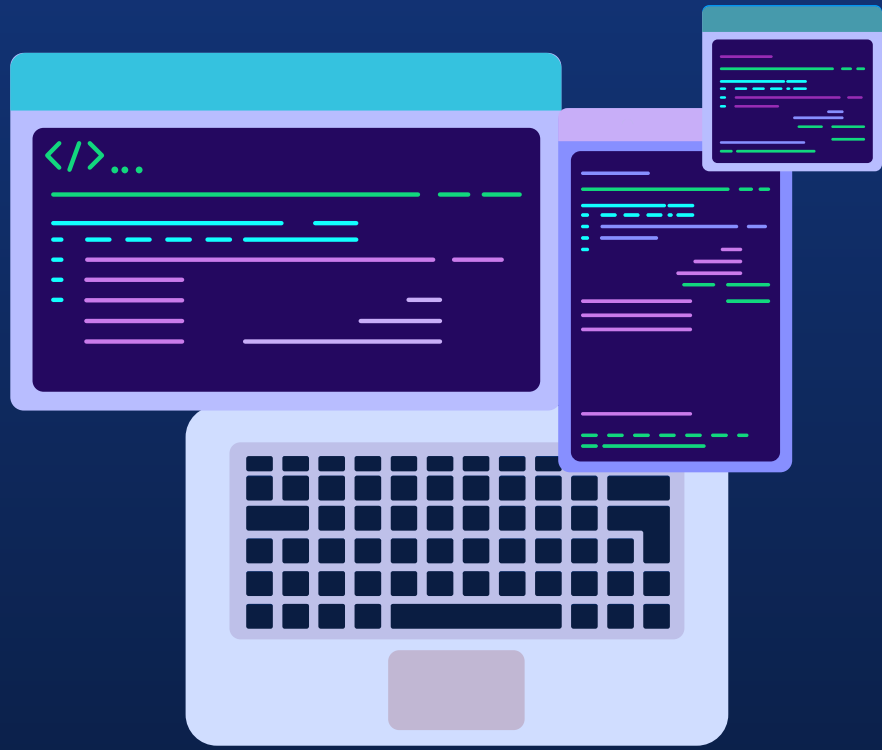
GitGrade

Plataforma de Apoio
às Avaliações de
Projetos no GitHub



Trabalho de Conclusão de Curso — Engenharia de Software PUC Minas — 1/2023

- Alunos:
  - Guilherme Gabriel Silva Pereira
  - Lucas Ângelo Oliveira Martins Rocha
- Stakeholder:
  - Responsável pelo levantamento de necessidades e requisitos: José Laerte Pires Xavier Junior
- Orientadores:
  - Orientador de conteúdo: Cleiton Silva Tavares
  - Orientador acadêmico: José Laerte Pires Xavier Junior

Sumário

01 Contextualização e Problema

02 Objetivo do Software

03 Escopo do Software

04 Principais Necessidades e Funcionalidades

05 Descrição dos Atores

06 Modelagem do Diagrama de Caso de Uso

07 Modelagem da Arquitetura do Software

08 Modelagem dos Principais Componentes

09 Projeto de Interfaces com o Usuário

10 Cronograma do TCC II

01

Contextualização e Problema

Contextualização e Problema

Contextualização

- Os professores do curso Engenharia de Software semestralmente lecionam disciplinas de Trabalho Interdisciplinar.
- Nessas disciplinas, as turmas são divididas em diversos grupos de alunos, que desenvolvem, ao longo do semestre, um trabalho interdisciplinar de software.
- Por parte do professor, o processo de avaliação consiste em verificar se todos os documentos, artefatos, arquivos e código-fonte foram entregues no repositório do trabalho na organização ICEI-PUC-Minas-PPLES-TI do GitHub.

Problema

- Atualmente a avaliação é realizada manualmente pelos docentes, tornando-se uma tarefa árdua.
- Consiste nas atividades feitas manualmente, abrir o repositório, validar entrega de artefatos (Exemplo: CITATION.cff, arquivo de documentação...), verificar as contribuições de *commits*, qualidade das descrições de *commits*, adições e remoções de linhas de código e arquivos, resoluções de *issues*, entre outras informações para cada integrante de cada trabalho, em períodos de tempos delimitados por *sprint*.

