
Documentação de Projeto

para o sistema

Plataforma de Apoio às Avaliações de Projetos GitHub

Versão 3.1

Projeto de sistema elaborado pelo(s) aluno(s) Guilherme Gabriel Silva Pereira e Lucas Ângelo Oliveira Martins Rocha e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor Cleiton Silva Tavares, orientação acadêmica do professor José Laerte Pires Xavier Junior e orientação de TCC II do professor (a ser definido no próximo semestre).

6 de maio de 2023.

Tabela de Conteúdo

| | |
|---|----|
| Tabela de Conteúdo | ii |
| Histórico de Revisões | ii |
| 1. Introdução | 1 |
| 2. Modelos de Usuário e Requisitos | 1 |
| 2.1 Descrição de Atores | 1 |
| 2.2 Modelos de Usuários | 1 |
| 2.3 Modelo de Casos de Uso e Histórias de Usuário | 1 |
| 2.3.1 Diagrama de Casos de Uso | 1 |
| 2.3.2 Histórias de Usuários | 1 |
| 2.4 Diagrama de Sequência do Sistema e Contrato de Operações | 1 |
| 3. Modelo de Projeto | 1 |
| 3.1 Diagrama de Classes | 1 |
| 3.2 Diagramas de Sequência | 1 |
| 3.3 Diagramas de Comunicação | 1 |
| 3.4 Arquitetura | 1 |
| 3.5 Diagramas de Estados | 1 |
| 3.6 Diagrama de Componentes | 1 |
| 4. Projeto de Interface com Usuário | 2 |
| 5. Modelo de Dados | 2 |
| 6. Modelo de Teste | 2 |

Histórico de Revisões

| Nome | Data | Razões para Mudança | Versão |
|-------------------------------------|----------------|--|--------|
| Guilherme Gabriel e Lucas Ângelo | 05/03/20 23 | <ul style="list-style-type: none"> • Criação do documento; • Inserindo título, nomes e datas; • Adicionando a Seção 1. Introdução; • Adicionando a Seção 2.3. Modelo de Casos de Uso e Histórias de Usuários; • Adicionando a Seção 2.3.1. Diagrama de Casos de Uso; • Adicionando a Seção 2.3.2. Histórias de Usuários. | 1.0 |
| Lucas Ângelo | 10/03/20 23 | <ul style="list-style-type: none"> • Corrigindo Diagrama de Casos de Uso, com adição dos atores secundários GitHub e Aluno; • Corrigindo erros gramaticais e removendo verbos no futuro. | 1.1 |
| Lucas Ângelo | 13/03/20 23 | <ul style="list-style-type: none"> • Adicionando a Seção 4. Projeto de Interface com Usuário. | 1.2 |

| | | | |
|-------------------------------------|----------------|--|-----|
| Guilherme Gabriel | 14/03/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 2. Modelos de Usuário e Requisitos; ● Adicionando a Seção 2.1. Descrição de Atores; ● Adicionando a Seção 2.2. Modelos de Usuários. | 1.3 |
| Lucas Ângelo | 07/04/20 23 | <ul style="list-style-type: none"> ● Corrigindo problemas apontados no diagrama de caso de uso, interfaces de usuário e erros gramaticais; ● Adicionando casos de uso faltantes; ● Adicionando a Seção 2.4. Diagrama de Sequência do Sistema e Contrato de Operações. | 2.0 |
| Lucas Ângelo | 11/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 3. Modelos de Projeto; ● Adicionando a Seção 3.1 Diagrama de Classes. | 2.1 |
| Guilherme Gabriel | 12/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 3.2 Diagramas de Sequência. | 2.2 |
| Guilherme Gabriel | 13/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 3.3 Diagramas de Comunicação. | 2.3 |
| Lucas Ângelo | 15/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 3.4 Arquitetura. | 2.4 |
| Lucas Ângelo | 23/04/20 23 | <ul style="list-style-type: none"> ● Atualizando versão do Diagrama de Classes. | 2.5 |
| Guilherme Gabriel | 27/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 3.5 Diagramas de Estados; ● Adicionando a Seção 3.6 Diagrama de Componentes e Implantação. | 2.6 |
| Lucas Ângelo | 28/04/20 23 | <ul style="list-style-type: none"> ● Corrigindo diagrama de pacotes; ● Removendo “subseção” do texto. | 2.7 |
| Lucas Ângelo | 29/04/20 23 | <ul style="list-style-type: none"> ● Adicionando a Seção 5 Glossário e Modelos de Dados. | 2.8 |
| Guilherme Gabriel | 04/05/20 23 | <ul style="list-style-type: none"> ● Adicionando a seção “Diagrama de Componentes e Implantação”; ● Corrigindo erros textuais e em imagens apontados pelos professores. | 2.9 |
| Lucas Ângelo | 04/05/20 23 | <ul style="list-style-type: none"> ● Atualizando numeração das figuras; ● Corrigindo erros gramaticais. | 3.0 |
| Guilherme Gabriel e Lucas Ângelo | 06/05/20 23 | <ul style="list-style-type: none"> ● Adicionando Seção 7.1 Cronograma | 3.1 |

1. Introdução

Este documento agrupa: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema Plataforma de Apoio às Avaliações de Projetos GitHub. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema. Tal especificação acompanha este documento. Anexo a este documento também se encontra o Glossário.

O sistema proposto neste trabalho consiste em uma plataforma web de apoio às avaliações de trabalhos no GitHub¹ que será utilizado pelos professores das disciplinas de “Trabalho Interdisciplinar”. Nessa plataforma os professores poderão cadastrar métodos avaliativos para cada oferta de disciplina e visualizar as informações resultantes dos repositórios dos trabalhos conforme o método avaliativo selecionado. A necessidade de uma plataforma com essas funções origina-se da carência de nenhuma outra aplicação que auxilie professores a avaliarem repositórios de código e os artefatos de documentação de trabalhos, por meio de filtros temporais e por integrantes.

O fluxo de funcionamento da plataforma inicia com o cadastro da oferta de uma disciplina em um semestre específico, com isso, o professor dessa disciplina deve selecionar quais são os repositórios dessa oferta. A partir disso, deve ser cadastrado um método avaliativo para essa disciplina, informando quais são os arquivos e seus respectivos diretórios onde devem estar criados e preenchidos. Por exemplo, na plataforma deve ser cadastrada a oferta da disciplina de “Trabalho Interdisciplinar 5: Aplicações Distribuídas 1º/2023” e seu método avaliativo contendo uma regra de consistência que exija a existência do arquivo “Documentacao/documento_de_arquitetura.md” contendo no mínimo 750 caracteres, a partir disso, todos os repositórios dessa oferta de disciplina verificam essa regra. Quando essa regra não for seguida por algum repositório de um trabalho com esse método avaliativo, é possível utilizar a opção de abrir uma *issue* padronizada no respectivo repositório.

Em relação ao controle temporal, também pode-se cadastrar sprints para ofertas da disciplina, o que possibilita avaliar contribuições de alunos em trabalhos em diferentes períodos temporais. Seguindo o mesmo exemplo da oferta da disciplina de “Trabalho Interdisciplinar 5: Aplicações Distribuídas 1º/2023”, podem ser cadastradas seis sprints para esse primeiro semestre de 2023, cada sprint com períodos pré-definidos de tempo, o que pode ser utilizado para filtrar contribuições de integrantes em trabalhos para cada uma dessas seis sprints. Por exemplo, para regra de consistência que exija a existência do arquivo “Documentacao/documento_de_arquitetura.md”, pode ser associada uma sprint que determina que até a data final da sprint esse documento deverá estar criado ou abrirá uma *issue* padronizada.

A partir da coleta dos dados dos repositórios do GitHub, resultados das validações das regras de consistências dos métodos avaliativos, contribuições de arquivos, linhas de código, *issues* e qualidade de código, a plataforma informa as contribuições no trabalho, por aluno e prazos de sprints para cada trabalho. Além disso, apresenta uma visualização de histórico de *commits* com filtro para alunos e sprint para cada repositório. Ademais, para cada repositório é apresentada a quantidade de *issues* abertas e fechadas por cada integrante de cada equipe, também sendo possível filtrar essa informação por sprint.

¹ <https://github.com/>

Esta plataforma conta com uma opção para efetuar uma inspeção da qualidade dos códigos por meio da ferramenta SonarQube² para cada repositório, essa inspeção calcula a qualidade do código do trabalho. Outrossim, com objetivo de facilitar o uso por meio dos professores, o sistema de autenticação e autorização utiliza o OAuth do próprio GitHub. Dessa forma, os professores são auxiliados na avaliação tanto qualitativa quanto quantitativa das entregas dos artefatos, documentação e qualidade de código dos trabalhos desenvolvidos pelas equipes de alunos.

2. Modelos de Usuário e Requisitos

Nesta seção são apresentados os modelos de usuário e requisitos do sistema. Mais especificamente, a Seção 2.1 descreve os atores do sistema, a Seção 2.2 discorre sobre o modelo de usuário, a Seção 2.3 sobre os casos de uso e histórias de usuário e a Seção 2.4, por fim, apresenta o diagrama de sequência do sistema e o contrato de operações. Os diagramas apresentados nesta seção foram modelados por meio da Linguagem de Modelagem Unificada (UML, do inglês *Unified Modeling Language*).

2.1 Descrição de Atores

Sobre os atores que interagem com a plataforma, pode-se dividi-los em primários e secundários. Os primários são aqueles que interagem diretamente com a plataforma, enquanto os secundários são aqueles que impactam ou são impactados indiretamente pelas ações realizadas dentro da plataforma. Portanto, pode-se definir que os atores da plataforma são os professores de disciplinas de trabalho interdisciplinar, os alunos das mesmas e o GitHub. A seguir são detalhados, individualmente, seu papel na plataforma.

O professor é o ator primário da plataforma, ou seja, o usuário que a utiliza diretamente. Ele é responsável por lecionar as disciplinas de trabalho interdisciplinar e, consequentemente, avaliar os repositórios dos alunos que cursam a disciplina. Seu objetivo principal na plataforma é acompanhar as entregas dos alunos.

O GitHub é um ator secundário do sistema. Ele é a plataforma de hospedagem de códigos que armazena os trabalhos dos alunos que cursam a disciplina. O próprio GitHub também é impactado pelo sistema, com a criação de *issues* a partir da detecção de erros nos repositórios dos alunos na plataforma.

O aluno é um ator secundário, impactado indiretamente pelas ações da plataforma. Eles cursam as disciplinas de trabalho interdisciplinar e realizam as entregas através do GitHub, sendo avaliados pelos professores. Além de alimentar os dados usados nas avaliações dentro da plataforma, ele recebe *feedbacks* através da criação de *issues* no GitHub.

2.2 Modelos de Usuários

Nesta seção são apresentados os modelos de usuários da plataforma. O modelo foi construído usando a estratégia de personas, no qual são especificados a biografia, personalidade, necessidades e frustrações. As personas são descrições de pessoas fictícias que representam o público alvo da

² <https://www.sonarsource.com/products/sonarqube/>

plataforma. A Figura 1 apresenta a persona que representa o usuário ativo da plataforma, o professor. Enquanto isso, a Figura 2 apresenta a persona que representa os alunos, impactados pelo uso do sistema.

Vanessa



IDADE 35
FORMAÇÃO Mestrado
CARGO Professora
LOCALIZAÇÃO Belo Horizonte

Personalidade

Entusiasta Comunicativa

Bio
Residente de Belo Horizonte, possui um mestrado na área de engenharia de software. Há 4 anos faz parte do corpo docente de Engenharia de Software na PUC Minas, lecionando a disciplina de Trabalho Interdisciplinar IV.

Necessidades

- Precisa acompanhar e avaliar as entregas dos alunos na disciplina que leciona
- Melhores ferramentas para auxiliar no processo de aprendizagem dos seus alunos.

Frustações

- O processo manual de avaliação das entregas é demorado.
- Falta de tempo hábil para formular retornos mais específicos para cada grupo sobre suas entregas.

Figura 1. Persona que representa um professor



Figura 2. Persona que representa um aluno

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta seção descreve os casos de uso e as histórias de usuário que contemplam as possíveis ações dos usuários da plataforma desenvolvida neste projeto. Para descrever os casos de uso, foi criado o Diagrama de Caso de Uso, o qual é responsável por descrever as interações entre a plataforma deste projeto e seus usuários ou outros sistemas, ele mostra as ações que um usuário ou sistema pode realizar em relação à plataforma em questão. As histórias de usuário tem objetivo de apresentar as necessidades do usuário em relação à plataforma de uma maneira clara e concisa para facilitar o entendimento das funcionalidades.

2.3.1 Diagrama de Casos de Uso

A Figura 3 ilustra o Diagrama dos Casos de Uso do sistema. Nesse diagrama é possível observar 3 atores da plataforma, sendo o ator primário Professor e os atores secundários GitHub e Aluno. O ator Professor representa os professores que acessam o sistema para efetuar as operações básicas de cadastro, atualização, deleção e visualização das funcionalidades de trabalhos, métodos avaliativos, regras de consistência, análise estática de código e *issues* padronizadas. Já o ator secundário GitHub é utilizado pela execução de tarefas cronometradas de busca e atualização da base de dados do sistema por meio da Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) do GitHub, também é utilizado para autenticação de usuários. Por fim, o ator Aluno é utilizado para análises de contribuições individuais nos repositórios de trabalhos.

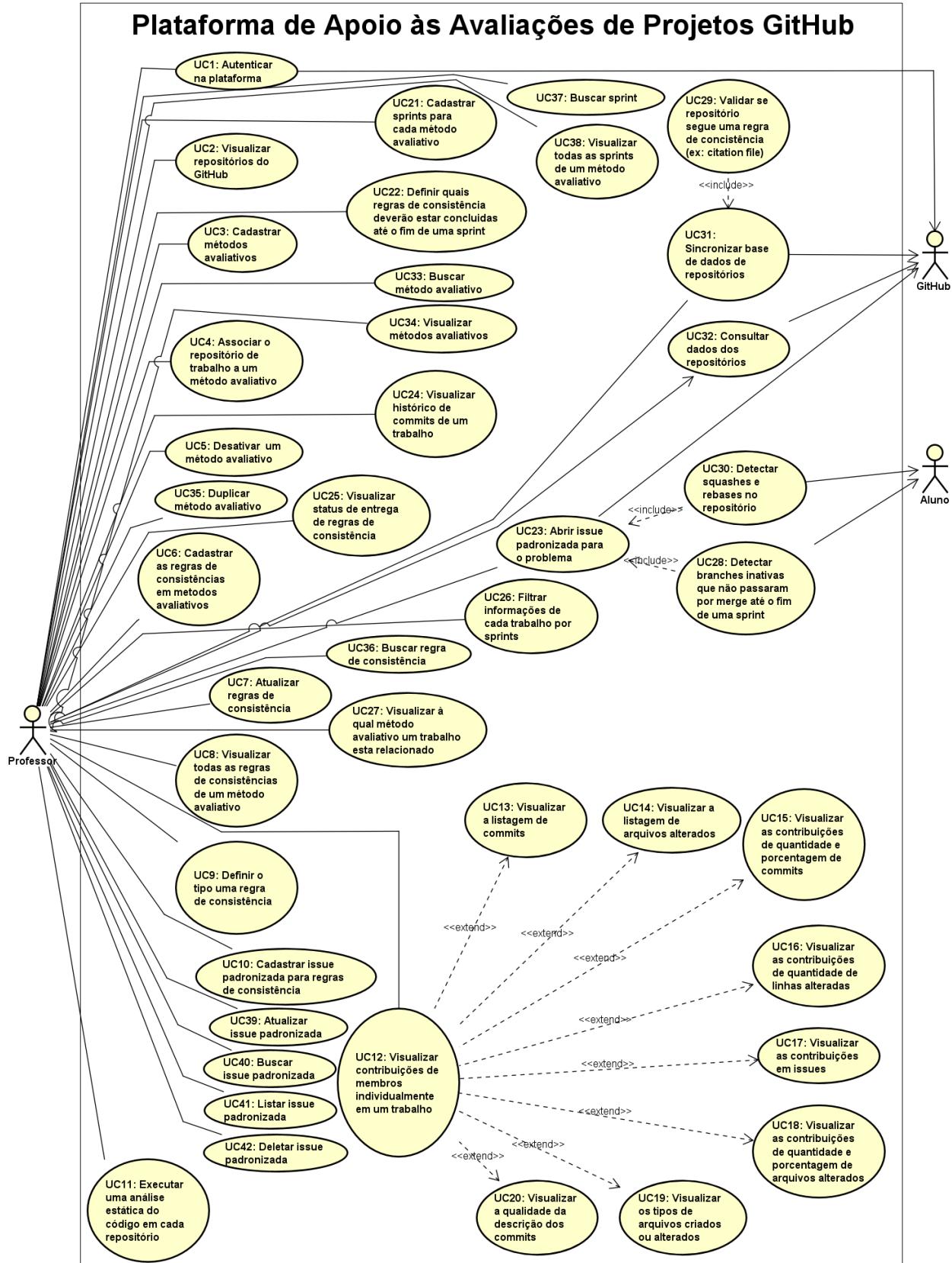


Figura 3. Diagrama de Caso de Uso

2.3.2 Histórias de Usuários

As histórias de usuário (US, do inglês *User Stories*) são uma forma de representar os requisitos do cliente. Trata-se de uma descrição simples, que informa quem realizará uma determinada ação, qual será essa ação, e a razão que ela é realizada. A seguir, são descritas as histórias de usuários levantadas para a plataforma, identificadas pela sigla US e uma numeração.

US01. Como professor, desejo me autenticar usando a conta do GitHub, para usar as funcionalidades do sistema;

US02. Como professor, desejo ver os repositórios que sou responsável, para poder avaliá-los;

US03. Como professor, desejo criar um método avaliativo para vincular aos repositórios que avaliarei;

US04. Como professor, desejo vincular um repositório, que sou responsável, a um método avaliativo, para o repositório ser avaliado de acordo com aquele método;

US05. Como professor, desejo inativar um método avaliativo obsoleto para que ele não possa mais ser vinculado a novos repositórios;

US06. Como professor, desejo criar uma regra de consistência de um artefato para que ele possa ser vinculado a um método avaliativo;

US07. Como professor, desejo criar uma regra de consistência do conteúdo de um artefato para que ele seja vinculado a um método avaliativo;

US08. Como professor, desejo vincular regras de consistência (pré-definidas ou criadas, do conteúdo ou não) a um método avaliativo, para que ele seja usado na avaliação dos repositórios vinculados;

US09. Como professor, desejo editar as regras de consistência para que eu possa corrigir informações anteriormente cadastradas;

US10. Como professor, desejo ver todas as regras de consistências cadastradas para um método avaliativo, para que eu possa saber como o conjunto de repositórios vinculados será avaliado;

US11. Como professor, desejo definir as sprints de um método avaliativo, para agrupar as entregas dos repositórios vinculados;

US12. Como professor, desejo ver quais as regras de consistência relacionadas a uma sprint, para poder saber o que deve ser entregue pelos alunos.

US13. Como professor, desejo ver quais regras de consistência foram cumpridas no prazo da sprint estipulada em um determinado repositório, para usar essa informação para o cálculo da nota do grupo;

US14. Como professor, desejo ver quais são os contribuidores do repositório, para descobrir os alunos que fazem parte daquele grupo;

US15. Como professor, desejo conferir os indicadores de contribuição (*commits*, linhas alteradas, *issues* fechadas) de um repositório, para poder avaliar o empenho geral do grupo;

US16. Como professor, desejo filtrar os indicadores de contribuição por contribuidor, para poder avaliar o empenho geral do integrante do grupo;

US17. Como professor, desejo conferir os indicadores de contribuição relativos por contribuidor (% de *commits*, % de linhas alteradas, % de *issues* fechadas), para poder avaliar o empenho relativo do integrante do grupo;

US18. Como professor, desejo filtrar os indicadores de contribuição por sprint, para poder avaliar o empenho naquela entrega;

US19. Como professor, desejo ver quais os tipos de arquivos foram alterados por cada integrante, em qual sprint, para ter uma visão mais específica do tipo contribuição que o integrante está fazendo;

US20. Como professor, desejo ver quais arquivos foram alterados por cada integrante, em qual sprint, para ter uma visão mais específica do tipo contribuição que o integrante está fazendo;

US21. Como professor, desejo disparar a execução de uma ferramenta de análise estática de código (SonarQube) no código do repositório, para avaliar a qualidade do código que está sendo produzidos pelos alunos;

US22. Como professor, desejo ver os resultados da execução de uma ferramenta de análise estática de código (SonarQube) no código do repositório, para avaliar a qualidade do código que está sendo produzidos pelos alunos;

US23. Como professor, desejo criar um *template* de *issue* em um método avaliativo para uma determinada regra de consistência, para ser usado na criação de *issues* nos repositórios;

US24. Como professor, desejo executar a criação de *issues* de *template*, para os alunos serem alertados de uma pendência na entrega;

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta seção, são apresentados os diagramas de sequência do sistema (DSS) com seus respectivos contratos de operações, sendo baseados nos casos de uso apresentados na Seção 2.3.1. Esses diagramas têm como finalidade descrever os fluxos de interação que ocorrem entre os usuários e o sistema desenvolvido.

A Figura 4 apresenta o DSS 1 relacionado ao fluxo de autenticar na plataforma. Nesse fluxo, o usuário Professor envia uma mensagem síncrona ao sistema com o objetivo de se autenticar na plataforma utilizando suas credenciais do GitHub. Ademais, esse fluxo também contata o sistema GitHub com intuito de validar as credenciais informadas. Por fim, o usuário recebe sua autorização para prosseguir utilizando a plataforma. Esse diagrama contempla o caso de uso UC1. Além disso, o contrato de operação desse diagrama é detalhado pela Tabela 1.

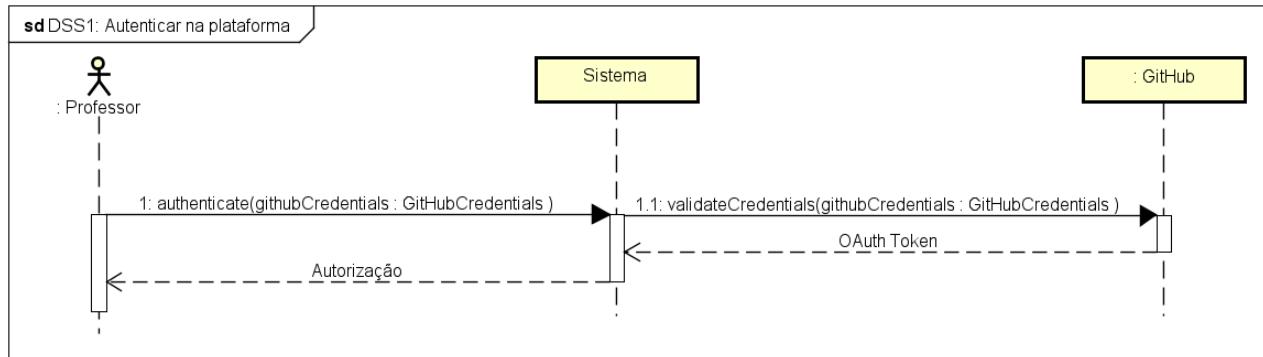


Figura 4. DSS 1: Autenticar na plataforma

| | |
|-----------------------------|---|
| Contrato | Autenticar. |
| Operação | authenticate(githubCredentials : GitHubCredentials) |
| Referências cruzadas | UC1: Autenticar na plataforma. |
| Pré-condições | Professor abre o sistema web sem estar autenticado. |
| Pós-condições | Iniciada uma sessão no sistema, o usuário está devidamente autenticado com uma autorização. |

Tabela 1. Contrato de operação para autenticar

A Figura 5 apresenta o DSS 2 relacionado ao fluxo de gerenciar métodos avaliativos. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, buscar, listar, desativar, duplicar ou adicionar um repositório a um método avaliativo. Dessa forma, o Professor consegue gerenciar os métodos avaliativos na plataforma. Esse diagrama contempla os casos de uso UC3, UC4, UC5, UC6, UC21, UC33, UC34 e UC35. Além disso, os contratos de operações desse diagrama são detalhados pelas tabelas Tabela 2, Tabela 3, Tabela 4, Tabela 5, Tabela 6, Tabela 7 e Tabela 8.

