UNIVERSIDADE FEDERAL DO PARANÁ

BIANCA RICI FRANCO

DANIEL MACHADO PINTOS

GIULIA CESAR PADOVANI

RENAN ROMANIO SANTOS

GITLAB CLASSROOM

CURITIBA

2020

BIANCA RICI FRANCO

DANIEL MACHADO PINTOS

GIULIA CESAR PADOVANI

RENAN ROMANIO SANTOS

GITLAB CLASSROOM

Trabalho referente à disciplina DS960- Trabalho de Conclusão de Curso II- apresentado para UFPR no curso Tecnologia em Análise e Desenvolvimento de Sistemas, como requisito para obtenção de diploma.

Professor orientador: Alexander Robert Kutzke

CURITIBA

2020

UNIVERSIDADE FEDERAL DO PARANÁ

ATA DE REUNIÃO

TERMO DE APROVAÇÃO

Bianca Rici Franco
Daniel Machado Pintos
Giulia Cesar Padovani
Renan Romanio Alves Santos

Gitlab Classroom

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)
Orientador – SEPT/UFPR

Prof Dr. Mauro Antonio Alves Castro (Membro)
SEPT/UFPR

Prof. Dr. João Eugênio Marynowski (Membro)
SEPT/UFPR

Curitiba, 14 de Dezembro de 2020.

Documento assinado eletronicamente por **ALEXANDER ROBERT KUTZKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/12/2020, às 21:08, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **MAURO ANTONIO ALVES CASTRO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 15/12/2020, às 10:24, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **JOAO EUGENIO MARYNOWSKI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 15/12/2020, às 15:01, conforme art. 1º, III, "b", da Lei 11.419/2006.

A autenticidade do documento pode ser conferida aqui informando o código verificador **3186599** e o código CRC **01AA15A8**.

Referência: Processo nº 23075.067329/2020-34 SEI nº 3186599

A todos os futuros usuários deste sistema, que nos deram uma razão para desenvolvê-lo e ao nosso Professor-Orientador Alexander Robert Kutzke, que confiou esta ideia maravilhosa ao nosso grupo e sem o qual não seria possível torná-la realidade.

AGRADECIMENTOS

Aos professores, que durante os anos de curso forneceram o conhecimento necessário para a elaboração deste trabalho segundo as normas atuais.

Aos nossos familiares, que forneceram o suporte necessário e foram compreensivos quanto aos eventos deixados de lado em prol do desenvolvimento deste projeto.

A todos aqueles que contribuíram de alguma forma para a concretização deste software.

“A tarefa não é tanto ver aquilo que ninguém viu,

mas sim, pensar o que ninguém ainda pensou

sobre aquilo que todo mundo vê.”

(Arthur Schopenhauer)

RESUMO

Ferramentas de versionamento de código se consolidam cada vez mais como um requisito básico ao invés de um diferencial nas empresas de análise e desenvolvimento de sistemas. Desta forma, para que os futuros profissionais estejam devidamente capacitados se faz necessária a implementação destes conceitos e domínio da ferramenta já durante a formação acadêmica. O Git lidera o mercado nesta área, como o software mais utilizado e acessível, possuindo uma grande linha de produtos. Um deles é o GitHub ClassRoom, sistema voltado para ensino, com um ambiente que simula uma turma e permite a interação entre aluno e professor, criação, realização e correção de tarefas e feedback de cada uma. Este sistema contudo não é inteiramente gratuito, o que acaba impedindo a utilização do mesmo por instituições como a UFPR. O software criado pela equipe, o GitLab Classroom, surge então como uma alternativa gratuita, integrada ao GitLab, apresentando funcionalidades modeladas às necessidades tanto do corpo docente quanto discente da universidade. O projeto aqui apresentado tem como base as premissas do crescente uso do Git pelo área de TI e a necessidade de um sistema que facilite o ensino e a aprendizagem de tais ferramentas em sala de aula

Palavras-chave: Git. GitLab Classroom. TI. Software. Universidade.

ABSTRACT

Code versioning tools are increasingly consolidated as a basic requirement rather than a differential in systems analysis and development companies. Thus, for future professionals to be properly trained, it is necessary to implement these concepts and master the tool already during academic training. Git leads the market in this area, as the most used and accessible software, with a wide range of products. One of them is GitHub ClassRoom, a system geared towards teaching, with an environment that simulates a class and allows interaction between student and teacher, creation, execution and revision of tasks and feedback for each one. This system, however, is not entirely free, which ends up preventing its use by institutions like UFPR. The software created by the team, GitLab Classroom, then appears as a free alternative, integrated with GitLab, presenting features tailored to the needs of both faculty and students of the university. The project presented here is based on the premises of the growing use of Git by the IT area and the need for a system that facilitates teaching and learning such tools in the classroom.

Keywords: Git. GitLab Classroom. TI. Software. University.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O sistema aqui apresentado é o GitLab Classroom e tem por objetivo ser uma alternativa *open source* ao GitHub Classroom, este último sendo uma plataforma que oferece automação na criação e controle de acesso a repositórios do GitHub voltada ao meio acadêmico.

O desenvolvimento deste software surgiu da necessidade de docentes do SEPT que utilizam o GitLab, cujo uso é gratuito, como plataforma de administração de atividades relacionadas a codificação e perceberam a necessidade de uma ferramenta que ofereça funcionalidades similares às ofertadas pelo GitHub Classroom, mas sendo integrada ao GitLab Tads.

O software, em suma, tem um funcionamento fundamentado na premissa de facilitar a delegação e correção de exercícios pelo professor e incentivar o uso da ferramenta Git por parte dos alunos. Dessa forma, torna-se uma interface que atua como mediadora na interação de ambos com o GitLab, enquanto ferramenta de versionamento, e como auxiliar do professor no que tange à administração das atividades da turma.

Partindo-se deste princípio, o sistema trabalha com dois tipos de usuário: o Professor, que possui privilégios que lhe permitem criar, administrar e avaliar turmas e tarefas, e o Aluno, que pode submeter suas respostas às atividades propostas pelo professor de sua turma. Seguindo um fluxo base de eventos do software, o professor e o aluno se cadastram na plataforma, fornecendo seu GRR ou e-mail e concedendo acesso ao seu perfil no GitLab. O professor cria então uma turma, nomeando-a e adicionando os alunos correspondentes que já estejam cadastrados no GitLab Classroom. Caso não estejam, o professor deve fornecer o usuário GitLab do aluno para que seja então enviado ao estudante um convite para que realize seu cadastro.

Com uma turma devidamente criada, o professor pode então criar tarefas, que são notificadas aos alunos cadastrados na turma, estes últimos podendo então realizar as atividades, enviando suas respostas. O professor pode ainda visualizar quantos e quais alunos realizaram cada tarefa, corrigir cada resposta e apontar plágios e inconsistências.

1.1 JUSTIFICATIVA

O trabalho aqui apresentado se justifica pela crescente demanda de profissionais de Tecnologia da Informação com conhecimento superior ao básico em ferramentas de versionamento código, como o Git, que acaba por implicar diretamente nos conteúdos e na forma com que devem ser ministrados nos cursos dessa área.

Assim como o domínio de uma segunda língua em boa parte das profissões, no desenvolvimento e análise de softwares a proficiência no uso de ferramentas para controle de versões de código nos últimos anos tem deixado de ser um diferencial para se configurar como um requisito na maioria das empresas desse ramo. Tendência essa que é até mesmo apontada por alguns como tardia, tendo em vista a importância de tal habilidade e o valor que agrega tanto ao profissional quanto ao desenvolvimento do produto final e à base de conhecimento da organização.

Logo, buscando oferecer uma formação de qualidade e que de fato prepare o indivíduo para o mercado de trabalho, os cursos de TI buscam introduzir o uso desta ferramenta logo nos primeiros semestres e avançar nas funcionalidades e possibilidades ao longo do curso. Assim como as demais disciplinas, o aprendizado é facilitado e mais eficiente se realizado através do uso de conceitos já familiares, de forma didática e através de uma plataforma amigável.

1.2 OBJETIVOS

1.2 Objetivo geral

Este trabalho tem por objetivo modelar e implementar um software de gerenciamento de repositórios que facilite o aprendizado e desenvolvimento de habilidades relacionadas ao uso de ferramentas de versionamento de código e que também forneça uma plataforma de administração de turmas.

1.2 Objetivos específicos

- Conceituar ferramentas de versionamento de código e seu crescente uso no mercado;
- Avaliar quais ações o aluno deveria executar na ferramenta;
- Identificar funcionalidades que auxiliem o professor a gerenciar as tarefas e a turma;
- Analisar quais as tecnologias mais adequadas para o desenvolvimento do software;
- Construir uma interface intuitiva e didática;
- Modelar o sistema de forma a atender os requisitos propostos;
- Implementar as funcionalidades estipuladas;
- Realizar teste que comprovem o funcionamento do sistema proposto.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda as bases teóricas que levaram à construção do software apresentado neste trabalho, bem como os principais conceitos e fundamentos que levaram à consolidação deste projeto.

2.1 FERRAMENTAS DE VERSIONAMENTO DE CÓDIGO EM SALA

A produção de softwares atualmente é uma atividade realizada em equipes, na qual cada indivíduo desenvolve a parte que lhe cabe de um mesmo projeto em seu próprio computador. Tendo isso em vista, algo que facilmente se tornaria um problema seria a combinação de cada uma dessas partes, desenvolvidas separadamente, em um software funcional. Visando evitar este e outros problemas, surgiram as ferramentas de versionamento de código.

Dentro deste gênero, a ferramenta Git ganhou muito espaço e é amplamente utilizada dentro de organizações de pequeno e grande porte para construção e manutenção dos códigos dos softwares que produzem, e até mesmo dos que são utilizados pela própria empresa. Assim sendo, a formação dos futuros profissionais da área de tecnologia acaba por ter a obrigação de contemplar o uso, ainda que básico, de tais ferramentas.

Segundo resultados obtidos durante um curso voltado ao desenvolvimento web na Universidade Aalto (HAARANEN e LEHTINEN, 2015), no qual foram introduzidos os conceitos e o uso do Git para uma turma cuja maioria não possuía conhecimento prévio de ferramentas de versionamento, grande parte dos alunos afirmaram ver a ferramenta de forma positiva. As respostas positivas ainda variam entre aquelas que relatam a satisfação em aprender uma nova tecnologia e outras que ressaltam a melhora na curva de aprendizado, que foi essencial para o dia-a-dia no ambiente de trabalho.

Alexandre Gouveia (2016) no artigo “Os serviços de versionamento de código” afirma que tais serviços são “Uma maneira mais simples e eficiente da equipe de desenvolvimento possa modificar o mesmo código base sem haver conflitos. Com ele ainda, é possível ver um histórico de todas as modificações feitas pela equipe, criar novas versões e recuperar versões antigas do projeto(...)”. O autor observa, assim, a aplicabilidade desta tecnologia no dia-a-dia dos desenvolvedores de software. Na busca de um diferencial para o trabalho aqui apresentado, a equipe se focou especialmente na parte final deste trecho, buscando explorar as possibilidades que a visualização das alterações realizadas no projeto pode oferecer quando aplicada ao cenário acadêmico.

2.2 GIT

Sistemas de controle de versão são utilizados principalmente no desenvolvimento de softwares, visando manter a integridade e usabilidade do código, assim como monitorar e visualizar o resultado final de cada alteração feita pelos diferentes usuários que o estiverem modificando.

O Git então, sendo um ferramenta desta categoria, funciona de forma que o código original é mantido em um repositório e são criadas cópias, chamadas de *branchs*, que podem ser modificadas segundo o que é desenvolvido por cada usuário em sua máquina e após a finalização são integradas ao todo por meio do gerenciador de versões, que identifica as diferenças, sinaliza conflitos e mantém um histórico das atualizações realizadas tanto na versão final quanto em cada uma das *branchs.*

Neste ramo já existem softwares há algum tempo, mas após o surgimento do Git, em 2005, quase nenhum consegue chegar perto de sua popularidade. Sua acessibilidade como uma ferramenta *open source*, interface amigável e estabilidade atraiu e tem atraído cada vez mais usuários e empresas, que na busca pelo aprendizado de novas tecnologias ou por uma melhor ferramenta para os desenvolvimentos de seus produtos se deparam com Git e suas variadas ofertas,

como o GitLab e o GitHub.

2.3 GITHUB CLASSROOM

Uma ferramenta que serve bem à proposta de incentivar o uso de ferramentas de versionamento por parte dos alunos e auxiliar na condução de atividades do gênero em sala, é o muito bem avaliado e conceituado GitHub Classroom.

“O workflow que desenvolvedores utilizam, tendo 5 estudantes ou 500(...) ”(GITHUB CLASSROOM, 2020), é como a própria ferramenta se apresenta na *home* de seu site, fazendo referência a sua alta escalabilidade, que é somente uma das funcionalidades que tem atraído um grande número de usuários para sua plataforma. Entre os demais atributos que acabaram por servir também como base para o trabalho aqui apresentado estão: a diminuição do tempo de configuração do ambiente de trabalho e maior visibilidade das atividades dos alunos.

A acima referida redução de tempo é ofertada de forma consistente tanto ao usuário com um perfil do tipo docente quanto discente. Ao professor são disponibilizadas funcionalidades e interfaces que simplificam ações como a criação de tarefas e a disponibilização de materiais de consulta para as turmas desejadas, enquanto ao aluno os mesmos atributos se encontram presentes em suas atividades principais, tais como a obtenção dos arquivos necessários para a realização da atividade proposta e de seu próprio repositório da tarefa.

O GitHub em si funciona com os perfis de “Professor” e “Aluno”, dentro do qual os primeiros criam suas turmas, tarefas e as administram, enquanto os estudantes são notificados das atividades, as realizam e recebem o feedback de cada uma. Outras funcionalidades que agregam imenso valor à ferramenta são o sistema de testes automatizado, que permite ao Aluno visualizar qual o resultado do código que escreveu a cada *commit* (FIGURA 1), e o *feedback* linha-a-linha, que permite ao Professor recomendar mudanças, deixar comentários ou um parecer sobre a tarefa a cada linha de código enviada(FIGURA 2).

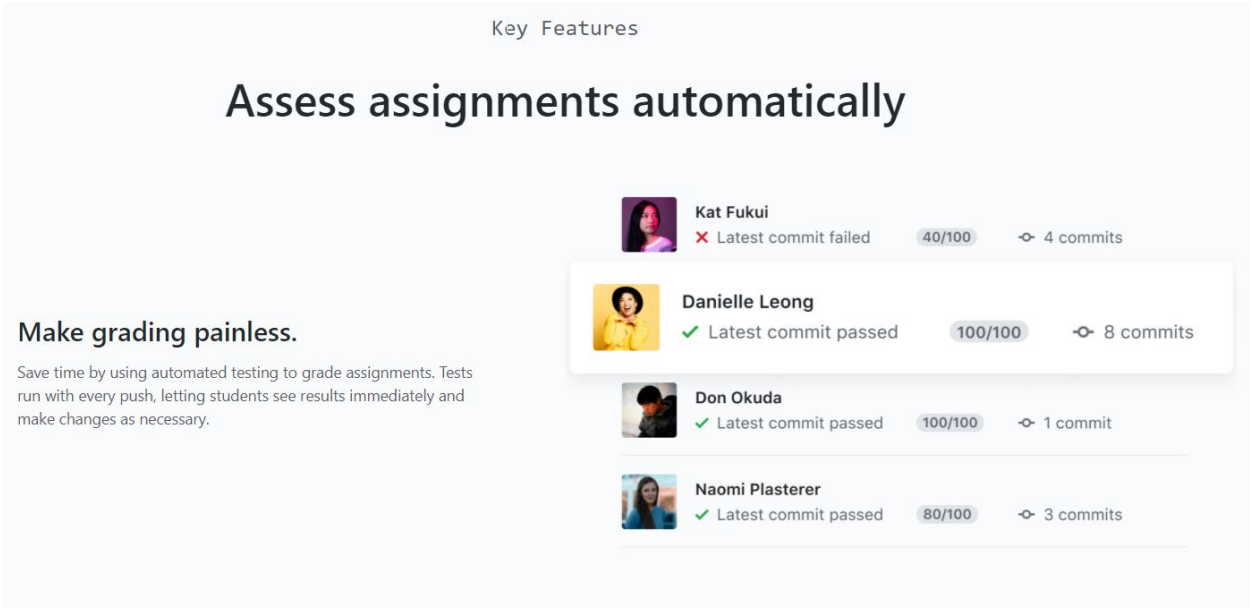


FIGURA 1 - SISTEMA DE TESTE GITHUB

FONTE: GitHub Classroom(2020).

FIGURA 2 - FEEDBACK GITHUB

FONTE: GitHub Classroom(2020).

Entre as semelhanças que existem entre o GitHub ClassRoom e o projeto aqui apresentado estão a dinâmica de dois tipos de usuário -Professor e Aluno- e a estrutura de turma que a plataforma fornece. Tendo-se em vista que o GitLab Classroom foi desenvolvido com a ferramenta como base e a mesma necessidade por parte do corpo docente, os softwares não poderiam ser intrinsecamente diferentes neste aspecto, pois é o que os caracteriza como uma plataforma de versionamento de código voltada para o meio acadêmico. O que o trabalho desenvolvido pela equipe traz de diferente é sua acessibilidade, visto que, ao contrário do produto GitHub, se propõe a ser uma ferramenta completamente gratuita e *open source*.

Uma das funcionalidades exclusivas da nova plataforma é o sistema de avaliação, que ao invés de possuir um processo automatizado de testes fornece ao Professor um meio para avaliar individualmente cada tarefa, atribuir uma nota de acordo com seu sistema de classificação e ainda controlar e computar as ocorrências de plágio, podendo assim destinar atenção especial a esta prática e discutir também o tópico com seus alunos. Outro requisito elaborado especificamente para o GitLab Classroom é a exibição de gráficos que ilustram a quantidade de alunos que realizaram as tarefas e a quantidade de *commits* por tarefa, fornecendo assim ao Professor uma visualização mais clara do desempenho dos alunos e dos resultados de cada tarefa.

3 MATERIAIS E MÉTODOS

O presente capítulo descreve as tecnologias utilizadas e o motivo pelo qual foram escolhidas, quais foram as atividades por cada membro do time e também quais as etapas que resultaram neste projeto e o cronograma no qual foram distribuídas.

3.1 HISTÓRICO E ETAPAS DO TRABALHO

Este trabalho teve início quando os membros do grupo e o orientador foram definidos. Posteriormente foram levantados os possíveis temas, que variaram entre ideias individuais, trabalhos iniciados em outras disciplinas e algumas sugestões de ferramentas com fins acadêmicos advindas do próprio orientador. Optou-se então por seguir com uma das sugestões do Professor Alexander, a qual é aqui apresentada.

O passo seguinte foi o estudo de ferramentas similares e das possibilidades que poderiam ser exploradas tanto no que tange à função da ferramenta quanto às tecnologias que seriam mais adequadas para sua construção. Seguido do alinhamento disso com as ideias dos membros e por fim a definição do software em si. Esta etapa consistiu em reuniões para discussão de ideias, esboços da interface planejada e de fluxos de atividade para as funcionalidades que foram elencadas. A fase foi finalizada com a entrega e defesa da documentação do planejamento do projeto, juntamente a um protótipo, durante a defesa do trabalho na disciplina TCC - 1.

A terceira etapa foi constituída, então, da codificação do trabalho, alinhada à elaboração da documentação requisitada na regulamentação da Universidade. Foram também realizadas as mudanças necessárias quanto às tecnologias planejadas e diagramas para que fosse possível elaborar um software atualizado, consistente e documentado de forma satisfatória. Esta fase é finalizada pela defesa final deste trabalho, tanto a ferramenta em si quanto a documentação, perante uma banca de docentes da própria instituição.

3.2 CRONOGRAMA

Seguindo o fluxo das fases acima descritas, o processo de elaboração deste trabalho se dividiu em semestres. Durante o primeiro, de Fevereiro a Julho de 2019, foram desenvolvidas as 2 primeiras etapas, durante as quais foram realizadas reuniões semanais nas quartas-feiras para revisão dos conceitos e objetivos do projeto, do que havia sido produzido e das entregas futuras.

Os semestres seguintes, compreendidos entre Agosto de 2019 e Novembro de 2020, foram dedicados à codificação do software, construção deste documento, revisões e ajustes necessários para conclusão destes anteriormente citados. Durante o ano de 2019 as reuniões eram realizadas a cada 15 dias, com a presença de todos nas dependências da Universidade. Já em 2020, devido à pandemia do COVID-19, as reuniões foram realizadas através da plataforma Microsoft Teams e ocorreram de forma semanal, às quintas ou segundas-feiras de acordo com a disponibilidade do Professor e da equipe.

Abaixo segue uma tabela com uma síntese das atividades desempenhadas ao longo do tempo. Os meses de Janeiro não foram utilizados para desenvolvimento por serem parte das férias, assim como Agosto de 2019, visando o bem-estar da equipe.

Atividades	Período
Definição do tema	Fevereiro/2019
Estudo do GitLab/GitHub	Março/2019
Desenvolvimento de diagramas	Março/2019 - Maio/2019
Estudo das tecnologias viáveis	Maio/2019 - Julho/2019
Construção de esboços da interface	Maio/2019 - Julho/2019
Elaboração do Documento TCC 1	Junho/2019 - Julho/2019
Defesa do TCC 1	Julho/2019
Codificação	Setembro/2019 - Novembro/2020
Revisão das Tecnologias	Novembro/2019 - Julho/2020
Elaboração do Documento Final	Setembro/2019 - Novembro/2020
Defesa do TCC	Dezembro/2020

TABELA 1 – CRONOGRAMA

FONTE: Os autores. (2020).

3.3 DIVISÃO DE RESPONSABILIDADES

A equipe foi composta por 4 membros, cada um com maior interesse e proficiência em uma área da tecnologia da informação. Estes foram os fatores determinantes na divisão das responsabilidades da execução deste trabalho, a qual foi estabelecida por meio de uma reunião entre as partes e posteriormente apresentada ao orientador.

Como responsável pelo estudo da API do GitLab, construção do *back-end* e também parte da integração entre as funcionalidades foi definida Giulia Padovani. Do restante da integração, design das telas e a construção do *front-end* ficaram incumbidos Daniel Machado e Renan Romanio. O quarto membro, Bianca Franco, ficou responsável pela definição do banco de dados, elaboração de diagramas e deste documento.

Contudo, visando a construção de um melhor sistema e a aplicação dos conhecimentos de todos os envolvidos, durante as reuniões e ao longo do processo, todos os membros participaram de todos os aspectos do projeto. Ou seja, os principais desenvolvedores atuaram também nas revisões de diagramas, a responsável pela documentação atuou nas revisões de bugs, fluxos do sistema e testes, assim como os que desenvolveram o *front-end* absorveram os conteúdos referentes ao funcionamento da API utilizada no *back-end* e vise-versa.

