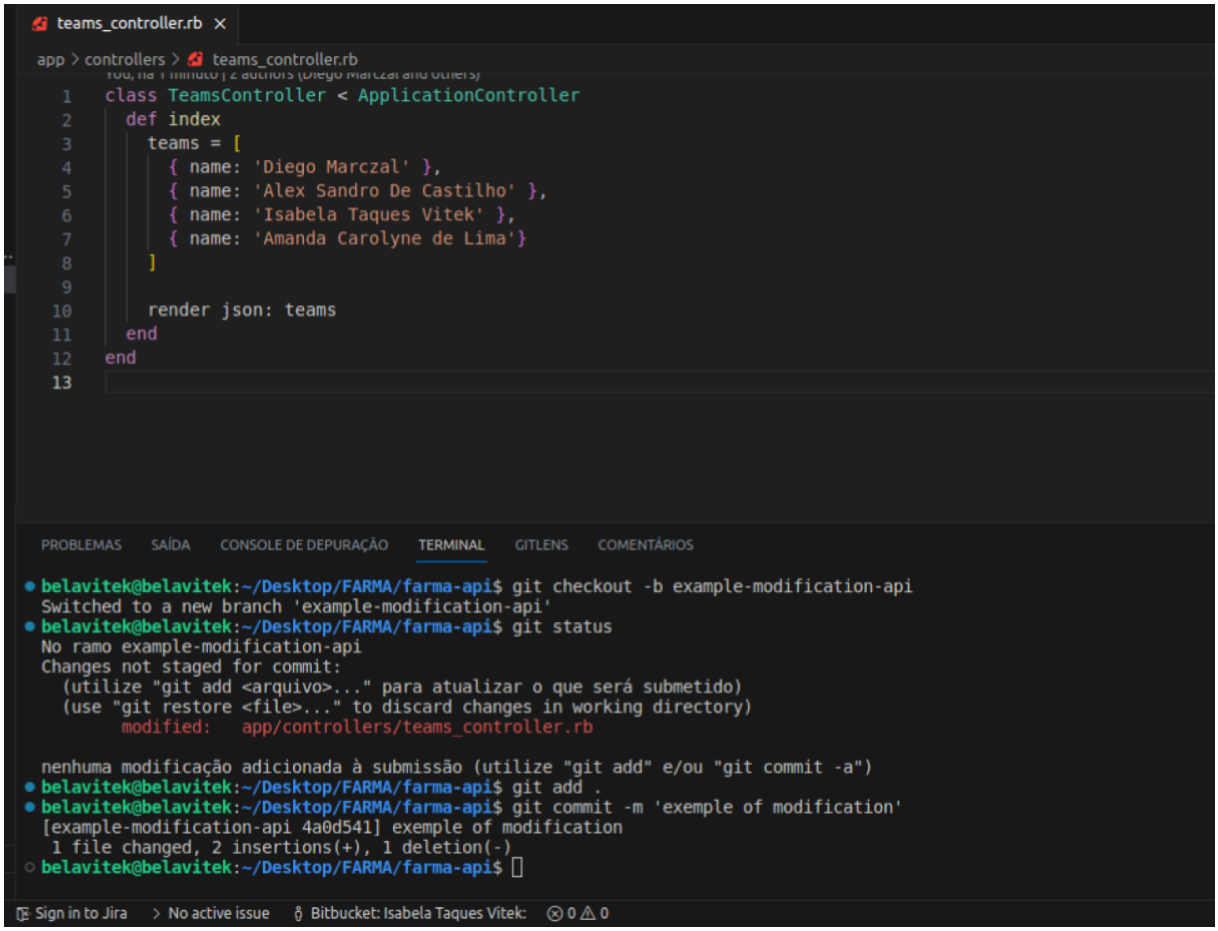


Figura 18 – Commit das modificações

Fonte: Autoria própria (2023).

Após realizar o commit foi executados os testes conforme a Figura 19, Figura 20, Figura 21 e Figura 22, onde possuí um exemplo, no primeiro teste executado, de uma falha que necessita de correção e respectivamente sua correção, exibindo em seguida o teste bem sucedido assim como os demais que foram executados na sequência.