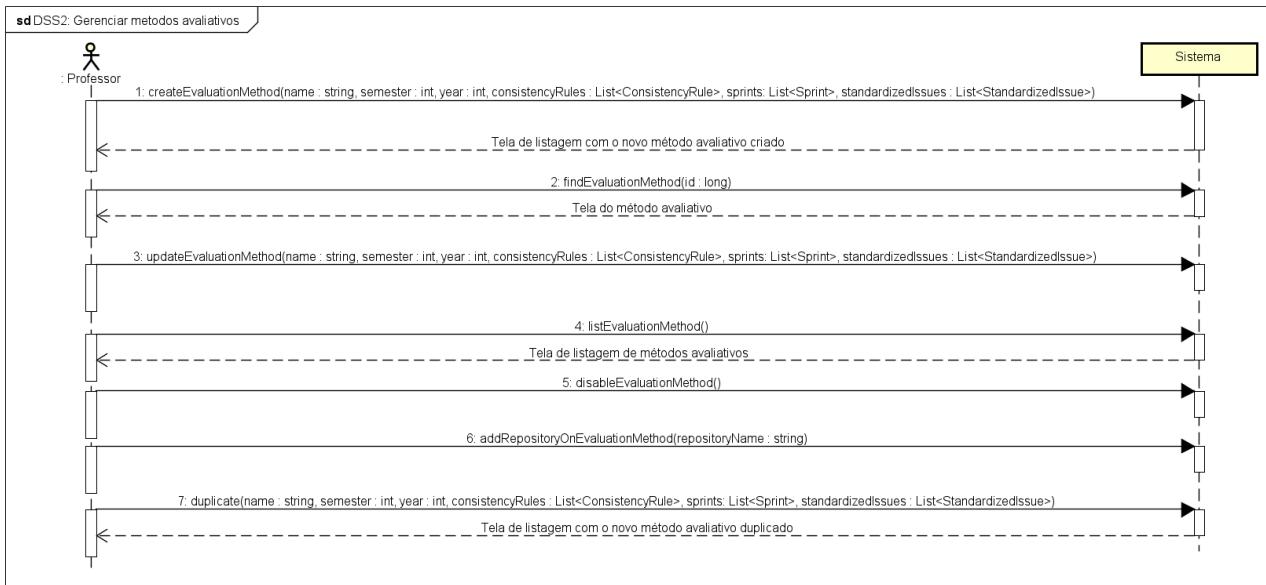


Figura 5. DSS 2: Gerenciar método avaliativos

| | |
|-----------------------------|--|
| Contrato | Criar um método avaliativo. |
| Operação | createEvaluationMethod(name : string, semester : int, year : int, consistencyRules : List<ConsistencyRule>, sprints: List<Sprint>, standardizedIssues : List<StandardizedIssue>) |
| Referências cruzadas | UC3: Cadastrar métodos avaliativos; UC6: Cadastrar as regras de consistências em métodos avaliativos; UC21: Cadastrar sprints para cada método avaliativo. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | O método avaliativo ser criado e a listagem de métodos atualizar. |

Tabela 2. Contrato de operação para criar um método avaliativo

| | |
|-----------------------------|--|
| Contrato | Buscar um método avaliativo. |
| Operação | findEvaluationMethod(id : long) |
| Referências cruzadas | UC33: Buscar método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | O método avaliativo deve ser apresentado. |

Tabela 3. Contrato de operação para buscar um método avaliativo.

| | |
|-----------------------------|--|
| Contrato | Atualizar um método avaliativo. |
| Operação | updateEvaluationMethod(name : string, semester : int, year : int, consistencyRules : List<ConsistencyRule>, sprints: List<Sprint>, standardizedIssues : List<StandardizedIssue>) |
| Referências cruzadas | UC6: Cadastrar as regras de consistências em métodos avaliativos; UC21: Cadastrar sprints para cada método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | O método avaliativo deve ser atualizado. |

Tabela 4. Contrato de operação para atualizar um método avaliativo

| | |
|-----------------------------|---|
| Contrato | Visualizar os métodos avaliativos. |
| Operação | listEvaluationMethod() |
| Referências cruzadas | UC34: Visualizar métodos avaliativos. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | Uma listagem dos métodos avaliativos ser apresentada. |

Tabela 5. Contrato de operação para visualizar os métodos avaliativos

| | |
|-----------------------------|--|
| Contrato | Desativar um método avaliativo. |
| Operação | disableEvaluationMethod() |
| Referências cruzadas | UC5: Desativar um método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | O método avaliativo deve ser desativado com deleção lógica. |

Tabela 6. Contrato de operação para desativar um método avaliativo

| | |
|-----------------------------|--|
| Contrato | Adicionar um repositório a um método avaliativo. |
| Operação | addRepositoryOnEvaluationMethod(repositoryName : string) |
| Referências cruzadas | UC4: Associar o repositório de trabalho a um método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | O repositório deve ser associado ao método avaliativo. |

Tabela 7. Contrato de operação para adicionar um repositório a um método avaliativo

| | |
|-----------------------------|---|
| Contrato | Duplicar um método avaliativo. |
| Operação | duplicate(name : string, semester : int, year : int, consistencyRules : List<ConsistencyRule>, sprints: List<Sprint>, standardizedIssues : List<StandardizedIssue>) |
| Referências cruzadas | UC35: Duplicar um método avaliativo; UC6: Cadastrar as regras de consistências em métodos avaliativos; UC21: Cadastrar sprints para cada método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | O método avaliativo deve duplicado juntamente com seus objetos associados. |

Tabela 8. Contrato de operação para duplicar um método avaliativo

A Figura 6 apresenta o DSS 3 relacionado ao fluxo de gerenciar regras de consistência. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, buscar e listar regras de consistência. Dessa forma, o Professor consegue gerenciar as regras de consistência dos métodos avaliativos na plataforma. Esse diagrama contempla os casos de uso UC6, UC7, UC8, UC9, UC22 e UC36. Além disso, os contratos de operações desse diagrama são detalhados pelas tabelas Tabela 9, Tabela 10, Tabela 11 e Tabela 12.

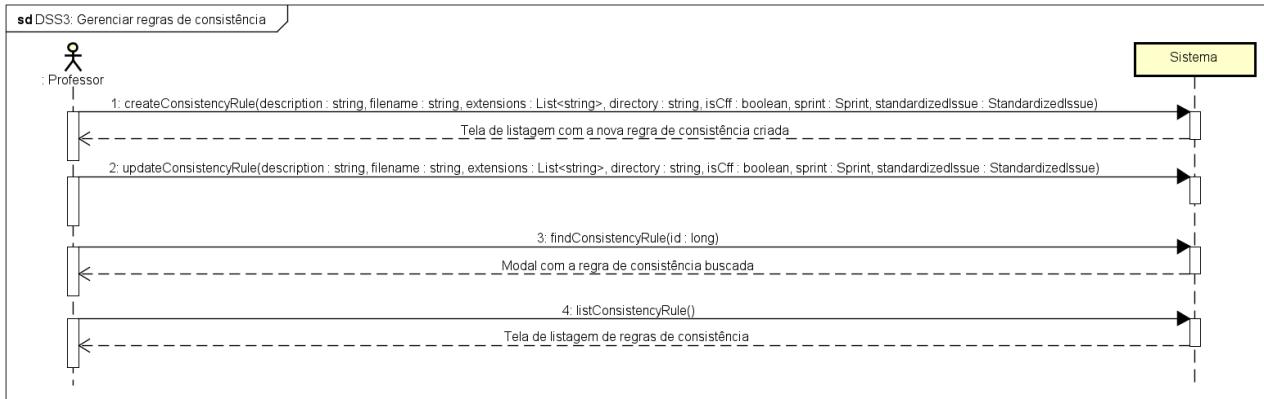


Figura 6. DSS 3: Gerenciar regras de consistência

| | |
|-----------------------------|---|
| Contrato | Criar uma regra de consistência. |
| Operação | createConsistencyRule(description : string, filename : string, extensions : List<string>, directory : string, isCff : boolean, sprint : Sprint, standardizedIssue : StandardizedIssue) |
| Referências cruzadas | UC6: Cadastrar as regras de consistências em métodos avaliativos; UC9: Definir o tipo uma regra de consistência; UC22: Definir quais regras de consistência deverão estar concluídas até o fim de uma sprint. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | A regra de consistência ser criada e a listagem de regras do método avaliativo correspondente atualizar. |

Tabela 9. Contrato de operação para criar uma regra de consistência

| | |
|-----------------------------|--|
| Contrato | Atualizar uma regra de consistência. |
| Operação | updateConsistencyRule(description : string, filename : string, extensions : List<string>, directory : string, isCff : boolean, sprint : Sprint, standardizedIssue : StandardizedIssue) |
| Referências cruzadas | UC7: Atualizar regras de consistência. |
| Pré-condições | Usuário estar autenticado e uma regra de consistência estar cadastrada. |
| Pós-condições | A regra de consistência ser atualizada e a listagem de regras do método avaliativo correspondente atualizar. |

Tabela 10. Contrato de operação para atualizar uma regra de consistência

| | |
|-----------------------------|---|
| Contrato | Buscar uma regra de consistência. |
| Operação | findConsistencyRule(id : long) |
| Referências cruzadas | UC36: Buscar regra de consistência. |
| Pré-condições | Usuário estar autenticado e uma regra de consistência estar cadastrada. |
| Pós-condições | A regra de consistência deve ser apresentada. |

Tabela 11. Contrato de operação para buscar uma regra de consistência

| | |
|-----------------------------|--|
| Contrato | Visualizar as regra de consistência. |
| Operação | listConsistencyRule() |
| Referências cruzadas | UC8: Visualizar todas as regras de consistências de um método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | Uma listagem das regras de consistência do método avaliativo correspondente ser apresentada. |

Tabela 12. Contrato de operação para visualizar as regras de consistência

A Figura 7 apresenta o DSS 4 relacionado ao fluxo de gerenciar sprints. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, buscar e listar sprints. Dessa forma, o Professor consegue gerenciar as sprints dos métodos avaliativos na plataforma. Esse diagrama contempla os casos de uso UC21, UC22, UC37 e UC38. Além disso, os contratos de operações desse diagrama são detalhados pelas tabelas Tabela 13, Tabela 14 e Tabela 15.

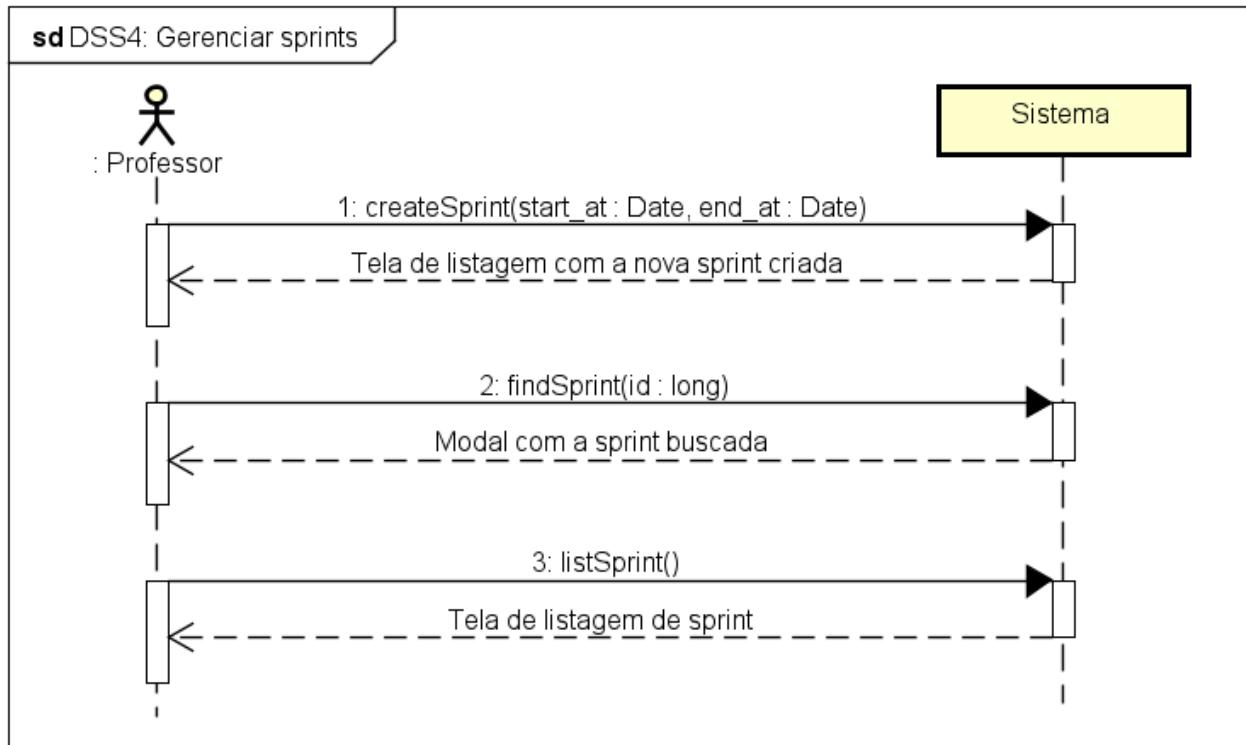


Figura 7. DSS 4: Gerenciar sprints

| | |
|-----------------------------|--|
| Contrato | Criar sprint. |
| Operação | <code>createSprint(start_at : Date, end_at : Date)</code> |
| Referências cruzadas | UC21: Cadastrar sprints para cada método avaliativo. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar criado. |

| | |
|----------------------|--|
| Pós-condições | A sprint ser criada e a listagem de sprints do método avaliativo correspondente atualizar. |
|----------------------|--|

Tabela 13. Contrato de operação para criar uma sprint

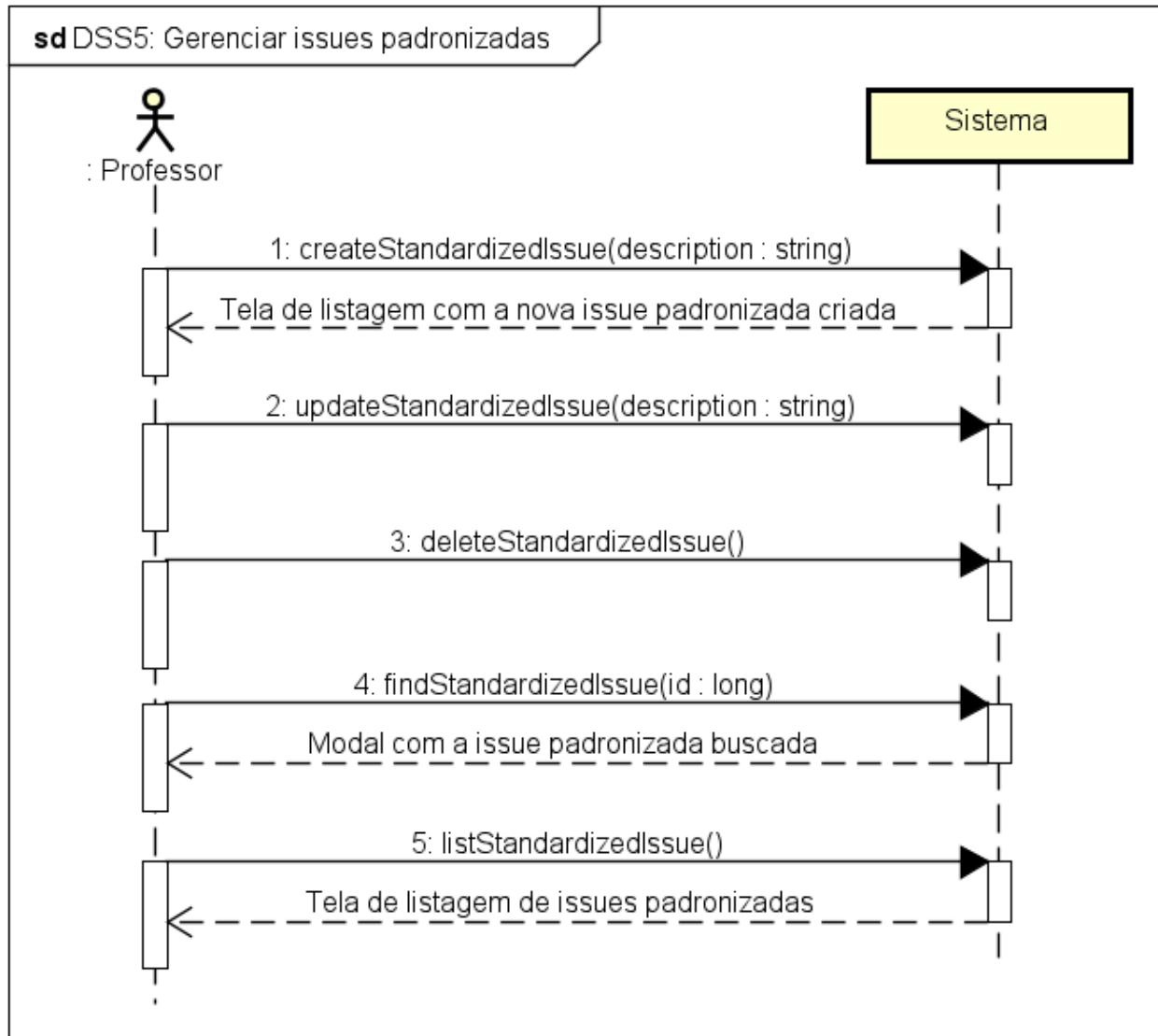
| | |
|-----------------------------|--|
| Contrato | Buscar uma sprint. |
| Operação | findSprint(id : long) |
| Referências cruzadas | UC37: Buscar sprint. |
| Pré-condições | Usuário estar autenticado e uma sprint estar criada. |
| Pós-condições | A sprint deve ser apresentada. |

Tabela 14. Contrato de operação para buscar uma sprint

| | |
|-----------------------------|---|
| Contrato | Visualizar as sprints. |
| Operação | listSprint() |
| Referências cruzadas | UC38: Visualizar todas as sprints de um método avaliativo |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | Uma listagem das sprints do método avaliativo correspondente ser apresentada. |

Tabela 15. Contrato de operação para visualizar as sprints

A Figura 8 apresenta o DSS 5 relacionado ao fluxo de gerenciar *issues* padronizadas. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, deletar, buscar e listar *issues* padronizadas. Dessa forma, o Professor consegue gerenciar as *issues* padronizadas dos métodos avaliativos na plataforma. Esse diagrama contempla os casos de uso UC10, UC39, UC40, UC41 e UC42. Além disso, os contratos de operações desse diagrama são detalhados pelas tabelas Tabela 16, Tabela 17, Tabela 18, Tabela 19 e Tabela 20.

Figura 8. DSS 5: Gerenciar *issues* padronizadas

| | |
|-----------------------------|---|
| Contrato | Criar <i>issue</i> padronizada. |
| Operação | createStandardizedIssue(description : string) |
| Referências cruzadas | UC10: Cadastrar <i>issue</i> padronizada para regras de consistência. |
| Pré-condições | Usuário estar autenticado, uma regra de consistência estar criada. |
| Pós-condições | A <i>issue</i> padronizada ser criada e a listagem de <i>issues</i> padronizadas do método avaliativo correspondente atualizar. |

Tabela 16. Contrato de operação para criar *issue* padronizada

| | |
|-----------------------------|---|
| Contrato | Atualizar <i>issue</i> padronizada. |
| Operação | updateStandardizedIssue(description : string) |
| Referências cruzadas | UC39: Atualizar <i>issue</i> padronizada. |
| Pré-condições | Usuário estar autenticado, uma <i>issue</i> padronizada estar criada. |

| | |
|----------------------|---|
| Pós-condições | A <i>issue</i> padronizada ser atualizada e a listagem de <i>issues</i> padronizadas do método avaliativo correspondente atualizar. |
|----------------------|---|

Tabela 17. Contrato de operação para atualizar *issue* padronizada

| | |
|-----------------------------|---|
| Contrato | Deletar <i>issue</i> padronizada. |
| Operação | deleteStandardizedIssue() |
| Referências cruzadas | UC42: Deletar <i>issue</i> padronizada. |
| Pré-condições | Usuário estar autenticado, uma <i>issue</i> padronizada estar criada. |
| Pós-condições | A <i>issue</i> padronizada deve ser deletada. |

Tabela 18. Contrato de operação para deletar *issue* padronizada

| | |
|-----------------------------|---|
| Contrato | Buscar <i>issue</i> padronizada. |
| Operação | findStandardizedIssue(id : long) |
| Referências cruzadas | UC40: Buscar <i>issue</i> padronizada. |
| Pré-condições | Usuário estar autenticado, uma <i>issue</i> padronizada estar criada. |
| Pós-condições | A <i>issue</i> padronizada deve ser apresentada. |

Tabela 19. Contrato de operação para buscar *issue* padronizada

| | |
|-----------------------------|--|
| Contrato | Listar <i>issues</i> padronizadas. |
| Operação | listStandardizedIssue() |
| Referências cruzadas | UC41: Listar <i>issue</i> padronizada. |
| Pré-condições | Usuário estar autenticado e um método avaliativo estar cadastrado. |
| Pós-condições | Uma listagem das <i>issues</i> padronizadas do método avaliativo correspondente ser apresentada. |

Tabela 20. Contrato de operação para listar *issues* padronizadas

A Figura 9 apresenta o DSS 6 relacionado ao fluxo de gerenciar repositórios de trabalhos. Nesse fluxo, o usuário Professor envia mensagens síncronas e assíncronas ao sistema com o objetivo de sincronizar, buscar, listar, atualizar, calcular métricas e qualidade do código, buscar histórico de *commits* e consistência das entregas de um repositório de trabalho. Dessa forma, o Professor consegue gerenciar os repositórios de trabalhos. Esse diagrama contempla os casos de uso UC2, UC11, UC12, UC24, UC25, UC26, UC29, UC31 e UC32. Além disso, os contratos de operações desse diagrama são detalhados pelas tabelas Tabela 21, Tabela 22, Tabela 23, Tabela 24, Tabela 25, Tabela 26 e Tabela 27.