3.4 TECNOLOGIAS UTILIZADAS

Como apontado anteriormente, cada membro da equipe ficou responsável por um aspecto do trabalho com o qual era mais familiarizado. Nesse sentido, cada um optou por utilizar ferramentas que atendessem aos requisitos estipulados e que proporcionassem também o aprendizado de novas tecnologias.

Para o *front-end* foi utilizado Vue.js (VUE.JS, 2020) com as seguintes dependências:

Dependência	Função
moment	Formatação de horários
chart.js	Criação de gráficos
vuex	Gerenciamento de estado e controle de dados
vue-mq	Formatação de diferentes tipos de *divises*
vue-router	Visualização e mapeamento de rotas
vuetify	Design, semelhante ao *Bootstrap*
vue-toasted	Criação de caixas de diálogo
vue-gravatar	Configuração de imagens de usuário
vue-the-mask	Formatação e aplicação de máscara em campos de inserção de dados

TABELA 2 – DEPENDÊNCIAS VUE.JS

FONTE: Os autores. (2020).

O *front-end*, como estabelecido para o próprio, ficou incumbido do design das telas, tratamento de mensagens de erro -de forma que fosse traduzidos em texto claros e objetivos-, animações e elaboração dos gráficos apresentados. Os membros responsáveis por esta parte do trabalho também atuaram na integração do sistema, correlacionando as ações em tela com os processos executados no *back-end.*

Devido à natureza do trabalho, no *back-end* foi utilizada a API do GitLab -uma interface que realiza solicitações ao GitLab para as ações correspondentes no sistema- juntamente com o *framework* PHP Laravel. As definições, nomenclaturas de *endpoints* e respostas de cada um deles estão documentados no apêndice X (Documentação da Giulia).

Para o armazenamento de dados optou-se por um banco MySQL, juntamente ao Redis e ao próprio GitLab, visando a consistência necessária entre as informações exibidas no GitLab Classroom e os artefatos e transações em tempo real nos repositórios do Git. Para toda a comunicação com o GitLab foi utilizado o pacote do Guzzle, uma biblioteca de comunicações HTTP do próprio PHP.

Quanto ao hardware, foram utilizados os materiais dos próprios alunos: um Macbook 16gb, com 256 GB de armazenamento, processador Intel Core i5, produzido em 2019 e um notebook Dell Inspiron 5548, 480 GB de memória e um processador i5.

3.5 ARQUITETURA DO SISTEMA E COMUNICAÇÃO COM O GITLAB

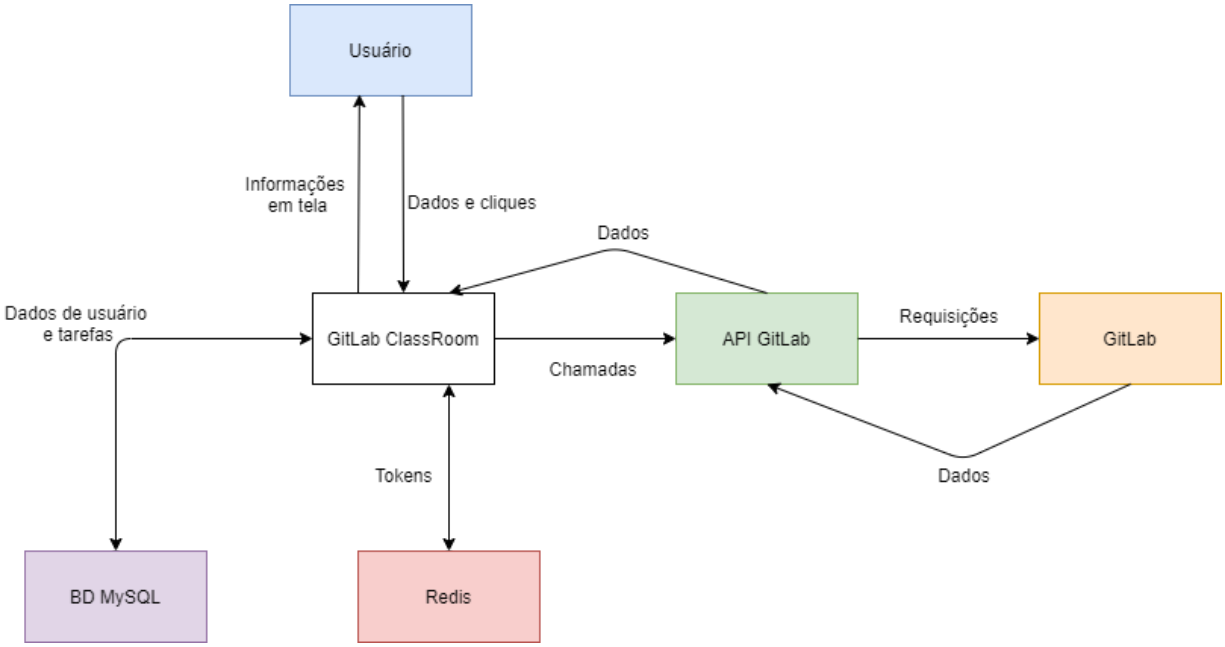


FIGURA 3 - DIAGRAMA DE ENTENDIMENTO DO SISTEMA

FONTE: Os autores(2020).

3.5.1 Login ao Sistema

O login do sistema se dá por meio do protocolo OAuth2, o que significa que para logar no GitLab Classroom é necessário ter uma conta na aplicação GitLab. Para essa funcionalidade, foi implementada uma integração direta com o GitLab usando o método Web Application Flow, onde o fluxo acontece da seguinte forma:

1. A nossa aplicação é cadastrada no GitLab como um provedor OAuth.
2. Encaminhamos o nosso usuário para um endpoint do GitLab que irá perguntar se ele permite que nossa aplicação acesse os dados dele do

GitLab.

3. O GitLab redireciona então o usuário para a página inicial do GitLab Classroom.

Durante esse fluxo existe a troca de duas informações entre o GitLab e nosso sistema. Para identificar o usuário que fez login no GitLab, ele retorna uma *hash* (uma sequência alfanumérica) que funciona como um código único identificador. A partir desse código recuperamos o *token* de acesso (uma chave que o GitLab usa para identificar seu usuário e permitir o acesso aos recursos privados) que é usado nas requisições da API do GitLab em todas as demais ações do sistema GitLab Classroom.

3.5.2 Armazenamento e Troca de Dados

Como mencionado anteriormente, foi tomada a decisão arquitetural de utilizar o próprio GitLab como uma base para armazenamento de dados, além das ferramentas MySQL e Redis para salvar dados adicionais. No sistema, esse uso se dá pela relação entre um artefato no GitLab Classroom com um artefato no GitLab: todos os artefatos criados pelo nosso sistema correspondem a um artefato no GitLab, de forma que quando uma nova turma é criada por um Professor também é gerado um grupo correspondente.

Assim sendo, o Git acaba por desempenhar um papel de “banco de dados”, pois consultar os dados que resultam das ações dos usuários em tempo real fornece uma maior acuracidade e é mais econômico do que realizar a cópia destas informações para um banco e utilizá-las a partir dele. Caso esse último fosse o caminho utilizado, precisaríamos ter uma operação de constante consulta e atualização entre um local e outro, além de gerar a necessidade de uma operação de conciliação para comparar os dados e garantir a integridade destes.

Essa relação se estende por todo o sistema. Abaixo se encontra uma tabela que indica a correspondência dos recursos entre os sistemas:

Git Classrooom	**GitLab**
Turma	Grupo
Tarefa Base	Subgrupo
Tarefa aceita	Projeto
Aluno	Usuário
Professor	Usuário
Contribuição	Commit

TABELA 3 – CORRESPONDÊNCIA DE RECURSOS

FONTE: Os autores. (2020).

Para que essa relação seja efetivada, cada ação do nosso sistema faz uma chamada para a API do GitLab. Nas ações referentes a uma tarefa, por exemplo, o usuário realiza a ação no GitLab Classroom para criação do recurso e a API dispara uma chamada para o GitLab criando um novo subgrupo. Ao visualizar uma tarefa, o sistema dispara uma chamada de consulta àquele subgrupo, obtendo assim os dados reais e atuais dela: se ainda está ativa, quais alunos já iniciaram e seus respectivos repositórios, se houveram *commits,* quantos e quais foram.

Abaixo temos uma tabela que descreve as chamadas feitas para a API do GitLab para cada funcionalidade do sistema.

Funcionalidade	Chamada da API
Login	Get Token By Code
	Get Access Token Info
	Get User
Cadastro	Get Token By Code
	Get Access Token Info
	Get User
Visualizar usuários	Get Access Token Info
	Get User
Visualizar usuário	Get Access Token Info
	Get User
Buscar usuário por GRR	Get Access Token Info
	Get User
Atualizar usuário	Get Access Token Info
	Get User
Deletar usuário	Get Access Token Info
	Get User
Visualizar turmas	Get Groups
Criar turma	Create Group
	Add Member To Group

	Get User
Visualizar detalhes de uma turma	Get Group Details
	Get Group Members
Deletar turma	Delete Group
Visualizar tarefas	Get Subgroups
Criar tarefa	Create Group
Aceitar tarefa	Get Access Token Info
	Get User
	Get Group Details
	Get Projects From Group
	Create Project
Visualizar detalhes de uma tarefa	Get Group Details
	Get Read Me (chamada fora da API do GitLab)
Deletar tarefa	Delete Group
	Delete Project
Visualizar a tarefa dos alunos	Get Group Details
	Get Commits
Fechar tarefa	Get Group Details
	Get Commits
Avaliar	-
Ver as avaliações	-

TABELA 4 – CHAMADAS DA API

FONTE: Os autores. (2020).

Ao banco de dados MySQL cabe armazenar as informações de usuário, realizando o cadastro deste na plataforma com as informações adicionais (que já não estão contidas no cadastro do usuário no próprio GitLab e, portanto, não podem ser obtidas através da API), e da avaliação das tarefas: quando a tarefa é finalizada pelo Professor, é registrado o status “Fechada” e assim não são mais contabilizadas as ações dos alunos após esse período; Após o fechamento, o Professor segue para a avaliação de cada tarefa e o banco armazena a nota atribuída pelo docente e, caso seja necessária, a anotação de plágio.

O anteriormente citado Redis, está incumbido de armazenar somente os *tokens* de acesso ao GitLab. Todas as vezes que o usuário realiza o login no sistema, é realizada uma chamada pela API ao GitLab para que seja obtido o *token* de acesso da pessoa (como demonstrado na tabela acima, na funcionalidade de Login) e este é armazenado como uma chave do Redis. Guardado na sessão do nosso sistema, estará o código único retornado pelo GitLab na integração de Login. A partir dele, recuperamos o token: ao realizar qualquer chamada para o GitLab, o sistema recupera esta chave do Redis e a utiliza na requisição.

3.6 ARTEFATOS DESENVOLVIDOS

Nas seções anteriores foram detalhadas as etapas, atividades realizadas, tecnologias utilizadas e a principal forma como se dão as ações do sistema, na relação entre ações em tela e chamadas da API ao GitLab. Neste tópico serão apontados e descritos de forma mais direta os artefatos elaborados durante a construção do GitLab Classroom.

Durante a etapa inicial do projeto foi criado o Diagrama de Casos de Uso, que ilustra quais os usuários, as funcionalidades do sistema e a quem estas

cabem. Com base nele foram então desenvolvidos e explicitados os Requisitos Funcionais do sistema e posteriormente a Especificação de Casos de Uso, esta última acabando por levar ao design das telas.

Para servir como base à codificação, foram construídos o Diagrama de Classes, buscando estabelecer as funções necessárias e atributos de cada entidade do sistema, e o Modelo Lógico de Banco de Dados, que permitiu à equipe então determinar a melhor forma de armazenar as informações do sistema e chegar à definição final de utilização do GitLab também como banco de dados. Ainda na construção do código, visto que a API do GitLab era uma tecnologia nova ao grupo, foi elaborado um documento que facilitasse o entendimento e ilustrasse quais parâmetros eram utilizados, qual o formato das requisições e também as respostas obtidas por elas.

De forma a demonstrar as ações do usuário em tela e as repercussões delas pelo sistema, foram desenvolvidos os Diagramas de Sequência. Estes diagramas indicam quais dados e cliques o professor ou o aluno realizam, em quais telas, qual classe e função são acionadas e o resultado delas para o fluxo de dados e no que é exibido ao usuário.

Todos os artefatos aqui descritos foram desenvolvidos tendo-se em mente esclarecer o funcionamento do sistema, com base nas anotações UML e conteúdos apresentados em sala e encontram-se na seção “Apêndices” deste documento.

4 APRESENTAÇÃO DO SISTEMA

Nesta seção são apresentadas as interfaces e funcionalidades nelas compreendidas, para que sejam melhor compreendidas as atividades que podem ser realizadas na plataforma. Como indicado ao longo deste documento, o software GitLab Classroom foi projetado para ser um sistema no qual os professores tenham maior controle e facilidade na administração de turmas e correção de atividades e os alunos, por sua vez, tenham um contato amigável e instrutivo com o GitLab.

O primeiro contato que o usuário tem com a ferramenta é através da *landing page* (FIGURA 4), que apresenta os objetivos, funcionalidades disponíveis para cada tipo de usuário e características do GitLab Classroom, ainda sem a necessidade de autenticação no sistema. Nesta página ainda são disponibilizados, e encorajados, o login e o cadastro.

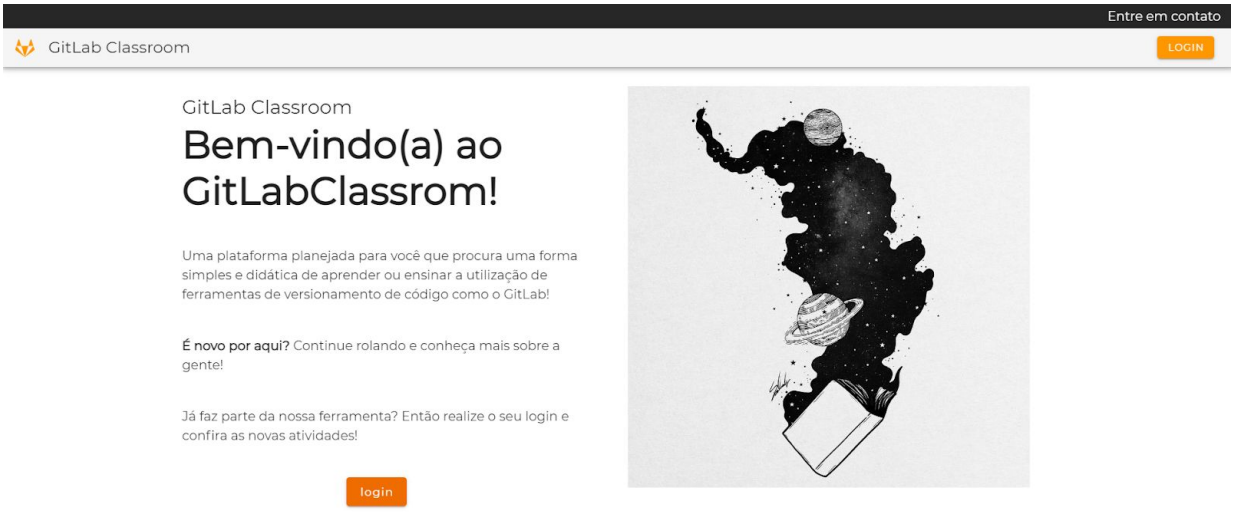


FIGURA 4 - LANDING PAGE

FONTE: Os autores(2020).

Caso o usuário decida então se autenticar, ao clicar em “Login” ele é redirecionado à tela correspondente (FIGURA 5), que valida o cadastro da pessoa no GitLab, solicitando a permissão para o uso na plataforma caso não possua cadastro no GitLab Classroom ou efetuando o login se é um usuário já cadastrado.

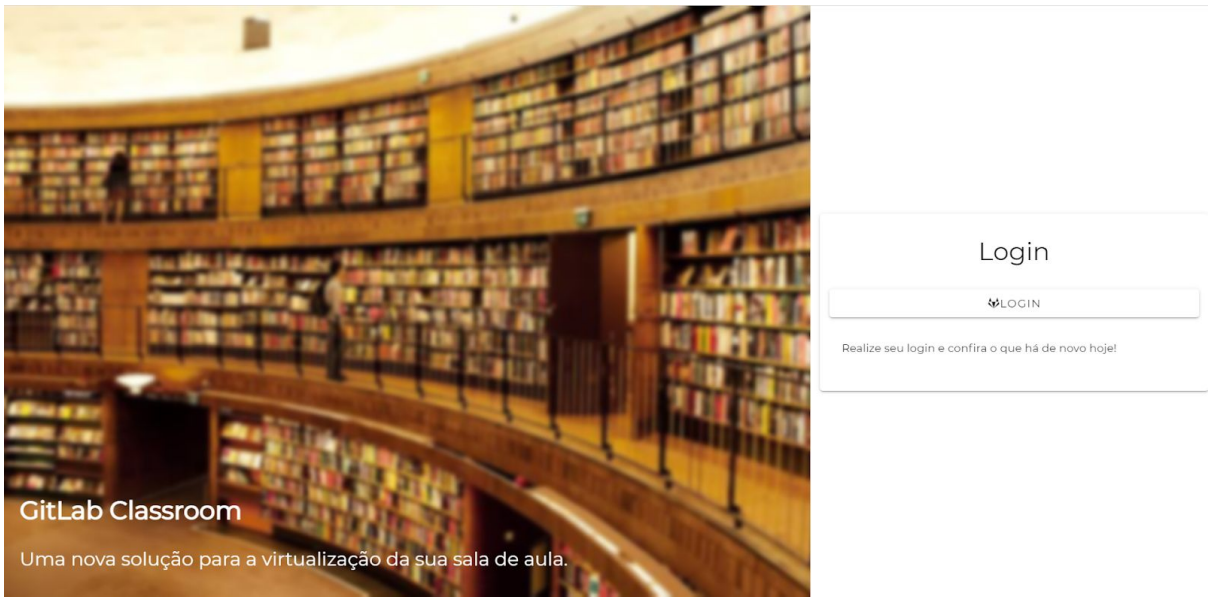


FIGURA 5 - LOGIN

FONTE: Os autores(2020).

Após autenticar-se, o sistema apresenta a *dashboard* (FIGURA 6 e FIGURA 7), disponibilizando as funcionalidades de administração -criação e edição de turmas e tarefas- caso seja um usuário professor e expondo as atividades a serem realizadas e as turmas nas quais está cadastrado para os estudantes. Ambos os usuários podem ainda definir se seu sistema irá funcionar no Modo Escuro (Dark Mode) ou no padrão e podem redefinir este aspecto a qualquer momento. Já logado, somente o próprio usuário pode realizar edições de seus dados.

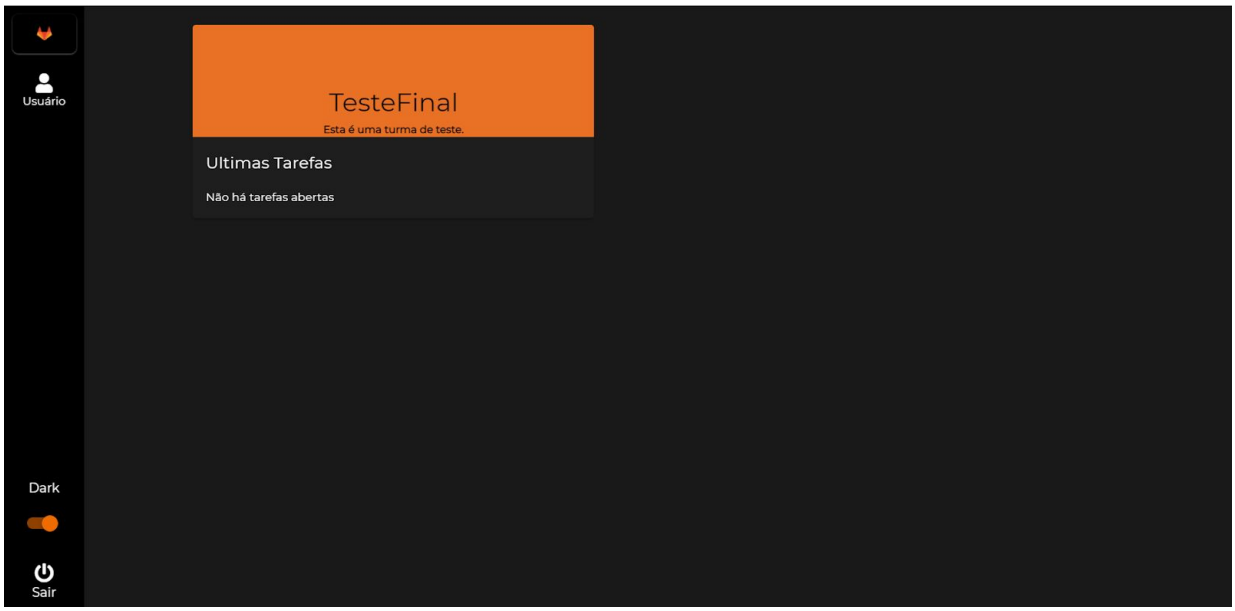


FIGURA 6 - DASHBOARD ALUNO

FONTE: Os autores(2020).

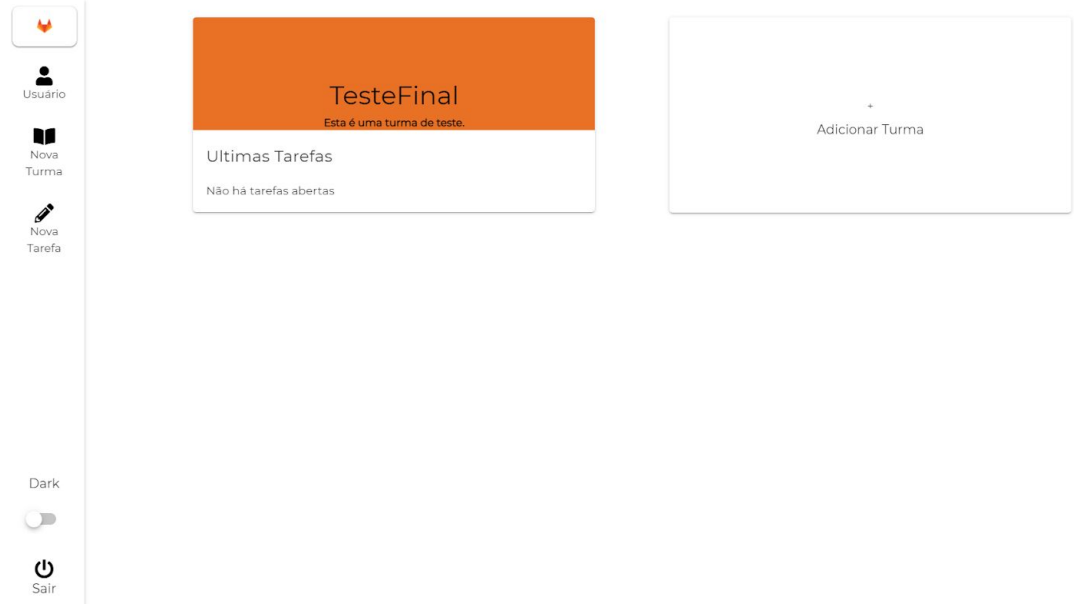


FIGURA 7 - DASHBOARD PROFESSOR

FONTE: Os autores(2020).