02

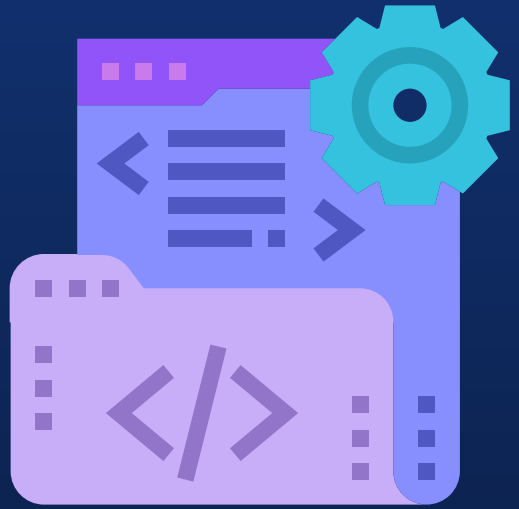
Objetivo do Software

Objetivo do Software

- Auxiliar os professores das disciplinas “Trabalho Interdisciplinar” no processo de avaliar as entregas dos alunos.
- Auxiliar nas avaliações quantitativas (contribuições), qualitativas (qualidade) e da consistência de entregas dos repositórios.
- Facilitar a comunicação e o *feedback* entre os docentes e os alunos

03

Escopo do Software



Escopo do software

Dentro do escopo

- Ver contribuições de linhas de código, *commits* e *issues* de trabalhos e filtrar por *sprint* e por contribuidor.
- Ver consistência de entregas cumpridas pelos grupos.
- Automatizar verificações de qualidade código a partir de ferramentas de análise estática de código.
- Criar *issues* padronizadas para problemas encontrados nos repositórios.
- Realizar configurações de métodos avaliativos de trabalhos, configurando entregas que devem ser feitas, sprints e *issues* padronizadas

Fora do escopo

- Gerenciar as turmas e integrar com o GitHub Classroom
- Gerar notas ou fazer integrações com o Canvas.
- Avaliar cumprimento de requisitos funcionais especificados por alunos.

04

Principais Necessidades e Funcionalidades

Principais Necessidades e Funcionalidades

Os professores precisão se autenticar pelo GitHub

- Se autenticar na plataforma usando as credenciais do GitHub
- Autenticar apenas usuários da organização

Avaliação de artefatos customizados para cada oferta de disciplina

- Criar métodos avaliativos
- Criar regras de consistência para os métodos
- Vincular repositórios aos métodos
- Validar estrutura para o caso de um arquivo “.cff”

Avaliação de entrega de artefatos por *sprints* em métodos avaliativos.

- Criar *sprints* em um método avaliativo
- Relacionar regras de consistência a *sprints* de entrega
- Ver quais regras foram entregues por um trabalho em determinada *sprint*

Principais Necessidades e Funcionalidades

Avaliação das entregas de um integrante de um trabalho

- Ver contribuidores do repositório
- Ver contribuições de *commit*, arquivos, linhas de código, *issues* e tipos de arquivos do repositório
- Filtrar exibições por *sprint* e/ou por contribuidor

Automação da avaliação da qualidade de código

- Executar uma análise estática de código do SonarQube em um repositório de um trabalho
- Ver resultados da execução da análise estática de código

Avaliação de trabalhos de forma geral

- Ver o método avaliativo que um repositório faz parte
- Ver quais regras de consistência devem ser entregues para um repositórios
- Ver histórico de *commits* de trabalhos

Principais Necessidades e Funcionalidades

Abertura de *issues* padronizadas para soluções de problemas detectados pelos métodos avaliativos

- Cadastrar *issues* padronizadas para problemas detectados em métodos avaliativos
- Realizar abertura de *issues* padronizadas quando forem detectados os problemas no repositório

Auxílio aos alunos sobre boas práticas do uso do GitHub

- Detectar *squashes* e *rebases* e abrir *issues* alertando do problema
- Detectar branches inativas e abrir *issues* alertando do problema

05

Descrição dos Atores

Descrição dos atores

O ator Professor:

- Professor das disciplinas de Trabalho Interdisciplinar;
- Ator primário da plataforma, vai utilizá-la diretamente;
- Avalia e acompanha as entregas dos alunos.