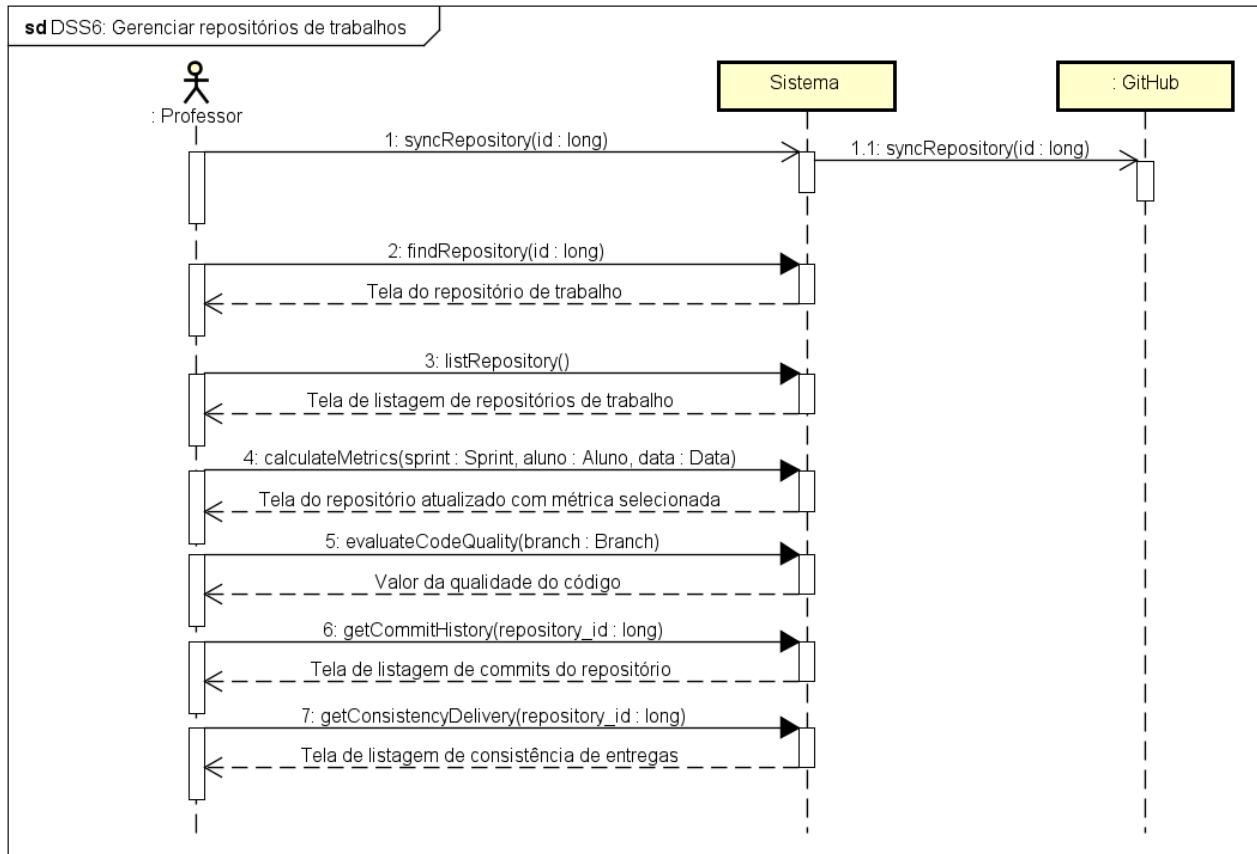


Figura 9. DSS 6: Gerenciar repositórios de trabalhos

| | |
|-----------------------------|---|
| Contrato | Sincronizar repositório de trabalho. |
| Operação | syncRepository(id : long) |
| Referências cruzadas | UC31: Sincronizar base de dados de repositórios. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | O repositório de trabalho ser sincronizado com a base de dados proveniente do GitHub. |

Tabela 21. Contrato de operação para sincronizar repositório de trabalho

| | |
|-----------------------------|--|
| Contrato | Buscar repositório de trabalho. |
| Operação | findRepository(id : long) |
| Referências cruzadas | UC32: Consultar dados dos repositórios. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | O repositório de trabalho ser apresentado. |

Tabela 22. Contrato de operação para buscar repositório de trabalho

| | |
|-----------------------------|---|
| Contrato | Listar repositórios de trabalhos. |
| Operação | listRepository() |
| Referências cruzadas | UC2: Visualizar repositórios do GitHub. |
| Pré-condições | Usuário estar autenticado. |

| | |
|----------------------|--|
| Pós-condições | A listagem de repositórios de trabalhos ser apresentada. |
|----------------------|--|

Tabela 23. Contrato de operação para listar repositórios de trabalhos

| | |
|-----------------------------|---|
| Contrato | Calcular métricas de um repositório de trabalho. |
| Operação | calculateMetrics(sprint : Sprint, aluno : Aluno, data : Data) |
| Referências cruzadas | UC12: Visualizar contribuições de membros individualmente em um trabalho; UC26: Filtrar informações de cada trabalho por sprints. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | As métricas do repositório de trabalho correspondente serem calculadas. |

Tabela 24. Contrato de operação para calcular métricas de um repositório de trabalho

| | |
|-----------------------------|--|
| Contrato | Calcular qualidade de código de um repositório de trabalho. |
| Operação | evaluateCodeQuality(branch : Branch) |
| Referências cruzadas | UC11: Executar uma análise estática do código em cada repositório. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | A qualidade de código do repositório de trabalho correspondente ser calculada. |

Tabela 25. Contrato de operação para calcular qualidade de código de um repositório de trabalho

| | |
|-----------------------------|--|
| Contrato | Buscar histórico de <i>commits</i> de um repositório de trabalho. |
| Operação | getCommitHistory(repository_id : long) |
| Referências cruzadas | UC24: Visualizar histórico de commits de um trabalho. |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | O histórico de <i>commits</i> do repositório de trabalho correspondente ser apresentada. |

Tabela 26. Contrato de operação para buscar histórico de commits de um repositório de trabalho

| | |
|-----------------------------|--|
| Contrato | Buscar consistência de entregas um repositório de trabalho. |
| Operação | getConsistencyDelivery(repository_id : long) |
| Referências cruzadas | UC25: Visualizar status de entrega de regras de consistência; UC29: Validar se repositório segue uma regra de concistênci (ex: citation file). |
| Pré-condições | Usuário estar autenticado. |
| Pós-condições | A consistência de entregas baseado nas regras de consistência do repositório de trabalho correspondente ser apresentada. |

Tabela 27. Contrato de operação para buscar consistência de entregas um repositório de trabalho

3. Modelos de Projeto

Nesta seção são apresentados os modelos de projeto por meio de diagramas na UML. A seção é composta pelo Diagrama de Classes, apresentados na Seção 3.1, os Diagramas de Sequência,

apresentados na Seção 3.2, os Diagramas de Comunicação na Seção 3.3, a Arquitetura apresentada na Seção 3.4, os Diagramas de Estado, apresentados na Seção 3.5 e os Diagramas de Componentes e Implantação na Seção 3.6.

3.1 Diagrama de Classes

Nesta seção, é apresentado o diagrama de classes que modela objetos e suas relações e seus respectivos relacionamentos utilizados para tratar as informações da plataforma. O propósito desta seção consiste em apresentar todos os modelos de dados implementados, com o intuito de fornecer uma visão geral da arquitetura das classes que compõem o sistema e assegurar o funcionamento correto da plataforma.

A Figura 10 apresenta o diagrama de classes de modelos do sistema, o qual contém as principais classes *Repository*, *Branch*, *Commit*, *File*, *CodeQuality*, *EvaluationMethod*, *ConsistencyRule*, *ConsistencyDelivery*, *Sprint*, *FileExtension*, *StandardizedIssue*, *User*, *ProfessorUser*, *StudentUser*, *Issue* e *PullRequest* responsáveis por armazenar e tratar os dados necessários inseridos na plataforma pelos professores e GitHub. Além disso, contém também classes secundárias *repository_has_contributor*, *issue_has_assigne_student* e *pullrequest_has_assigne_student* necessárias para tratar relações entre as respectivas classes principais. Todas essas classes são tratadas na plataforma durante a execução do sistema, utilizadas para permitir o tratamento das informações das interfaces internas da plataforma referentes as funcionalidades dos professores e interfaces externas da plataforma necessárias para sincronização de dados com o GitHub.

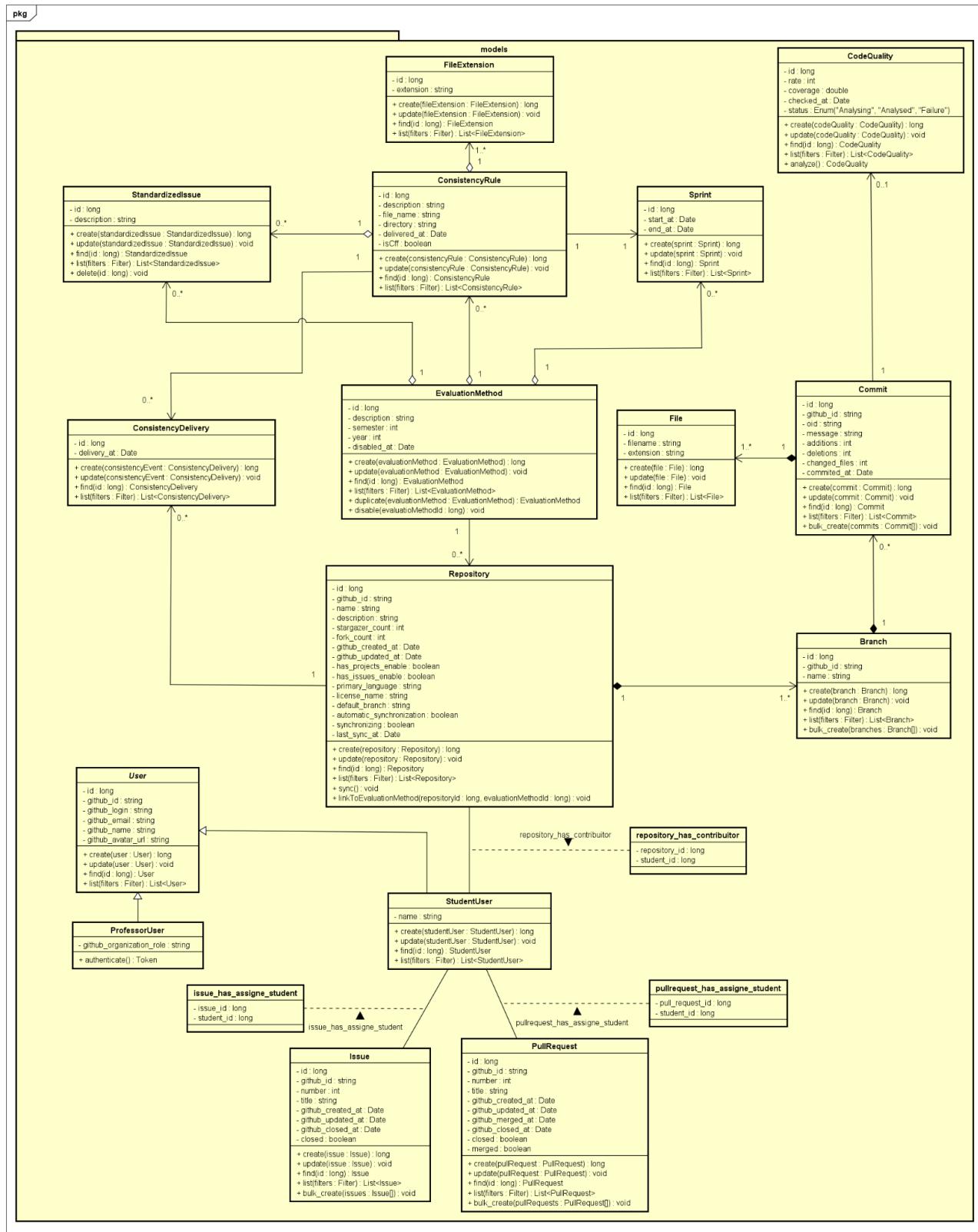


Figura 10. Diagrama de Classes de modelos do sistema

A Figura 11 apresenta o diagrama de classes de serviços do sistema, o qual contém as classes *AuthService*, *RepositoryService*, *EvaluationMethodService*, *ConsistencyRuleService*, *StandardizedIssueService* e *SprintService* responsáveis manipular as funções das classes de modelo do sistema. Esse diagrama representa as classes de *services* invocadas pelas classes de *controllers*, esses serviços manipulam diretamente apenas os dados de modelos internos da plataforma, sem envolver informações de classes do sistema externo GitHub.

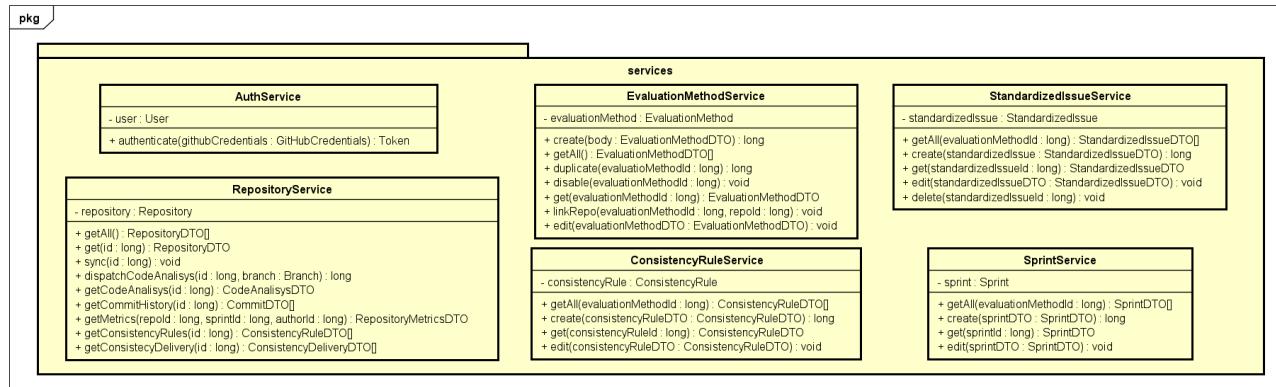


Figura 11. Diagrama de Classes de serviços do sistema

A Figura 12 apresenta o diagrama de classes de controladores do sistema, o qual contém as classes *AuthController*, *RepositoryController*, *EvaluationMethodController*, *ConsistencyRuleController*, *StandardizedIssueController* e *SprintController*. Elas são responsáveis por receber e tratar os dados de requisições do sistema. Esse diagrama representa as classes de *controllers* invocadas por chamadas da API, tratando as mensagens e encaminhando para o *service* correspondente, fazendo também o tratamento do retorno das informações.

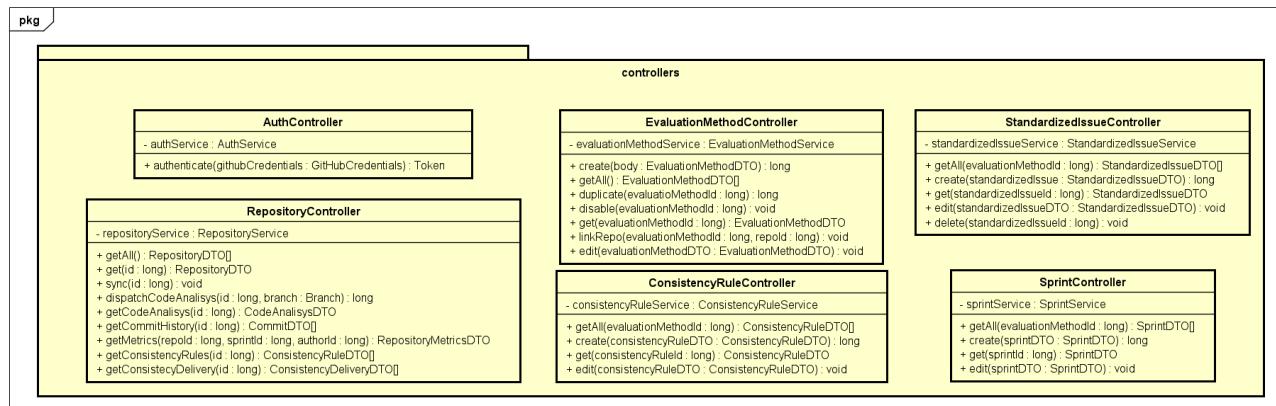


Figura 12. Diagrama de Classes de controladores do sistema

3.2 Diagramas de Sequência

Nesta seção, são apresentados diagramas de sequência referentes aos casos de uso do sistema implementado. Esses diagramas demonstram o fluxo entre os componentes para cada cenário de

uso, dessa forma, eles apresentam como funcionam os fluxos de chamadas entre as entidades que participam de uma interação. As entidades desses diagramas foram apresentadas no diagrama de classes das Figura 10, Figura 11 e Figura 12.

A Figura 13 representa o diagrama de sequência responsável pelo fluxo de autenticação na plataforma. Nesse fluxo, o usuário Professor envia uma mensagem síncrona ao sistema com o objetivo de se autenticar na plataforma utilizando suas credenciais do GitHub. A partir disso, a mensagem chega ao componente de execução *AuthController* que irá tratar os dados da requisição, com isso, enviar para o sistema externo GitHub com intuito de validar as credenciais informadas. Ao sucesso da validação, a autorização é gerada pelo componente de execução *AuthService* e o usuário pode prosseguir utilizando a plataforma. Esse diagrama contempla o caso de uso UC1.

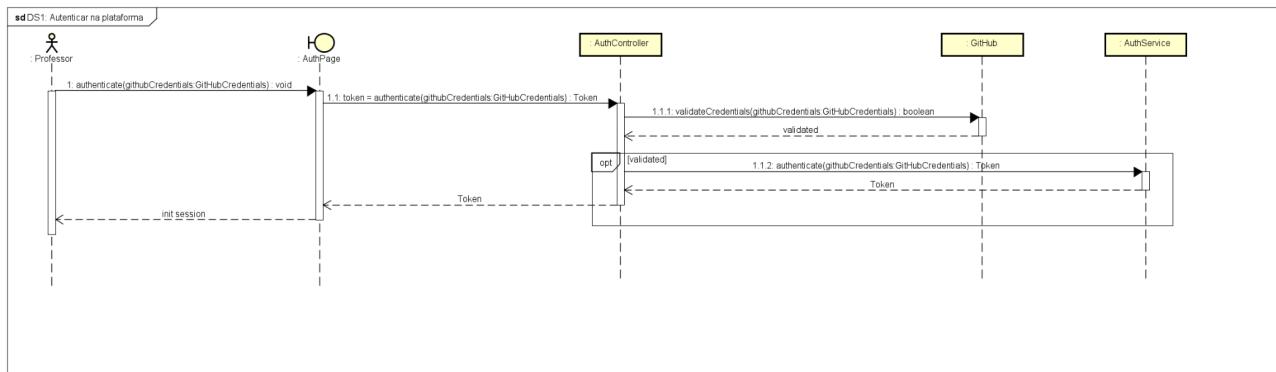


Figura 13. DS 1: Autenticar na plataforma

A Figura 14 representa o diagrama de sequência responsável pelo fluxo de gerenciar métodos avaliativos. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, buscar, listar, desativar, duplicar ou adicionar um repositório a um método avaliativo. A partir disso, as mensagens chegam ao componente de execução *EvaluationMethodController* que trata os dados da requisição e, com isso, enviar para o componente de execução *EvaluationMethodService* que manipula a entidade *EvaluationMethod* dentro do sistema conforme o objetivo requerido pelo usuário. Esse diagrama contempla os casos de uso UC3, UC4, UC5, UC6, UC21, UC33, UC34 e UC35.

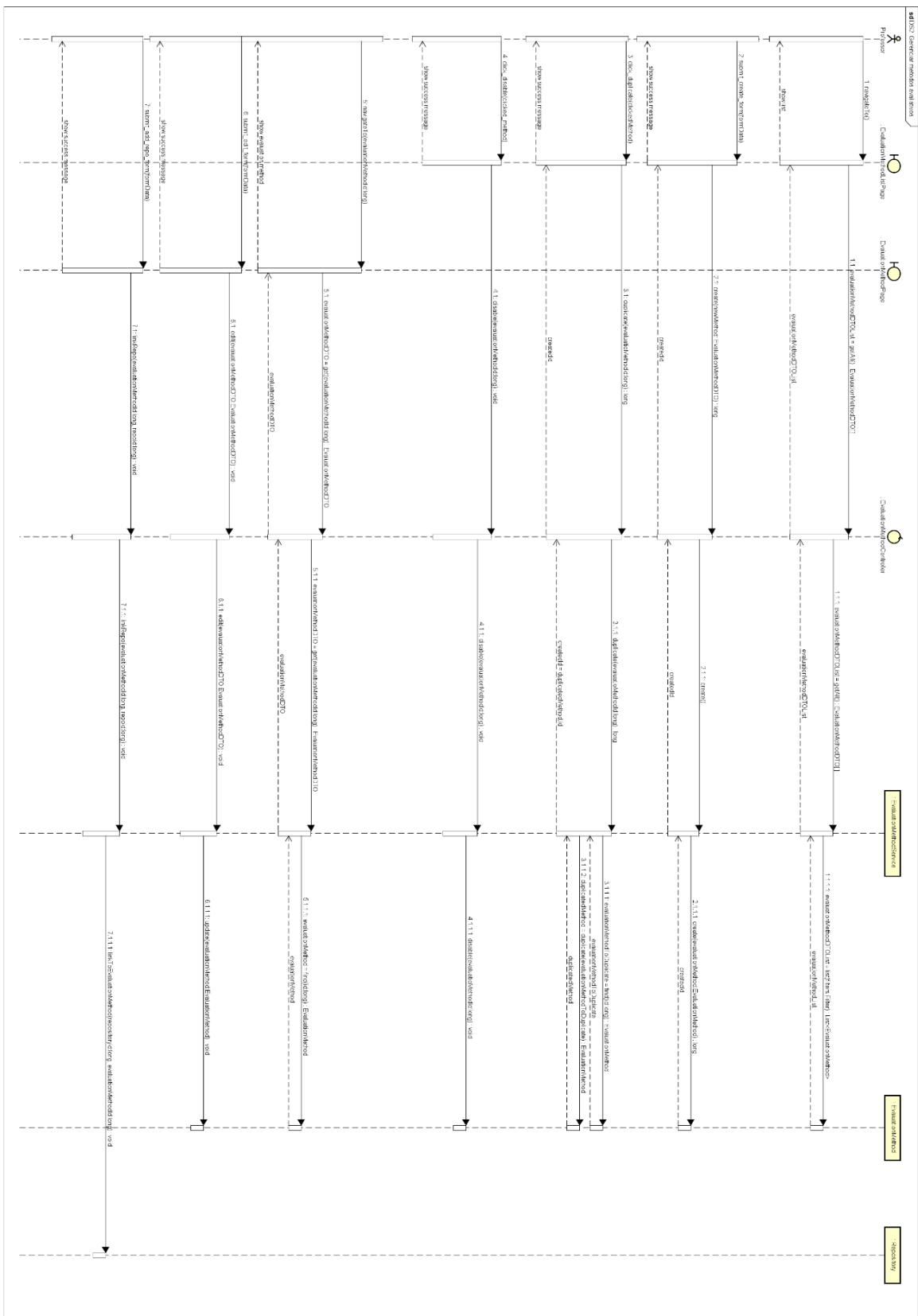


Figura 14. DS 2: Gerenciar métodos avaliativos

A Figura 15 representa o diagrama de sequência responsável pelo fluxo de gerenciar regras de consistência. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, buscar e listar regras de consistência. A partir disso, as mensagens chegam ao componente de execução *ConsistencyRuleController* que trata os dados da requisição e, com isso, enviar para o componente de execução *ConsistencyRuleService* que manipula a entidade *ConsistencyRule* dentro do sistema conforme o objetivo requerido pelo usuário. Esse diagrama contempla os casos de uso UC6, UC7, UC8, UC9, UC22 e UC36.