Para que se possa seguir no fluxo de atividades, é necessário que o professor realize a criação de uma turma, o que pode ser realizado na tela “Nova Turma” (FIGURA 8), onde serão preenchidos campos referentes ao nome, descrição e privacidade. Correspondentemente no GitLab, é criado um grupo, que servirá como base para as relações de alunos e tarefas.

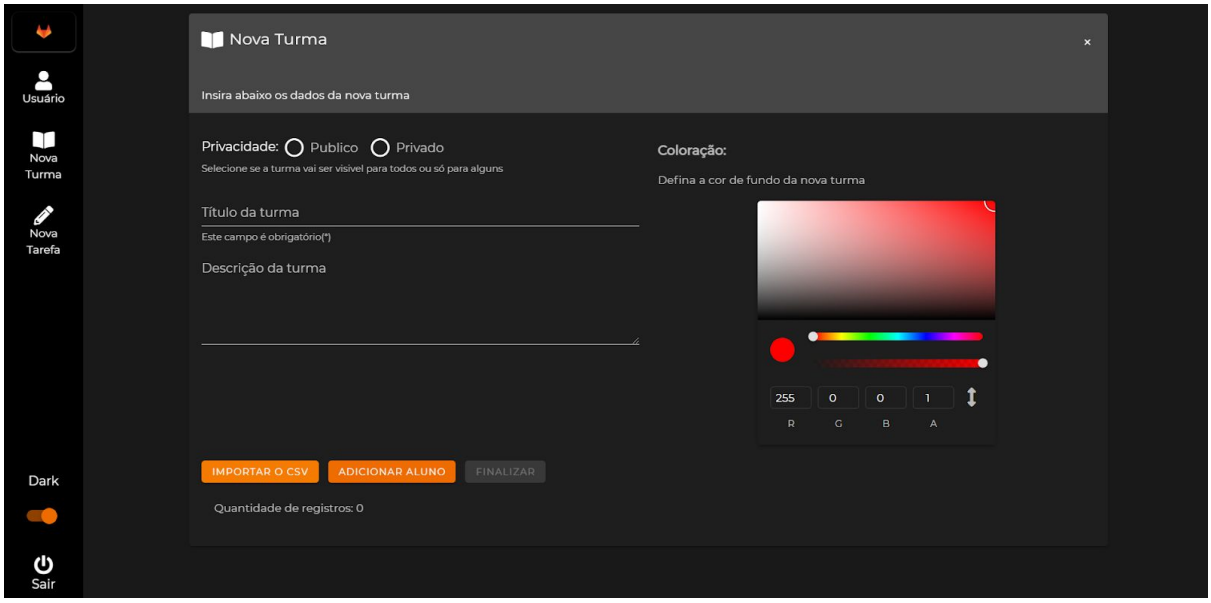


FIGURA 8 - NOVA TURMA

FONTE: Os autores(2020).

Durante a criação acima, podem ser adicionados alunos à turma (FIGURA 9), fornecendo o GRR que será pesquisado na base caso o aluno já esteja cadastrado na plataforma, e caso contrário informando o endereço de e-mail e demais informações deste, que receberá um e-mail notificando-o de sua inclusão na turma. Alunos podem ser adicionados também por meio de um arquivo no formato CSV, que será fornecido pelo professor e carregado clicando em “Importar CSV”(FIGURA 10). Esta ação pode ser realizada também em uma turma já criada ao clicar em “Adicionar Alunos”.

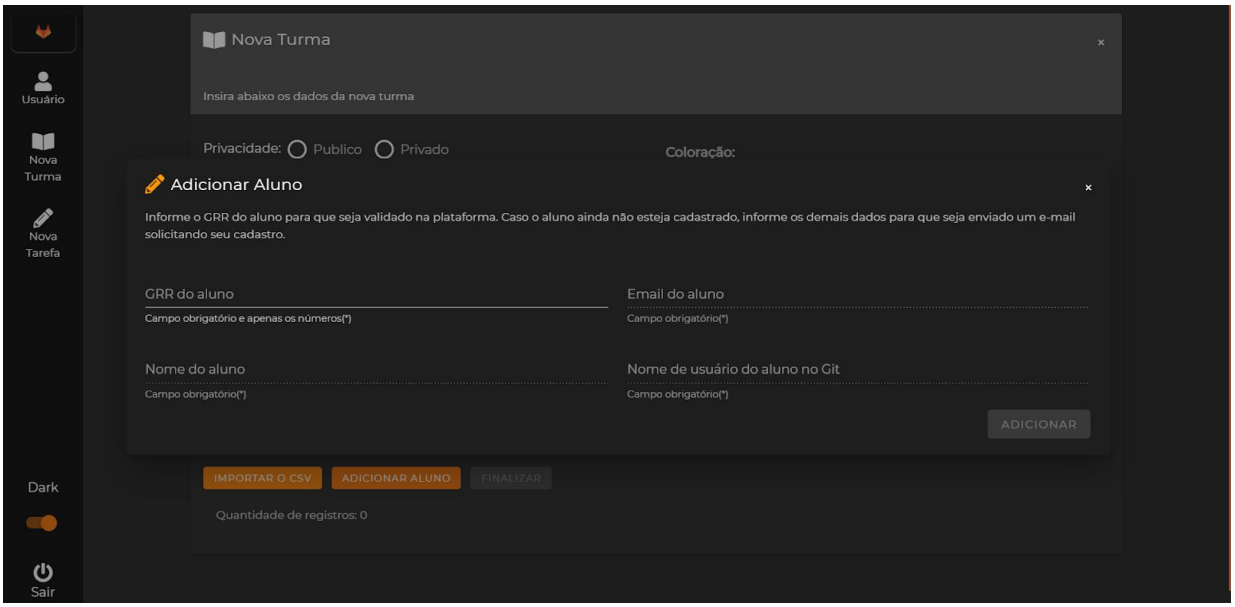


FIGURA 9 - ADICIONAR ALUNO

FONTE: Os autores(2020).

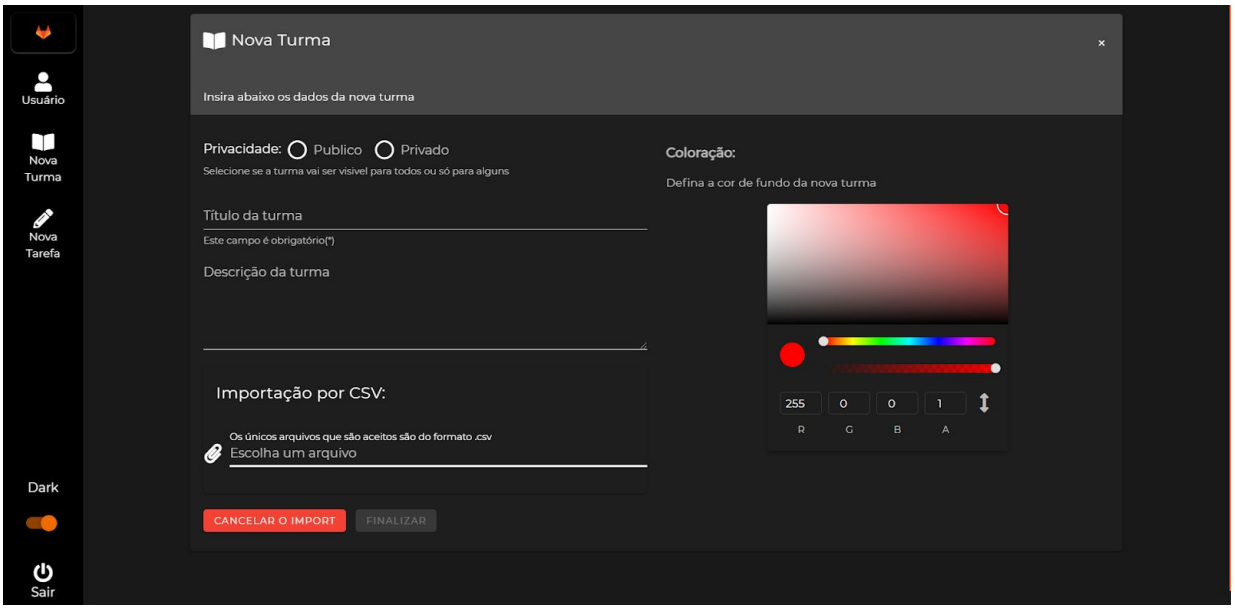


FIGURA 10 - ADICIONAR ALUNO CSV

FONTE: Os autores(2020).

Dentro de uma turma o professor pode então realizar a criação de tarefas, definindo o título, descrição e data limite. O professor pode ainda fornecer o link de um repositório para que este seja utilizado como base para a tarefa, fazendo uso de todos os arquivos e anotações. (FIGURA 11). De forma lógica, o professor também pode excluir uma turma ou tarefa, clicando no botão “Excluir” correspondente e confirmando sua ação.

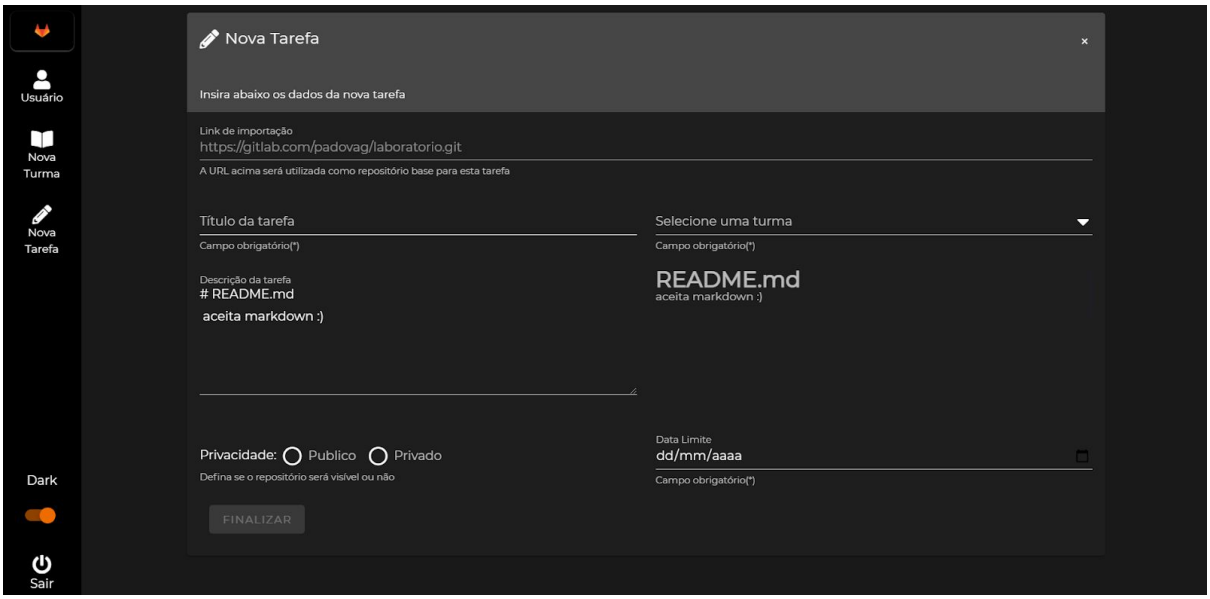


FIGURA 11 - NOVA TAREFA

FONTE: Os autores(2020).

Assim que uma nova tarefa é criada, os alunos cadastrados na turma na qual ela foi aberta podem visualizá-la ao clicar em “Ver Mais” (FIGURA 12). O sistema o direciona então para a atividade e o usuário pode clicar em “Aceitar Tarefa” para que seja gerado um repositório correspondente. O processo consiste na execução de um *fork* do repositório base no qual o aluno irá realizar *commits* da sua resolução. A quantidade de *commits* bem como a quantidade de alunos que realizaram a tarefa são disponibilizadas para o professor em formato de gráfico, tendo como função fornecer novos dados para avaliação do desempenho da turma. É exibida ao aluno uma

relação de instruções sobre como prosseguir (FIGURA 13), que podem ser consultadas ao clicar no botão “Rever Instruções”.

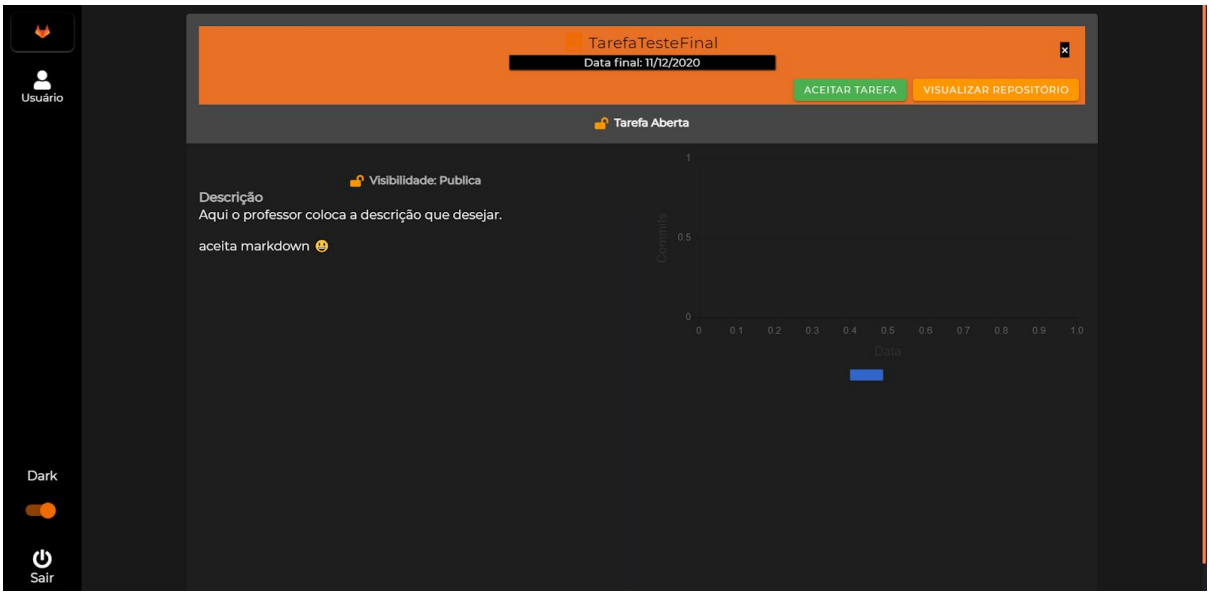


FIGURA 12 - TAREFA ALUNO

FONTE: Os autores(2020).

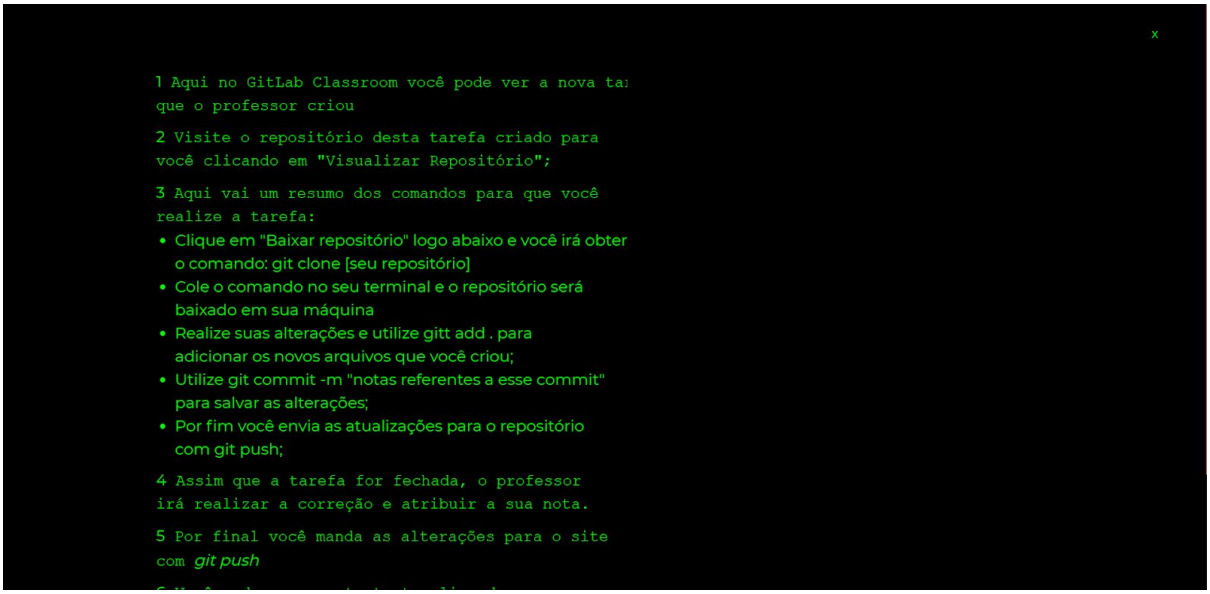


FIGURA 13 - INSTRUÇÕES

FONTE: Os autores(2020).

Quando o prazo final de entrega é atingido, o professor realiza o fechamento da tarefa e pode passar a realizar as correções (FIGURA 14). Este processo ocorre por meio da avaliação dos repositórios de cada aluno, e posteriormente da atribuição de uma nota à versão final apresentada pelo estudante. Caso seja identificado plágio, o professor pode iniciar a correção e clicar no botão “Plágio” e a atividade do aluno receberá esta marcação e a nota será automaticamente zero. O professor pode também inspecionar um commit específico em sua correção, clicando sobre o número de *commits* da tarefa e posteriormente selecionando um dos registros da lista de *commits* que será apresentada (FIGURA 15).

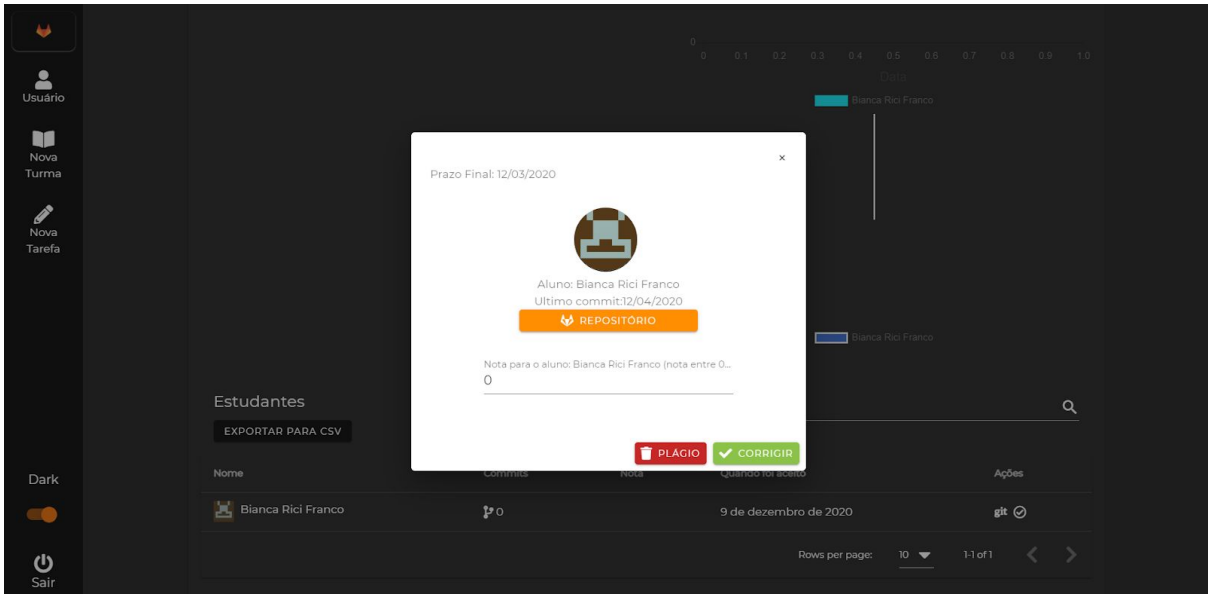


FIGURA 14 - CORREÇÃO TAREFA

FONTE: Os autores(2020).

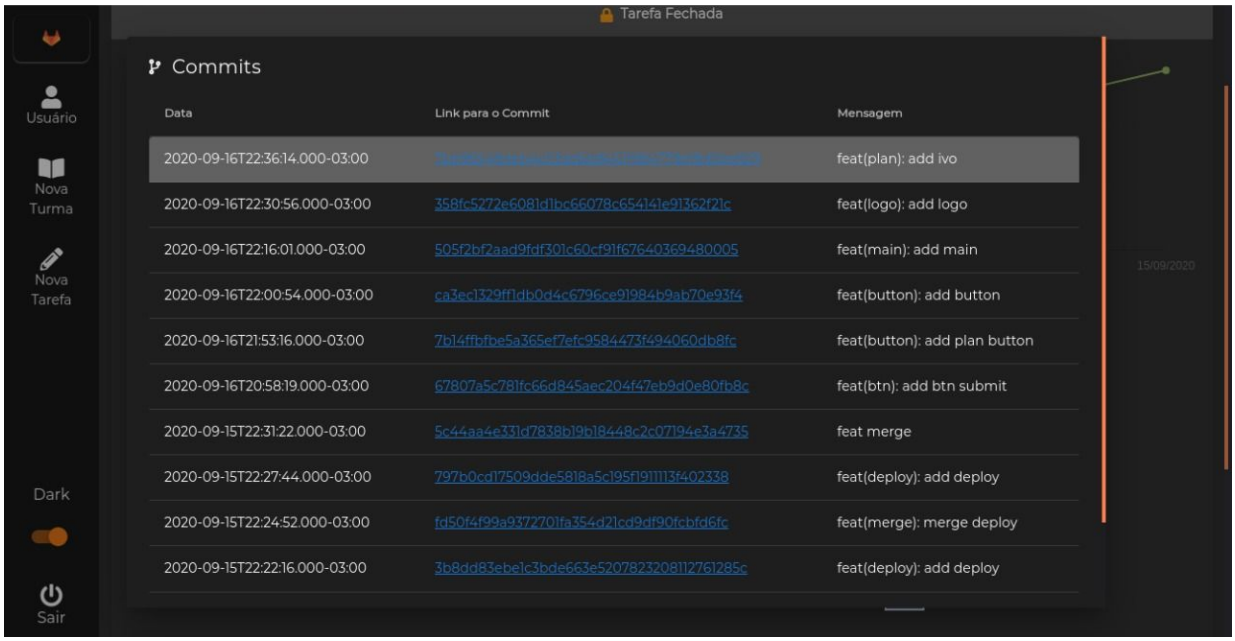


FIGURA 15 - LISTA DE COMMITS

FONTE: Os autores(2020).