O ator GitHub:

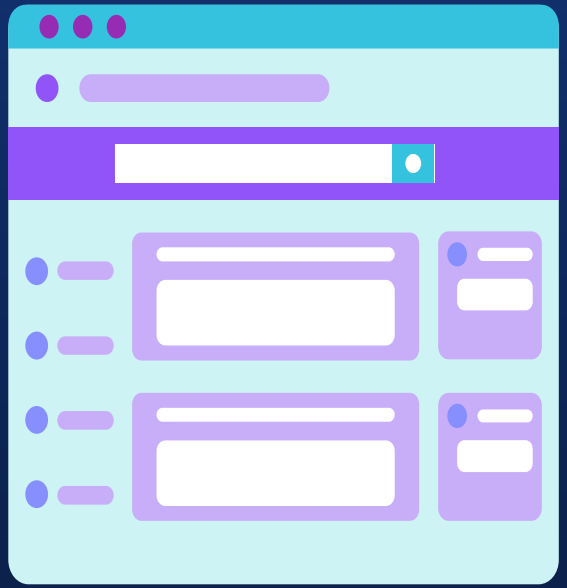
- Plataforma que hospeda os repositórios de código dos trabalhos dos alunos;
- Ator secundário da plataforma;
- É impactado pela criação de *issues* padronizadas dos problemas encontrados nos repositórios.

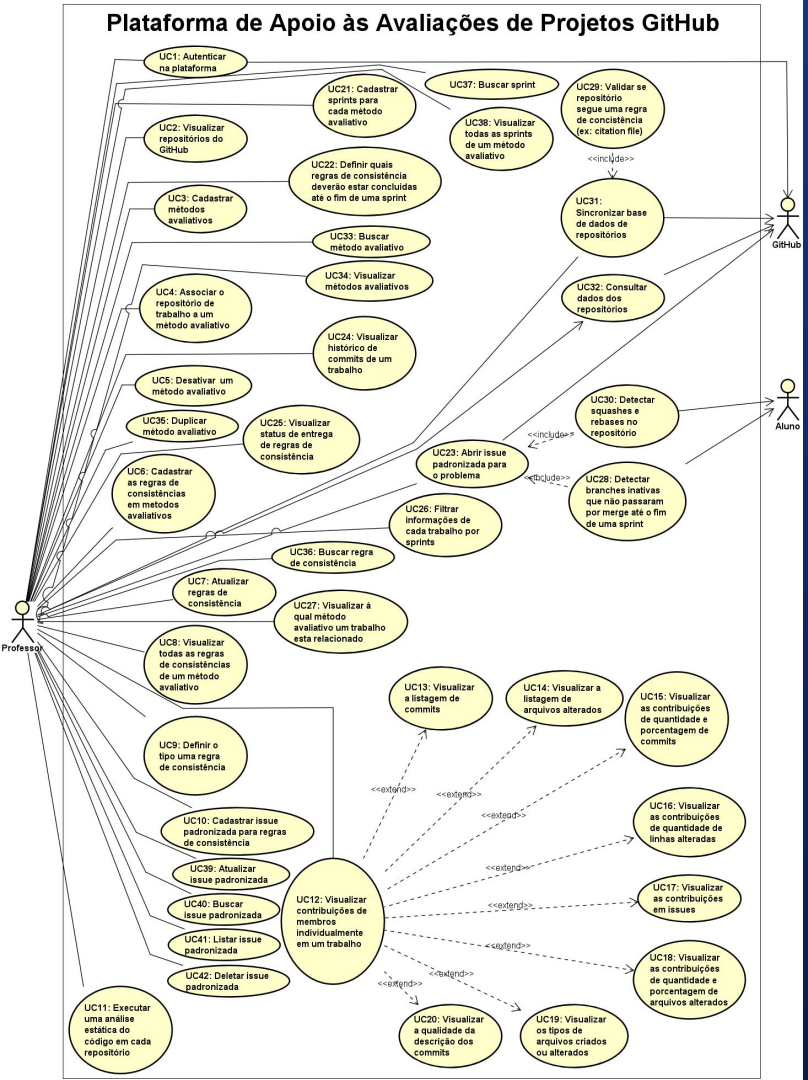
O ator Aluno:

- Aluno que cursa uma disciplina de Trabalho Interdisciplinar;
- Ator secundário, impacta a plataforma ao gerar e subir o código que será avaliado;
- É impactado pelos *feedbacks* que a plataforma gera a partir da criação de *issues*.