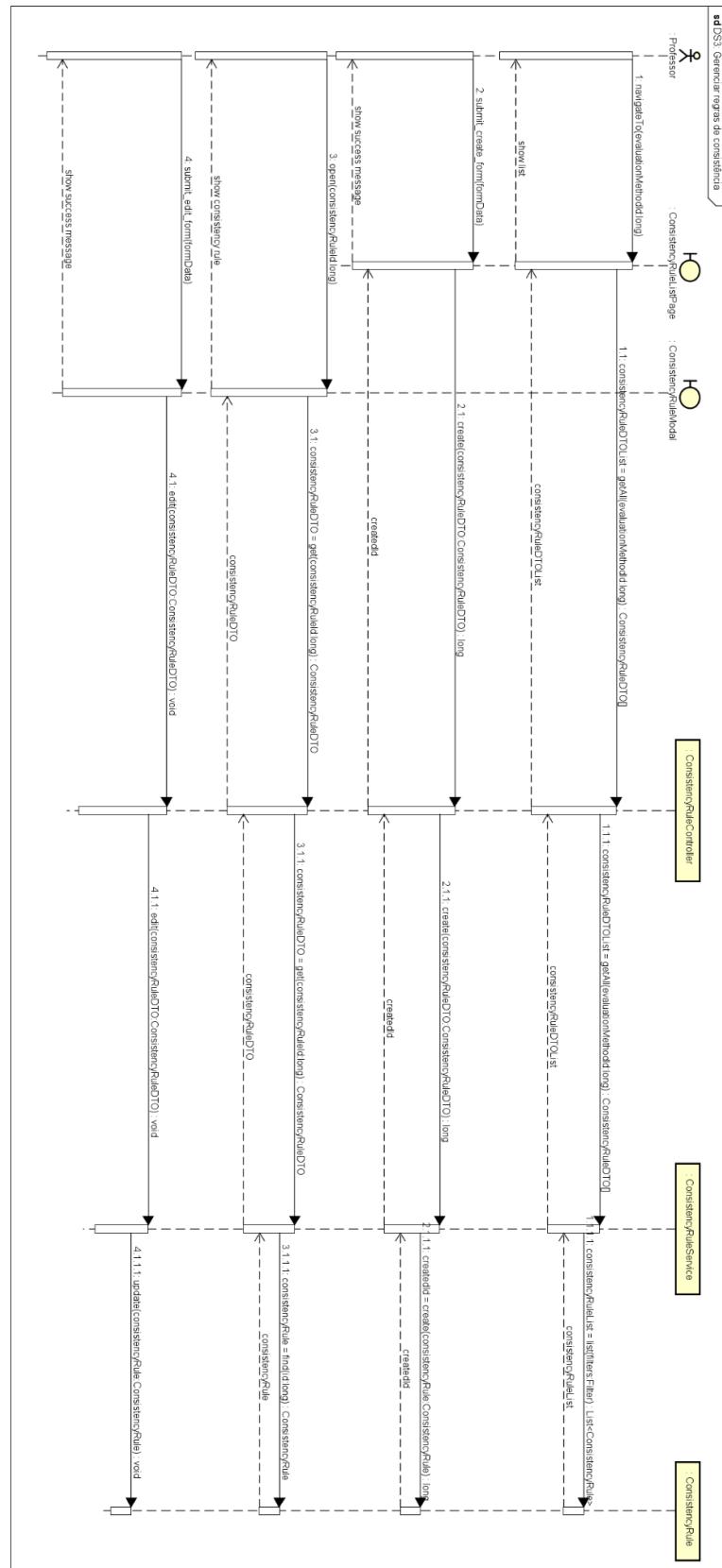


Figura 15. DS 3: Gerenciar regras de consistência

A Figura 16 representa o diagrama de sequência responsável pelo fluxo de gerenciar sprints. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, buscar e listar sprints. A partir disso, as mensagens chegam ao componente de execução *SprintController* que trata os dados da requisição e, com isso, enviar para o componente de execução *SprintService* que manipula a entidade *Sprint* dentro do sistema conforme o objetivo requerido pelo usuário. Esse diagrama contempla o caso de uso UC21, UC22, UC37 e UC38.

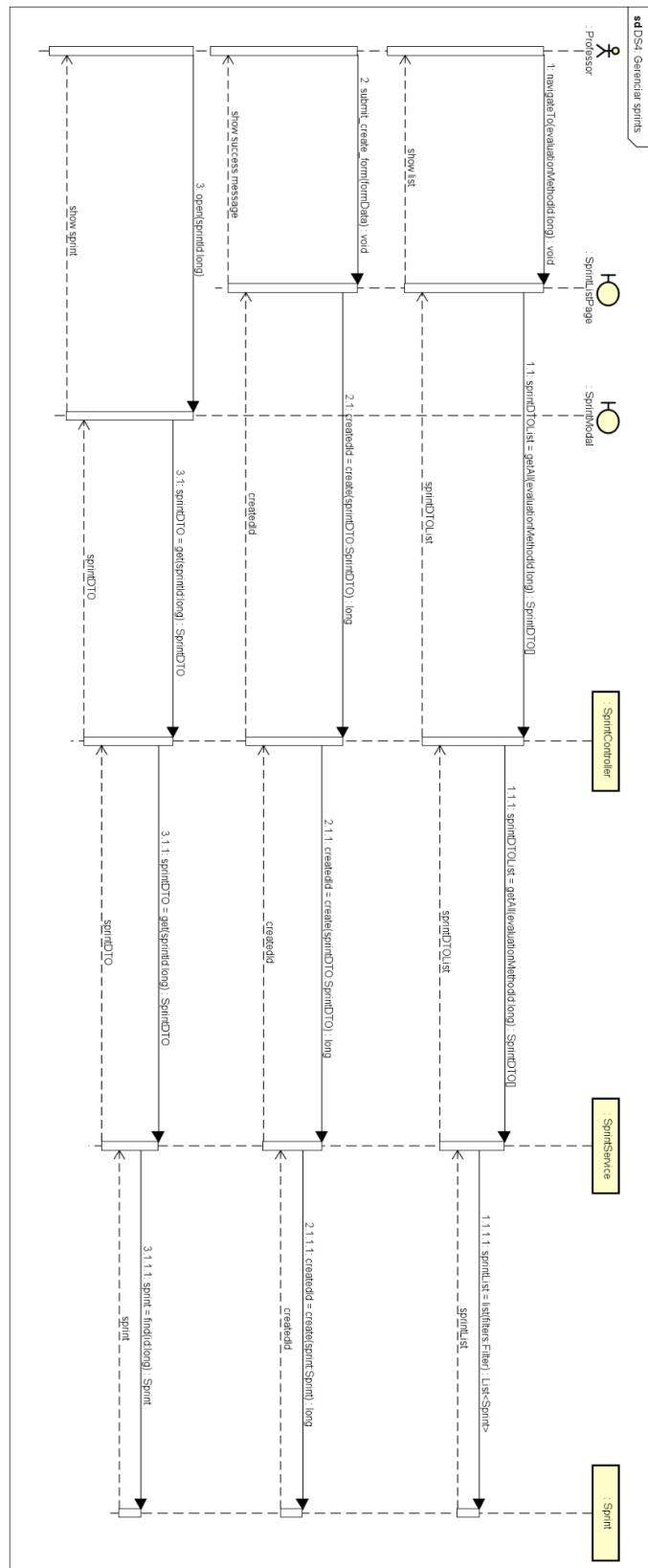


Figura 16. DS 4: Gerenciar sprints

A Figura 17 representa o diagrama de sequência responsável pelo fluxo de gerenciar *issues* padronizadas. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de criar, atualizar, deletar, buscar e listar *issues* padronizadas. A partir disso, as mensagens chegam ao componente de execução *StandardizedIssueController* que trata os dados da requisição e, com isso, enviar para o componente de execução *StandardizedIssueService* que manipula a entidade *StandardizedIssue* dentro do sistema conforme o objetivo requerido pelo usuário. Esse diagrama contempla os casos de uso UC10, UC39, UC40, UC41 e UC42.

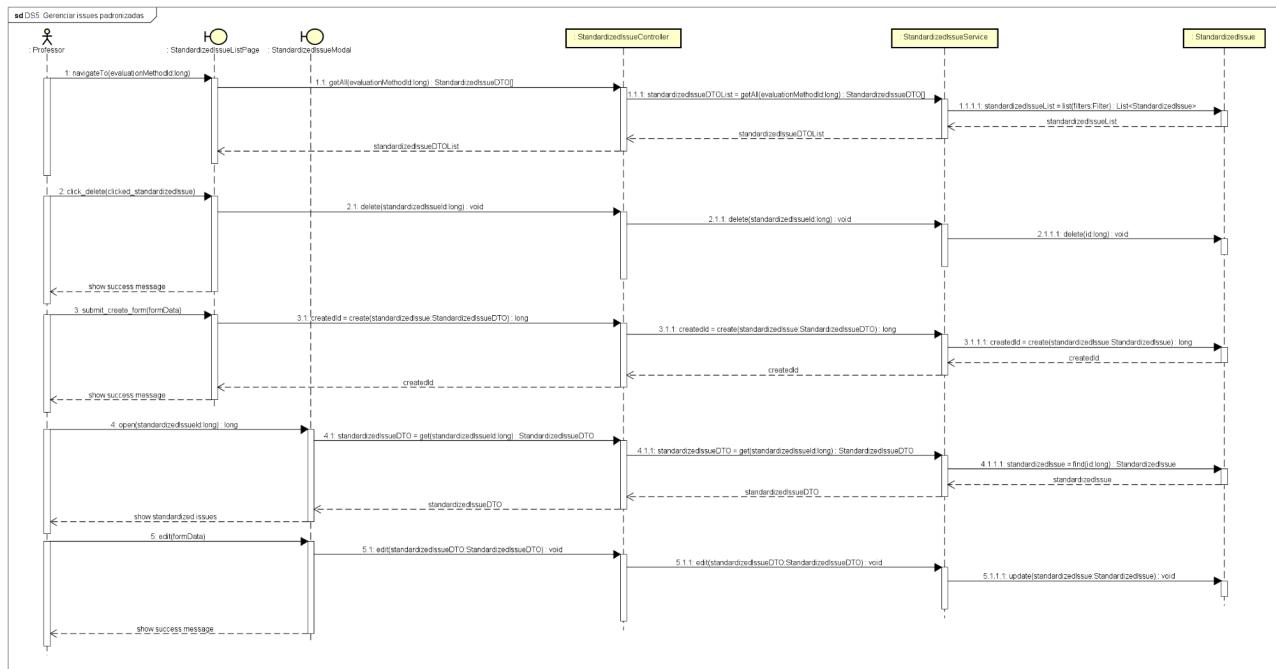


Figura 17. DS 5: Gerenciar *issues* padronizadas

A Figura 18 representa o diagrama de sequência responsável pelo fluxo de gerenciar repositórios de trabalhos. Nesse fluxo, o usuário Professor envia mensagens síncronas ao sistema com o objetivo de sincronizar, buscar, listar, atualizar, calcular métricas e qualidade do código, buscar histórico de commits e consistência das entregas de um repositório de trabalho. A partir disso, as mensagens chegam ao componente de execução *RepositoryController* que trata os dados da requisição e, com isso, enviar para o componente de execução *RepositoryService* que manipula a entidade *Repository* dentro do sistema conforme o objetivo requerido pelo usuário. Esse diagrama contempla o caso de uso UC2, UC11, UC12, UC24, UC25, UC26, UC29, UC31 e UC32.

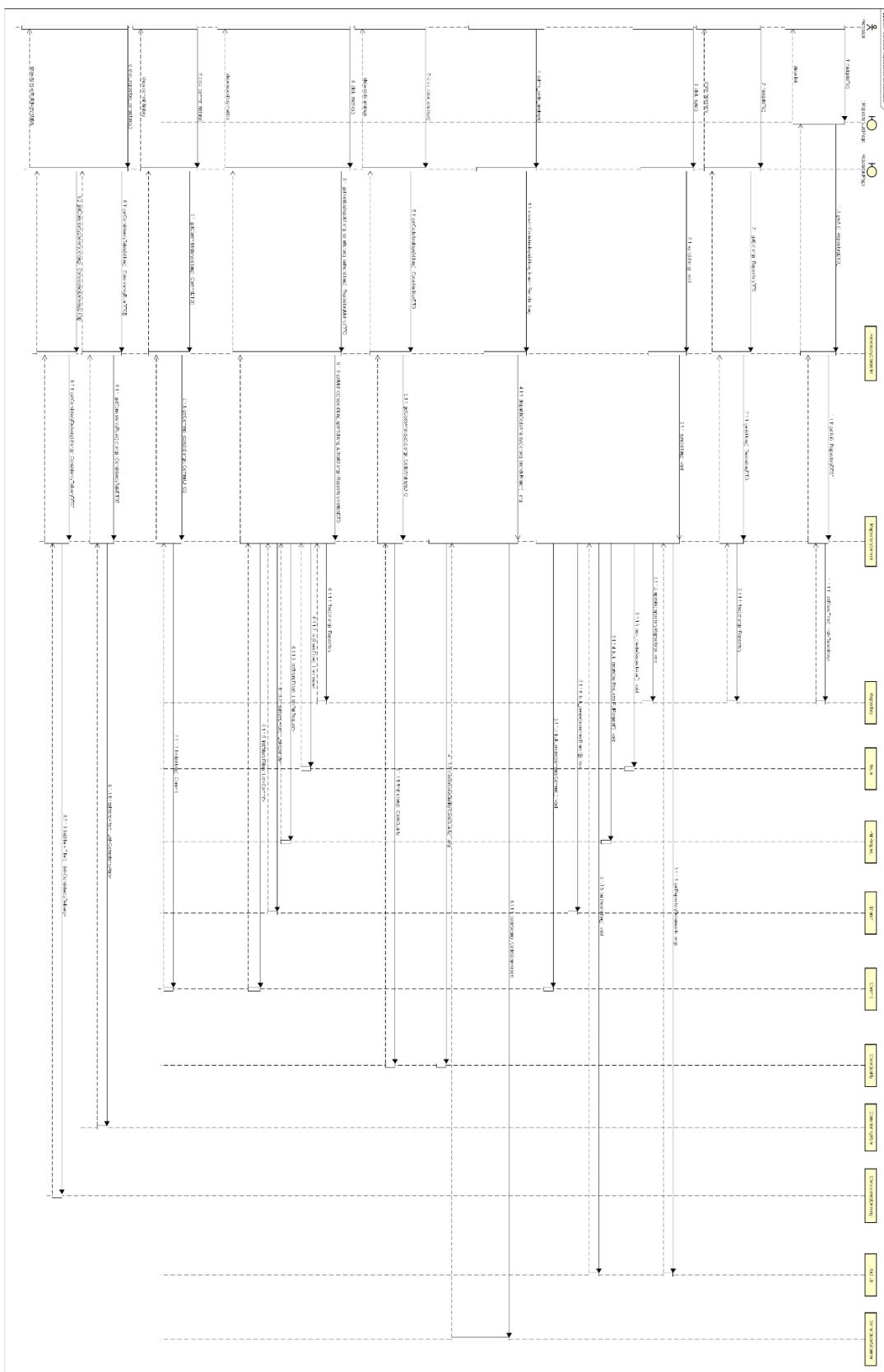


Figura 18. DS6: Gerenciar repositórios de trabalhos

3.3 Diagramas de Comunicação

Nesta seção, são apresentados diagramas de comunicação que modelam as trocas de mensagens dos casos de uso do sistema implementado. Esses diagramas representam as interações entre componentes do sistema para realizar uma determinada funcionalidade. Dessa forma, os principais fluxos da aplicação são representados como meio de documentar as comunicações que os compõem.

A Figura 19 representa o diagrama de comunicação responsável pelo fluxo de autenticar na plataforma. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *AuthPage* com o objetivo de autenticar-se na plataforma por meio de uma autorização. Diante disso, a comunicação é recebida pelo *AuthController* que irá se comunicar com sistema externo GitHub com intuito de validar as credenciais informadas. Ao sucesso da validação, a autorização é por uma solicitação feita ao *AuthService*, assim, permitindo o usuário *ProfessorUser* continuar a utilização da plataforma. Esse diagrama contempla o caso de uso UC1.

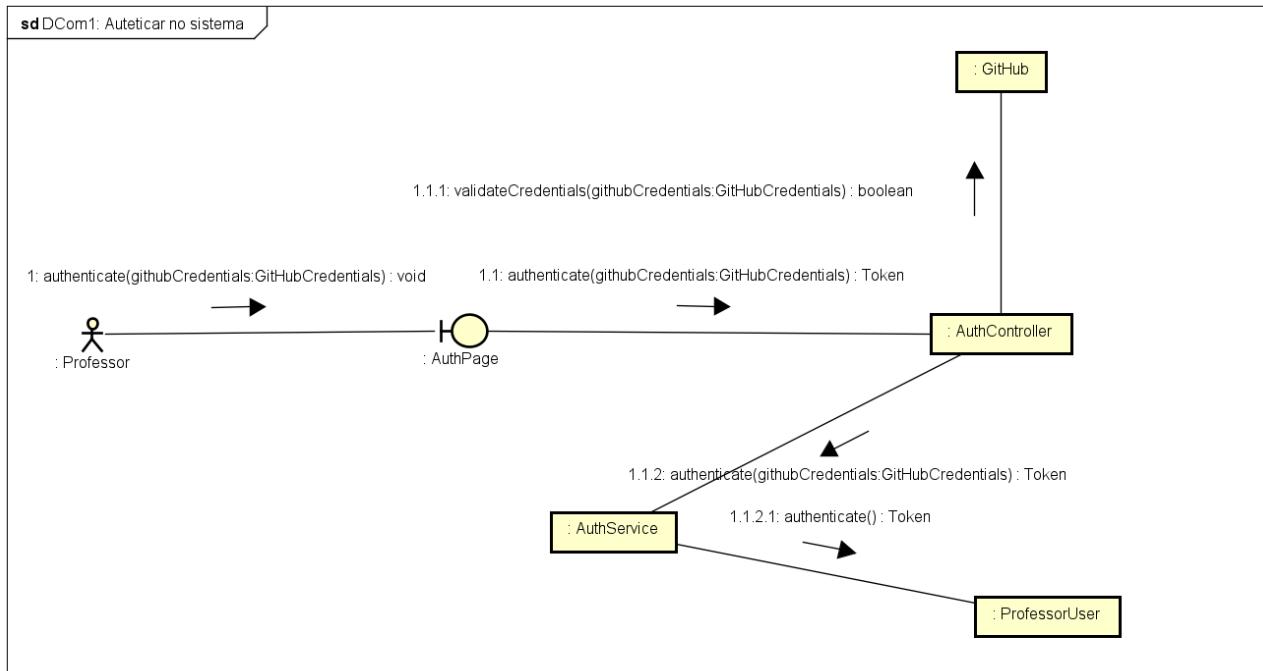


Figura 19. DCom 1: Autenticar na plataforma

A Figura 20 representa o diagrama de comunicação responsável pelo fluxo de gerenciar métodos avaliativos. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *EvaluationMethodListPage* ou *EvaluationMethodPage* com o objetivo de criar, atualizar, buscar, listar, desativar, duplicar ou adicionar um repositório a um método avaliativo. Diante disso, a comunicação é recebida pelo *EvaluationMethodController* que irá validar os dados e detectar qual ação que deve ser realizada. Ao definir qual ação, a comunicação com o *EvaluationMethodService* é iniciada, assim, executando a ação requerida pelo Professor. Esse diagrama contempla os casos de uso UC3, UC4, UC5, UC6, UC21, UC33, UC34 e UC35.

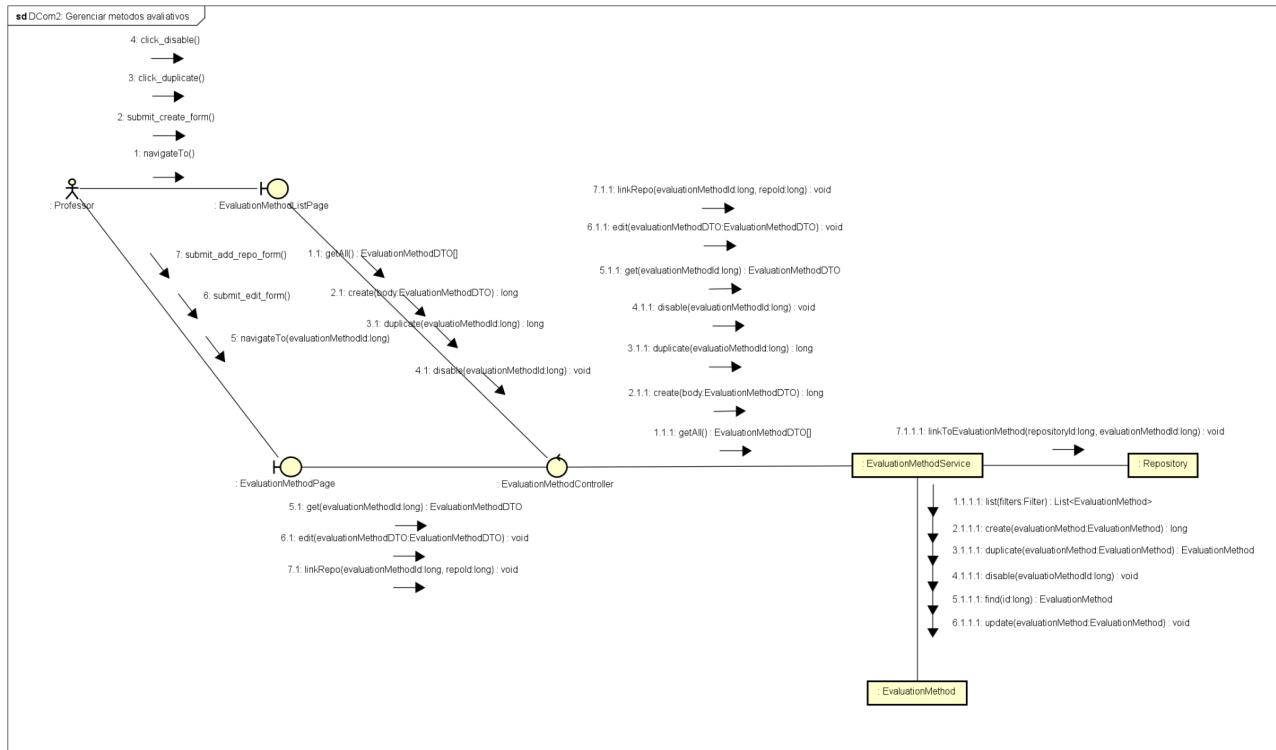


Figura 20. DCom 2: Gerenciar métodos avaliativos

A Figura 21 representa o diagrama de comunicação responsável pelo fluxo de gerenciar regras de consistência. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *ConsistencyRuleListPage* ou *ConsistencyRuleModal* com o objetivo de criar, atualizar, buscar e listar regras de consistência. Diante disso, a comunicação é recebida pelo *ConsistencyRuleController* que irá validar os dados e detectar qual ação que deve ser realizada. Ao definir qual ação, a comunicação com o *ConsistencyRuleService* é iniciada, assim, executando a ação requerida pelo Professor. Esse diagrama contempla os casos de uso UC6, UC7, UC8, UC9, UC22 e UC36.