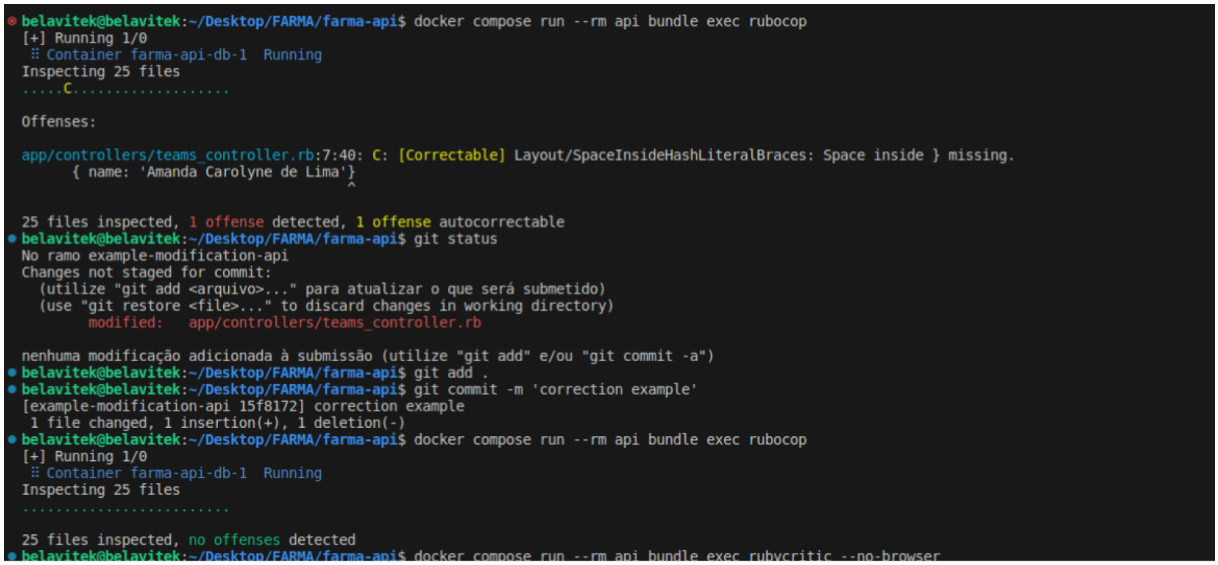


Figura 19 – Teste Rubocop e exemplo de uma correção

Fonte: Autoria própria (2023).

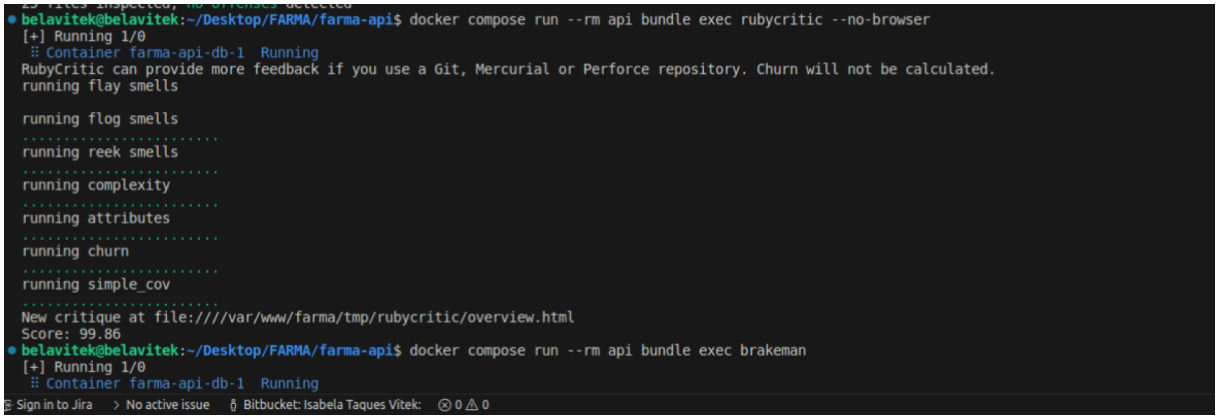


Figura 20 – Teste Ruby Critic

Fonte: Autoria própria (2023).

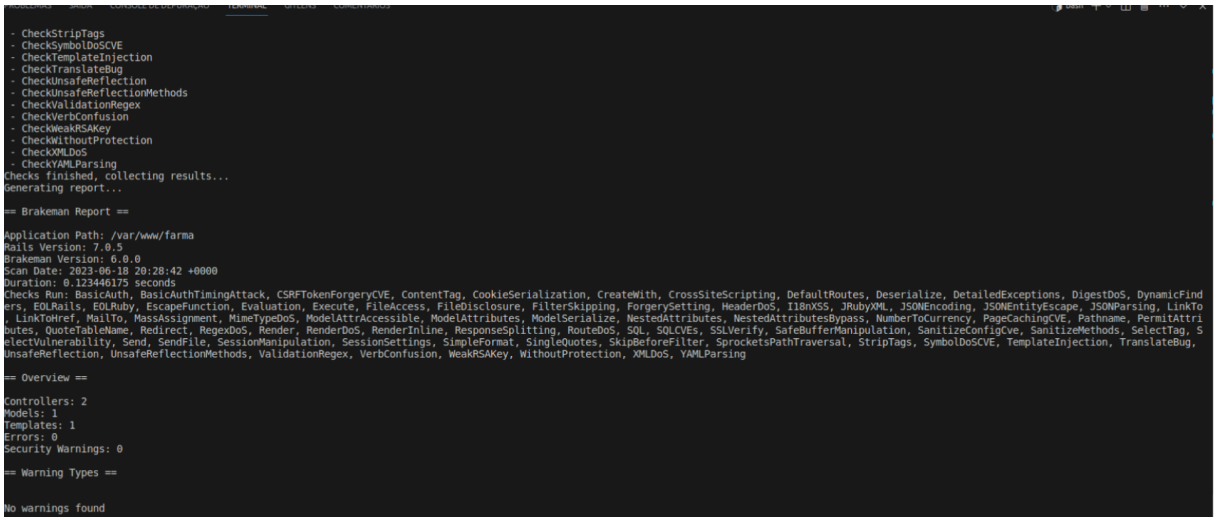


Figura 21 – Teste Brakeman

Fonte: Autoria própria (2023).