As ações por todo o fluxo estão correlacionadas com entidades e ações dentro GitLab, de forma que acabam por ser versões espelhadas, possibilitando assim o atingimento do objetivo deste projeto, que é ser uma interface de fácil entendimento e utilização de ferramentas de versionamento de código, bem como uma plataforma que facilite para os professores a administração de turmas e a avaliação de atividades e desempenho de seus alunos.

5 CONSIDERAÇÕES FINAIS

O GitLab Classroom foi desenvolvido com o objetivo de proporcionar uma plataforma gratuita, amigável e completa que atendesse à necessidade dos docentes: ter um ambiente em que fosse possível administrar suas turmas de disciplinas que envolvessem codificação, que facilitasse a revisão e correção de tarefas e funcionasse como um sistema intermediário que viria a inspirar os alunos a expandir suas habilidades no que tange a ferramentas de versionamento de código como o Git.

Desde a concepção deste projeto até a finalização da documentação foram aplicados conceitos adquiridos pela equipe nas demais disciplinas do curso de Tecnologia em Análise e Desenvolvimento de Sistemas. Durante a elaboração das interfaces levou-se em consideração a experiência do usuário e a ergonomia que deveria estar presente em cada tela; No desenvolvimento de diagramas e da documentação como um todo foram aplicados os conceitos e regras estabelecidos para cada tipo de artefato; E a codificação foi executada de forma a construir um código conciso, de fácil entendimento e que constituísse um sistema estável e que apresentasse o projeto idealizado pela equipe.

5.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Durante as etapas finais do trabalho surgiram sugestões de melhoria e foram notadas funcionalidades adicionais que poderiam vir a agregar valor ao projeto e a facilitar o uso da ferramenta. Como a expansão das ações que podem ser realizadas por e-mail, como o envio de links para as tarefas assim que as mesmas forem disponibilizadas pelo Professor e a obtenção de relatórios de desempenho da turma ao fechamento de cada mês. A implementação de um sistema de notificações dentro da interface do GitLab Classroom também foi uma melhoria percebida durante a finalização do projeto.

Uma outra funcionalidade que pode vir a ser ampliada é a edição de dados de usuário e a tela de usuário como um todo. A criação de uma tela específica para a

exibição de dados do usuário ofereceria a ele a possibilidade de edição de outros campos fora seu nome e uma relação mais refinada de suas ações na plataforma. Ao Professor, seriam apresentados gráficos e dados tratados acerca do desempenho de cada aluno, proporcionando assim uma nova visão sobre sua turma e cada um dos integrantes.

Após algum tempo de discussão com enfoque neste tópico, percebeu-se que, assim como para a maioria dos projetos, as possibilidades de melhoria são quase infinitas e ainda que a equipe esteja feliz com o resultado final obtido, com o uso de diferentes recursos, cronogramas e ideias é possível expandir as funcionalidades do sistema e o GitLab Classroom pode vir a oferecer muito mais a seus usuários.

REFERÊNCIAS

API Docs. Disponível em: https://docs.gitlab.com/ee/api/README.html. Acesso em: 9 dez.2020.

GITHUB Classroom.Disponível em: https://classroom.github.com/. Acesso em: 9 dez. 2020.

GLASSEY, Richard. Adopting Git/Github within Teaching: A Survey of Tool Support. **CompEd '19: Proceedings of the ACM Conference on Global Computing Education**, Chengdu, p. 143–149, maio 2019. Disponível em: https://doi.org/10.1145/3300115.3309518. Acesso em: 9 dez. 2020.

GOMES, José Ferreira (2014). **A tecnologia na sala de aula**. Novas tecnologias e educação. Porto: Biblioteca Digital da Faculdade de Letras da Universidade do Porto. Pp. 17-44.

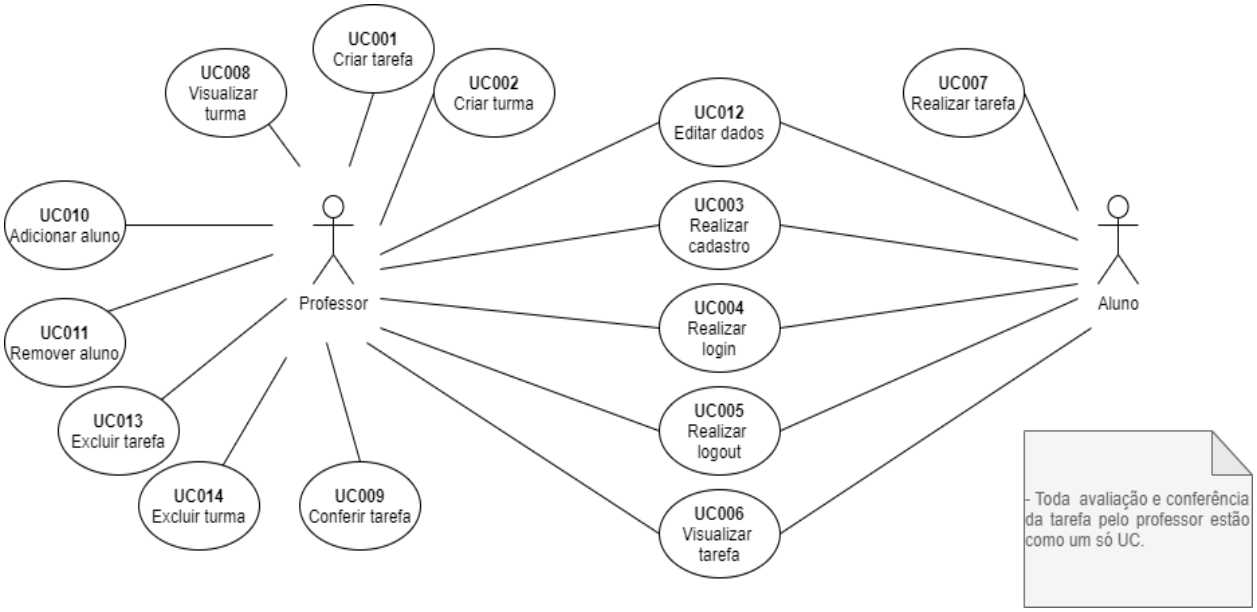
GOUVEIA, A. Os serviços de versionamento de código. **Universidade Positivo.** Curitiba, 2016. Disponível em: https://www.up.edu.br/blogs/engenharia-da-computacao/2016/03/18/os-servicos-de-versionamento-de-codigo/. Acesso em: 9 dez. 2020.

HAARANEN, L; LEHTINEN, T.Teaching Git on the Side: Version Control System as a Course Platform.**ITiCSE '15: Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education,** Vilnius, p. 87-92, jul. 2015.. Disponível em: https://doi.org/10.1145/2729094.2742608. Acesso em: 9 dez. 2020.

VUE-CHARTJS. Disponível em: https://vue-chartjs.org/. Acesso em: 9 dez.2020.

VUETIFY. Disponível em: https://vuetifyjs.com/en/. Acesso em: 9 dez.2020.

APÊNDICE 1 – DIAGRAMA DE CASOS DE USO



APÊNDICE 2 – REQUISITOS FUNCIONAIS

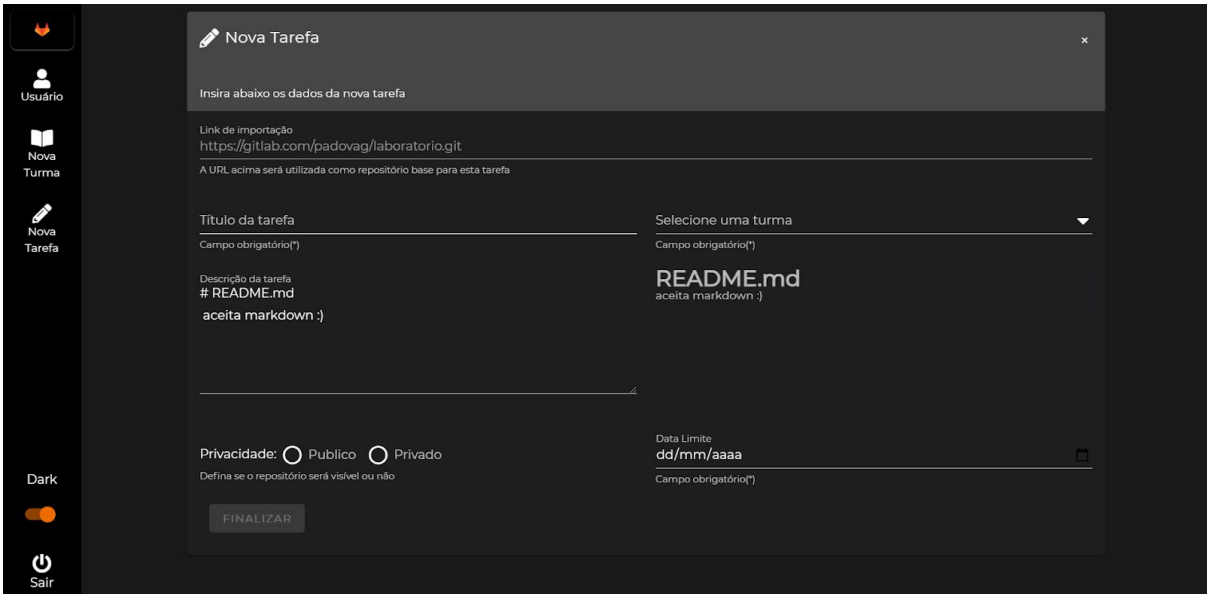
Requisito: Criar Tarefa	**UC001**
O Professor pode criar uma tarefa clicando em “Nova Tarefa”, informando o nome da tarefa, data limite de entrega e um link de base para o repositório.	
Requisito: Criar Turma	**UC002**
O Professor pode criar uma turma clicando em “Nova Turma”, informando o nome, descrição, privacidade, coloração e os usuários dos alunos, representados por seus GRR’s.	
Requisito: Realizar Cadastro	**UC003**
O Professor/Aluno pode realizar o cadastro na plataforma informando seu GRR e concedendo o acesso da plataforma ao seu GitLab.	
Requisito: Realizar Login	**UC004**
O Professor/Aluno pode realizar o login na plataforma, estando logado no GitLab e clicando em “Login”.	
Requisito: Realizar Logout	**UC005**
O Professor/Aluno pode realizar o logout na plataforma clicando em “Sair” no menu inferior.	
Requisito: Visualiza Tarefa	**UC006**
O Professor/Aluno pode visualizar uma tarefa atribuída a sua turma, o Professor podendo também ter acesso à quantidade e aos usuários dos alunos que realizaram a tarefa.	

Requisito: Realizar Tarefa	**UC007**
O Aluno pode realizar a tarefa clicando em “Aceitar Tarefa, assim é criado um repositório para que o aluno poste sua resposta à atividade proposta.	
Requisito: Visualizar Turma	**UC008**
O Professor pode visualizar as turmas que administra, os alunos por elas compostas, as tarefas a elas atribuídas e um gráfico de desempenho de cada turma.	
Requisito: Conferir Tarefa	**UC009**
O Professor pode conferir quais alunos realizaram a tarefa e pode corrigir esta tarefa, clicando no repositório referente ao aluno desejado, observando o código, classificando ou não como plágio e avaliando com uma nota de 0 a 10.	
Requisito: Adicionar Aluno	**UC010**
O Professor pode adicionar novos alunos a uma turma já criada, clicando em “Adicionar Alunos” ao lado da lista de alunos na tela da turma desejada.	
Requisito: Remover Aluno	**UC011**
O Professor pode remover um aluno de sua turma clicando no símbolo de remoção ao lado do registro do aluno na tela da turma desejada.	
Requisito: Editar Dados	**UC012**
O Professor/Aluno pode editar seus dados em sua tela de usuário, editando os campos em sua tela de usuário.	
Requisito: Excluir Tarefa	**UC013**

O Professor pode excluir uma tarefa clicando em “Excluir” na tarefa desejada.	
Requisito: Excluir Turma	**UC014**
O Professor pode excluir uma turma clicando em “Remover” e depois confirmando com “Apagar Turma”.	

APÊNDICE 3 – ESPECIFICAÇÃO DE CASO DE USO

UC001 - CRIAR TAREFA
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma como “Professor”.
REGRAS DE NEGÓCIO: Somente um Professor pode criar uma tarefa.
INTERFACE:



FLUXO PRINCIPAL:

- Professor clica em “Nova Tarefa” na barra lateral(A1);
- Sistema abre a tela “Nova Tarefa”;
- Professor preenche os campos referentes ao Nome, Turma, Descrição, Privacidade, Data Limite(E1)(E2) e Link Base(A2);
- Professor clica em "Finalizar";
- Sistema realiza a chamada de API para criar o artefato no GitLab;
- Sistema armazena o registro na tabela assignments;
- Sistema disponibiliza a tarefa para que os alunos a realizem.

FLUXOS ALTERNATIVOS:

A1 - Professor cria tarefa a partir da própria turma:

- Professor clica em “Nova Tarefa” dentro de uma turma;
- Sistema pré-seleciona a turma atual no campo “Turma”;
- Retorna-se ao Fluxo Principal.

A2 - Não é informada URL no campo “Link Base”:

- Professor não preenche o campo “Link Base”;
- Retorna-se ao Fluxo Principal.

FLUXO DE EXCEÇÃO:

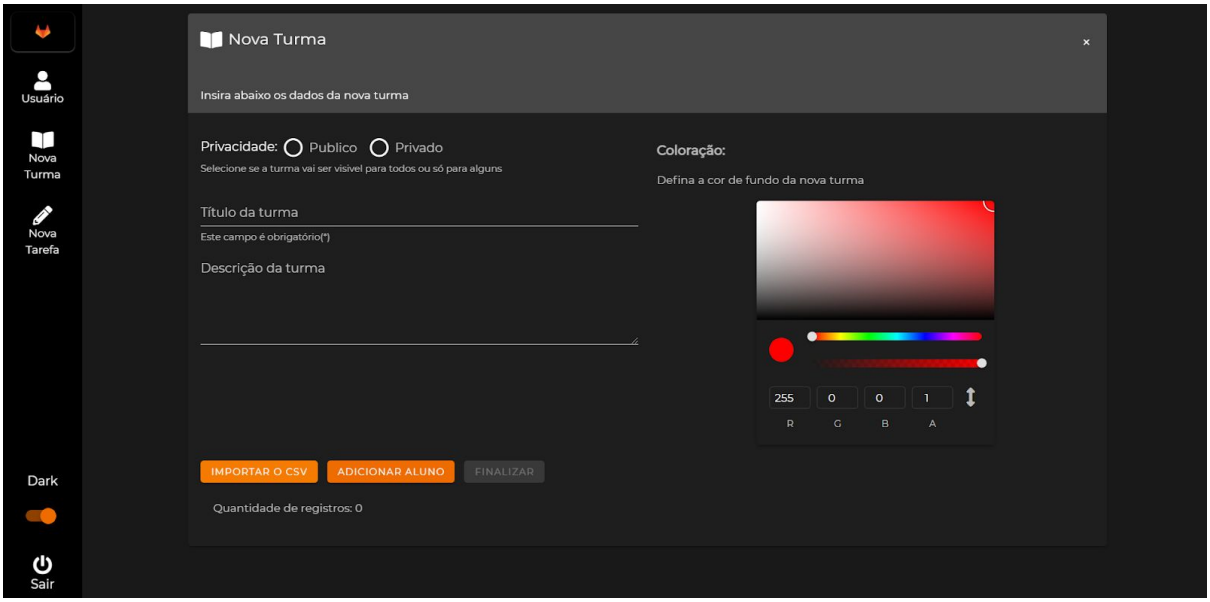
E1 - Campos obrigatórios são deixados em branco:

- O campo “Data Limite” é deixado em branco;
- Professor prossegue e clica em “Criar”;
- Mensagem “A Data Limite deve ser preenchida” aparece na tela;
- Professor fecha a mensagem, preenche o campo e retorna-se ao Fluxo Principal.

E2 - O texto informado no Link Base é inválido:

- O texto informado no Link Base não corresponde a um repositório git;
- Mensagem “O formato de importação é inválido” aparece na tela;
- Professor fecha a mensagem, preenche o campo com um link válido e retorna-se ao Fluxo Principal.

UC002 - CRIAR TURMA
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma como “Professor”.
REGRAS DE NEGÓCIO: Somente um Professor pode criar uma turma.
INTERFACE:



FLUXO PRINCIPAL:
- Professor clica em “Nova Turma” na seção de turmas de sua tela principal;
- Sistema abre a tela “Nova Turma”;
- Professor preenche os campos referentes ao Nome, Descrição, Privacidade e seleciona a Coloração(E1);
- Professor clica em “Adicionar Aluno”(A1)(A2);
- Sistema abre o pop up “Novo Aluno”;
- Professor insere o GRR do aluno no campo “GRR do aluno”;
- Sistema verifica que aluno possui cadastro na plataforma e retorna as demais informações nos campos “Nome do aluno”, “E-mail do aluno”, “Usuário no Git” (E2);
- Professor clica em “Adicionar”;
- Professor clica em “Finalizar”;
- Sistema armazena o registro na tabela assignment;
- O sistema cria a turma e os alunos são notificados de sua inclusão por e-mail.

FLUXOS ALTERNATIVOS:

A1 - Professor adiciona alunos pelo arquivo .CSV
- Professor clica em “Importar CSV”;
- Professor seleciona o arquivo no formato CSV que contém a relação de alunos em sua máquina;

- Sistema exibe as informações do arquivo;
- Professor clica em “Finalizar”;
- Retorna-se ao fluxo principal.

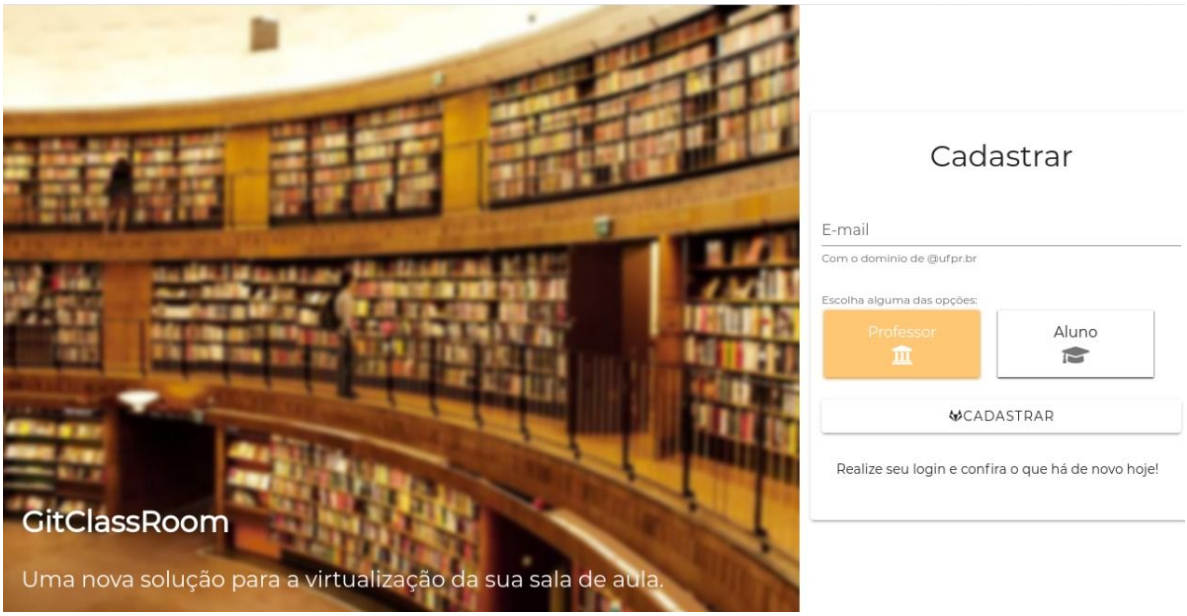
A2 - Professor cria uma turma sem aluno:
- Professor não adiciona alunos à turma;
- Professor clica em “Finalizar”;
- Sistema cria a turma na plataforma.

FLUXO DE EXCEÇÃO:

E1 - Campos são deixados em branco:
- O campo “Nome” é deixado em branco;
- O botão “Finalizar” não se torna clicável;
- Professor preenche os campos;
- Retorna-se ao fluxo principal.

E2 - Usuário não possui cadastro na plataforma:
- Sistema informa que o aluno não possui cadastro;
- Professor preenche o restante das informações;
- Retorna-se ao fluxo principal.

UC003 - REALIZAR CADASTRO
ATOR: Professor/Aluno.
PRÉ-CONDIÇÕES: O usuário deve possuir uma conta ativa na plataforma GitLab.
REGRAS DE NEGÓCIO: Uma conta do GitLab pode estar associada a somente uma conta no GitLab Classroom.
INTERFACE:



FLUXO PRINCIPAL:
- Usuário acessa a plataforma e clica em “Login”(A1);
- Sistema verifica se o usuário possui uma sessão ativa no GitLab(E1);
- O sistema realiza a validação e constata que o usuário não possui cadastro na plataforma;
- Sistema redireciona o usuário para a tela de autorização no GitLab;
- Usuário clica em “Authorize”(A1);
- Sistema retorna à tela de Login;
- Usuário é um aluno e seleciona Aluno;
- Usuário preenche o campo “GRR”(A2);
- Usuário clica em “Cadastrar”;
- Sistema realiza a inclusão do registro na tabela users;
- Sistema abre a tela “Dashboard”.

FLUXOS ALTERNATIVOS:
A1 - Usuário não autoriza o uso de seus dados do GitLab na plataforma:
- Usuário clica em “Deny”;
- Sistema retorna à tela de login.