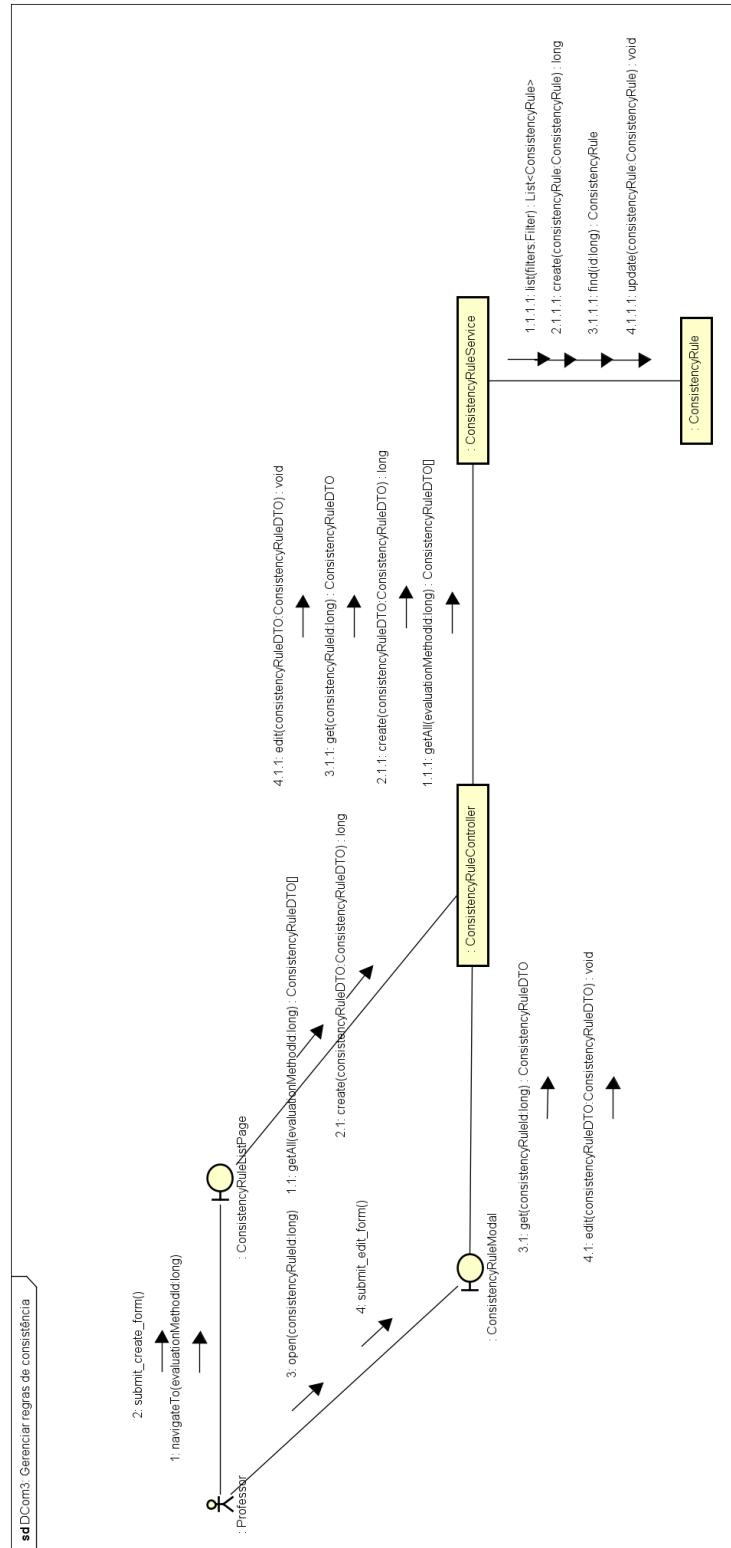


Figura 21. DCom 3: Gerenciar regras de consistência

A Figura 22 representa o diagrama de comunicação responsável pelo fluxo de gerenciar sprints. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *SprintListPage* ou *SprintModal* com o objetivo de criar, buscar e listar sprints. Diante disso, a comunicação é recebida pelo *SprintController* que irá validar os dados e detectar qual ação que deve ser realizada. Ao definir qual a ação, a comunicação com o *SprintService* é iniciada, assim, executando a ação requerida pelo Professor. Esse diagrama contempla os casos de uso UC21, UC22, UC37 e UC38.

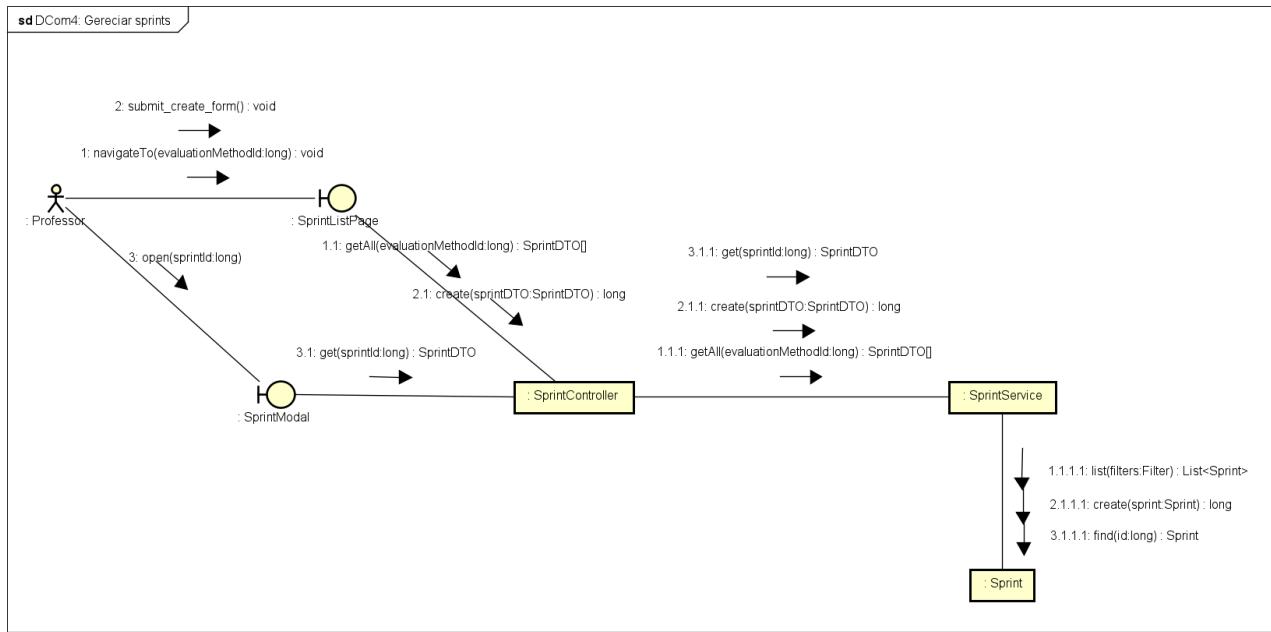
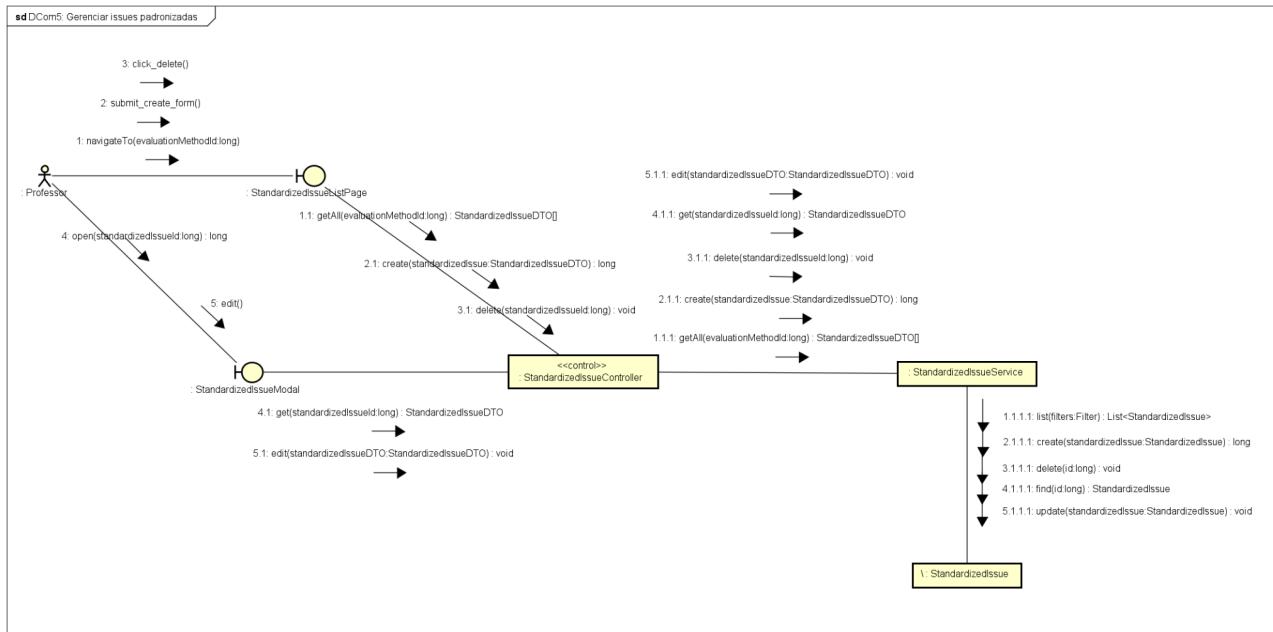


Figura 22. DCom 4: Gerenciar sprints

A Figura 23 representa o diagrama de comunicação responsável pelo fluxo de gerenciar *issues* padronizadas. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *StandardizedIssueListPage* ou *StandardizedIssueModal* com o objetivo de criar, atualizar, deletar, buscar e listar *issues* padronizadas. Diante disso, a comunicação é recebida pelo *StandardizedIssueController* que irá validar os dados e detectar qual ação que deve ser realizada. Ao definir qual a ação, a comunicação com o *StandardizedIssueService* é iniciada, assim, executando a ação requerida pelo Professor. Esse diagrama contempla os casos de uso UC10, UC39, UC40, UC41 e UC42.

Figura 23. DCom 5: Gerenciar *issues* padronizadas

A Figura 24 representa o diagrama de comunicação responsável pelo fluxo de gerenciar repositórios de trabalhos. Nesse fluxo, o usuário Professor inicia a comunicação com o sistema pela *RepositoryListPage* ou *RepositoryPage* com o objetivo de criar, atualizar, buscar e listar regras de consistência. Diante disso, a comunicação é recebida pelo *RepositoryController* que irá validar os dados e detectar qual ação que deve ser realizada. Ao definir qual ação, a comunicação com o *RepositoryService* é iniciada, assim, executando a ação requerida pelo Professor. Esse diagrama contempla os casos de uso UC2, UC11, UC12, UC24, UC25, UC26, UC29, UC31 e UC32.

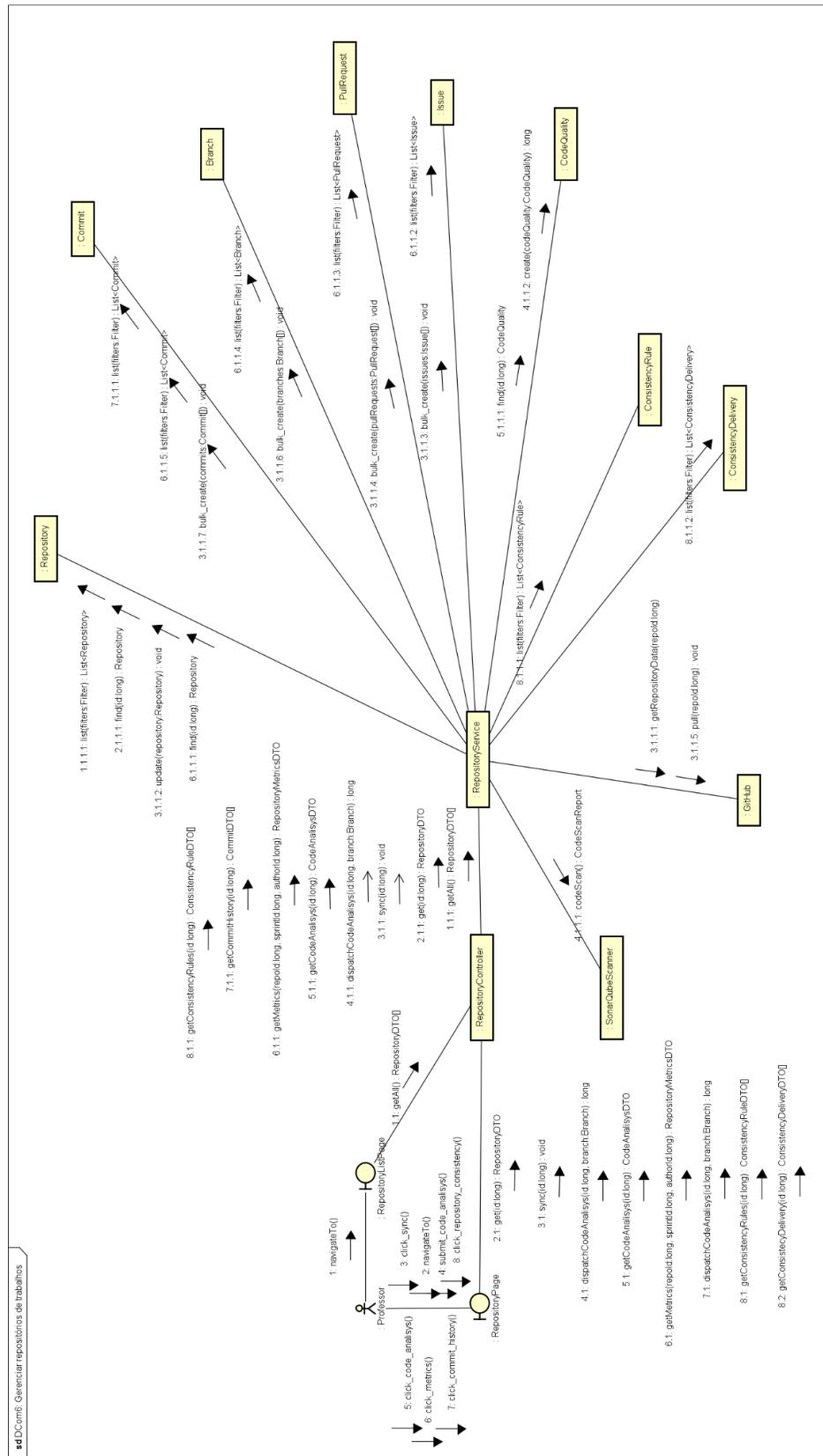


Figura 24. DCom 6: Gerenciar repositórios de trabalhos

3.4 Arquitetura

Nesta seção, é apresentada a modelagem da arquitetura lógica do sistema. Essa modelagem foi feita por meio do diagrama de pacotes, o qual descreve a estrutura organizacional do sistema, mostrando como os elementos do sistema estão organizados e hierarquizados em pacotes e como esses pacotes se relacionam entre si.

A Figura 25 representa o diagrama de pacotes da plataforma, que possui o pacote DockerCompose o qual contém os quatro principais pacotes do sistema, sendo eles o SupportPlataformAPI, View, SonarQube e JobScheduler. O DockerCompose refere-se ao pacote da ferramenta de administração de contêineres internos da plataforma. Já o pacote SupportPlataformAPI é o responsável pelas funções de manipulação dos dados dentro da plataforma, esse pacote depende do pacote SonarQube para efetuar as operações de análise estática de código. O pacote View possui as interfaces que os usuários finais utilizam, sendo dependente direto do SupportPlataformAPI para manipular os dados. Por fim, o pacote JobScheduler é responsável pela sincronização da base de dados com as informações provenientes do GitHub, também sendo dependente direto do SupportPlataformAPI para manipular os dados.

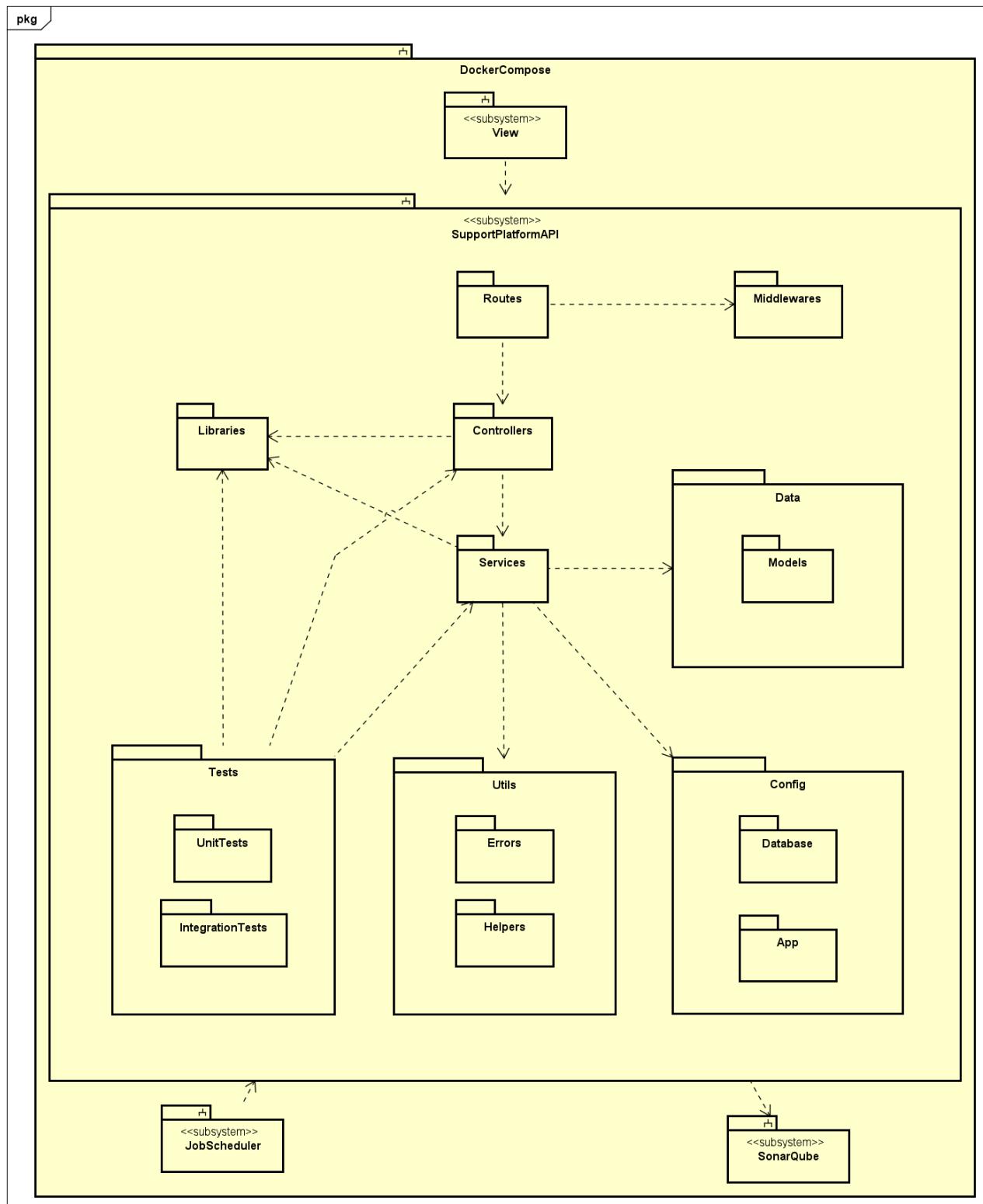


Figura 25. Diagrama de Pacotes do sistema

3.5 Diagramas de Estados

Nesta seção, são apresentados os diagramas de estados do sistema implementado. Esses diagramas modelam os possíveis estados que um objeto de uma determinada entidade pode estar, e especificam as transições entre eles.

A Figura 26 apresenta o diagrama de estados dos objetos da entidade *Code Quality*. Nesse diagrama, é apresentado que o objeto se inicia com o *status* “analisando”. Assim que a análise de qualidade de código é finalizada, seu *status* se altera para “analisado”. Caso ocorra algum erro durante a análise, o *status* passa a ser “falhou”. Nos dois últimos casos, esse é o estado final que o objeto pode assumir, não havendo nenhuma transição possível uma vez que ele chega em um desses estados.