06

Modelagem do Diagrama de Caso de Uso



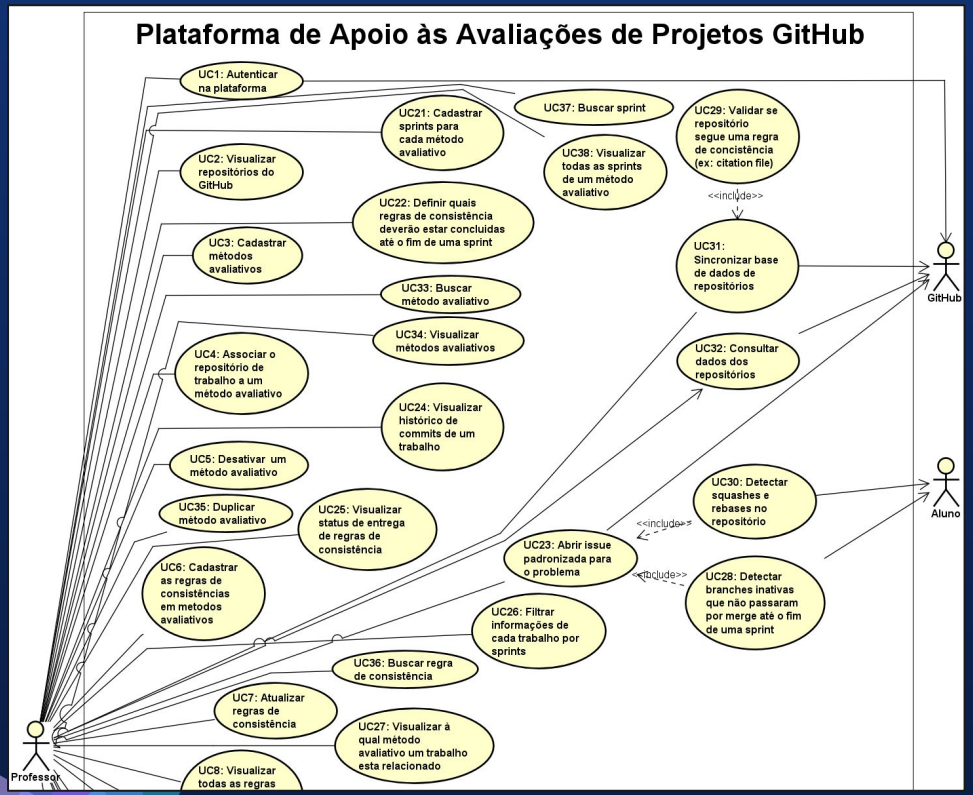


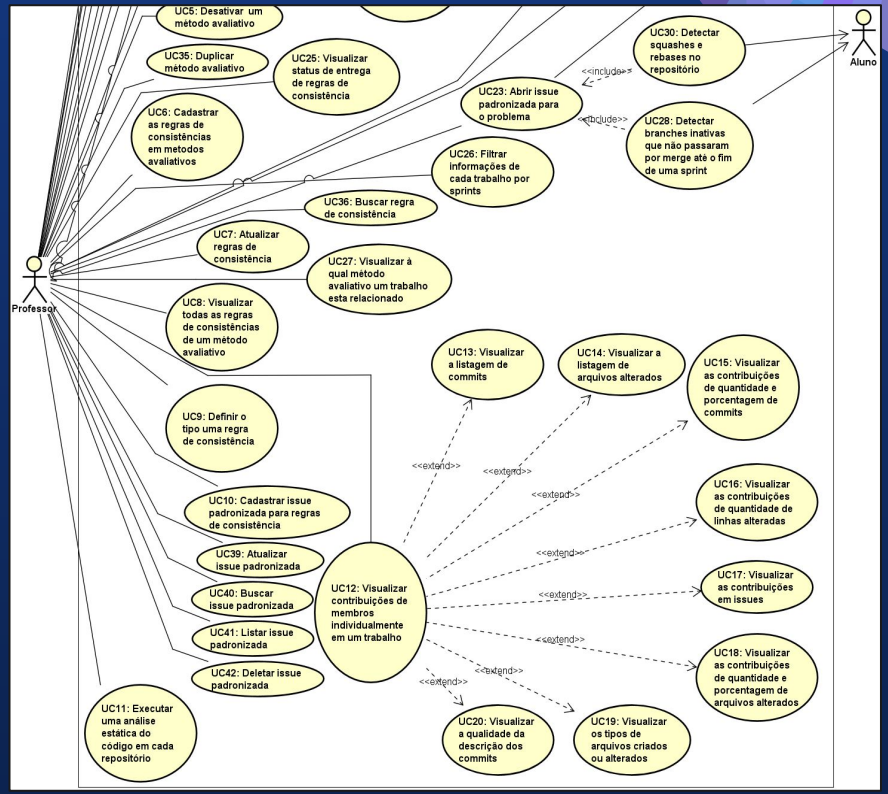
Visão geral do Diagrama de Caso de Uso

- O sistema possui 3 atores:
  - Ator primário:
    - Professor
  - Atores secundários:
    - Aluno e GitHub

- Ao todo são 42 casos de uso

O Diagrama de Caso de Uso





Funções gerais dos atores

O ator Professor:

- representa os professores que acessam a plataforma para avaliar os repositórios de trabalhos
- efetuam 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.

O ator GitHub:

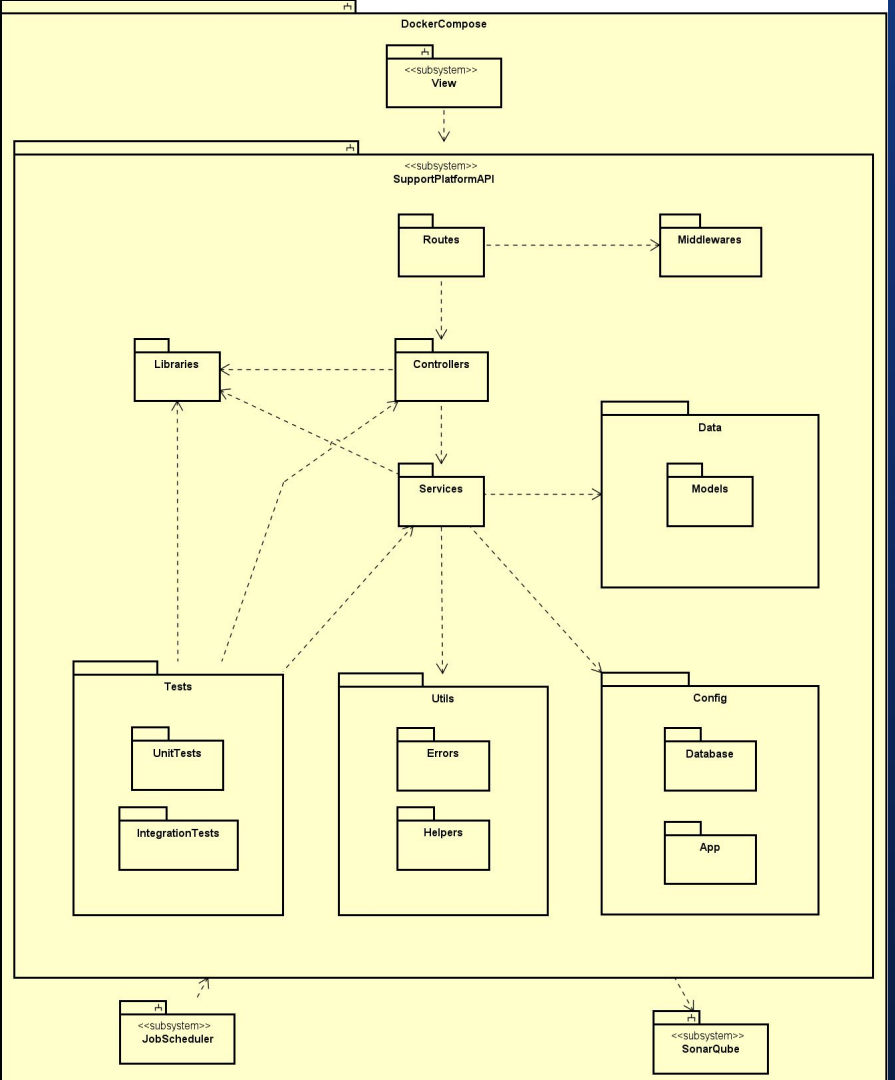
- utilizado pela execução de tarefas cronometradas de busca e atualização da base de dados do sistema por meio da API do GitHub, também é utilizado para autenticação de usuários.