A2 - Usuário é um professor:

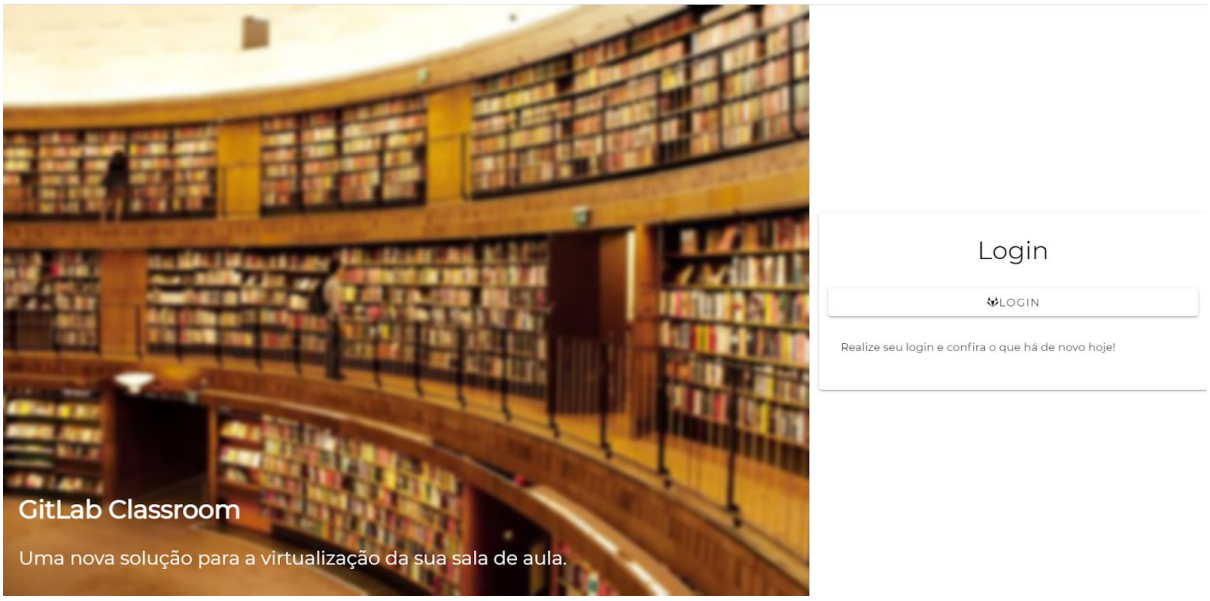
- Usuário é um Professor e seleciona “Professor”;
- Usuário preenche o campo “E-mail”;
- Retorna-se ao fluxo principal.

FLUXO DE EXCEÇÃO:

E1 - Usuário não possui sessão ativa no GitLab:

- Sistema redireciona o usuário para a tela de login do GitLab;
- Usuário insere os dados e realiza login no GitLab;
- Retorna-se ao fluxo principal.

UC004 - REALIZAR LOGIN
ATOR: Professor/Aluno.
PRÉ-CONDIÇÕES: O usuário deve possuir uma conta ativa na plataforma GitLab Classroom.
REGRAS DE NEGÓCIO: Não há.
INTERFACE:



FLUXO PRINCIPAL:
- Usuário acessa a tela de Login e clica em “Login”;
- Sistema verifica se o usuário possui uma sessão ativa no GitLab(E1);
- Sistema verifica que o usuário da sessão ativa possui cadastro na plataforma(E2);
- Sistema obtém o token de acesso do usuário;
- Sistema armazena o token no Redis;
- Sistema abre a “Home” .

FLUXOS ALTERNATIVOS:
Não há.

FLUXO DE EXCEÇÃO:
E1 - Usuário não possui sessão ativa no GitLab:
- Sistema redireciona o usuário para a tela de login do GitLab;
- Usuário insere os dados e realiza login no GitLab;
- Retorna-se ao fluxo principal.

E2 - Usuário não possui cadastro na plataforma GitLab ClassRoom:
- Segue-se o fluxo UC003 - REALIZAR CADASTRO;
- Retorna-se ao fluxo principal.

UC005 - REALIZAR LOGOUT
ATOR: Professor/Aluno.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom.
REGRAS DE NEGÓCIO: Não há.
INTERFACE:

FLUXO PRINCIPAL:
- Usuário clica em “Sair”;
- Sistema realiza o logout;
- Sistema apresenta a tela de Login.

FLUXOS ALTERNATIVOS:
Não há.

FLUXO DE EXCEÇÃO:
Não há.

UC006 - VISUALIZAR TAREFA

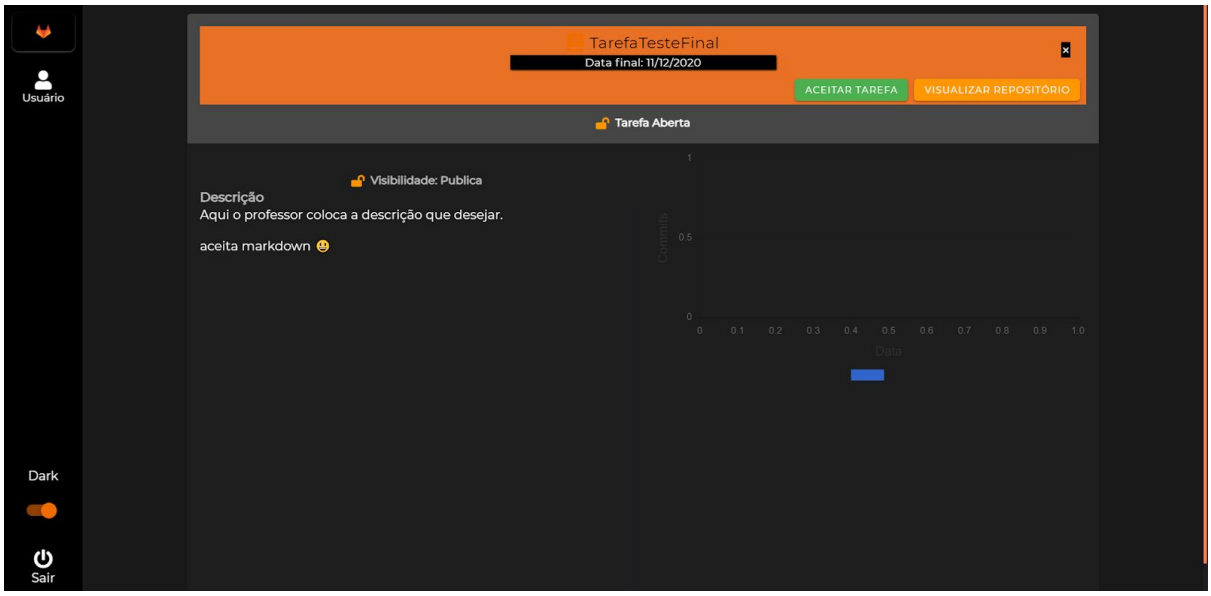
ATOR: Professor/Aluno.

PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom e relacionado à turma na qual a tarefa foi criada.

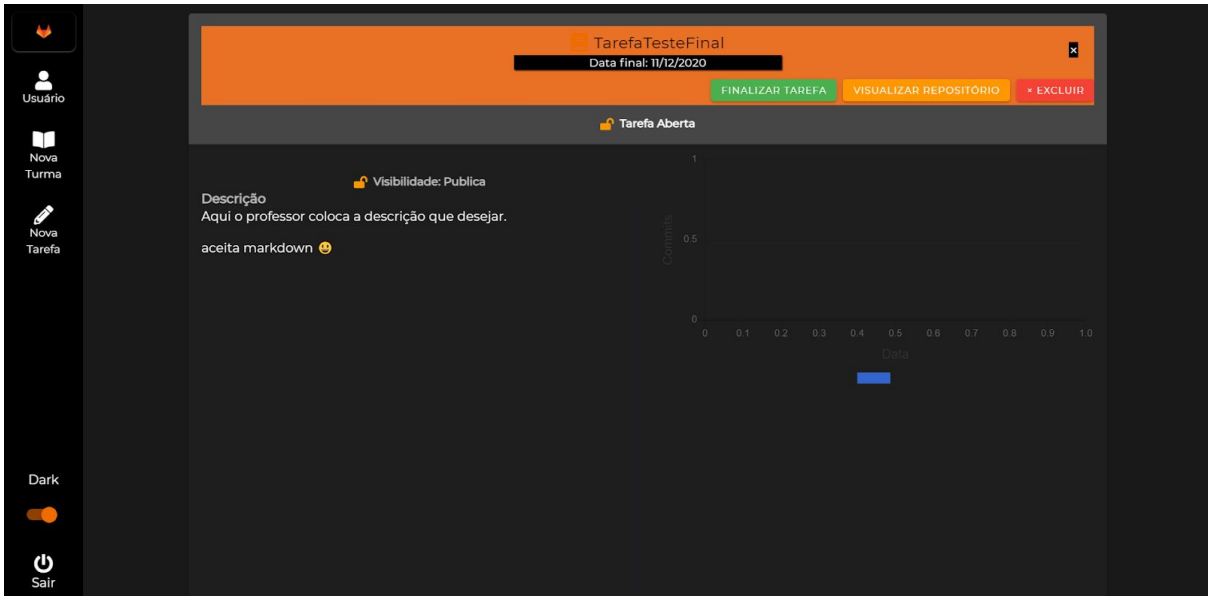
REGRAS DE NEGÓCIO: Cada usuário só pode ver tarefas que pertençam a turmas as quais esteja relacionado.

INTERFACE:

ALUNO:



PROFESSOR:



FLUXO PRINCIPAL:

- Usuário clica em uma turma de sua tela;
- Sistema redireciona para uma tela com as tarefas daquela turma em ordem de publicação na plataforma(A1);
- Usuário clica em uma tarefa;
- Sistema aciona a API para retornar os dados da turma no GitLab;
- Sistema exibe uma tela com as informações sobre a respectiva tarefa.

FLUXOS ALTERNATIVOS:

A1 - Usuário é do tipo Professor:

- Sistema exibe uma tela com as informações da tarefa: o número de alunos que a realizou, a relação de repositórios dos alunos e notas da turma.

FLUXO DE EXCEÇÃO:

Não há.

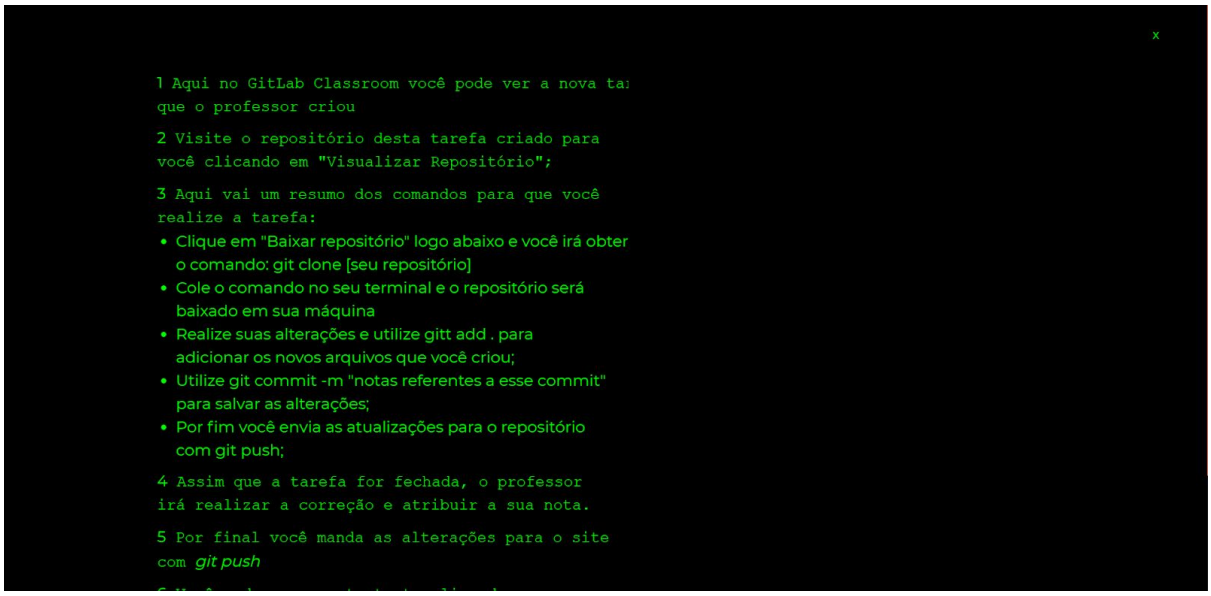
UC007 - REALIZAR TAREFA

ATOR: Aluno.

PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como “Aluno” e estar relacionado à turma na qual a tarefa foi criada.

REGRAS DE NEGÓCIO: Cada Aluno só pode realizar tarefas que pertençam a turmas as quais esteja relacionado; Cada Aluno só pode realizar uma mesma tarefa uma vez.

INTERFACE:



FLUXO PRINCIPAL:

- Aluno clica em “Ver Mais” na tarefa que deseja realizar na turma;
- Sistema abre a tela com a tarefa correspondente;
- Aluno clica em “Aceitar Tarefa” (A1);
- Sistema cria um repositório no GitLab para aquela tarefa e vincula este à plataforma;
- Sistema exibe a tela com instruções;
- Aluno clica em “Baixar repositório”;
- Sistema adiciona a sua área de transferência o comando para baixar a tarefa em sua máquina;
- Aluno realiza sua tarefa em seu computador e a publica dentro do prazo estipulado(A2);

FLUXOS ALTERNATIVOS:

A1 - Aluno confere o repositório:

- O Aluno clica em “Visualizar Repositório”;
- Sistema redireciona o usuário para o repositório de sua tarefa no GitLab.

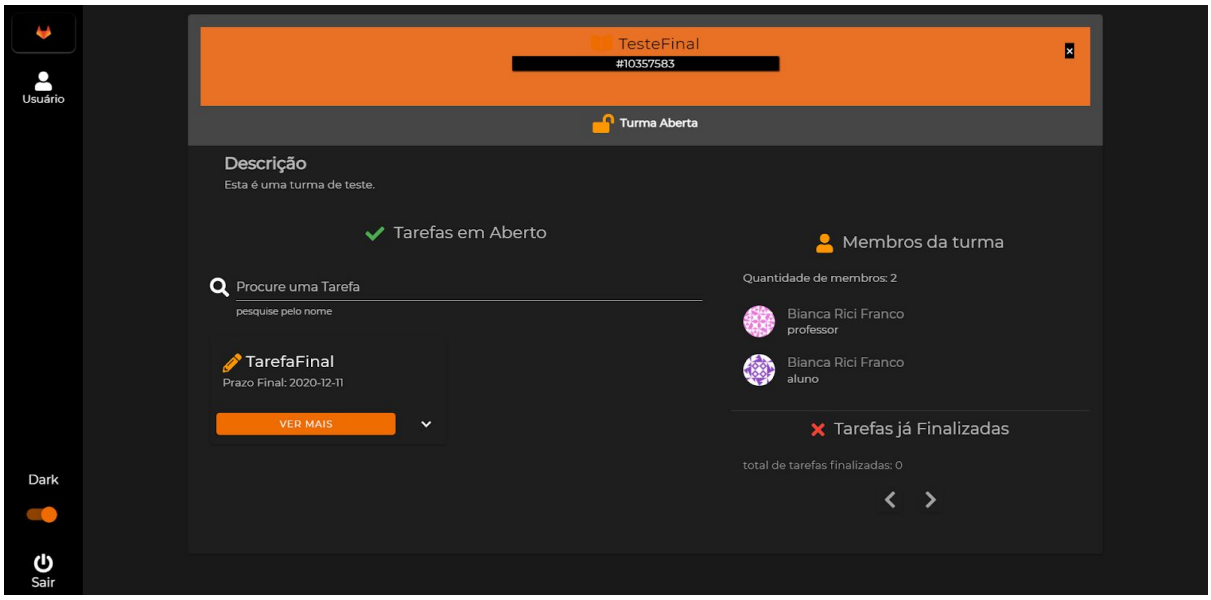
A2 - Aluno não publica sua tarefa:

- O Aluno não realiza sua tarefa;
- O sistema verifica que a data limite foi atingida, atualiza o status e atribui Nota 0(zero) àquela atividade.

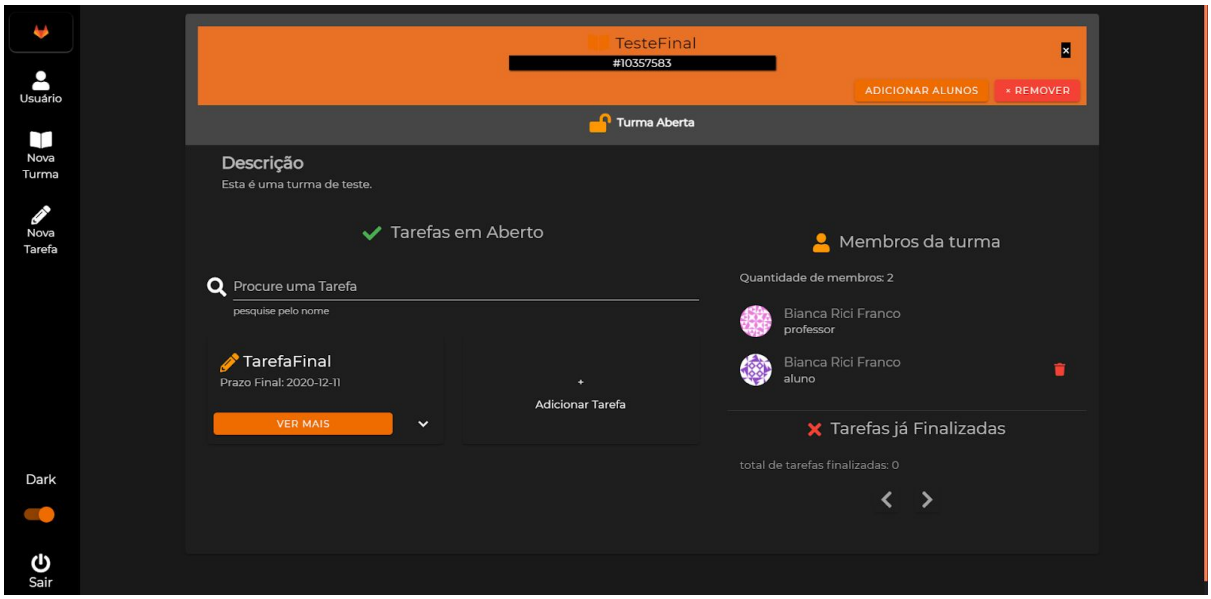
FLUXO DE EXCEÇÃO:

Não há.

UC008 - VISUALIZAR TURMA
ATOR: Professor/Aluno
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom e estar relacionado à turma.
REGRAS DE NEGÓCIO: Não há.
INTERFACE:
ALUNO:



PROFESSOR:



FLUXO PRINCIPAL:

- Usuário clica na turma;
- Sistema aciona a API para retornar os dados da turma no GitLab;
- Sistema abre a tela referente àquela turma;
- É exibida na tela uma lista dos alunos daquela disciplina, tarefas em aberto, tarefas fechadas e um gráfico de desempenho de cada turma.

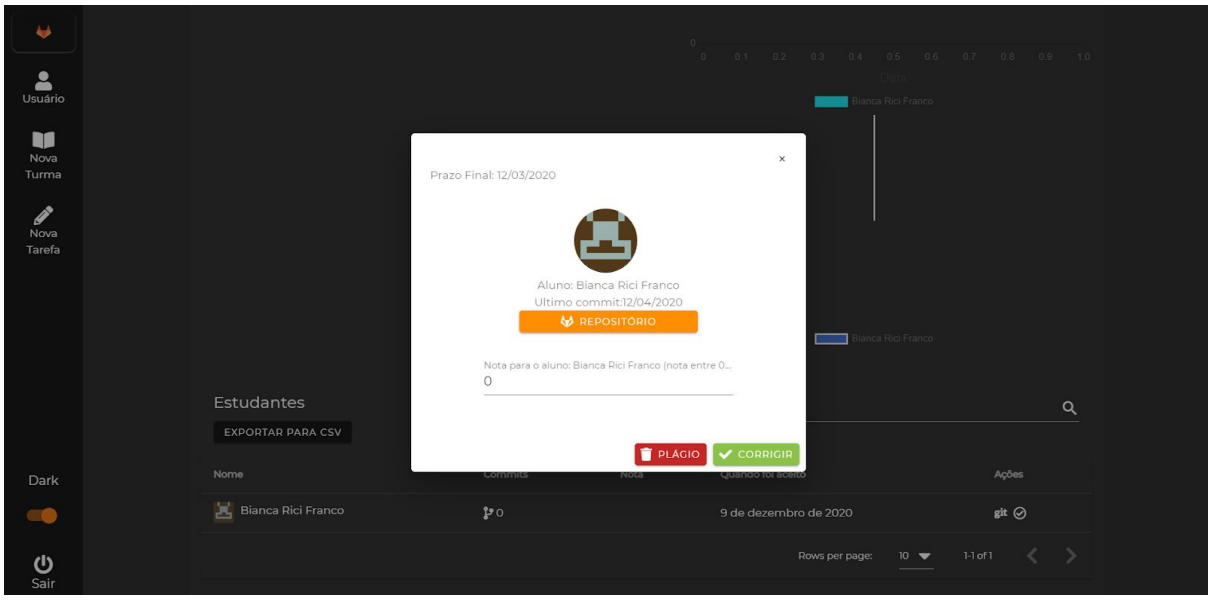
FLUXOS ALTERNATIVOS:

Não há.

FLUXO DE EXCEÇÃO:

Não há.

UC009 - CONFERIR TAREFA
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como "Professor", estar relacionado à turma na qual a tarefa foi criada e esta deve ter tido seu repositório criado pelo aluno.
REGRAS DE NEGÓCIO: Somente um Professor pode conferir/avaliar uma tarefa.
INTERFACE:



FLUXO PRINCIPAL:
- Professor clica em "Ver mais" na tarefa que deseja conferir;
- Sistema abre a tela da tarefa em questão com a relação das respostas dos alunos;
- Professor clica em "Finalizar Tarefa";
- Sistema disponibiliza então a atribuição de notas;
- Professor clica sobre a tarefa de um aluno(A1);
- Sistema abre o PopUp de correção;
- Professor clica em "Repositório" (A1)na tarefa do aluno que deseja conferir;
- Sistema redireciona o usuário para o repositório vinculado àquela tarefa no GitLab.

FLUXOS ALTERNATIVOS:

A1 - Professor deseja corrigir a tarefa:
- Professor preenche o campo "Nota" com a nota do aluno(E1)(A2);
- Professor clica em "Corrigir";
- Sistema atualiza a tabela assignments com a nota;
- Sistema salva as informações e retorna à tela de Tarefas.

A2 - Professor sinaliza plágio:

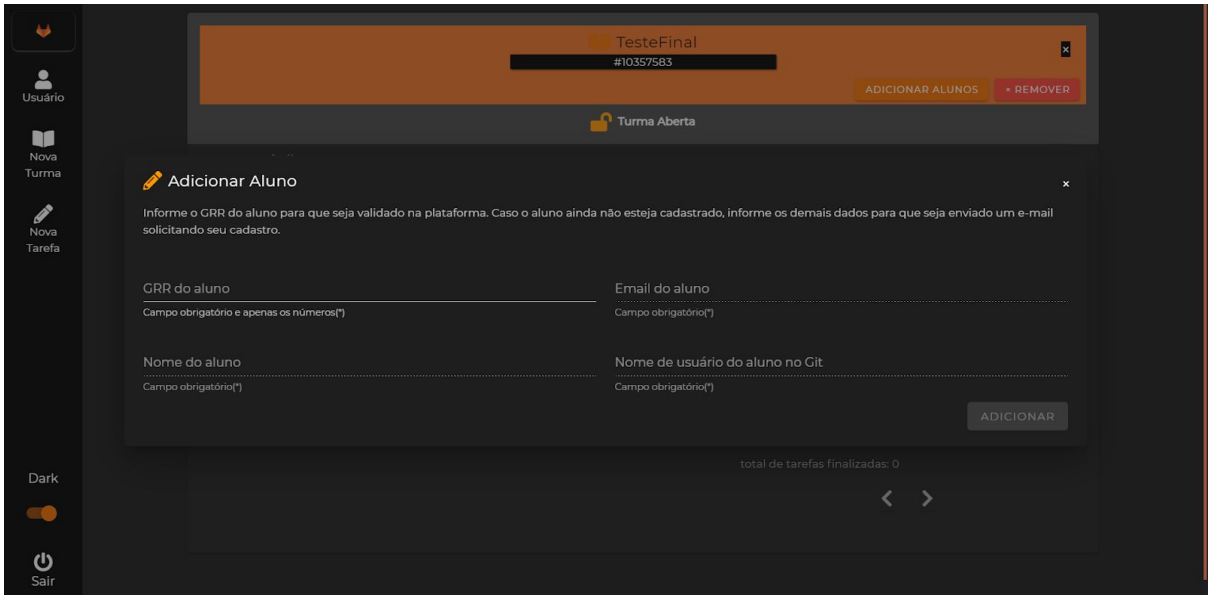
- Professor identifica plágio na tarefa;
- Professor clica em “Plágio”;
- O sistema marca a tarefa como “Plágio” e atribui nota 0(zero);
- Sistema salva as informações e retorna à tela de Tarefas.