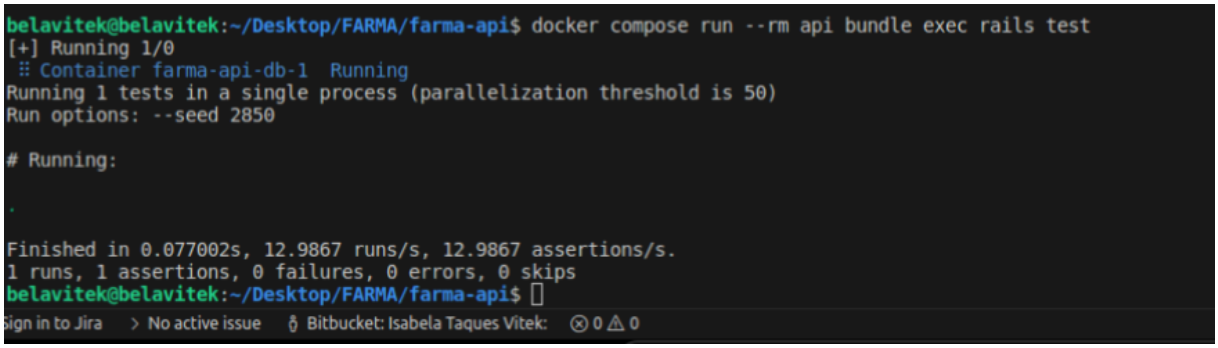


Figura 22 – Rails teste
Fonte: Autoria própria (2023).

Com os testes bem sucedidos a *branch* foi enviada para o Github e vai ser criada uma PR para essa *branch*, conforme a Figura 23, que será submetida para revisão na Figura 24 e para os testes de CI previamente configurados, ilustrados na Figura 25.

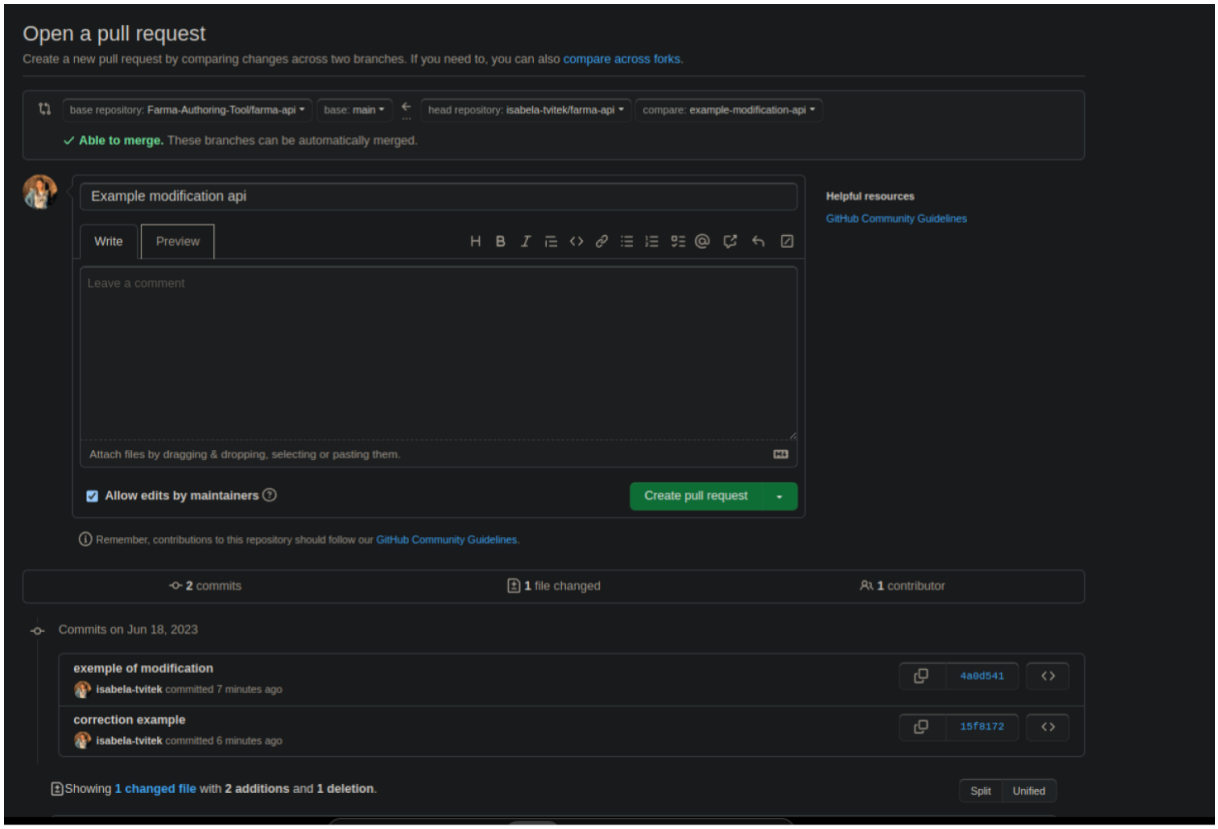


Figura 23 – Criação da PR
Fonte: Autoria própria (2023).