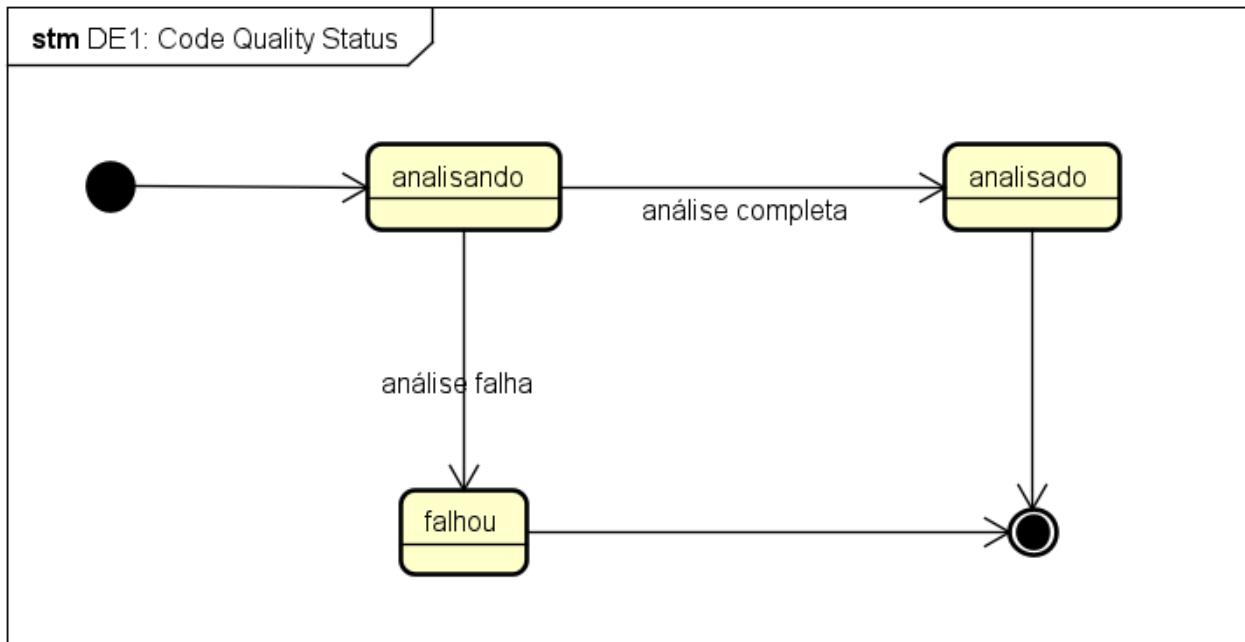
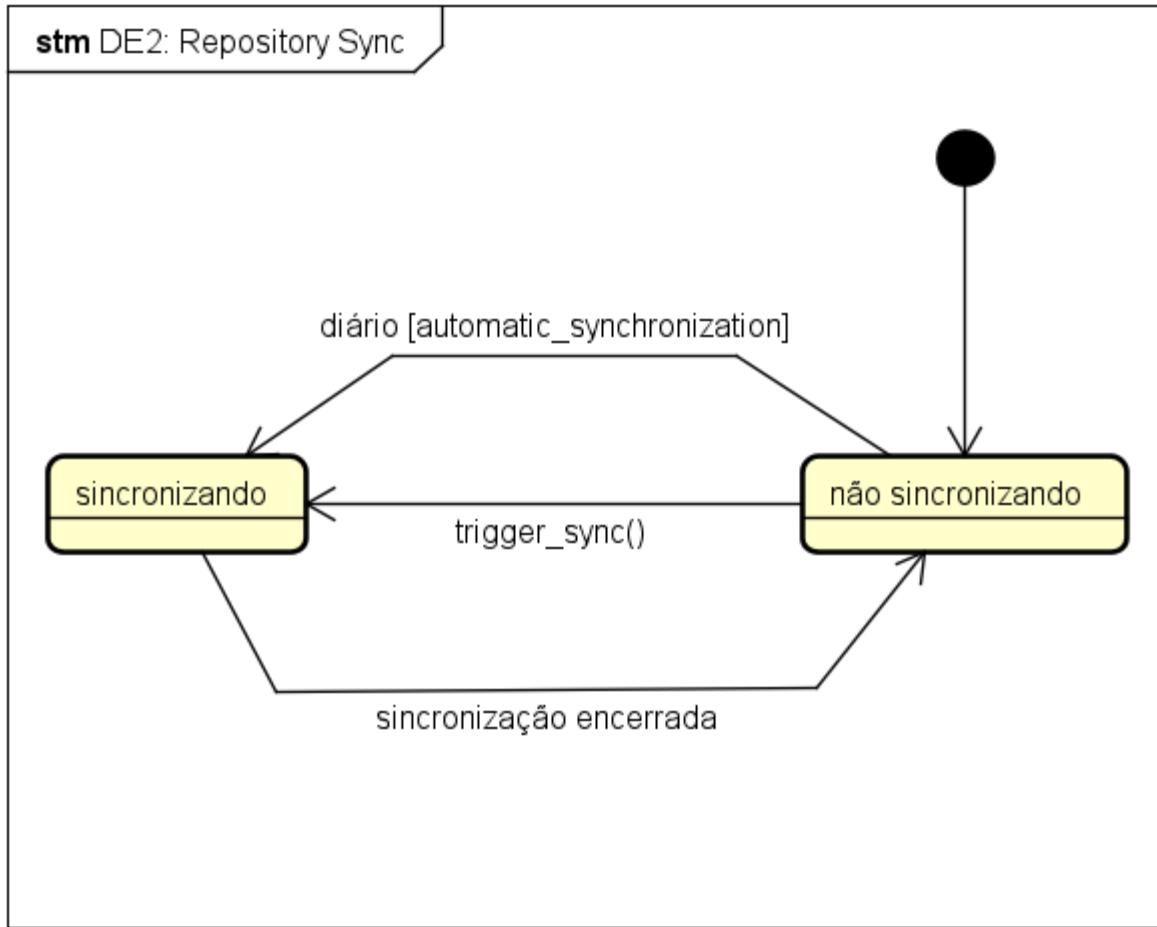


Figura 26. Diagrama de Estados da Entidade *Code Quality*

A Figura 27 apresenta o diagrama de estados dos objetos da entidade *Repository*. Nesse diagrama, é apresentado que o objeto inicia com a propriedade *sync* como “não sincronizando”. Há dois eventos que provocam a transição para o estado “sincronizando”. O primeiro é o disparo manual da função “*trigger_sync*”, e o segundo ocorre uma vez por dia, caso o atributo “*automatic_synchronization*” seja verdadeiro. Uma vez que se encontra no estado “sincronizando”, ele retorna ao estado “não sincronizando” assim que a sincronização é encerrada.

Figura 27. Diagrama de Estados da Entidade *Repository*

3.6 Diagrama de Componentes e Implantação

Nesta seção são apresentados os diagramas de componentes e implantação da plataforma. O diagrama de componentes modela os principais componentes do sistema, apontando suas composições e dependências, bem como as interfaces que eles consomem e que eles provedem. Enquanto isso, o diagrama de implantação representa os recursos físicos e de software necessários para o sistema ser implantado adequadamente.

Na Figura 28 é apresentado o diagrama de componentes da plataforma. O principal componente apresentado é o Web Server, que se relaciona com todos os outros componentes. Ele é o componente que provê a API para a aplicação web, comunica-se com o banco de dados, com a API externa do GitHub, além de requisitar à instância do SonarQube e ser chamado pelo JobScheduler, que agenda rotinas de execução.

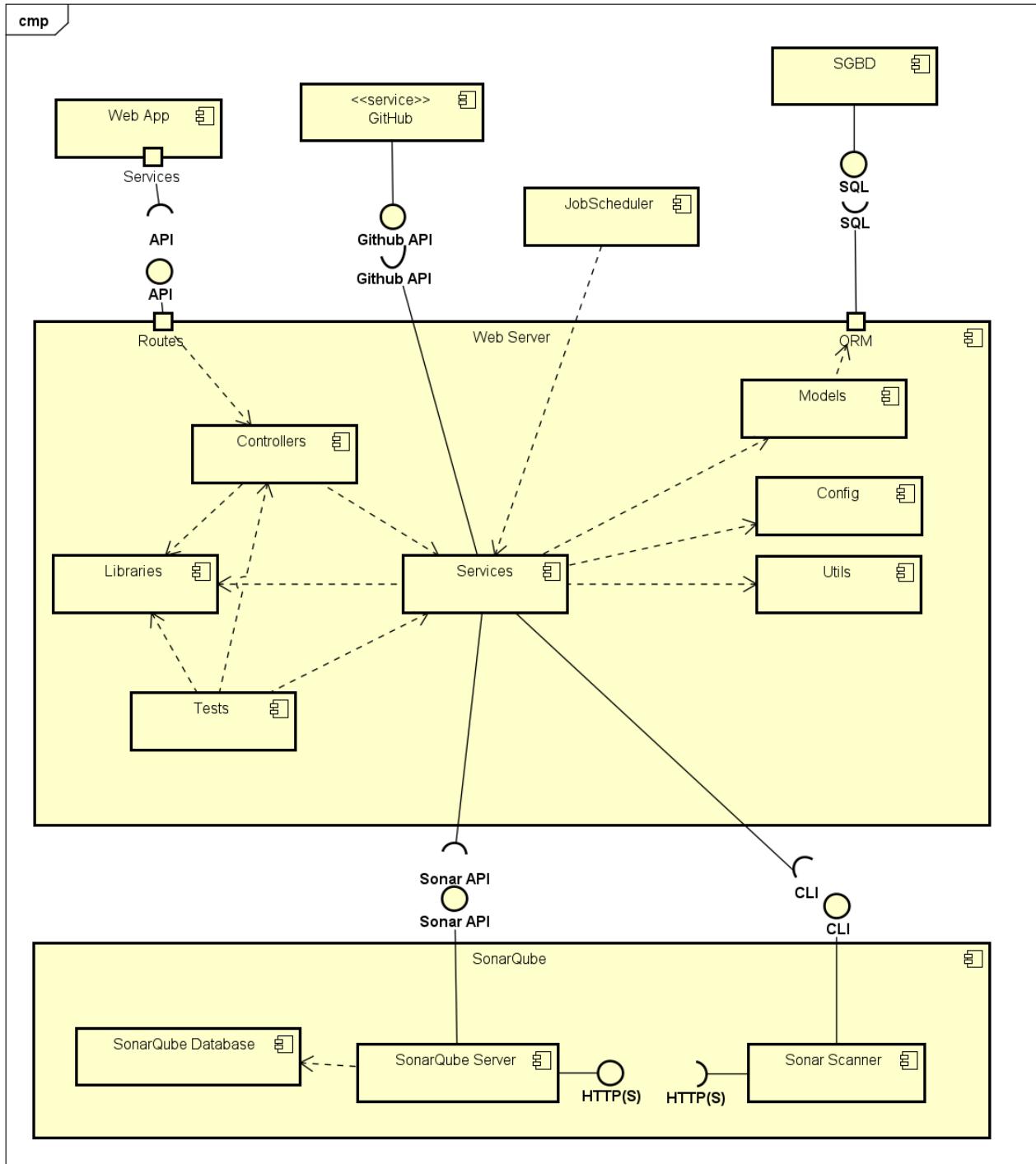


Figura 28. Diagrama de Componentes

Na Figura 29 é apresentado o diagrama de implantação da plataforma. Nele, são apresentados os nós de processamento para implantação e uso da mesma. A aplicação web, executada usando um *web browser* na máquina cliente, irá baixar os arquivos estáticos do Web Server e fazer requisições à Application Server, através do protocolo HTTPS. Os dois servidores estão em máquinas isoladas

um do outro. Outras instâncias necessárias são o SonarQube e o banco de dados. Essas também são isoladas, e ambas são requisitadas pelo Application Server através do protocolo TCP/IP.

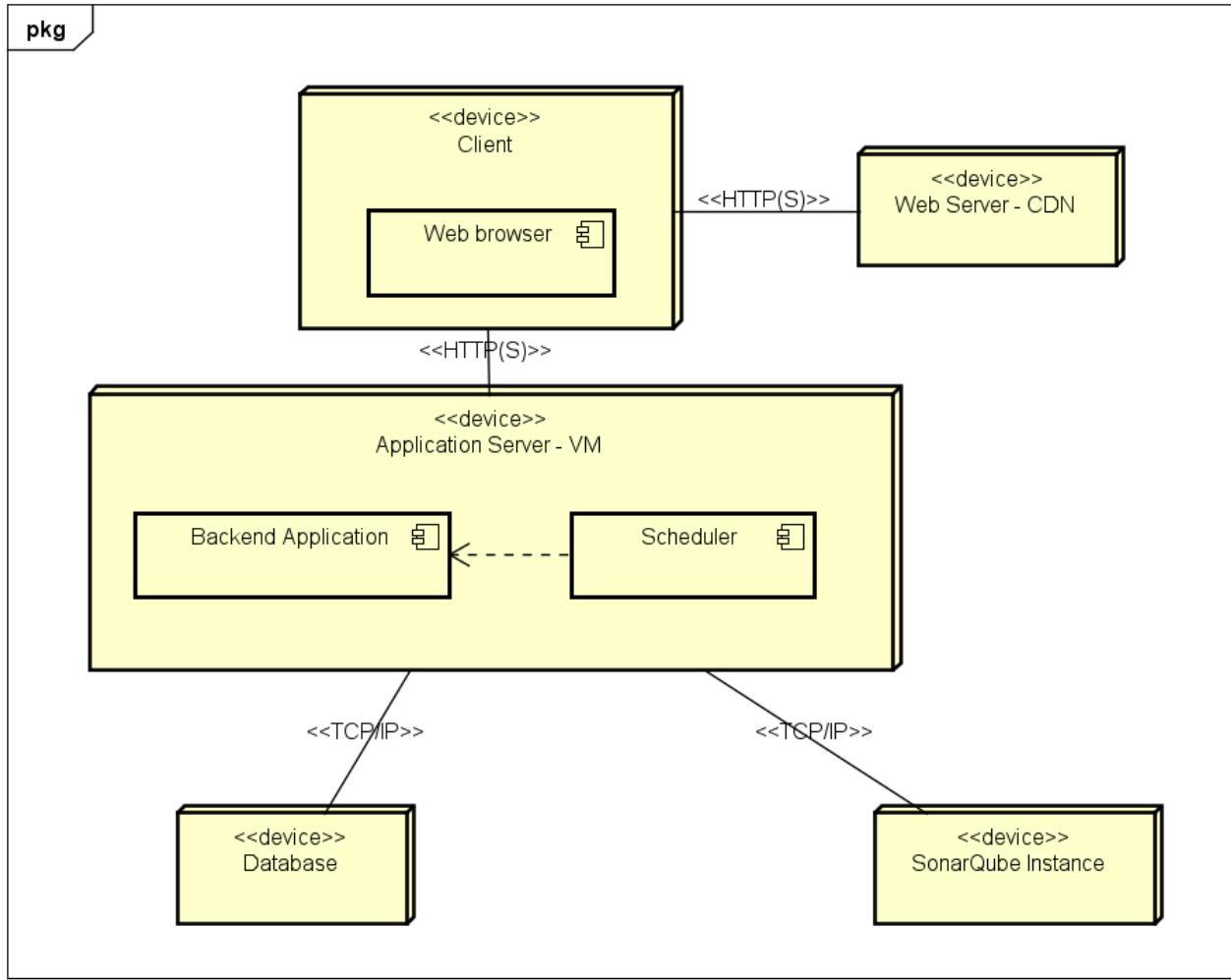


Figura 29. Diagrama de Implantação

4. Projeto de Interface com Usuário

Esta seção visa apresentar e descrever as principais interfaces de usuário da plataforma desenvolvida neste trabalho. Para isso, foram desenvolvidos *mockups* de alta fidelidade por meio da ferramenta Figma³ e as bibliotecas de *design* Primer⁴ e ChartJs⁵. Essas bibliotecas foram escolhidas, por permitirem a criação de elementos gráficos semelhantes ao GitHub. Diante disso, as interfaces

³ <https://www.figma.com/>

⁴ <https://primer.style/>

⁵ <https://www.chartjs.org/>

foram projetadas relacionando-as com os casos de uso definidos na Seção 2.3.1 com objetivo de mapear todas as funcionalidades necessárias para cumprimento dos requisitos especificados.

A Figura 30 representa a tela de *login* para usuário não autenticadas na aplicação. Essa tela redireciona o professor para a tela da Figura 31 após uma autenticação com sucesso utilizando as credenciais do GitHub. Essa interface contempla o caso de uso UC1.

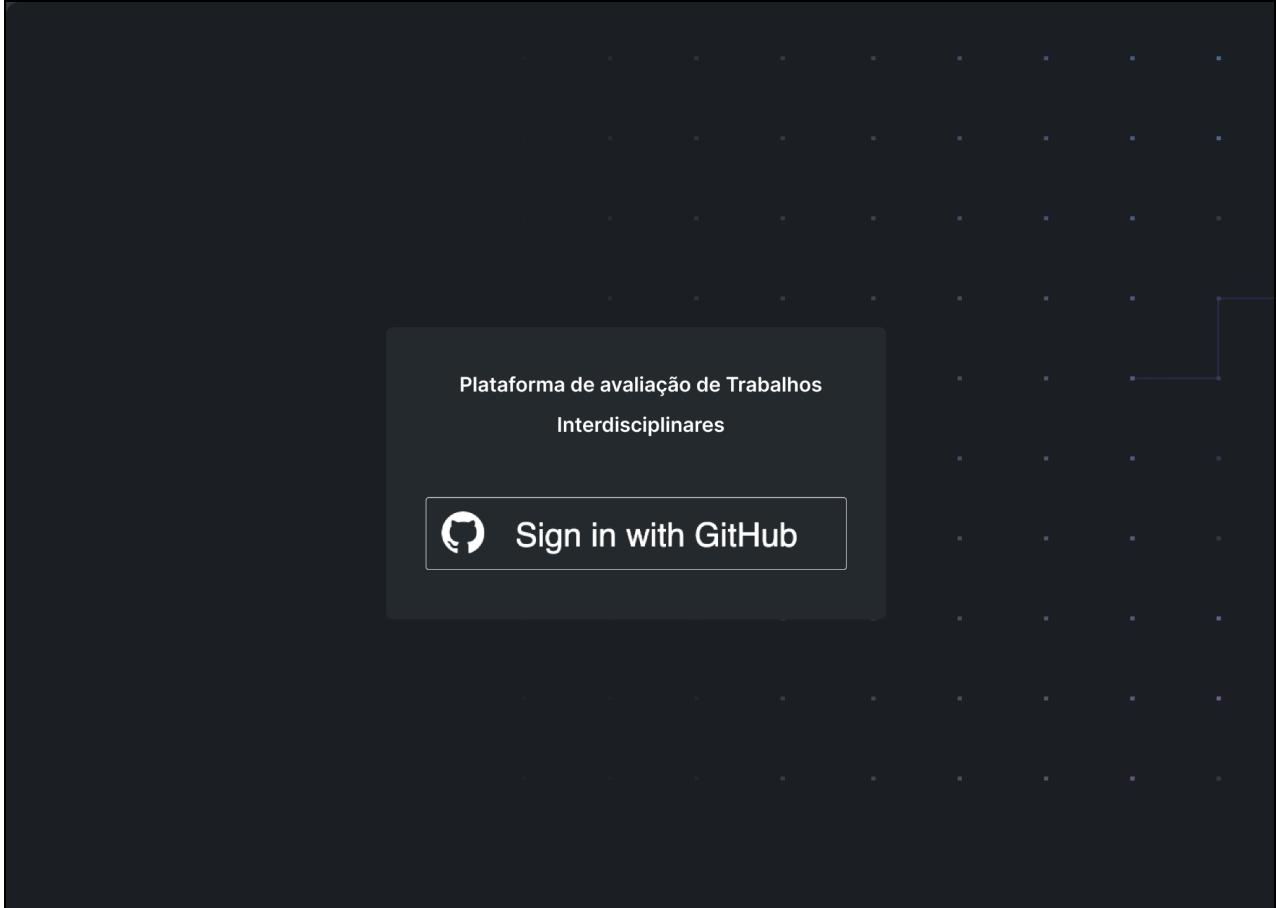


Figura 30. Tela de *login* na plataforma

A Figura 31 é a página inicial da plataforma que será aberta para os professores logo após a autenticação do usuário. Essa e todas as páginas do sistema possuem um *header* padrão com o ícone da plataforma, uma barra de pesquisa para buscar repositório de um trabalho pelo nome e os botões para visualizar a lista de repositórios de trabalhos e lista de métodos avaliativos, também possui um *footer* padrão com o link do repositório do GitHub deste TCC. Além disso, nessa página há a listagem de métodos avaliativos cadastrados na plataforma, sendo possível filtrar pelo nome do método avaliativo, abri-los e duplicá-los para facilitar na reutilização das configurações em vários semestres letivos. Essa interface contempla os casos de uso UC3, UC33 e UC35.

Buscar trabalho... Repositórios de trabalhos Métodos Avaliativos

Filtrar Q

Duplicar Editar

Anterior 1 2 3 4 Próxima Repositório

Figura 31. Tela de listagem de métodos avaliativos

A Figura 32 representa a página de gerenciamento dos repositórios de um método avaliativo. Nessa página é possível adicionar e filtrar quais repositórios estão sendo avaliados por esse método e duplicar o método avaliativo. Além disso, é possível navegar na tela para visualizar as regras de consistência, sprints e *issues* padronizadas do método avaliativo que está sendo gerenciado. Essa interface contempla os caso de usos UC5, UC34 e UC35.

Método Avaliativo: Trabalho Interdisciplinar1 - 01/2023

Duplicar

Repositórios

Regras de consistência Sprints Issues padronizadas

Filtrar

nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep nome-do-rep

Sincronizar Abrir Sincronizar Abrir

1 2 3 4 Próxima >

Figura 32. Tela dos repositórios de um método avaliativo

A Figura 33 representa a página de gerenciamento de um método avaliativo, mais especificamente o gerenciamento de suas regras de consistência. Nessa página é possível ver, editar e cadastrar as regras de consistência do método avaliativo. Essa interface contempla os casos de uso UC6, UC7, UC8, UC9 e UC36.

Método Avaliativo: Trabalho Interdisciplinar1 – 01/2023

Regras de consistência

Criar regra de consistência

Repositórios

Regras de consistência

Sprints

Issues padronizadas

Filtrar

citation.cff

Apresentacao/video.mp4

Documentacao/arquivo_de_arquitetura.md

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

diretório/arquivo.extensão

Sprint 1

Editor

< Anterior

1

2

3

4

Próxima >

© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub.

Repositório

Figura 33. Tela das regras de consistência de um método avaliativo

A Figura 34 apresenta o *modal* de cadastro de uma nova regra de consistência em um método avaliativo na plataforma. Esse *modal* será aberto ao clicar no botão “Criar regra de consistência” presente na Figura 33. Nesse modal é necessário inserir o diretório e nome do arquivo que será validado, a sprint de entrega desse arquivo, a *issue* padronizada caso a regra de consistência tenha sido quebrada e quais as possíveis extensões de arquivos aceitas por essa regra, sendo válido observar que caso tenha a extensão “cff” aparece o alerta que será validado a estrutura de arquivo de citação. Além disso, o *modal* dessa tela possui dois botões para melhorar a usabilidade, sendo eles para criar sprints e issues padronizadas sem ser necessário trocar de tela. Essa interface contempla os casos de uso UC9, UC10, UC22 e UC29.