FLUXO DE EXCEÇÃO:

E1 - Professor não preenche o campo “Nota”:

- Professor deixa o campo Nota em branco;
- Professor clica em “Corrigir”**;**
- Sistema sinaliza “Campo obrigatório” abaixo do campo “Nota”;
- Segue-se o Fluxo Principal.

UC010 - ADICIONAR ALUNO
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como “Professor” e estar relacionado à turma na qual os alunos estão inseridos.
REGRAS DE NEGÓCIO: Somente um Professor pode adicionar um aluno à sua turma.
INTERFACE:



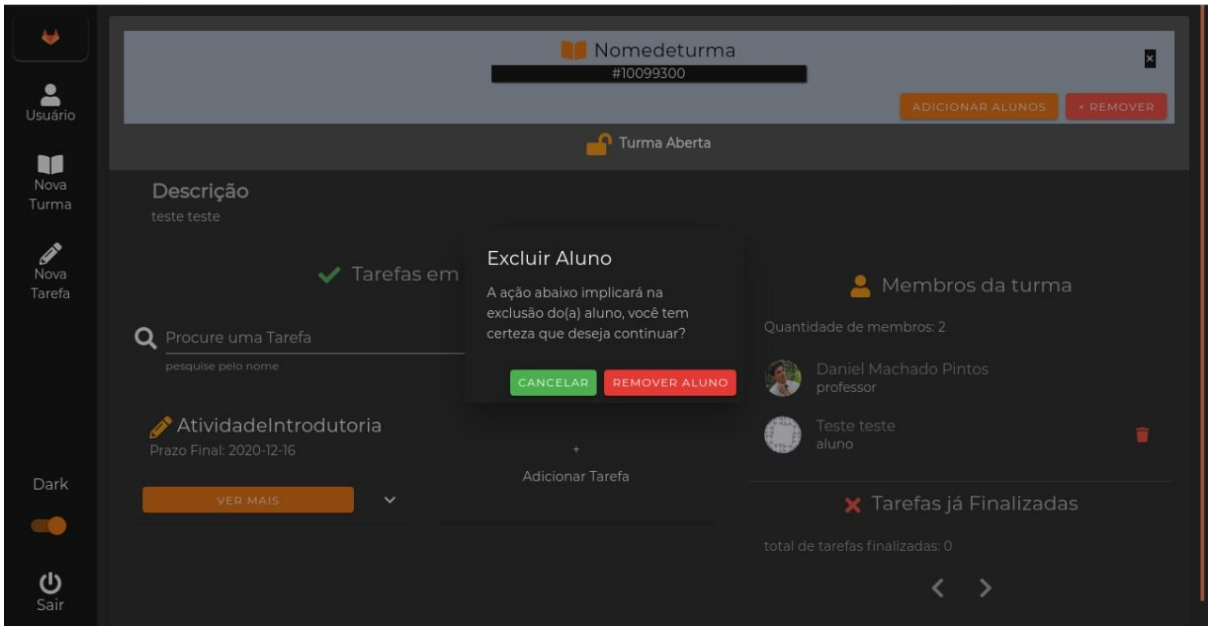
FLUXO PRINCIPAL:
- Professor clica em “Adicionar Alunos” na tela de uma turma;
- Sistema abre o pop up “Novo Aluno”;
- Professor insere o GRR do aluno no campo “GRR do aluno”;
- Sistema consulta a tabela users;
- Sistema verifica que aluno possui cadastro na plataforma e retorna as demais informações nos campos “Nome do aluno”, “E-mail do aluno”, “Usuário no Git” (E1);
- Professor clica em “Adicionar”;
- O sistema adiciona o aluno à turma e o notifica de sua inclusão na turma.

FLUXOS ALTERNATIVOS:
Não há.

FLUXO DE EXCEÇÃO:

E1 - Usuário não possui cadastro na plataforma:
- Sistema informa que o aluno não possui cadastro;
- Professor preenche o restante das informações;
- Retorna-se ao fluxo principal.

UC011 - REMOVER ALUNO
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como “Professor” e estar relacionado à turma na qual os alunos estão inseridos.
REGRAS DE NEGÓCIO: Somente um Professor pode remover um aluno de sua turma.
INTERFACE:

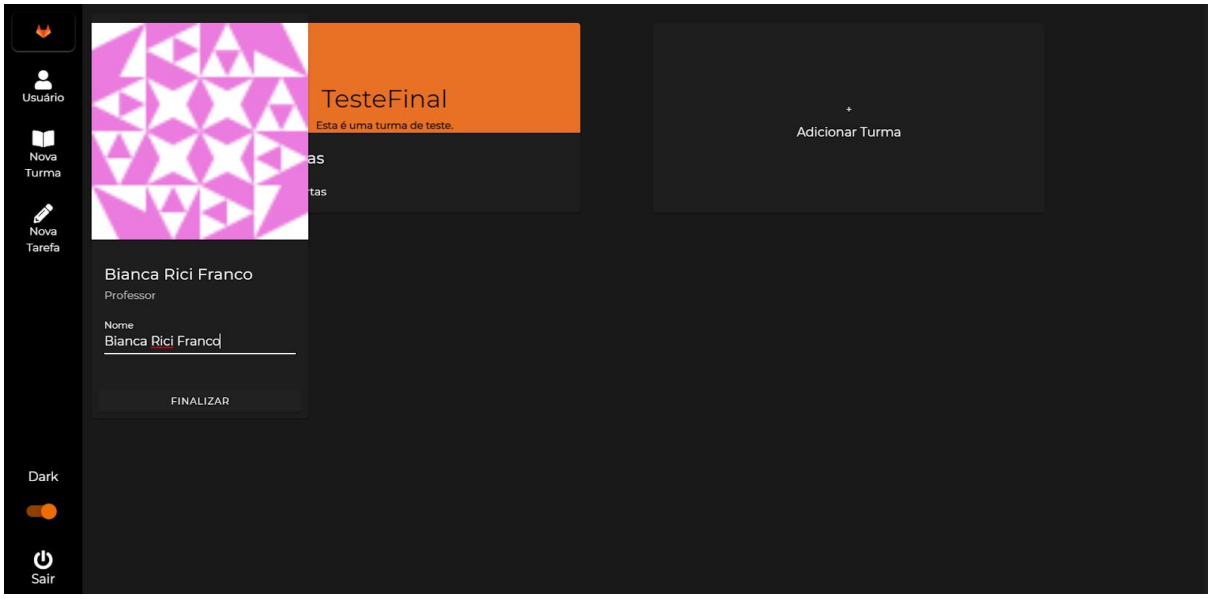


FLUXO PRINCIPAL:
- Professor clica no símbolo de remoção ao lado do registro do aluno, na tela da turma, que deseja remover;
- Sistema abre uma tela de confirmação da remoção;
- Professor clica em “Remover Aluno”(A1);
- SIstema aciona a API para desvincular o aluno da turma no GitLab;
- Sistema fecha a tela de confirmação;

FLUXOS ALTERNATIVOS:
A1 - Professor cancela a ação de remoção:
- Professor clica em “Cancelar”;
- Sistema fecha a tela de confirmação.

FLUXO DE EXCEÇÃO:
Não há.

UC012 - EDITAR DADOS
ATOR: Professor/Aluno.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom.
REGRAS DE NEGÓCIO: Somente o próprio usuário pode editar seus dados.
INTERFACE:

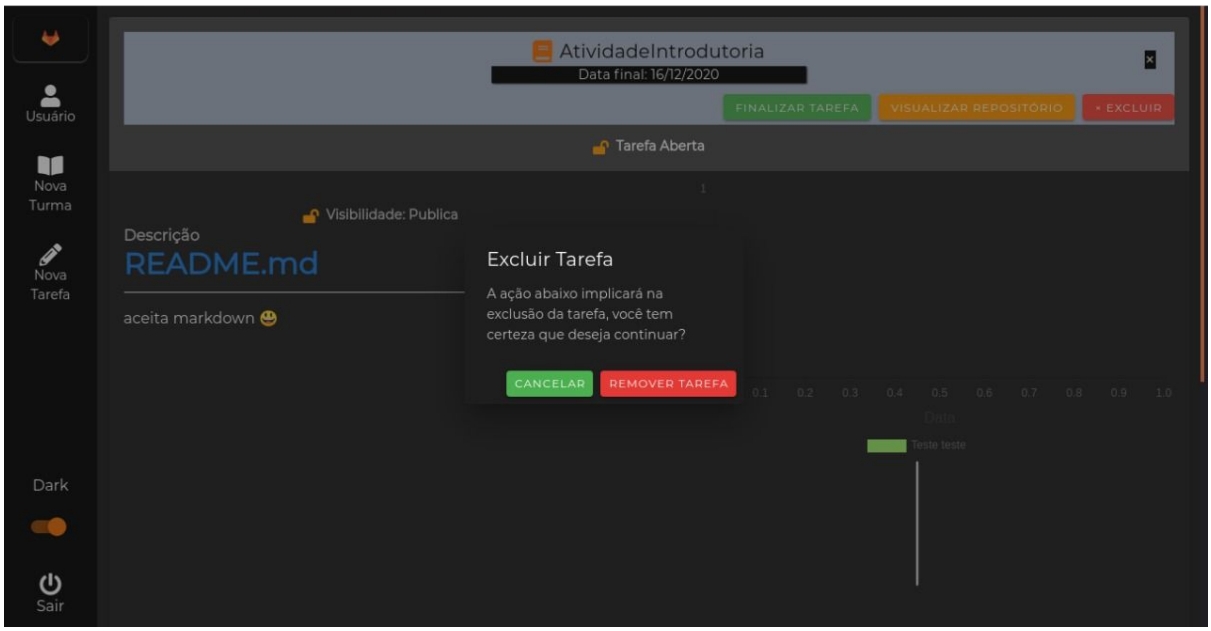


FLUXO PRINCIPAL:
- Usuário clica em “Usuário” na barra lateral;
- Usuário clica em “Editar”;
- Usuário altera seus dados e clica em “Finalizar”;
- Sistema atualiza a tabela users;
- Sistema salva as alterações.

FLUXOS ALTERNATIVOS:
Não há.

FLUXO DE EXCEÇÃO:
Não há.

UC013 - EXCLUIR TAREFA
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como “Professor” e estar relacionado à turma na qual os alunos estão inseridos.
REGRAS DE NEGÓCIO: Somente um Professor pode remover uma tarefa de sua turma.
INTERFACE:

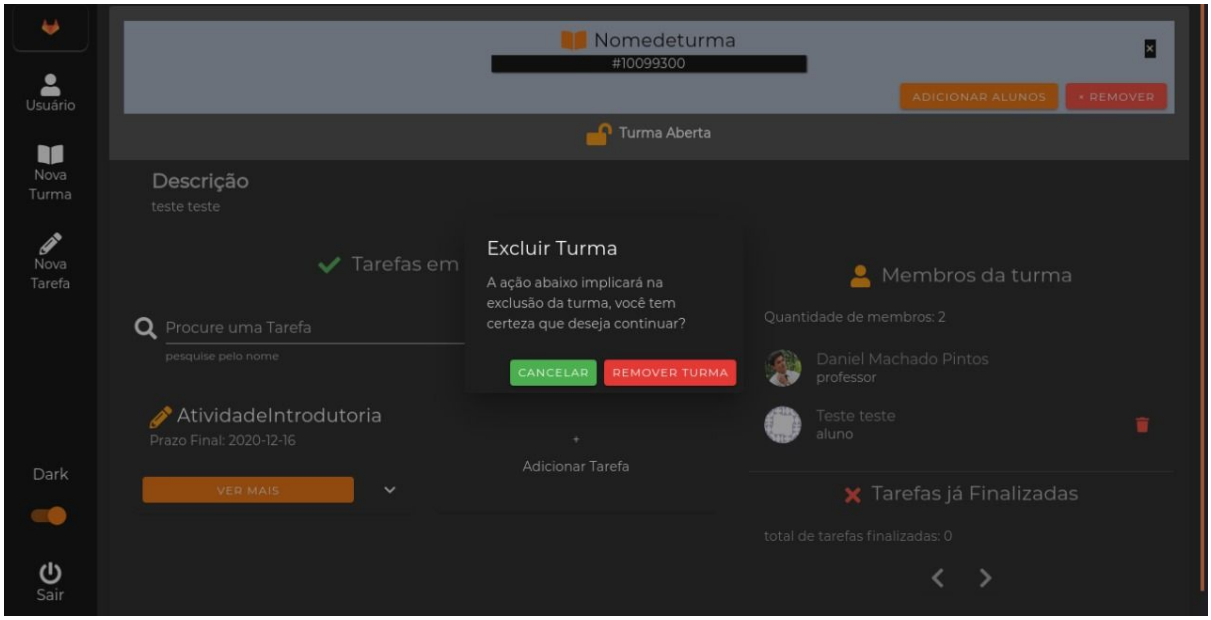


FLUXO PRINCIPAL:
- Professor clica em “Excluir”
- Sistema abre uma tela de confirmação da remoção;
- Professor clica em “Remover tarefa”(A1);
- Sistema fecha a tela de confirmação;
- Sistema chama a API para excluir a tarefa do GitLab;
- Sistema deleta os registros da tabela assignments;

FLUXOS ALTERNATIVOS:
A1 - Professor cancela a ação de remoção:
- Professor clica em “Cancelar”;
- Sistema fecha a tela de confirmação.

FLUXO DE EXCEÇÃO:
Não há.

UC014 - EXCLUIR TURMA
ATOR: Professor.
PRÉ-CONDIÇÕES: O usuário deve estar logado na plataforma GitLab Classroom como “Professor” e estar relacionado à turma na qual os alunos estão inseridos.
REGRAS DE NEGÓCIO: Somente um Professor pode remover uma turma.
INTERFACE:



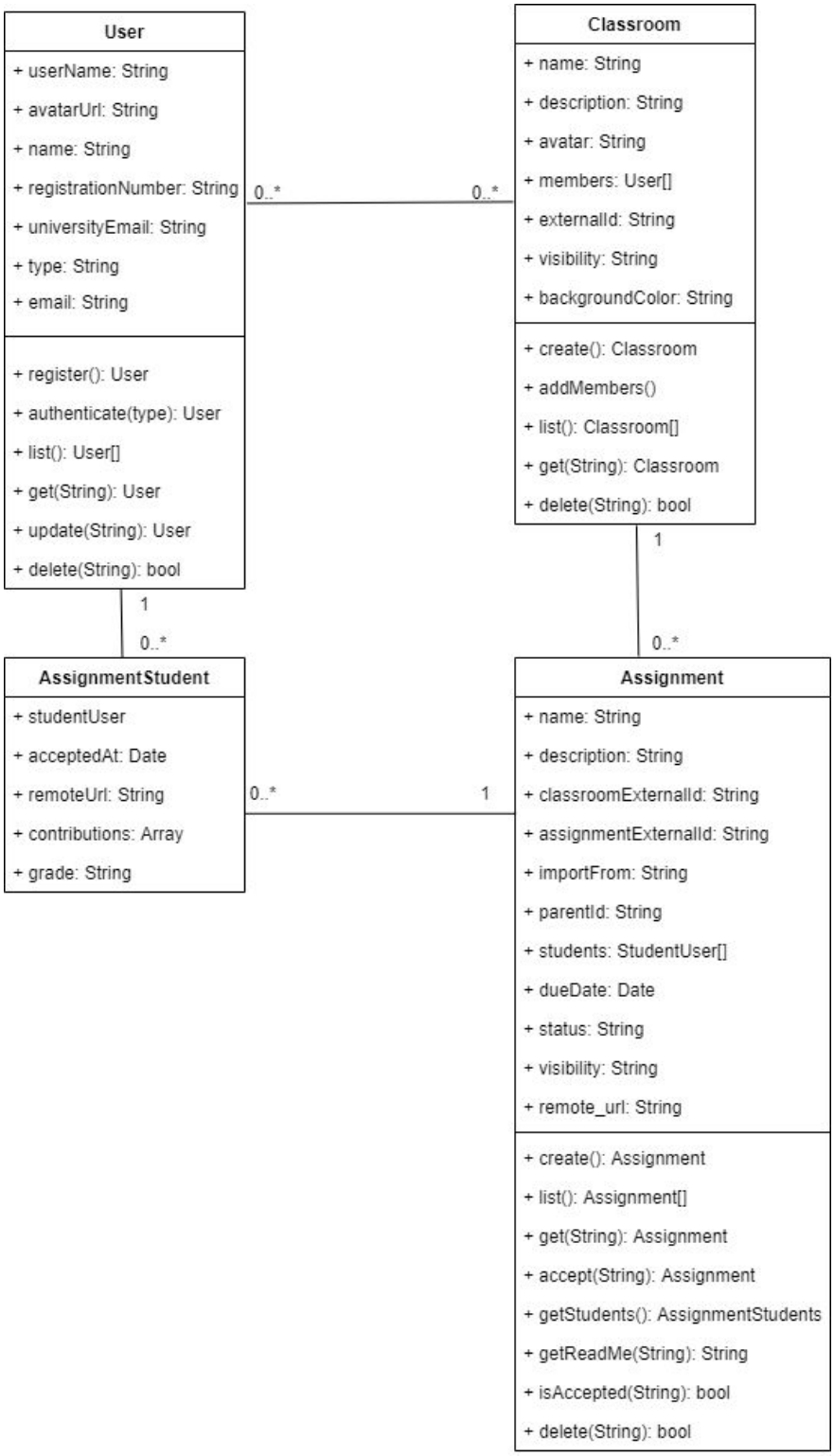
FLUXO PRINCIPAL:
- Professor clica em “Remover”
- Sistema abre uma tela de confirmação da remoção;
- Professor clica em “Remover turma”(A1);
- Sistema fecha a tela de confirmação;
- Sistema chama a API para excluir a turma do GitLab;

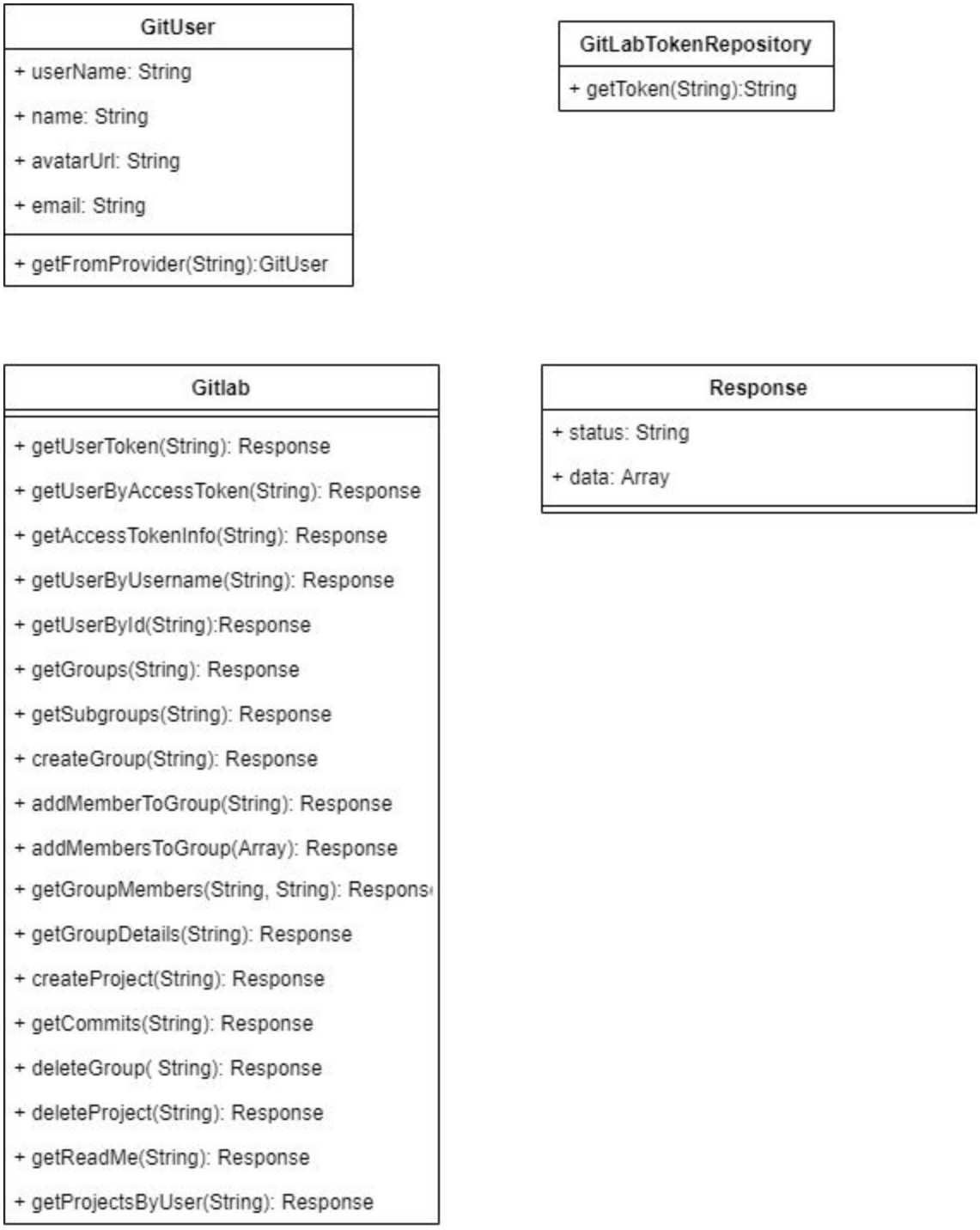
FLUXOS ALTERNATIVOS:
A1 - Professor cancela a ação de remoção:
- Professor clica em “Cancelar”;
- Sistema fecha a tela de confirmação.