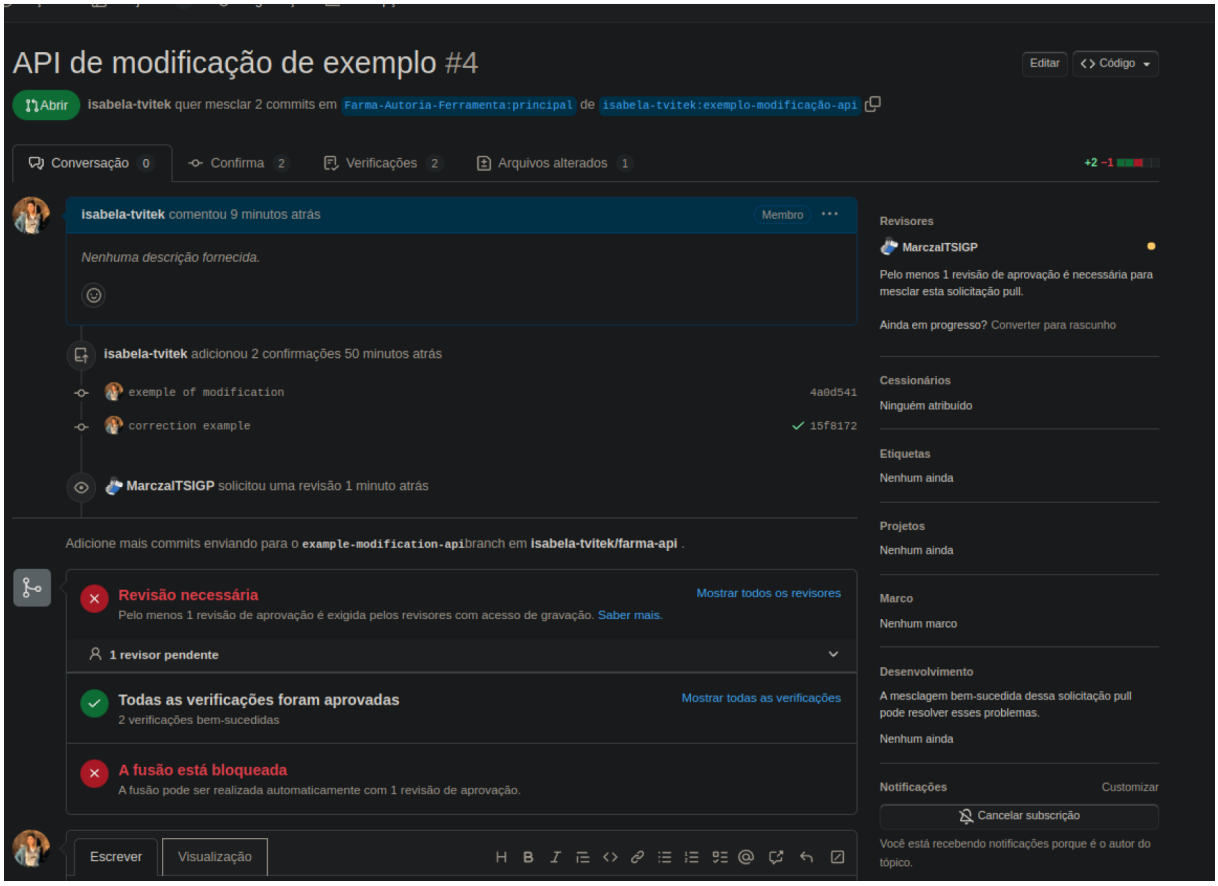


Figura 24 – Visualização da PR
Fonte: Autoria própria (2023).