O ator Aluno:

- utilizado para análises de contribuições individuais nos repositórios de trabalhos.

07

Modelagem da Arquitetura do Software

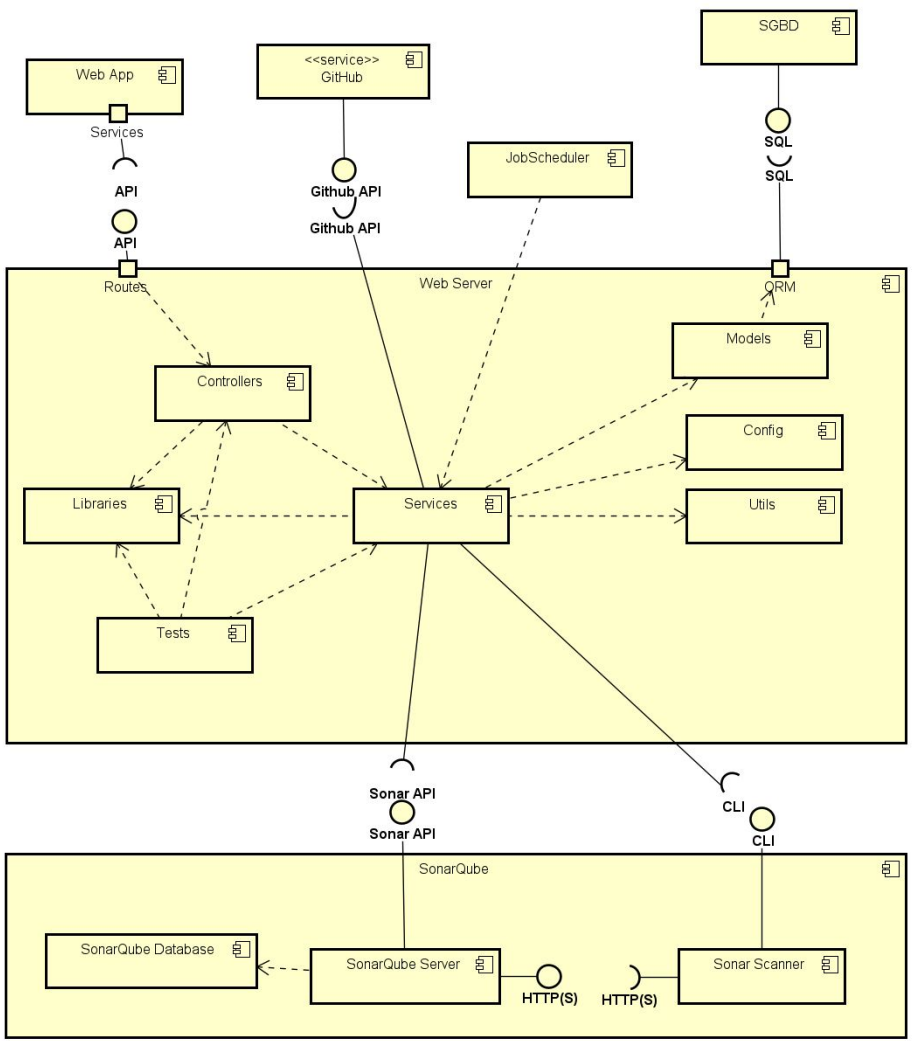


Visão geral do Diagrama de Pacotes

- O pacote de administração de contêineres DockerCompose contém os quatro principais pacotes do sistema:
    - SupportPlataformAPI: manipula dados na plataforma
    - View: possui as interfaces dos usuários
    - SonarQube: efetua operações de análise estática de código
    - JobScheduler: sincroniza base de dados com as informações provenientes do GitHub