FLUXO DE EXCEÇÃO:
Não há.

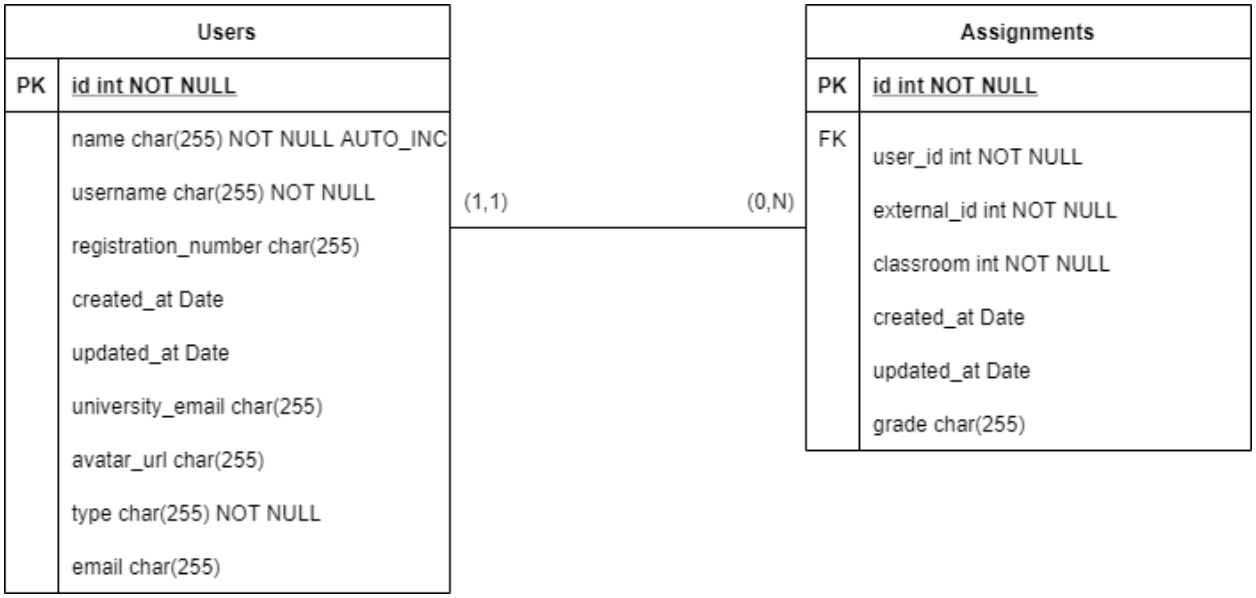
APÊNDICE 4 – DIAGRAMA DE CLASSES

Abaixo temos a representação das classes do sistema GitLab Classroom. Todas as classes que representam um artefato da aplicação (Classroom, Assignment, User) utilizam-se da classe GitLab, que possui todos os métodos de comunicação com essa aplicação. A classe GitLab não tem de fato relação com as demais, de forma que é somente consultada para que efetue as requisições à API, que por sua vez as transmite à aplicação GitLab.


User
+ userName: String
+ avatarUrl: String
+ name: String
+ registrationNumber: String
+ universityEmail: String
+ type: String
+ email: String
+ register(): User
+ authenticate(type): User
+ list(): User[]
+ get(String): User
+ update(String): User
+ delete(String): bool
0..*
0..*
Classroom
+ name: String
+ description: String
+ avatar: String
+ members: User[]
+ externalId: String
+ visibility: String
+ backgroundColor: String
+ create(): Classroom
+ addMembers()
+ list(): Classroom[]
+ get(String): Classroom
+ delete(String): bool
1
0..*
AssignmentStudent
+ studentUser
+ acceptedAt: Date
+ remoteUrl: String
+ contributions: Array
+ grade: String
0..*
1
Assignment
+ name: String
+ description: String
+ classroomExternalId: String
+ assignmentExternalId: String
+ importFrom: String
+ parentId: String
+ students: StudentUser[]
+ dueDate: Date
+ status: String
+ visibility: String
+ remote_url: String
+ create(): Assignment
+ list(): Assignment[]
+ get(String): Assignment
+ accept(String): Assignment
+ getStudents(): AssignmentStudents
+ getReadMe(String): String
+ isAccepted(String): bool
+ delete(String): bool


GitUser
+ userName: String
+ name: String
+ avatarUrl: String
+ email: String
+ getFromProvider(String):GitUser
GitLabTokenRepository
+ getToken(String):String
Gitlab
+ getUserToken(String): Response
+ getUserByAccessToken(String): Response
+ getAccessTokenInfo(String): Response
+ getUserByUsername(String): Response
+ getUserById(String):Response
+ getGroups(String): Response
+ getSubgroups(String): Response
+ createGroup(String): Response
+ addMemberToGroup(String): Response
+ addMembersToGroup(Array): Response
+ getGroupMembers(String, String): Respons
+ getGroupDetails(String): Response
+ createProject(String): Response
+ getCommits(String): Response
+ deleteGroup(String): Response
+ deleteProject(String): Response
+ getReadMe(String): Response
+ getProjectsByUser(String): Response
Response
+ status: String
+ data: Array