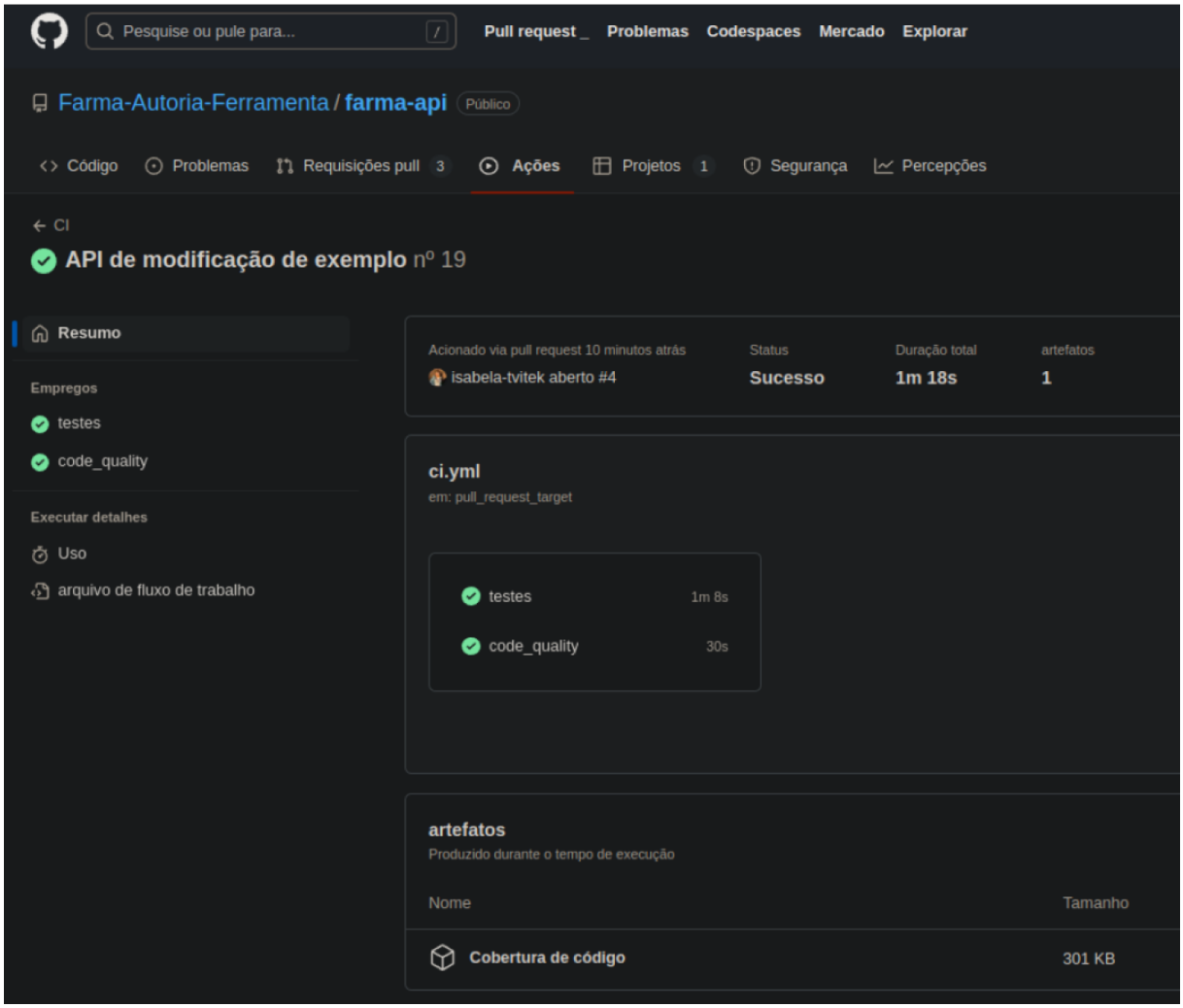


Figura 25 – Execução do CI
Fonte: Autoria própria (2023).

Em seguida o revisor deverá aprovar esta PR conforme é solicitado na Figura 26, o mesmo pode solicitar mudanças e correções adicionando comentários sobre o que deve ser alterado conforme a Figura 27. Com todas as modificações corretas nesta PR é realizada a aprovação conforme a Figura 28 e já está pronta para ser mergeada.

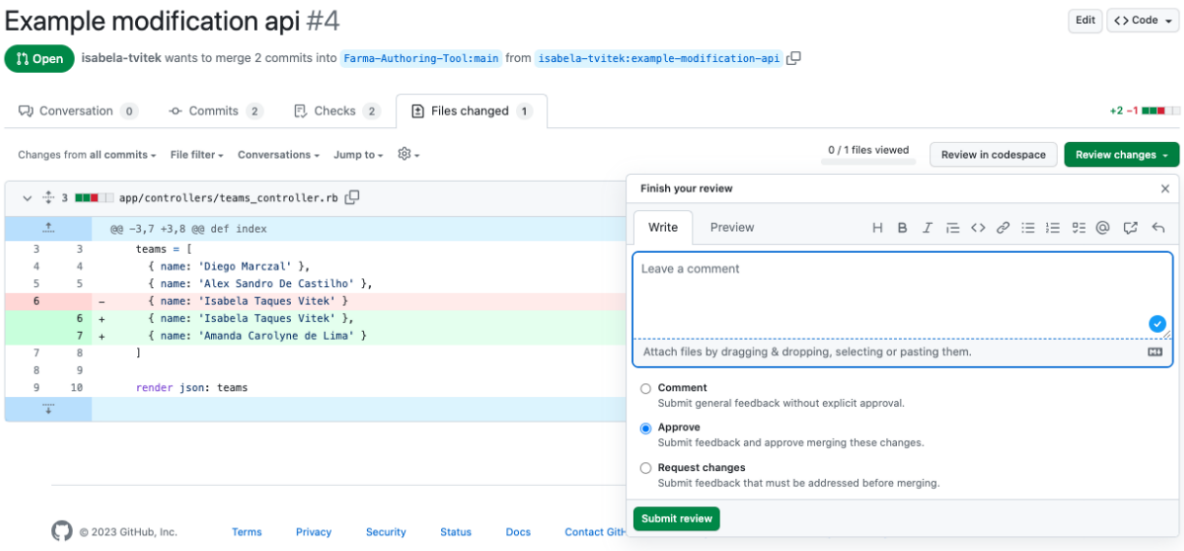


Figura 26 – Aprovação da PR

Fonte: Autoria própria (2023).