08

Modelagem dos Principais Componentes

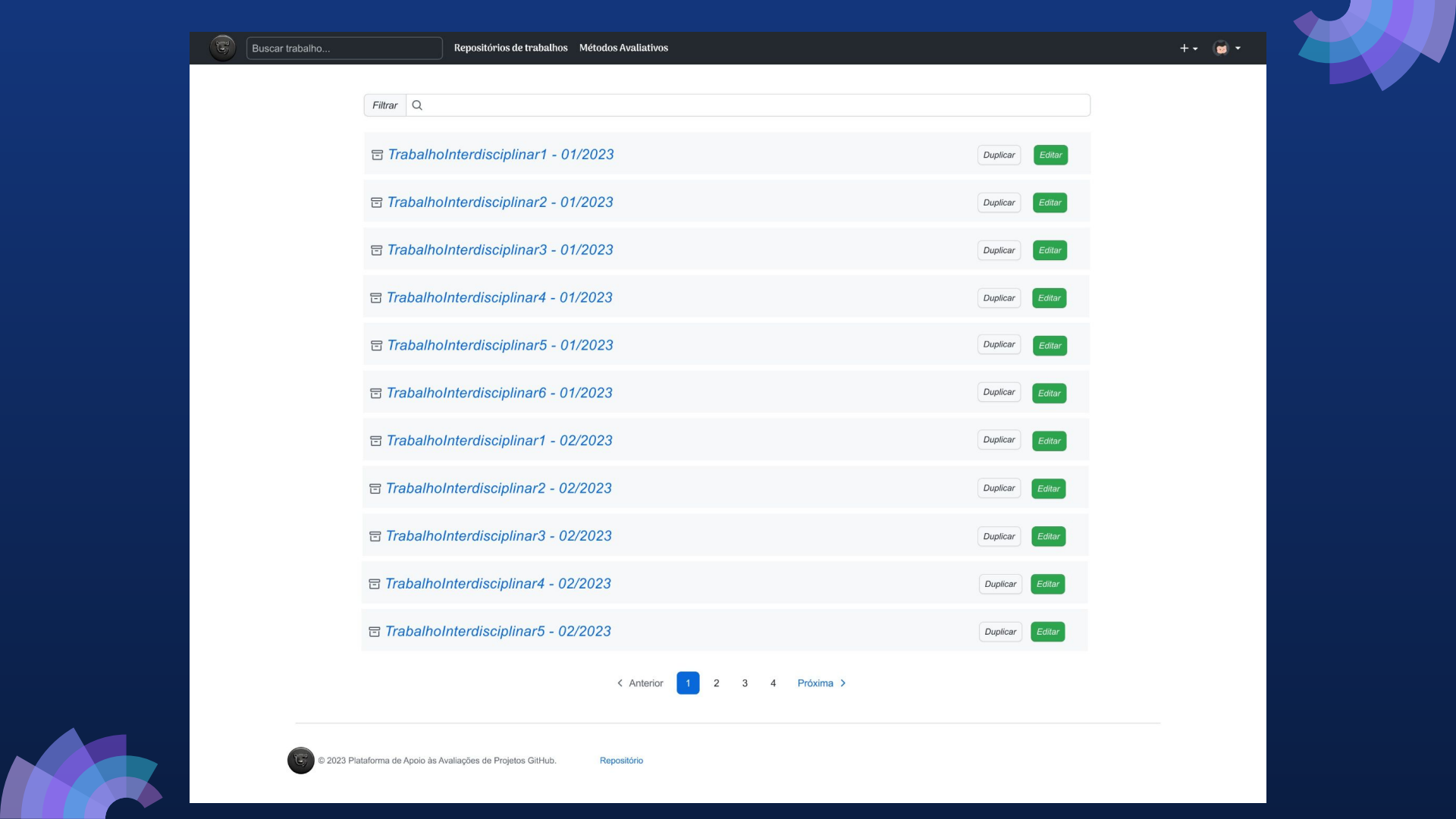