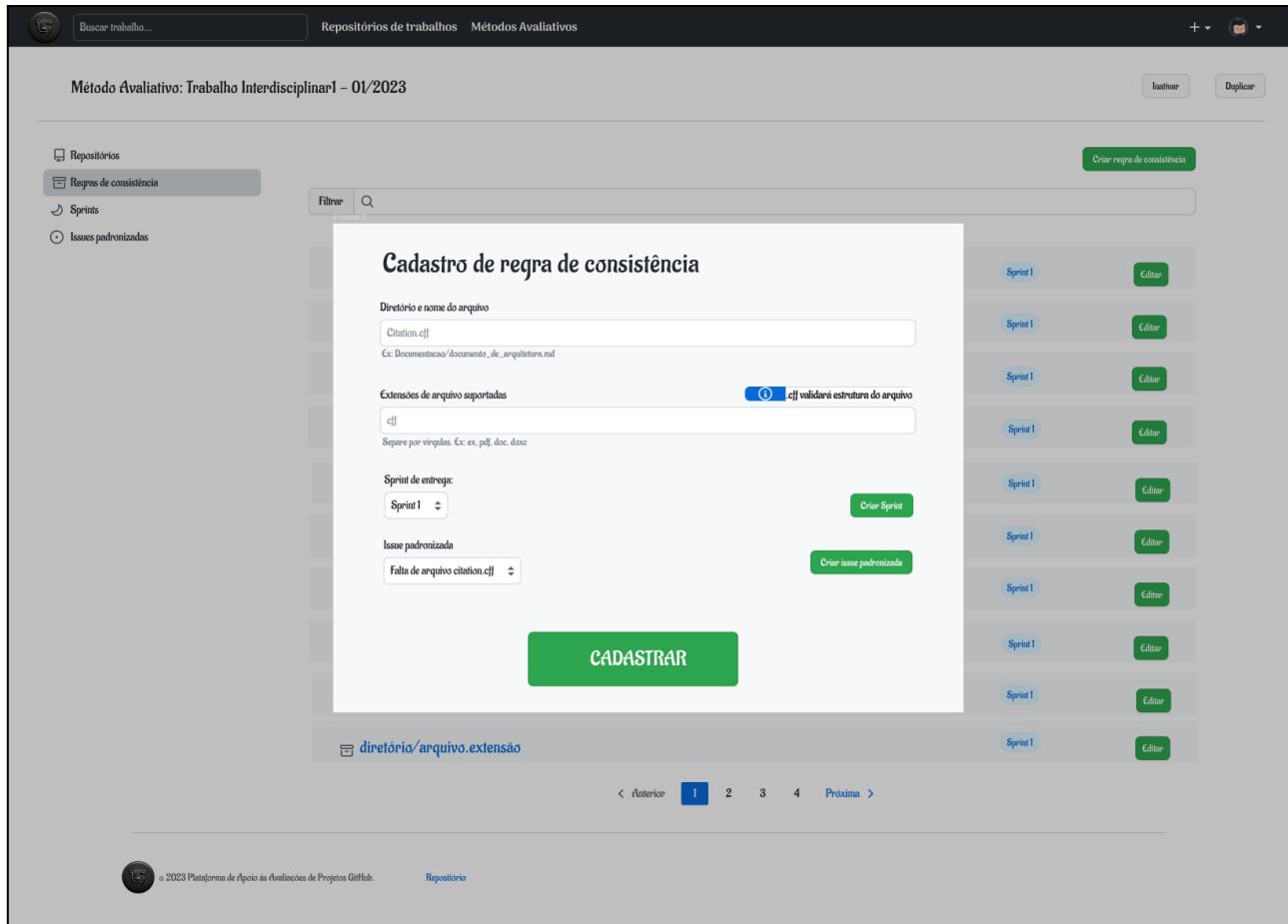


Figura 34. Tela de *modal* de cadastro de regra de consistência

A Figura 35 representa a página de gerenciamento das sprints de um método avaliativo. Nessa página é possível criar, editar e filtrar as sprints do método avaliativo. Essa interface contempla o caso de uso UC21.

The screenshot shows a web interface for managing sprints within a GitHub project evaluation method. At the top, there are navigation links for 'Repositórios de trabalhos' and 'Métodos Avaliativos'. Below the header, the title 'Método Avaliativo: Trabalho Interdisciplinar - 01/2023' is displayed, along with 'Iniciar' and 'Duplicar' buttons. On the left, a sidebar lists 'Repositórios', 'Regras de consistência', 'Sprints' (which is selected and highlighted in grey), and 'Issues padronizadas'. A search bar with 'Filtrar' and a magnifying glass icon is positioned above the sprint list. To the right of the search bar is a green 'Criar Sprint' button. The main area displays a list of 10 sprints, each with a blue circular icon and a name: 'Sprint 1', 'Sprint 2', 'Sprint 3', 'Sprint 4', 'Sprint 5', 'Sprint 6', 'Sprint 7', 'Sprint 8', 'Sprint 9', and 'Sprint 10'. Each sprint entry includes a green 'Editar' button. At the bottom of the list is a navigation bar with buttons for '< Anterior', '1' (selected), '2', '3', '4', and 'Próxima >'.

Figura 35. Tela de sprints de um método avaliativo

A Figura 36 representa a página de gerenciamento das *issues* padronizadas de um método avaliativo. Nessa página é possível criar, editar e filtrar as *issues* padronizadas do método avaliativo. Essa interface contempla os casos de uso UC10, UC23, UC39, UC40, UC41 e UC42.

Método Avaliativo: Trabalho Interdisciplinar1 – 01/2023

Issues padronizadas

| Issue Title | Status | Action |
|---------------------------|----------|-------------------------|
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |
| nome da issue padronizada | Sprint 1 | <button>Editar</button> |

< Anterior 1 2 3 4 Próxima >

© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub. Repertório

Figura 36. Tela de *issues* padronizadas de um método avaliativo

A Figura 37 representa a página onde são listados os repositórios pertencentes à organização ICEI-PUC-Minas-PPLES-TI do GitHub, possibilitando os professores visualizarem os nomes dos repositórios e dos seus respectivos métodos avaliativos. Por fim, também há os botões de Sincronizar para atualizar a base de dados de cada repositório com do GitHub e o Abrir que possibilita abrir a visão geral específica do repositório de um trabalho. Essa interface contempla os casos de uso UC2, UC27 e UC31.

The screenshot displays a list of work repositories. Each item in the list includes a small thumbnail icon, the repository name 'nome-do-rep' in blue text, a field labeled 'Nome do Método Avaliativo' (Assessment method name) with placeholder text 'Nome do Método Avaliativo', a 'Sincronizar' (Sync) button, and an 'Abrir' (Open) button. Above the list is a search bar with the query 'Q: ma.TI5-2023'. At the bottom of the page is a navigation bar showing page 1 of 4, with links to 'Anterior' (Previous), 'Próxima' (Next), and other pages.

Figura 37. Tela de listagem de repositórios de trabalhos

A Figura 38 representa a tela de visão geral das informações de um trabalho. Nessa tela os professores podem visualizar as métricas por tipo de informação, *branches*, sprints de cada contribuidor do repositório ou do repositório inteiro. Além disso, poderá ativar e desativar a sincronização automática, sincronizar manualmente, redirecionar para o repositório, visualizar contribuidores e método avaliativo. Ademais, nessa tela é possível mudar a visualização entre métricas do repositório, qualidade do código, históricos de *commits*, conformidade com regras de consistência e configurações do trabalho. Quando acionado o menu de métricas, são apresentados diversos indicadores e gráficos a respeito da contribuição do grupo e dos alunos nas entregas, com a possibilidade de filtrar por sprint e pela *branch* específica. Essa interface contempla os casos de uso UC26, UC27, UC12, UC13, UC14, UC15, UC16, UC17, UC18, UC19 e UC20.

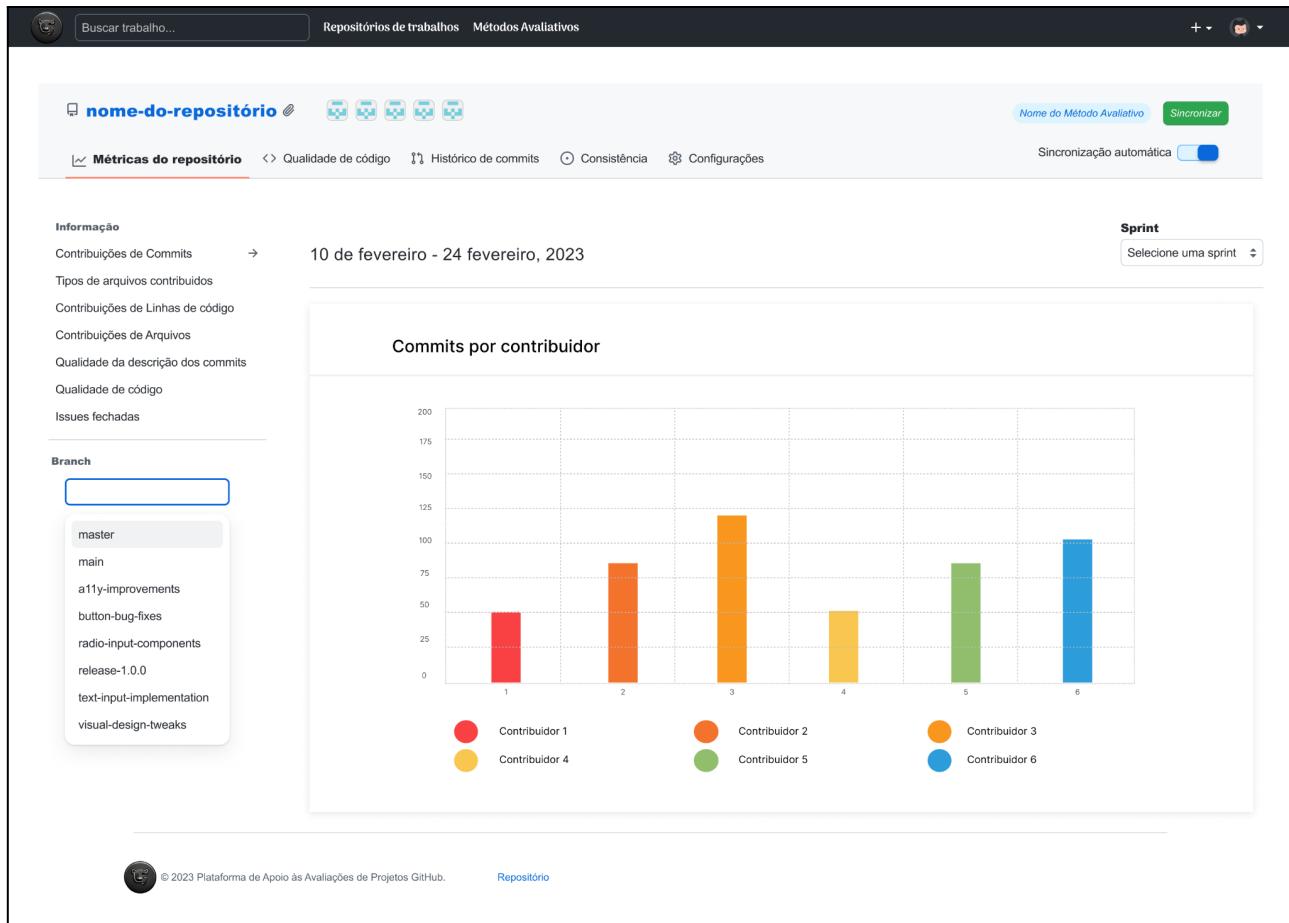


Figura 38. Tela de métricas de um trabalho

A Figura 39 apresenta a tela de visualização do trabalho, com a visualização de qualidade de código ativa. Nessa tela, é possível acionar uma análise estática de código, e, quando completa, são exibidos os resultados da mesma, provindos do SonarQube. Essa interface contempla o caso de uso UC11.

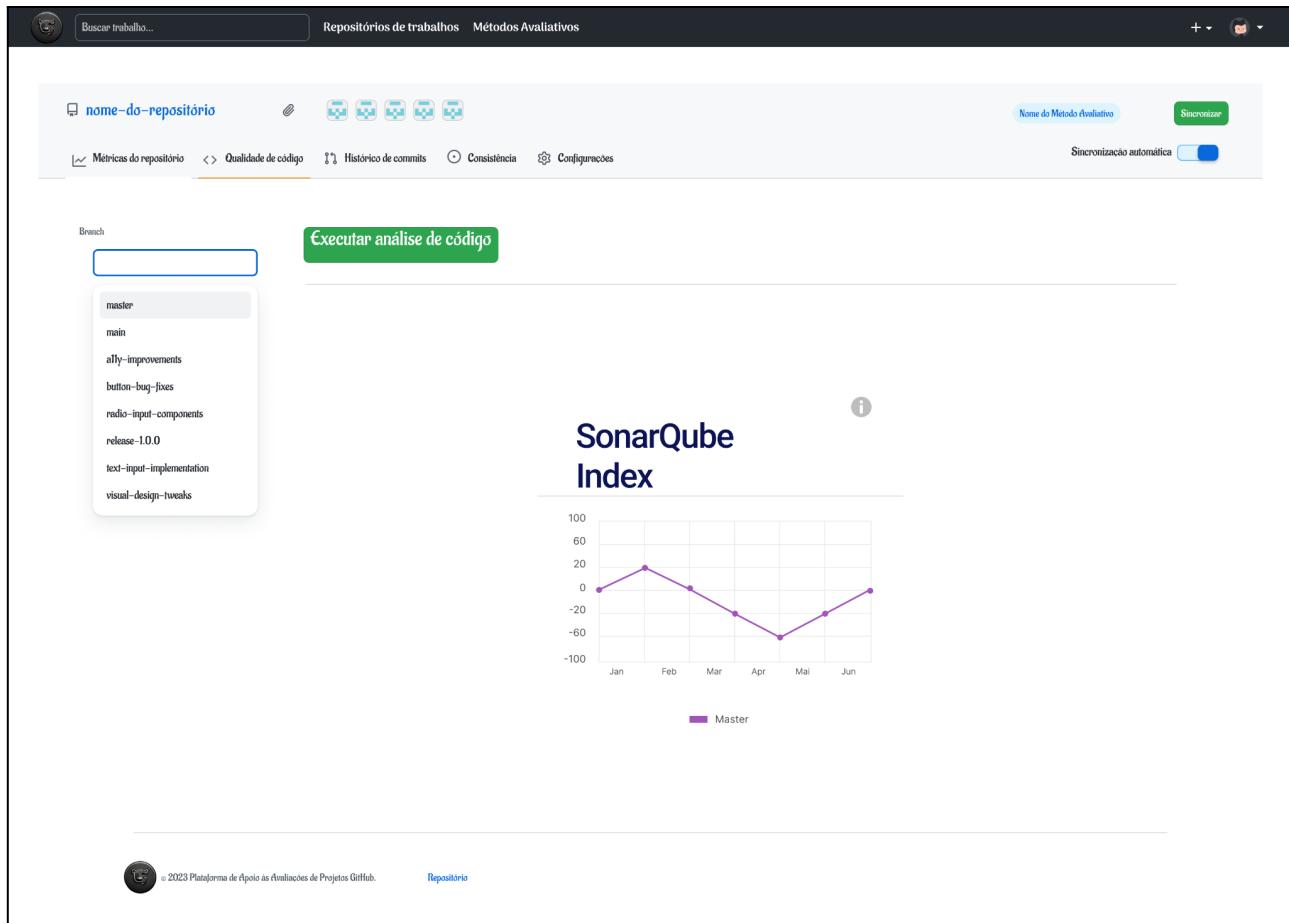
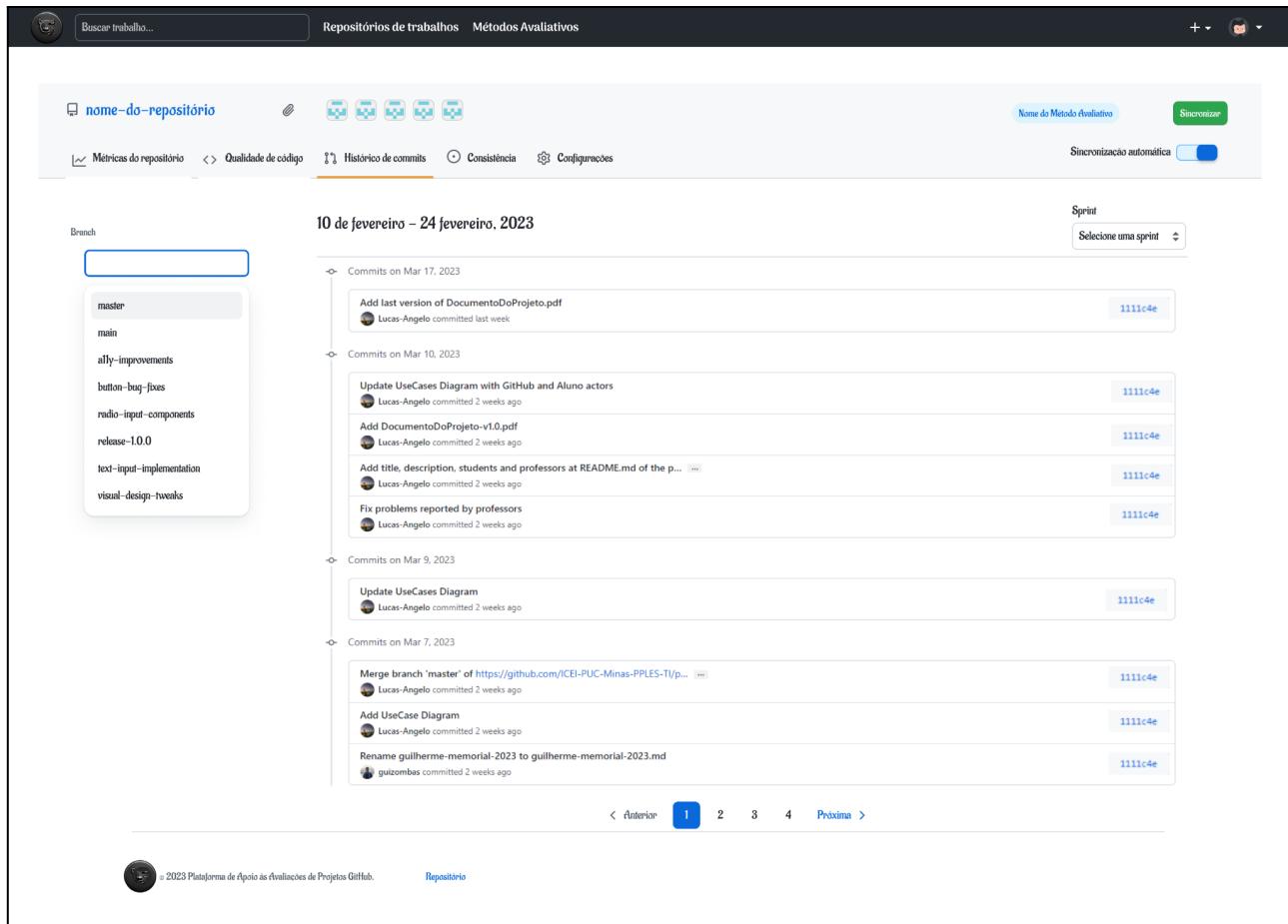


Figura 39. Tela de qualidade de código de um trabalho

A Figura 40 apresenta a tela de visualização do trabalho, com a visualização de histórico de *commits* ativa. Nessa tela, é possível visualizar uma lista de *commits* realizados no repositório, indicando quem o realizou e quando o fez, além de ser possível redirecionar para o *commit* no próprio site do GitHub. Essa interface contempla os casos de uso UC23 e UC24.

Figura 40. Tela do histórico de *commits* de um trabalho

A Figura 41 apresenta a tela de visualização do trabalho, com a visualização de consistência ativa. Nessa tela, é possível visualizar as regras de consistência configuradas para o trabalho a partir do método avaliativo vinculado. Cada regra apresenta uma cor indicando se ela foi entregue ou não, e se foi entregue com atraso. Além disso, é possível ainda filtrar a visualização das regras por sprint de entrega. Essa interface contempla os casos de uso UC25, UC28, UC29 e UC23.

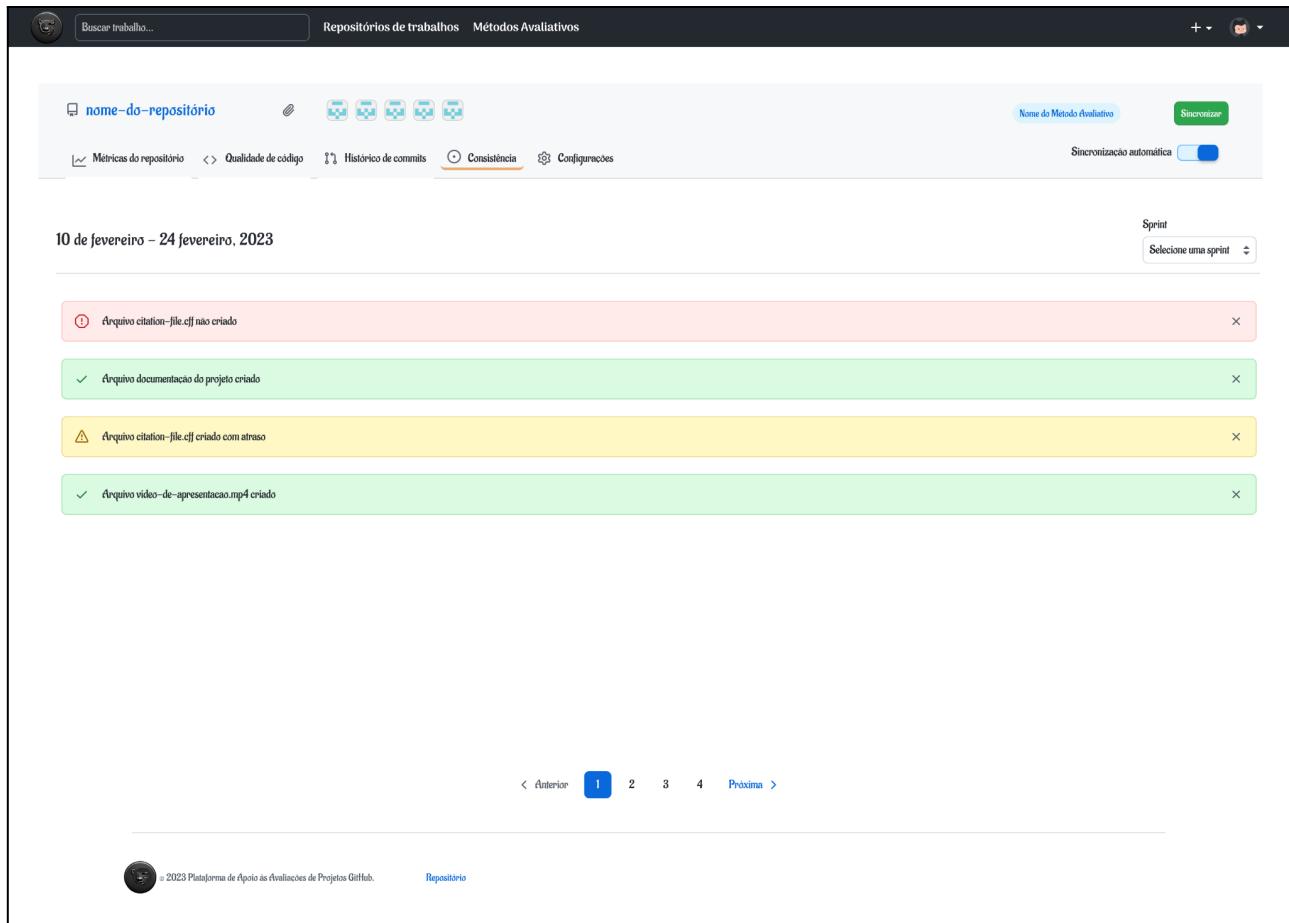


Figura 41. Tela da consistência de entrega de um trabalho

A Figura 42 apresenta a tela de visualização do trabalho, com a visualização de configurações do trabalho ativa. Nessa tela, é possível alterar a qual método avaliativo aquele trabalho pertence. Essa interface contempla o caso de uso UC4.