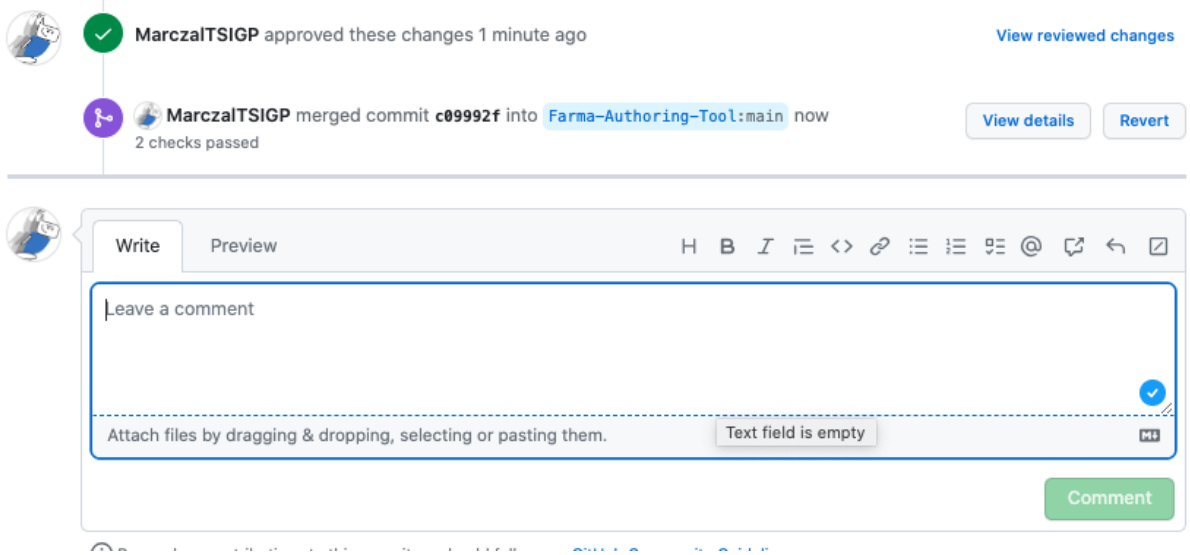


Figura 27 – Comentários na PR

Fonte: Autoria própria (2023).

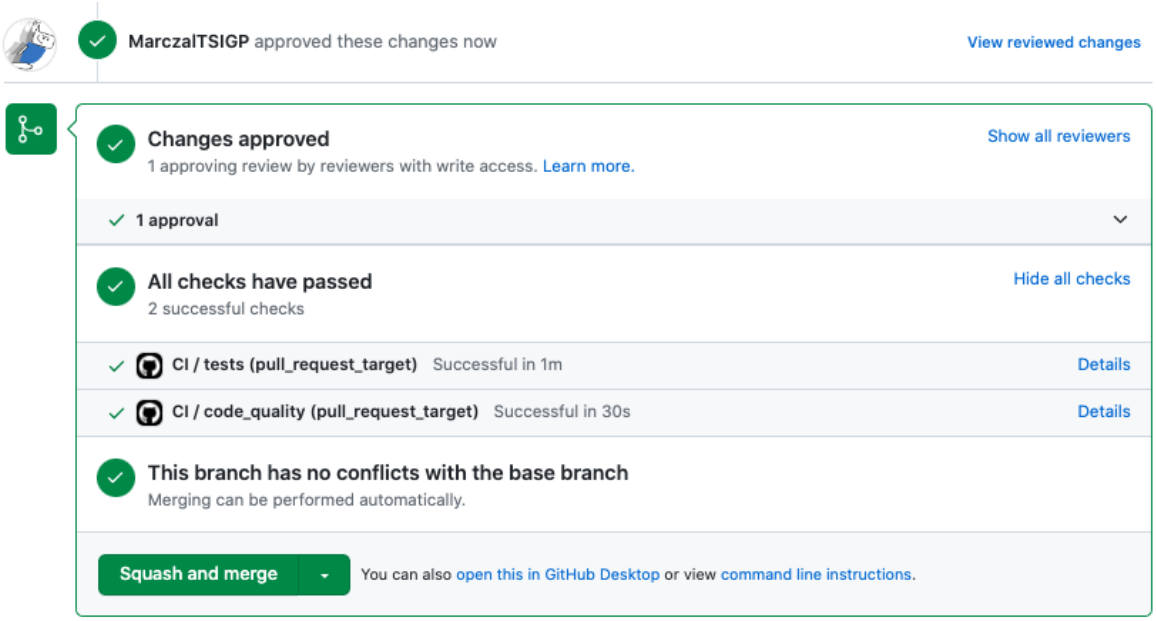


Figura 28 – PR aprovada

Fonte: Autoria própria (2023).