APÊNDICE 5 – MODELO LÓGICO DE BANCO DE DADOS



APÊNDICE 6 – DIAGRAMAS DE SEQUÊNCIA

DIAGRAMA 1 - REALIZAR CADASTRO

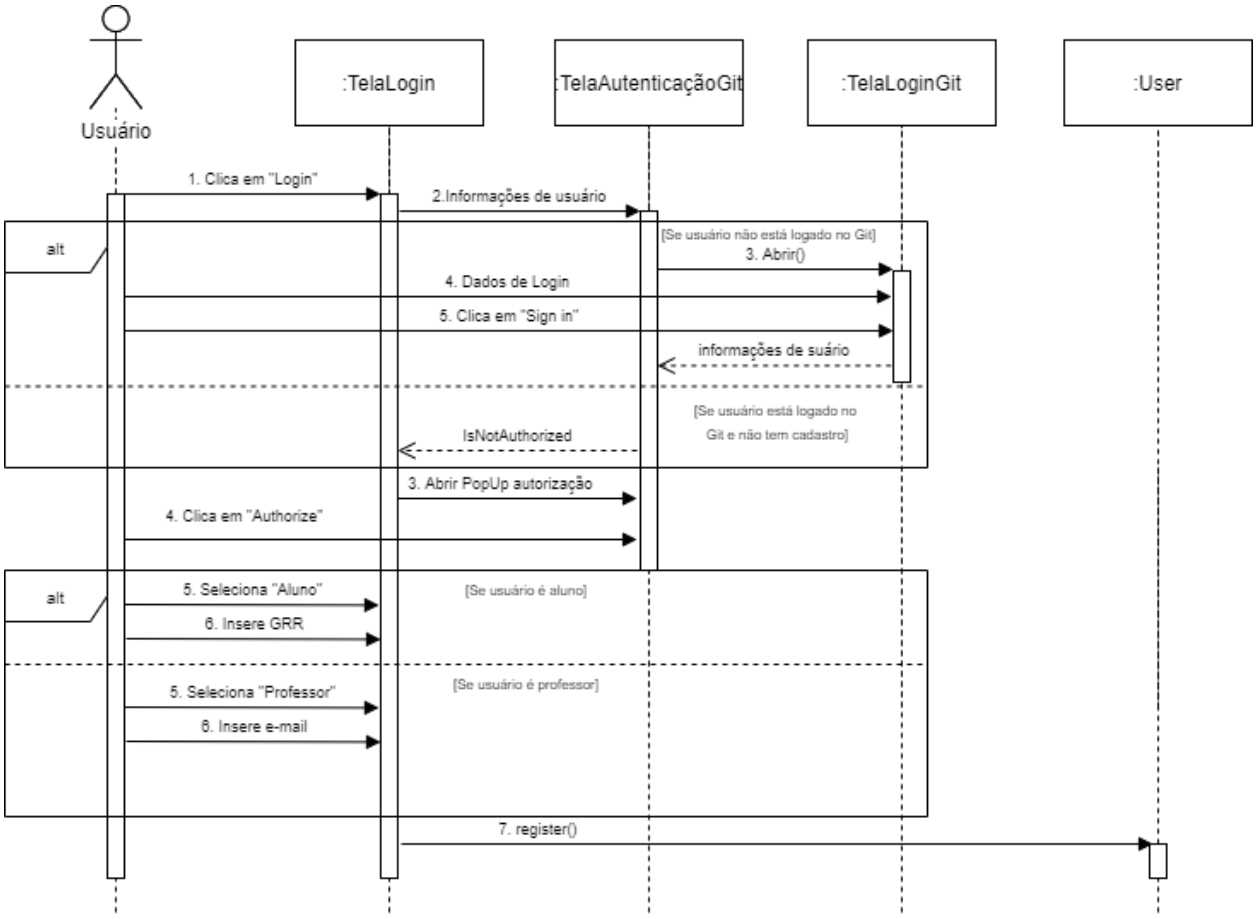


DIAGRAMA 2 - LOGIN

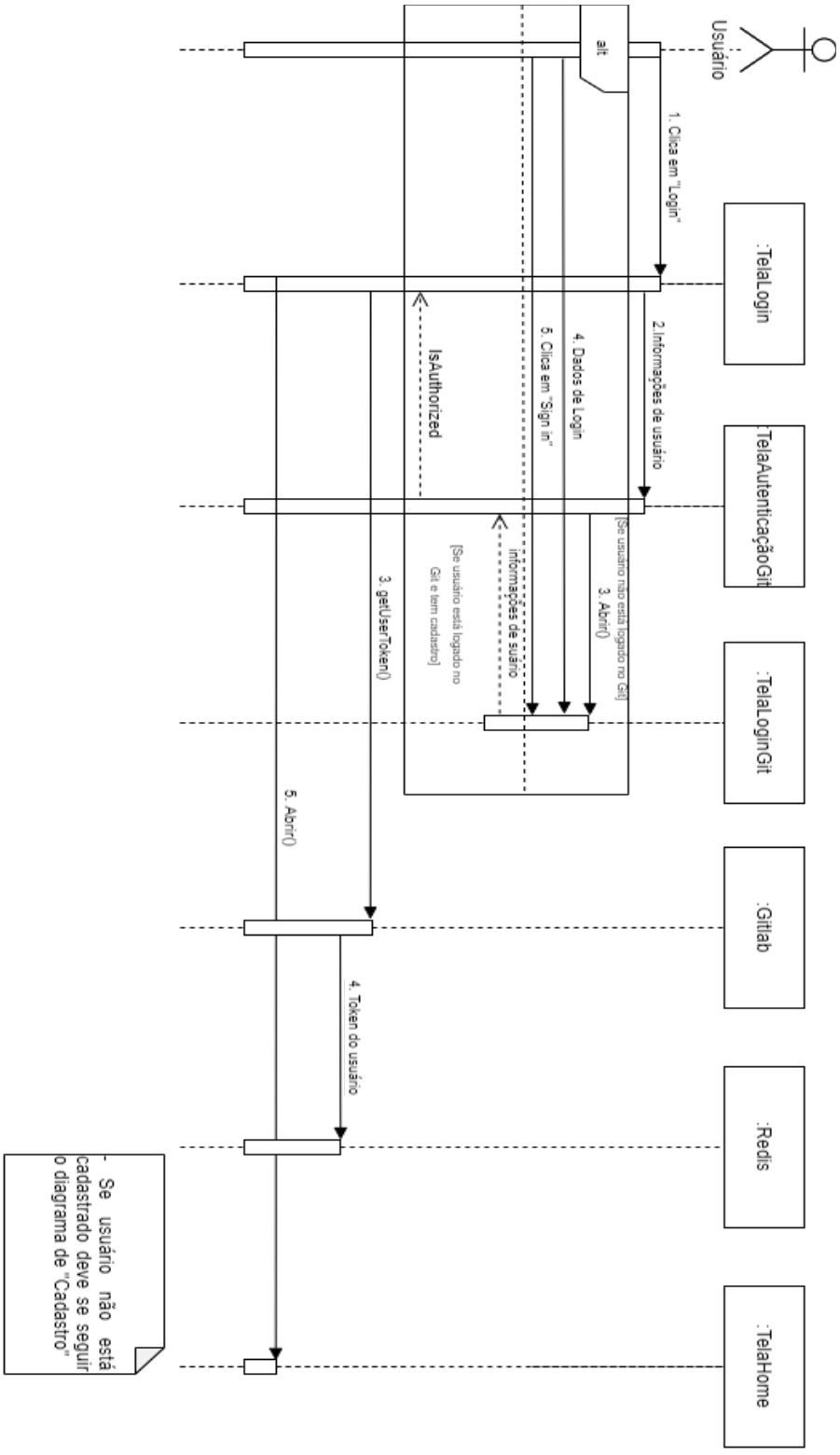


DIAGRAMA 3 - CRIAR TURMA

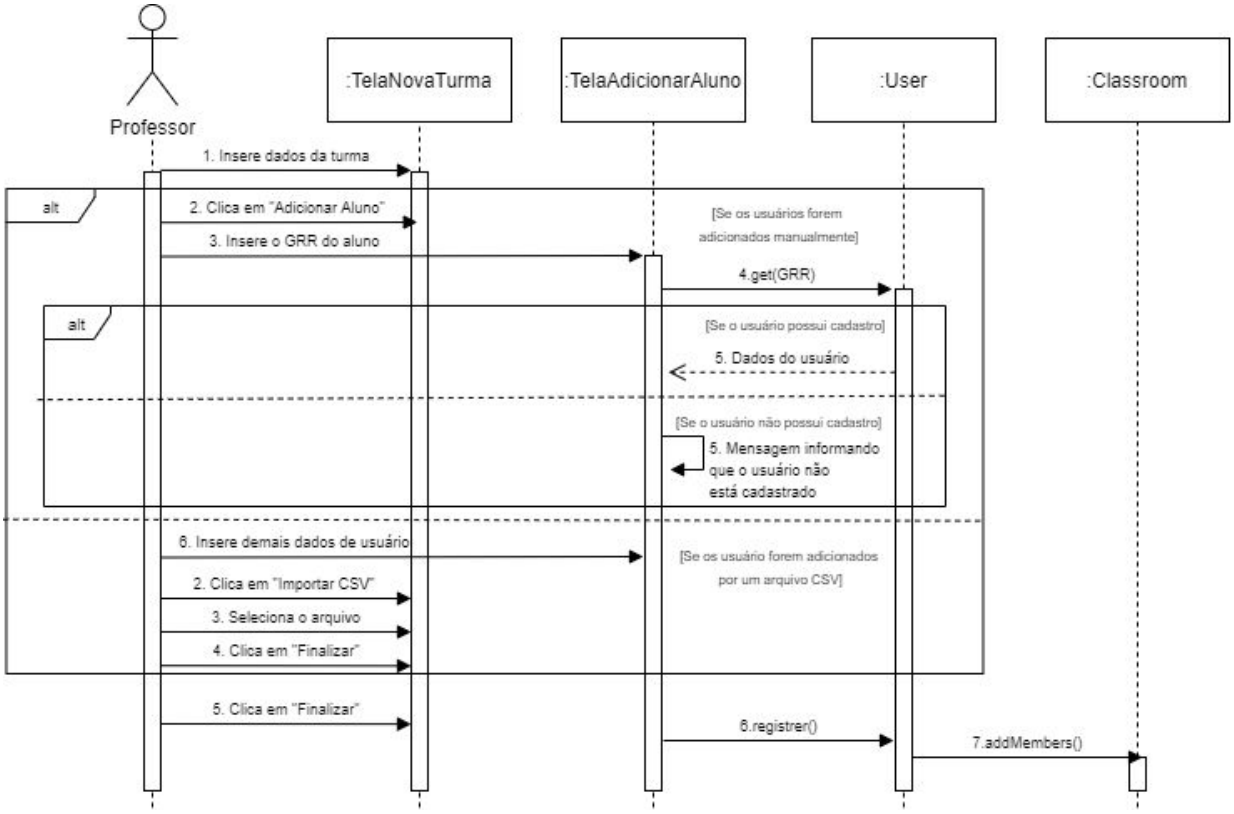


DIAGRAMA 4 - CRIAR TAREFA

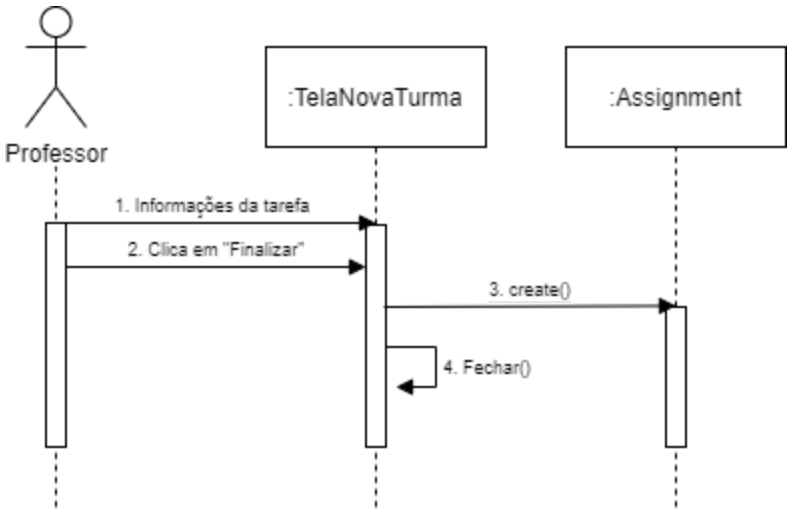


DIAGRAMA 5 - REALIZAR TAREFA

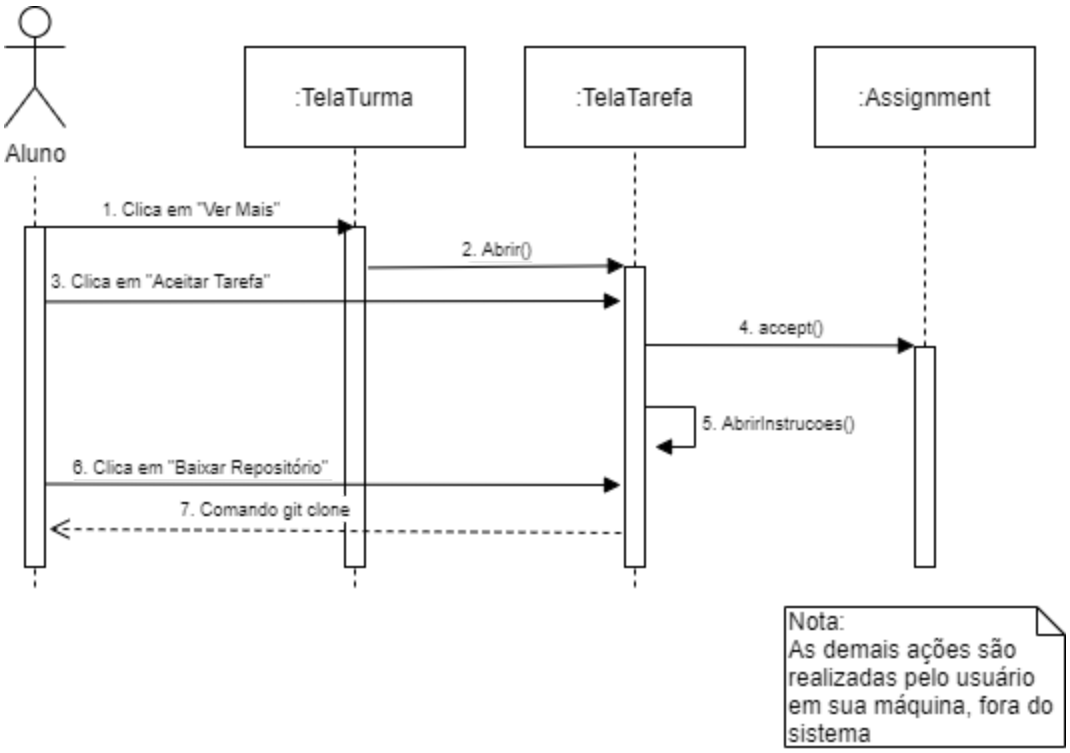


DIAGRAMA 6 - CONFERIR TAREFA

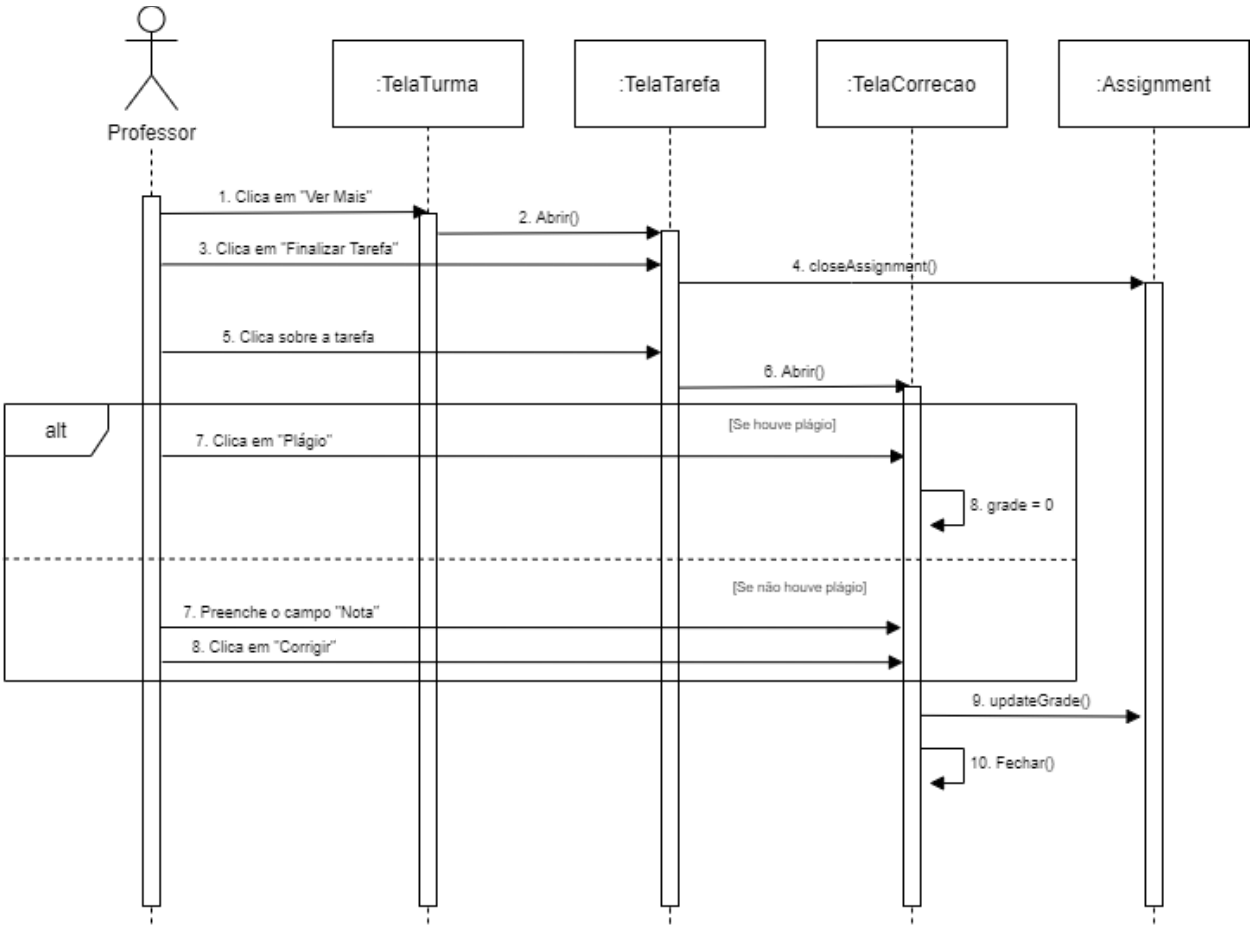


DIAGRAMA 7 - LOGOUT

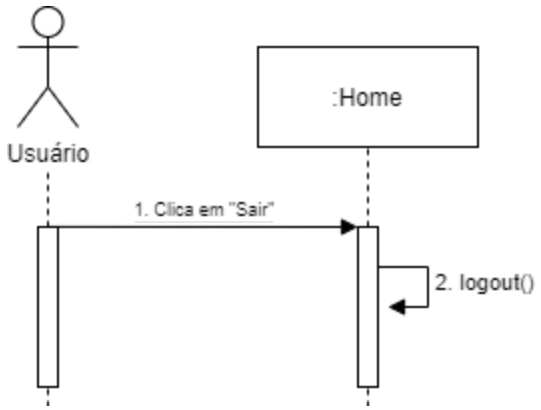


DIAGRAMA 8 - ADICIONAR ALUNO

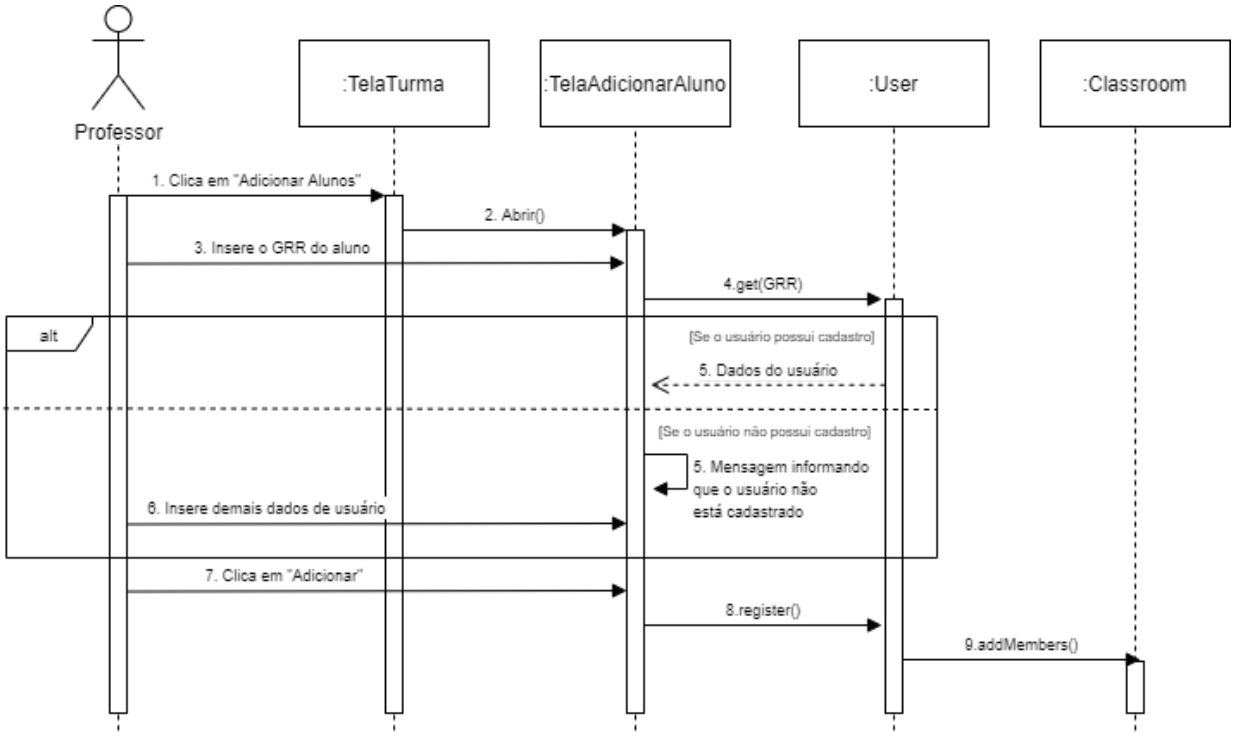


DIAGRAMA 9 - REMOVER ALUNO

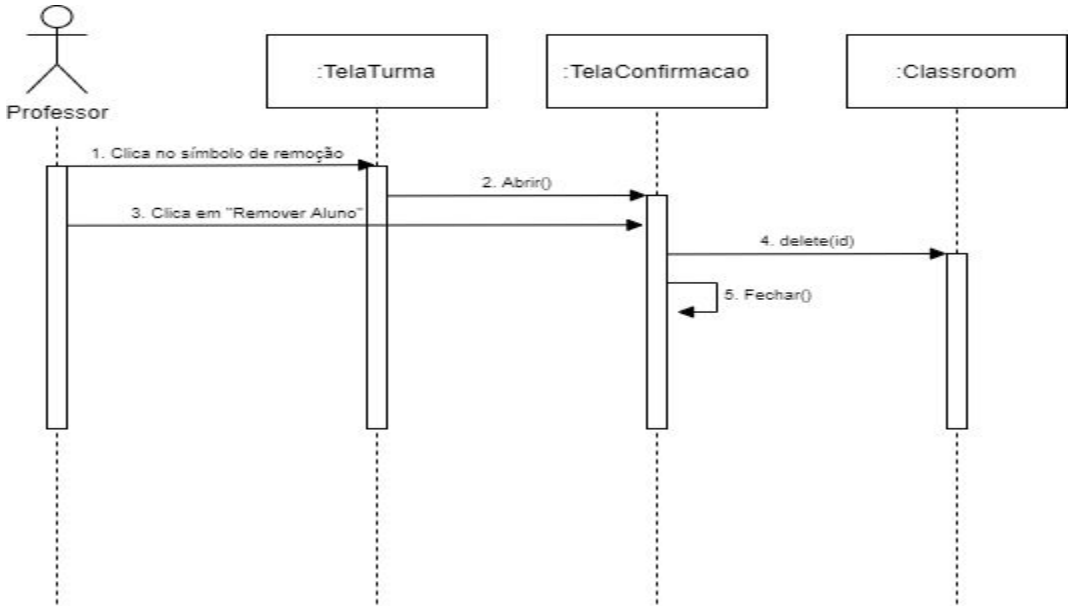


DIAGRAMA 10 - EXCLUIR TAREFA

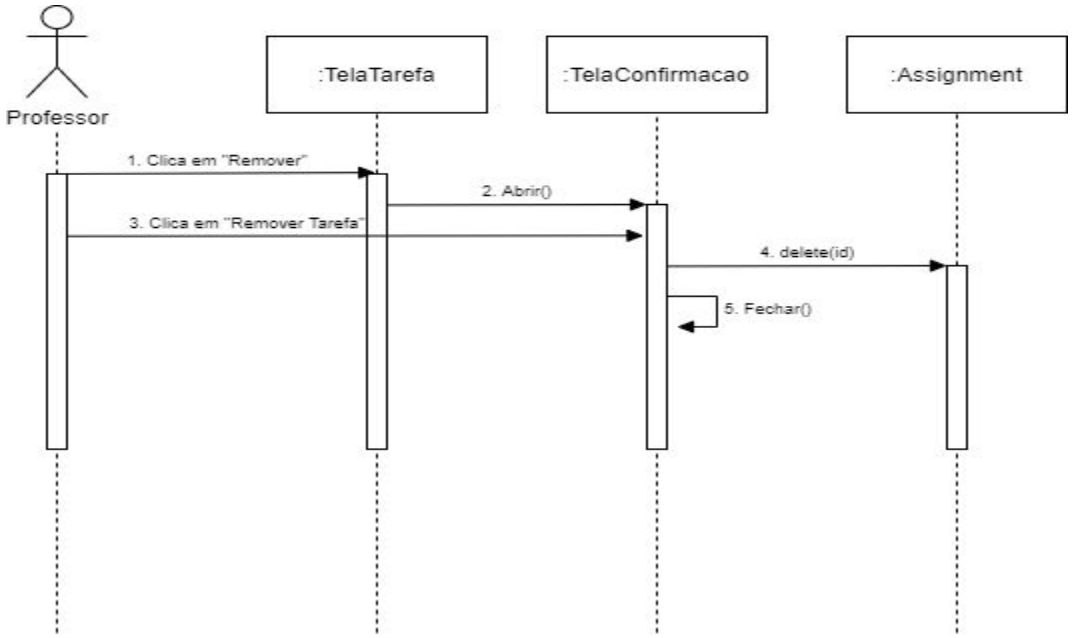
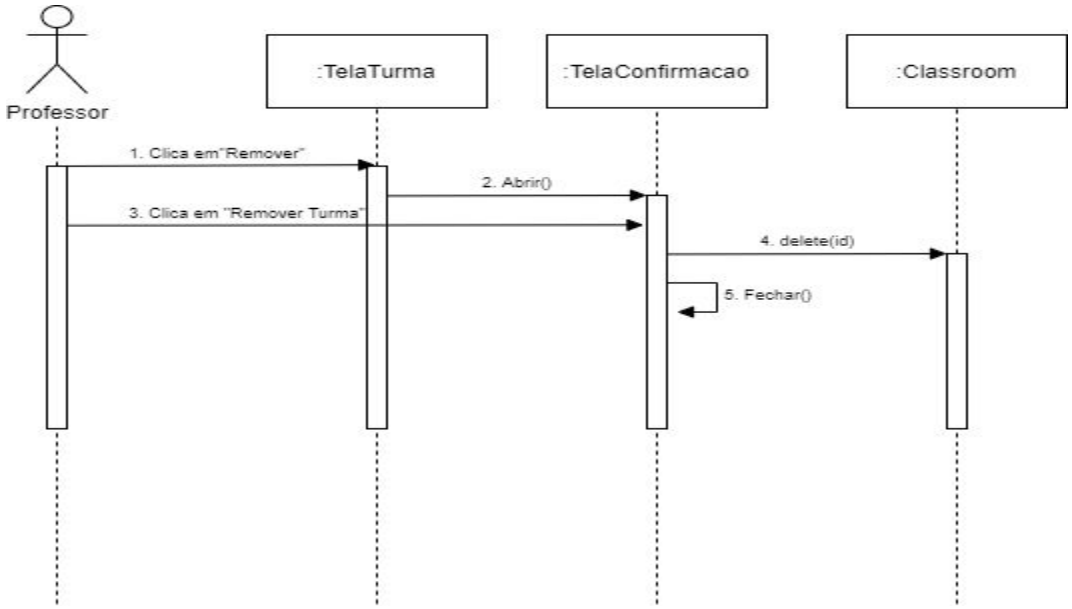


DIAGRAMA 11 - EXCLUIR TURMA



APÊNDICE 7 – DOCUMENTAÇÃO DA API

LABORATÓRIO
API

A API do GitLab Classroom foi intitulada Laboratório.

Chaves

Essas são as chaves que você deve gerar para autenticar e usar a API Laboratório

Login

Na aplicação *front-end*, deve ser implementado o Fluxo de Aplicação Web descrito na documentação da API GitLab.

POST laboratorio/api/user/auth

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código retornado do fluxo GitLab oauth2

Resposta:


```
{
   "response_status": "success",
   "data": {
      "user": {
         "id": 2,
         "name": "gi lab",
         "username": "gigilab",
         "registration_number": "grr20163127",
         "created_at": "2020-08-07 00:23:36",
         "updated_at": "2020-08-07 00:23:36",
         "university_email": null,
         "avatar_url":
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
         "type": "STUDENT"
      },
```

```
    "registration_error": null
  }
}
```

Register

Para que os usuários tenham acesso integral à API Laboratório eles devem estar cadastrados. Depois que realizam o login pela primeira vez com sua conta GitLab, será requisitado que se cadastrem na aplicação informando seu registro acadêmico (GRR) ou o e-mail da universidade (no caso de professores).

POST laboratorio/api/user

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	Para se autenticar na aplicação
registration_number	não	string	O registro acadêmico do estudante na universidade ou instituição. Perfis de estudantes devem possuir um, professores não.
university_email	não	string	O e-mail da universidade utilizado pelo professor.

Resposta:

```
{
  "response_status": "success",
  "data": {
    "username": "gigilab",
    "name": "gi lab",
    "avatar_url":
```

```
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
        "registration_number": null,
        "university_email": "giulia.padovani@ebanx.com",
        "type": "TEACHER",
        "updated_at": "2020-08-16 17:41:43",
        "created_at": "2020-08-16 17:41:43",
        "id": 4
    }
}
```


Users

Get all

Lista todos os usuários cadastrados na API.

GET laboratorio/api/users

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	Para se autenticar na aplicação

Resposta:


```
{
    "response_status": "success",
    "data": {
        "users": [
            {
                "id": 4,
                "name": "gi lab",
                "username": "gigilab",
                "registration_number": null,
                "created_at": "2020-08-16 17:41:43",
                "updated_at": "2020-08-16 17:41:43",
                "university_email": "giulia.padovani@ebanx.com",
                "avatar_url":
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
```

```
            "type": "TEACHER"
        }
    ]
  }
}
```

Get current

Retorna toda a informação sobre o usuário que está acessando a API no momento.

```
POST laboratorio/api/user/current
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	Para se autenticar na aplicação

Resposta:

```
{
   "response_status": "success",
   "data": {
      "user": {
            "id": 4,
            "name": "gi lab",
            "username": "gigilab",
            "registration_number": null,
            "created_at": "2020-08-16 17:41:43",
            "updated_at": "2020-08-16 17:41:43",
            "university_email": "giulia.padovani@ebanx.com",
            "avatar_url":
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
            "type": "TEACHER"
      }
   }
}
```

Get current user's classrooms

Retorna toda a informação sobre a turma do usuário que está acessando a API no momento.

GET laboratorio/api/user/current/classrooms

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	Para se autenticar na aplicação

Resposta:


```
{
   "response_status": "success",
   "data": [
      {
         "name": "Estrutura",
         "description": "teste",
         "avatar": null,
         "members": null,
         "external_id": "8018163"
      }
```


Get by Registration Number

Retorna toda a informação sobre um usuário informando seu registro acadêmico.

GET laboratorio/api/user

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	Para se autenticar na aplicação
registration_number	sim	string	GRR

Resposta:

```
{
   "response_status": "success",
   "data": {
      "user": {
            "id": 4,
            "name": "gi lab",
            "username": "gigilab",
            "registration_number": null,
            "created_at": "2020-08-16 17:41:43",
            "updated_at": "2020-08-16 17:41:43",
            "university_email": "giulia.padovani@ebanx.com",
            "avatar_url":
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
            "type": "TEACHER"
      }
   }
}
```


Classroom

A Turma/Classroom é o principal recurso compartilhado pelo professor e estudantes. Um professor pode criar e adicionar estudantes a ela. As tarefas estarão sempre vinculadas a uma turma.

Create

Cria um novo recurso.

```
POST laboratorio/api/classroom
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.
name	sim	string	Nome.
description	não	string	Descrição da turma.

members	não	string	Usuários a serem adicionados, separados por vírgula.
visibility	não	string	public | private

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "Nome do grupo",
      "description": "aaa",
      "members": null,
      "external_id": "8005774"
   }
}
```


Add members

Adiciona membros à turma.

```
POST laboratorio/api/classroom/members
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.
members	sim	string	Usuários a serem adicionados, separados por vírgula.

classroom_external_id	sim	int	O id da turma que relaciona-a ao grupo no GitLab.


```
{
   "response_status": "success",
   "data": [
      "renanromanio"
   ]
}
```


Get all

Consulta todas as turmas.

```
GET laboratorio/api/classroom/
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.

Resposta:


```
{
   "response_status": "success",
   "data": [
      {
         "name": "Estrutura",
         "description": "teste",
         "avatar": "https://gitlab.com/uploads/-/system/group/avatar/8018163/flo.jpg",
         "members": [],
         "external_id": "8018163"
      }
   ]
}
```

Get one

Busca a turma por seu id com as informações sobre ela, como os membros e a cor de fundo.

```
GET laboratorio/api/classroom/id
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "NovoNovoTeste",
      "description": "Turma de TCC2",
      "avatar": null,
      "members": [
         {
            "id": 3,
            "name": "gi lab",
            "username": "gigilab",
            "registration_number": "grr20163127",
            "created_at": "2020-08-16 00:36:43",
            "updated_at": "2020-08-16 00:36:43",
            "university_email": null,
            "avatar_url":
"https://secure.gravatar.com/avatar/34b5c51b1e2e9c87789af27829218a3d?s=80&d=identicon",
            "type": "STUDENT"
         }
      ],
      "external_id": "8701974"
   }
}
```

Assignments

Create

Cria o recurso.

```
POST laboratorio/api/assignment
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.
name	sim	string	O nome da tarefa
classroom_external_id	sim	string	O Id que identifica a turma à qual a tarefa pertence.
description	não	string	Descrição da tarefa
import_from	não	string	URL do repositório que serve como base para a tarefa. Todas as tarefas aceitas serão criadas a partir deste repositório.
due_date	sim	string	Qualquer data formatada.
visibility	não	string	public | private

Resposta:

```
{
```

```
  "response_status": "success",
  "data": {
    "name": "NovaComDataFormatadaDiferente",
    "description": "Entrega 1 do TCC2",
    "classroom_external_id": "8947443",
    "assignment_external_id": "8947458",
    "import_from": "https://gitlab.com/padovag/laboratorio.git",
    "parent_id": "8947443",
    "students": null,
    "due_date": "30-08-2020",
    "status": "OPENED"
  }
}
```


Delete

Deleta o recurso.

```
DELETE laboratorio/api/assignment/id
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.

Resposta:


```
{
  "response_status": "success",
  "data": {
    "deleted_assignment": "9856778"
  }
}
```


Accept

O estudante aceita a tarefa e é criada a tarefa-filha.

```
POST laboratorio/api/assignment/id/accept
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.

Resposta:

```
{
    "response_status": "success",
    "data": {
        "name": "Giulia-Tarefa20",
        "description": null,
        "classroom_external_id": null,
        "assignment external_id": 1234,
        "import_from": null,
    }
}
```

IsAccepted

Retorna se o estudante aceitou a tarefa baseado no Id da tarefa-pai informada na URL.

```
GET laboratorio/api/assignment/id/is_accepted
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição

code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "Giulia-Tarefa20",
      "description": null,
      "classroom_external_id": null,
      "assignment external_id": 1234,
      "import_from": null,
   }
}
```


Get All

Retorna todas as tarefas de uma turma.

```
GET laboratorio/api/classroom/id/assignments
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.
filter_by	não	string	CLOSED | OPENED

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "Giulia-Tarefa20",
      "description": null,
      "classroom_external_id": null,
      "assignment external_id": 1234,
      "import_from": null,
   }
}
```


Get

Retorna os dados de uma tarefa.

POST laboratorio/api/assignment/id

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O token de usuário no GitLab para se autenticar na aplicação e realizar as requisições ao GitLab.

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "Giulia-Tarefa20",
      "description": null,
      "classroom_external_id": null,
      "assignment external_id": 1234,
      "import_from": null,
   }
```

```
}
```

Get Students

Retorna os estudantes da turma e suas respostas.

```
GET laboratorio/api/assignment/id/students
```

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.
assignment_status	não	string	Status atual da tarefa.

Resposta:


```
{
   "response_status": "success",
   "data": {
      "name": "Entrega1",
      "description": "Entrega 1 do TCC2",
      "classroom_external_id": "8579370",
      "assignment_external_id": "8579380",
      "import_from": "https://gitlab.com/padovag/laboratorio.git",
      "parent_id": "8579370",
      "students": [
         {
            "student_user": "Giulia Padovani",
            "accepted_at": "2020-07-16T22:24:02.049Z",
            "remote_url": "https://gitlab.com/TCC222-group/Entrega1-group/giulia-padovani-entrega1",
            "contributions": [
               {
```

```
                "message": "Merge branch 'patch-1' into 'master'\n\nUpdate README.md\n\nSee merge request
TCC222-group/Entrega1-group/giulia-padovani-entrega1!1",
                "date": "2020-07-23T03:26:59.000+00:00"
            },
            {
                "message": "Update README.md",
                "date": "2020-07-23T03:25:03.000+00:00"
            },
            {
                "message": "Add README.md",
                "date": "2020-05-30T21:17:08.000+00:00"
            }
        ]
    },
    {
        "student_user": "giulia",
        "accepted_at": "2020-07-16T21:42:04.203Z",
        "remote_url": "https://gitlab.com/TCC222-group/Entrega1-group/giulia-entrega1",
        "contributions": []
    }
  ]
 }
}
```


Close

O professor deve escolher fechar a tarefa antes de começar a avaliá-la. Esta função irá salvar os dados da tarefa na base de dados.

GET laboratorio/api/assignment/id/close

Parâmetros:

Parâmetro	Obrigatório	Tipo	Descrição
code	sim	string	O código do usuário no GitLab para se autenticar na aplicação e realizar requisições ao GitLab.

Resposta:


```
{
   "response_status": "success",
   "data": [
      {
         "external_id": "8947444",
         "user_id": 1,
         "classroom_id": "8947443",
         "grade": null,
         "updated_at": "2020-09-28 03:07:59",
         "created_at": "2020-09-28 03:07:59",
         "id": 10
      },
      {
         "external_id": "8947444",
         "user_id": 2,
         "classroom_id": "8947443",
         "grade": 0,
         "updated_at": "2020-09-28 03:07:59",
         "created_at": "2020-09-28 03:07:59",
         "id": 11
      }
   ]
}
```


Grade

```
POST laboratorio/api/assignment/id/grade
```

Parâmetros:

```
{
          "grades": [
                    {"student_name": "giulia","grade": 10},
                    {"student_name": "Giulia Padovani","grade": 10}
          ]
}
```

Resposta:

```
{
   "response_status": "success",
   "data": [
      "The assignments have been graded"
   ]
}
```


Get Grades

```
GET laboratorio/api/assignment/id/grades
```

Parâmetros:
Nenhum.

Resposta:


```
{
   "response_status": "success",
   "data": [
      {
         "student": "giulia",
         "grade": "10"
      },
      {
         "student": "Giulia Padovani",
         "grade": "10"
      },
      {
         "student": "giulia",
         "grade": null
      }
   ]
}
```