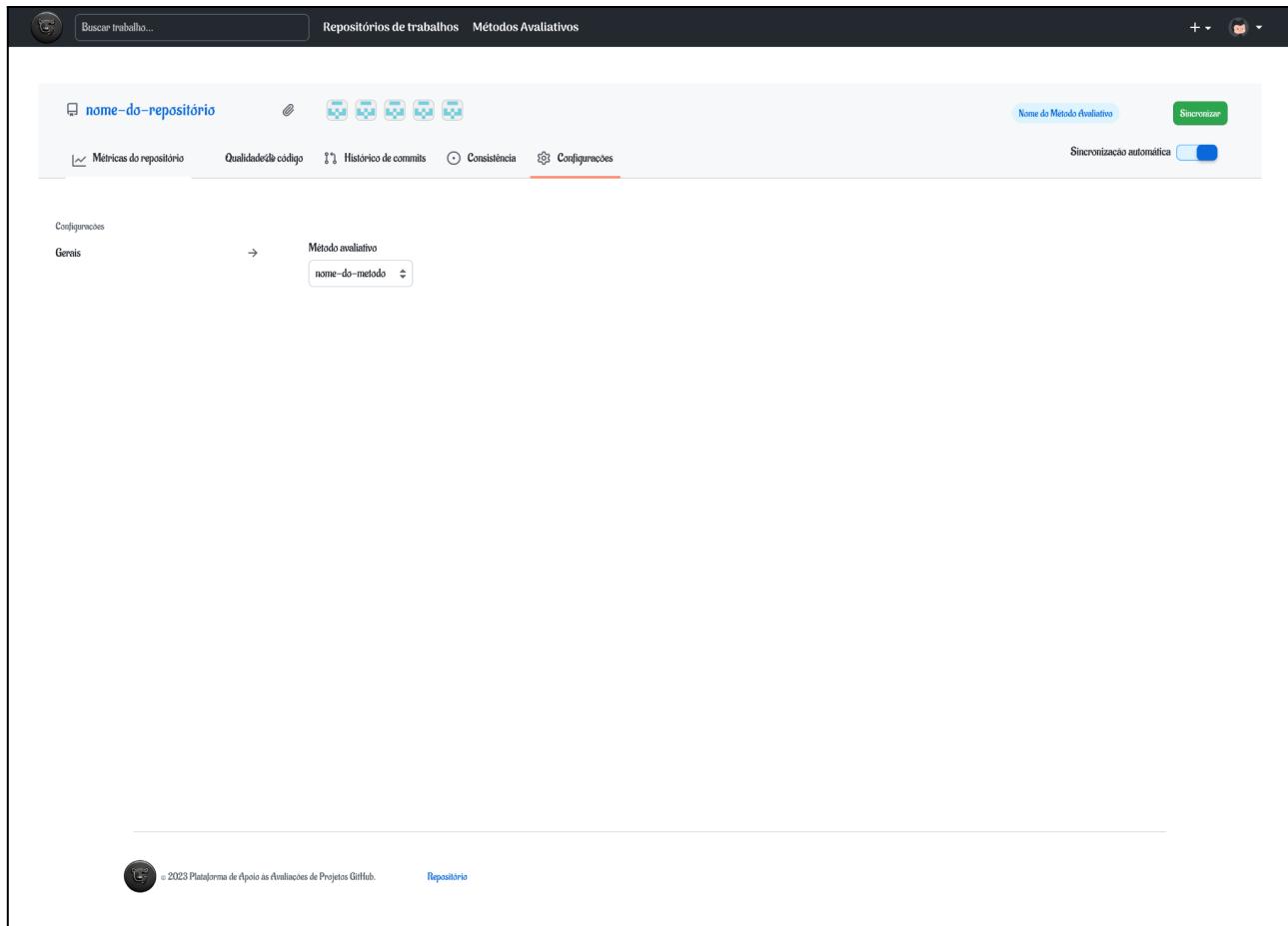


Figura 42. Tela de configurações de um trabalho

5. Glossário e Modelos de Dados

Esta seção tem o objetivo de apresentar e detalhar o glossário e modelo de dados do sistema, fornecendo uma compreensão mais abrangente sobre os termos técnicos utilizados e a estrutura de organização dos dados dentro da plataforma. Assim, a Tabela 28 especifica os atributos de entrada de dados no sistema, enquanto a Tabela 29 lista os atributos de saída de dados, permitindo uma compreensão mais clara das informações inseridas e geradas pela plataforma. Da mesma forma, a Figura 43 ilustra o Diagrama de Entidades e Relacionamentos (DER) do sistema, cujo objetivo é representar a camada de banco de dados da plataforma.

| Autenticação na plataforma | | |
|----------------------------|---------|---|
| Atributo | Formato | Descrição |
| E-mail | Texto | E-mail do usuário do professor no GitHub. |

| | | |
|--|-------------------|--|
| Senha | Texto | Senha do usuário do professor no GitHub. |
| Gerenciar método avaliativo | | |
| Descrição | Texto | Texto descritivo para caracterizar a disciplina à qual o método avaliativo se refere. Exemplo: “Trabalho Interdisciplinar 5: Aplicações Distribuídas”. |
| Semestre | Número | O semestre da oferta da disciplina do método avaliativo, sendo possível inserir 1 ou 2. |
| Ano | Número | O ano da oferta da disciplina do método avaliativo. |
| Desativado | Booleano | Indica se o método avaliativo está desativado ou não. |
| Gerenciar regra de consistência | | |
| Descrição | Texto | Texto descritivo para caracterizar a qual é a regra de consistência. Exemplo: “Regra para arquivo de documentação do projeto” |
| Diretório e nome do arquivo | Texto | Texto referente ao diretório e nome do arquivo da regra de consistência. Exemplo: “Documentacao/arquivo_de_arquitetura.md”. |
| Extensões de arquivo suportadas | Lista de texto | Lista de extensões de arquivos suportados pela regra de consistência. Exemplo: “md, pdf, docx”. |
| Sprint de entrega | Sprint | A sprint que esta regra de consistência deve ser entregue. |
| Issue padronizada | Issue padronizada | A issue padronizada que será aberta no repositório dos trabalhos que não seguirem a regra de consistência delimitada para cada sprint. |
| Gerenciar sprint | | |

| | | |
|------------------------------------|-------|--|
| Data do início | Data | Data do início da sprint. |
| Data do fim | Data | Data do fim da sprint. |
| Gerenciar issue padronizada | | |
| Título da issue padronizada | Texto | Texto referente ao título da issue que será aberta nos repositórios de trabalho. |
| Descrição da issue padronizada | Texto | Texto referente à descrição da issue que será aberta nos repositórios de trabalho. |

Tabela 28. Atributos de entrada de dados no sistema

| Visualização de repositório de trabalho | | |
|--|----------------|---|
| Atributo | Formato | Descrição |
| Nome do repositório | Texto | Texto referente ao nome do repositório de trabalho no GitHub. |
| Link do repositório | Texto | Texto do link do GitHub referente ao repositório do trabalho. |
| Método Avaliativo | Texto | Nome do método avaliativo que está avaliando o repositório. |
| <i>Branches</i> | Lista de texto | Nomes das <i>branches</i> que o repositório possui no GitHub. |
| <i>Commits</i> | Lista de texto | Nomes dos <i>commits</i> que o repositório possui que cada <i>branch</i> no GitHub. |
| Quantidade de linhas de código | Número | Número referente a quantidade de linhas que o repositório possui no GitHub. |
| Quantidade de arquivos | Número | Número referente a quantidade de arquivos que o repositório possui no GitHub. |
| Qualidade de código | Número | Número referente à qualidade do código calculado por meio do |

| | | |
|--|--|--|
| | | SonarQube para cada <i>commit</i> do repositório no GitHub. |
| Quantidade de <i>issues</i> | Número | Número de <i>issues</i> que o repositório possui no GitHub. |
| Quantidade de <i>pull requests</i> | Número | Número de <i>pull requests</i> que o repositório possui no GitHub. |
| Sincronização automática | Booleano | Indica se a sincronização diária de dados do repositório com o GitHub está ativada ou não. |
| Sincronizando | Booleano | Indica se o repositório está sincronizando os dados com o GitHub no momento ou não. |
| Alunos contribuidores | Número | Número de alunos que contribuíram com o repositório no GitHub. |
| Entrega de regra de consistência | Lista de entregas de regra de consistência | Refere-se a lista das entregas de regra de consistência entregues ou não por cada repositório do trabalho. |
| Visualização de método avaliativo | | |
| Descrição | Texto | Texto descritivo para caracterizar a qual disciplina o método avaliativo refere-se. Exemplo: “Trabalho Interdisciplinar 5: Aplicações Distribuídas”. |
| Semestre | Número | O semestre da oferta da disciplina do método avaliativo, sendo possível inserir 1 ou 2. |
| Ano | Número | O ano da oferta da disciplina do método avaliativo. |
| Desativado | Booleano | Se o método avaliativo está desativado. |
| Repositórios | Lista de repositórios | Lista dos repositórios avaliados pelo método avaliativo. |
| Regras de consistência | Lista de regras de consistência | Lista das regras de consistência de um método avaliativo. |

| | | |
|---------------------|-------------------------------------|--|
| Sprints | Lista de sprints | Lista das sprints de um método avaliativo. |
| Issues Padronizadas | Lista de <i>issues</i> padronizadas | Lista das issues padronizadas de um método avaliativo. |

Tabela 29. Atributos de saída de dados no sistema

A Figura 43 é o DER, que representa visualmente a estrutura do banco de dados do sistema e exibe um total de 18 tabelas, sendo 15 tabelas de entidades e 3 tabelas de relacionamento muitos-para-muitos. Esse diagrama foi modelado utilizando a ferramenta MySQL Workbench⁶. O símbolo de chave amarela define que aquela é a chave primária da tabela, já o losango vermelho são as chaves estrangeiras, o losango azul preenchido refereência atributos que não aceita valores nulos, por fim, o losango azul não preenchido são atributos que aceitam valores nulos. Ademais, esse modelo foi feito para possuir compatibilidade com o Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL Server⁷ na versão 5.7.

⁶ <https://www.mysql.com/products/workbench/>

⁷ <https://dev.mysql.com/downloads/mysql/5.7.html>

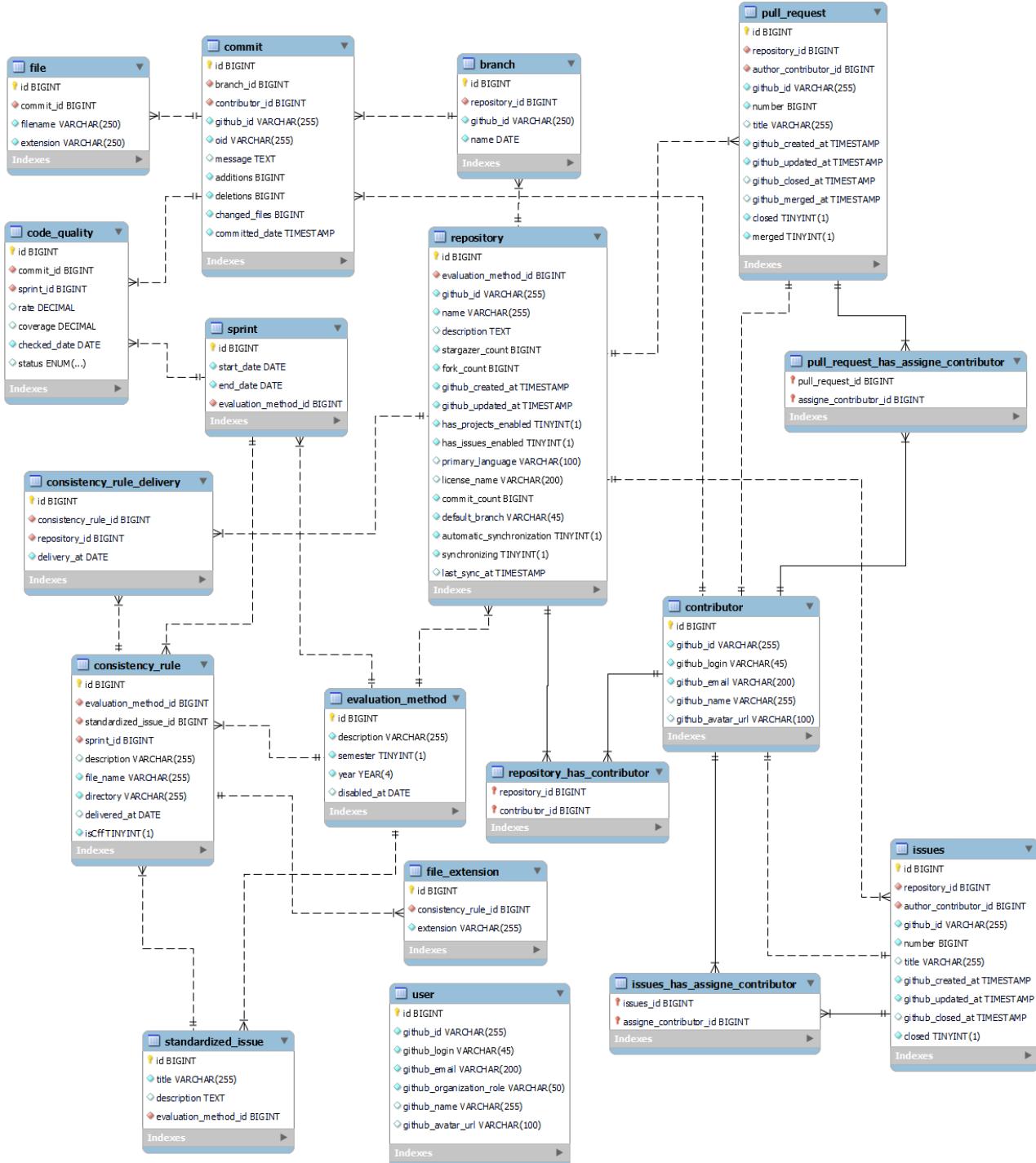


Figura 43. Diagrama de Entidade e Relacionamento

6. Casos de Teste

Esta seção descreve os casos de teste de aceitação e integração realizados conforme a especificação do sistema. Na Seção 6.1 são detalhados os casos de testes de aceitação, elaborados com base nas necessidades descritas no documento de visão do projeto, esses testes foram projetados para assegurar que as funcionalidades desenvolvidas atendam às necessidades dos usuários no sistema de forma satisfatória e confiável. Já na Seção 6.2 são descritos os casos de teste de integração elaborados visando analisar como o sistema se integra com os diversos componentes externos. Esses testes foram planejados para garantir que todas as funcionalidades do sistema operem de maneira harmoniosa e confiável em conjunto com os componentes externos envolvidos. Cada caso de teste apresenta uma pré-condição, isto é, o estado no qual o sistema se encontra antes da execução do teste, além das ações que serão realizadas pelo usuário para interagir com o sistema durante a execução do teste. Por fim, é apresentado o resultado esperado, ou seja, o estado final que o sistema deverá apresentar após a execução do teste.

6.1 Teste de Aceitação

[texto será devidamente contextualizado] Necessidades mapeadas no documento de visão

1. Autenticar usando GitHub;
2. Ver entregas de um trabalho;
3. Ver entregas de trabalho por sprint;
4. Ver contribuições de um integrante do trabalho;
5. Executar avaliação de qualidade de código do trabalho;
6. Ver métricas de contribuição do trabalho;
7. Abertura de *issues* padronizadas;
8. Detectar más práticas do Git nos trabalhos.

6.2 Testes de Integração

Uma descrição de casos de teste de Integração para validação do sistema.

7. Cronograma e Processo de Implementação

Uma descrição do cronograma para implementação do sistema e do processo que será seguido durante a implementação.

7.1 Cronograma

Uma descrição do cronograma para implementação do sistema.

| Nome | Período | Atividades | |
|----------|-----------------------------------|--|--|
| | | Guilherme Gabriel | Lucas Ângelo |
| Sprint 1 | 01/08/2023 – 15/08/2023 (15 dias) | <ul style="list-style-type: none"> ● Criar estrutura de <i>backend</i> do SupportPlatformAPI: <ul style="list-style-type: none"> ○ Configurar diretórios, bibliotecas e variáveis de ambiente; ○ Configurar o Docker para implantação. ● Criar estrutura do <i>frontend View</i>: <ul style="list-style-type: none"> ○ Configurar diretórios, bibliotecas e variáveis de ambiente; ○ Configurar Docker para implantação. ● Implementar interface de lista de repositórios; ● Implementar <i>backend</i> da listagem de repositórios da SupportPlatformAPI; ● Implementar <i>backend</i> de métricas de repositório da SupportPlatformAPI: <ul style="list-style-type: none"> ○ Quantidade e porcentagem de <i>commits</i>; ○ Quantidade de linhas alteradas. ● Casos de uso relacionados: UC2, UC15, UC16 e UC32. | <ul style="list-style-type: none"> ● Implementar sincronização de repositórios no JobScheduler: <ul style="list-style-type: none"> ○ Configurar diretórios, bibliotecas e variáveis de ambiente; ○ Configurar o Docker para implantação; ○ Mapear entidades referente a dados de repositórios do banco de dados; ○ Desenvolver <i>script</i> de integração dados de repositórios com API do GitHub; ○ Implementar sincronização em massa de dados do GitHub com o banco de dados; ○ Codificar <i>cronjob</i> para sincronização automática; ○ Configurar GitHub Action para integração contínua. ● Casos de uso relacionados: UC31 e UC32. |
| Sprint 2 | 16/08/2023 – | <ul style="list-style-type: none"> ● Implementar <i>backend</i> de métricas de repositório da | <ul style="list-style-type: none"> ● Desenvolver <i>backend</i> da autenticação integrada com o |

| | | | |
|----------|--------------------------------------|--|---|
| | 31/08/2023 (16 dias) | <ul style="list-style-type: none"> ● SupportPlataformAPI; ● Implementar interface de repositório. ● Implementar interface de métricas de repositório <ul style="list-style-type: none"> ○ Quantidade e porcentagem de <i>commits</i> ○ Linhas alteradas ○ Tipos de arquivos contribuídos ○ Contribuições em <i>issues</i> ○ Quantidade e porcentagem de arquivos ● Casos de uso relacionados: UC15, UC16, UC17, UC18, UC19 e UC31. | <ul style="list-style-type: none"> ● GitHub no SupportPlataformAPI; ● Implementar <i>backend</i> de métodos avaliativos no SupportPlataformAPI: <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ○ Testes unitários e de integração. ● Implementar <i>backend</i> de sprints no SupportPlataformAPI: <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ○ Testes unitários e de integração. ● Casos de uso relacionados: UC1, UC3, UC4, UC5, UC21, UC27, UC33, UC34, UC35, UC37 e UC38. |
| Sprint 3 | 01/09/2023 – 15/09/2023 (15 dias) | <ul style="list-style-type: none"> ● Implementar interface de login integrado com GitHub da plataforma; ● Implementar interfaces de gerenciar método avaliativo <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ● Implementar interface de gerenciar repositórios de um método avaliativo; ● Implementar interface de gerenciar sprints de um método avaliativo. <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ● Casos de uso relacionadas: UC1, UC3, UC4, UC5, UC21, UC33, UC34, UC35, UC37 e UC38 | <ul style="list-style-type: none"> ● Implementar <i>backend</i> de regras de consistência no SupportPlataformAPI: <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ○ Testes unitários e de integração. ● Implementar <i>backend</i> de entrega de regra de consistência no SupportPlataformAPI: <ul style="list-style-type: none"> ○ Criação, atualização, leitura e deleção; ○ Testes unitários e de integração. ● Casos de uso relacionados: UC6, UC7, UC8, UC9, UC22, UC25, UC36. |
| Sprint 4 | 16/09/2023 – 30/09/2023 (15 dias) | <ul style="list-style-type: none"> ● Implementar interface de gerenciar regras de consistência de um método avaliativo; | <ul style="list-style-type: none"> ● Implementar <i>backend</i> de <i>issue padronizada</i> no SupportPlataformAPI: <ul style="list-style-type: none"> ○ Criação, atualização, |

| | | | |
|----------|-----------------------------------|---|---|
| | | <ul style="list-style-type: none"> ● Implementar interface de gerenciar <i>issues</i> padronizadas de um método avaliativo; ● Implementar interface de histórico de <i>commits</i> do repositório; ● Implementar filtros por sprint na interface de métricas de repositório. ● Casos de uso relacionados: UC6, UC7, UC9, UC10, UC12, UC22, UC24, UC26, UC37, UC39, UC40, UC41 e UC42. | <ul style="list-style-type: none"> leitura e deleção; ○ Testes unitários e de integração. ● Casos de uso relacionados: UC10, UC39, UC40, UC41 e UC42. |
| Sprint 5 | 01/10/2023 – 15/10/2023 (15 dias) | <ul style="list-style-type: none"> ● Implementar interface de configurações do repositório; ● Implementar interface de entregas de regra de consistência de um repositório; ● Casos de uso relacionados: UC25 e UC27 | <ul style="list-style-type: none"> ● Implementar <i>cronjob</i> para validar regras de consistência em repositórios no JobScheduler: <ul style="list-style-type: none"> ○ Criação de entrega de regra de consistência; ○ Detecção de <i>squashes</i> e <i>rebases</i>; ○ Detecção de <i>branches</i> inativas ao fim de sprints. ○ Integração para abertura de <i>issue</i> padronizada no GitHub. ● Casos de uso relacionados: UC23, UC28, UC29 e UC30. |
| Sprint 6 | 16/10/2023 – 31/10/2023 (16 dias) | <ul style="list-style-type: none"> ● Implementar interface de análise estática de código do repositório; ● Casos de uso relacionados: UC11; | <ul style="list-style-type: none"> ● Implementar serviço de análise estática de repositório com SonarQube: <ul style="list-style-type: none"> ○ Definir banco de dados; ○ Configurar imagem do SonarQube no Docker; ○ Integrar serviços do SonarQube com o SupportPlataformAPI. |

| | | | |
|----------|--|---|---|
| | | | <ul style="list-style-type: none"> • Caso de uso relacionado: UC11. |
| Sprint 7 | 01/11/2023 – 15/11/2023 (15 dias) | <ul style="list-style-type: none"> • Correções e aprimoramentos com relação ao <i>feedback</i> da primeira versão estável da plataforma; • Pagamento de dívidas técnicas não internacionais; • Execução de testes regressivos da plataforma. | <ul style="list-style-type: none"> • Correções e aprimoramentos com relação ao <i>feedback</i> da primeira versão estável da plataforma; • Pagamento de dívidas técnicas não internacionais; • Execução de testes regressivos da plataforma. |
| Sprint 8 | 16/11/2023 – 30/11/2023 (15 dias) | <ul style="list-style-type: none"> • Documentar introduções de implantação e utilização; • Implantar sistema em produção; • Escrita do Post-mortem do projeto. | <ul style="list-style-type: none"> • Documentar introduções de implantação e utilização; • Implantar sistema em produção; • Escrita do Post-mortem do projeto. |

7.2 Processo de Implementação

Uma descrição do processo que será seguido durante a implementação.