5 CONCLUSÃO

Conforme explicado no Capítulo 2 o sistema já possuí versões anteriores que ainda apresentam carências e seus recursos não são unificados em apenas uma versão. Portanto, o trabalho proposto tem como objetivo dar continuidade em um projeto já existente, onde será desenvolvido uma nova versão que englobe todas as funcionalidades existentes e preencha as demandas de melhorias de forma incremental. O foco principal é desenvolver a área de professor, para que os docentes consigam criar os OA's e gerir os mesmos, contando com uma interface agradável e bonita sem complexidade de uso.

No desenvolvimento deste trabalho deverão ser atendidos os requisitos descritos na Seção 4.1. Nesta Seção é detalhado as funcionalidades esperadas para a área do professor neste sistema. Na Seção 4.3 é possível entender como foi realizado as configurações do ambiente inicial da API, com todas a tecnologias utilizadas e um exemplo de como é o fluxo para cada resolução de tarefas. Este projeto está cumprindo os prazos propostos para a entrega de cada etapa, conforme estabelecidos na Tabela 2, com exceção da prototipação de telas que não possuí prioridade, em primeiro momento, frente as demais atividades que foram realizadas e preencheram o tempo disponível.

Tabela 2 – Cronograma de Atividades.

Atividades	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
1. Escrita da Proposta										
2. Defesa da Proposta										
3. Entrega da proposta revisada de acordo com os apontamentos da banca										
4. Escolha do framework Css com os orientadores										
5. Configuração do ambiente de desenvolvimento										
6. Prototipar algumas telas do sistema										
7. Escrita do Projeto										
8. Defesa do Projeto										
9. Entrega do projeto revisado de acordo com os apontamentos da banca										
10. Desenvolvimento do sistema										
11. Implantação										
12. Escrita da Monografia										
13. Defesa da Monografia										
14. Entrega da monografia revisada de acordo com os apontamentos da banca										

Fonte: Autoria própria (2023).

REFERÊNCIAS

AGUIAR, E. V. B.; FLÔRES, M. L. P. Objetos de aprendizagem: conceitos básicos. **Objetos de aprendizagem: teoria e prática. Porto Alegre: Evangraf**, p. 12–28, 2014.

KENSKI, V. M. **Educação e tecnologias: o novo ritmo da informação**. [*S.l.*]: Papirus editora, 2003.

KUTZKE, A. R.; DIRENE, A. Informática educacional e a mediação do erro na educação: um estudo teórico-crítico e uma proposta de instrumento computacional. *In*: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [*S.l.*: *s.n.*], 2016. v. 5, n. 1, p. 12.

LEITE, M. D. Arquitetura para remediação de erros baseada em múltiplas representações externas. 2013.

LORENZO, C. O consentimento livre e esclarecido e a realidade do analfabetismo funcional no brasil: uma abordagem para a norma e para além da norma. **Revista Bioética**, Conselho Federal de Medicina, v. 15, n. 2, p. 268–282, 2007.

MARCZAL, D.; DIRENE, A. I. Um arcabouço que enfatiza a retroação a contextos de erro na solução de problemas. **Revista Brasileira de Informática na Educação**, v. 19, n. 01, p. 63, 2011.

MARCZAL, D. *et al.* Farma: Uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos. Universidade Federal do Paraná, 2014.

MOURA, V. A. B. de; PERES, L. M. Avaliação do impacto da retroação na aprendizagem apoiada por uma ferramenta educacional. **Revista Brasileira de Informática na Educação**, v. 25, n. 01, p. 60, 2017.

SANTOS, A. G. d. **Desenvolvimento do módulo de estatísticas da ferramenta de autoria de objetos de aprendizagem FARMA**. 2022. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2022.

SANTOS, J. A.; FRANÇA, K. V.; SANTOS, L. S. B. d. Dificuldades na aprendizagem de matemática. **Monografia de Graduação em Matemática. São Paulo: UNASP**, 2007.

SILVA, R. C.; DIRENE, A. I. **Sequenciamento Adaptativo de Exercıcios Baseados na Correspondência entre a Dificuldade da Soluçao e o Desempenho Dinâmico do Aprendiz**. 2015. Tese (Doutorado) — Tese de doutorado, Universidade Federal do Paraná, Curitiba, 2015.

SILVA, R. F. d.; CORREA, E. S. Novas tecnologias e educação: a evolução do processo de ensino e aprendizagem na sociedade contemporânea. **Educação e Linguagem, ano**, v. 1, n. 1, p. 23–25, 2014.

TAROUCO, L. M. R. *et al.* Objetos de aprendizagem: teoria e prática. Evangraf, 2014.

VICENTE, J. J. N. B.; ROCHA, M. F. L. da. Sobre o ato de educar: um breve olhar. **Saberes: Revista interdisciplinar de Filosofia e Educação**, v. 22, n. 1, p. 78–93, 2022.

APÊNDICE A – Casos de uso dos usuários do sistema FARMA

A.1 Relação dos usuários.

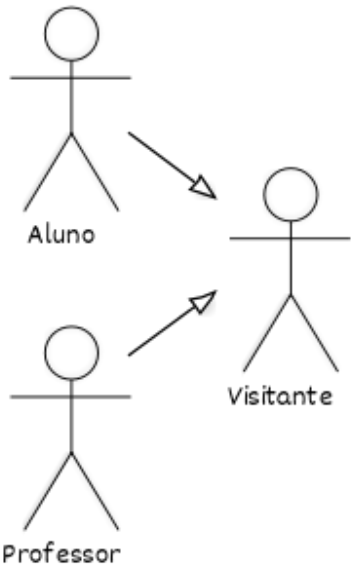


Figura 29 – Relação dos usuários
Fonte: Autoria própria (2023).

A.2 Caso de Uso Visitante.



Figura 30 – Caso de uso visitante
Fonte: Autoria própria (2023).

A.3 Caso de Uso Aluno.

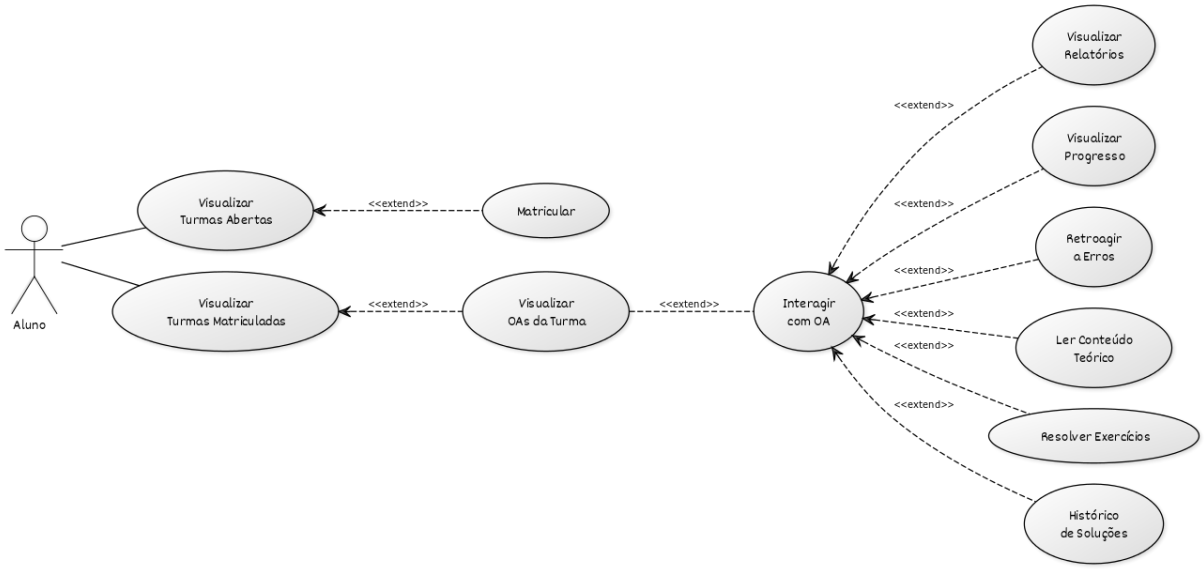


Figura 31 – Caso de uso aluno
Fonte: Autoria própria (2023).

A.4 Caso de Uso Professor.

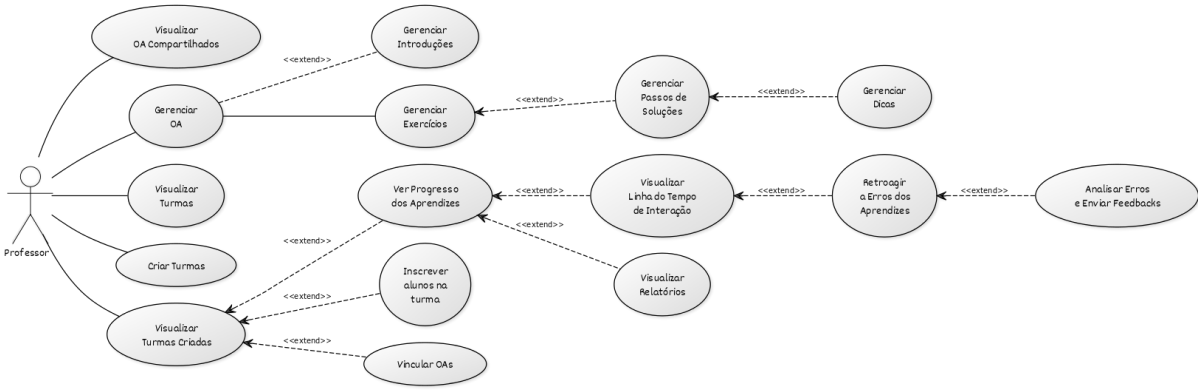


Figura 32 – Caso de uso professor
Fonte: Autoria própria (2023).