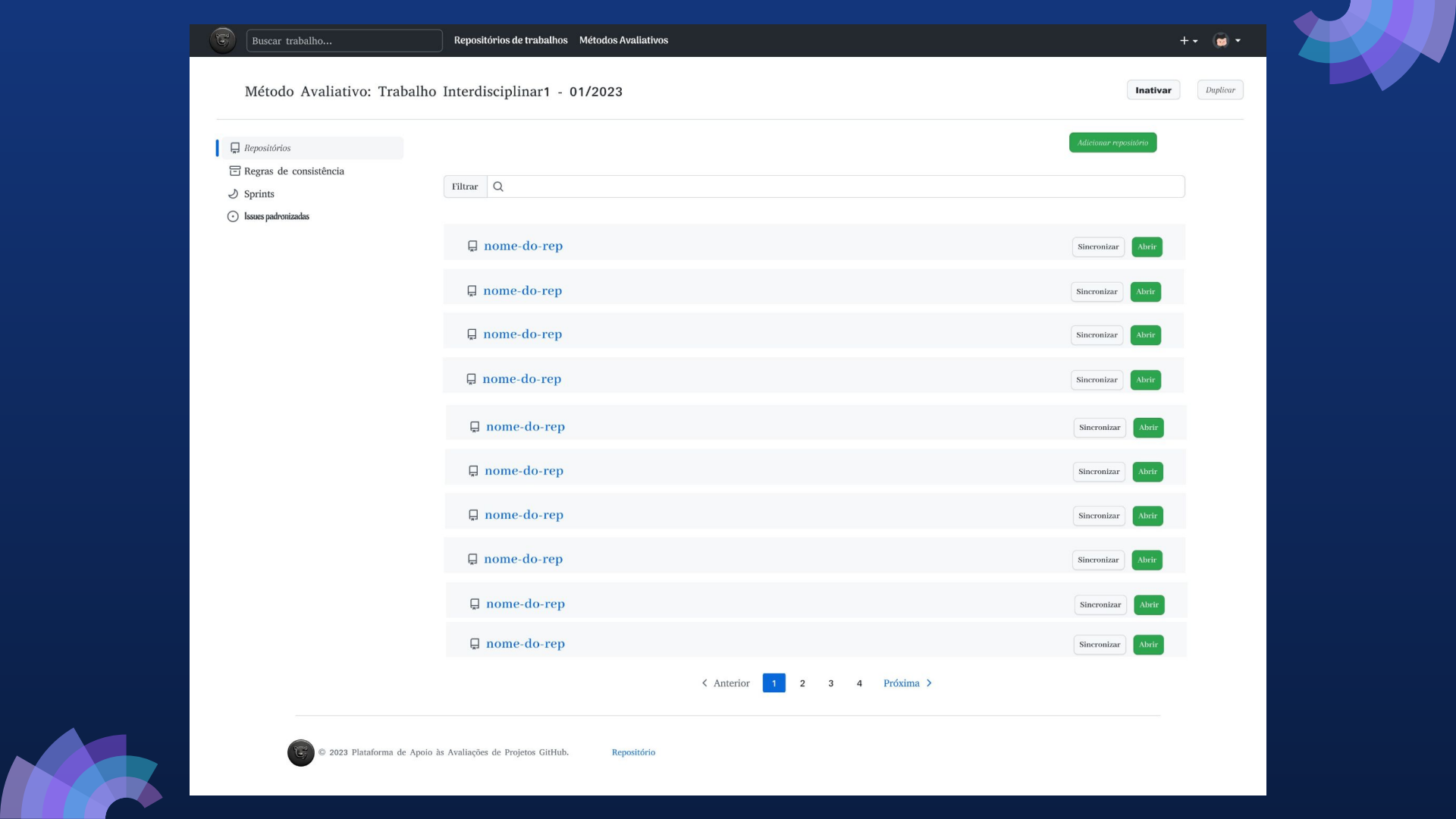
Visão geral do Diagrama de Componentes

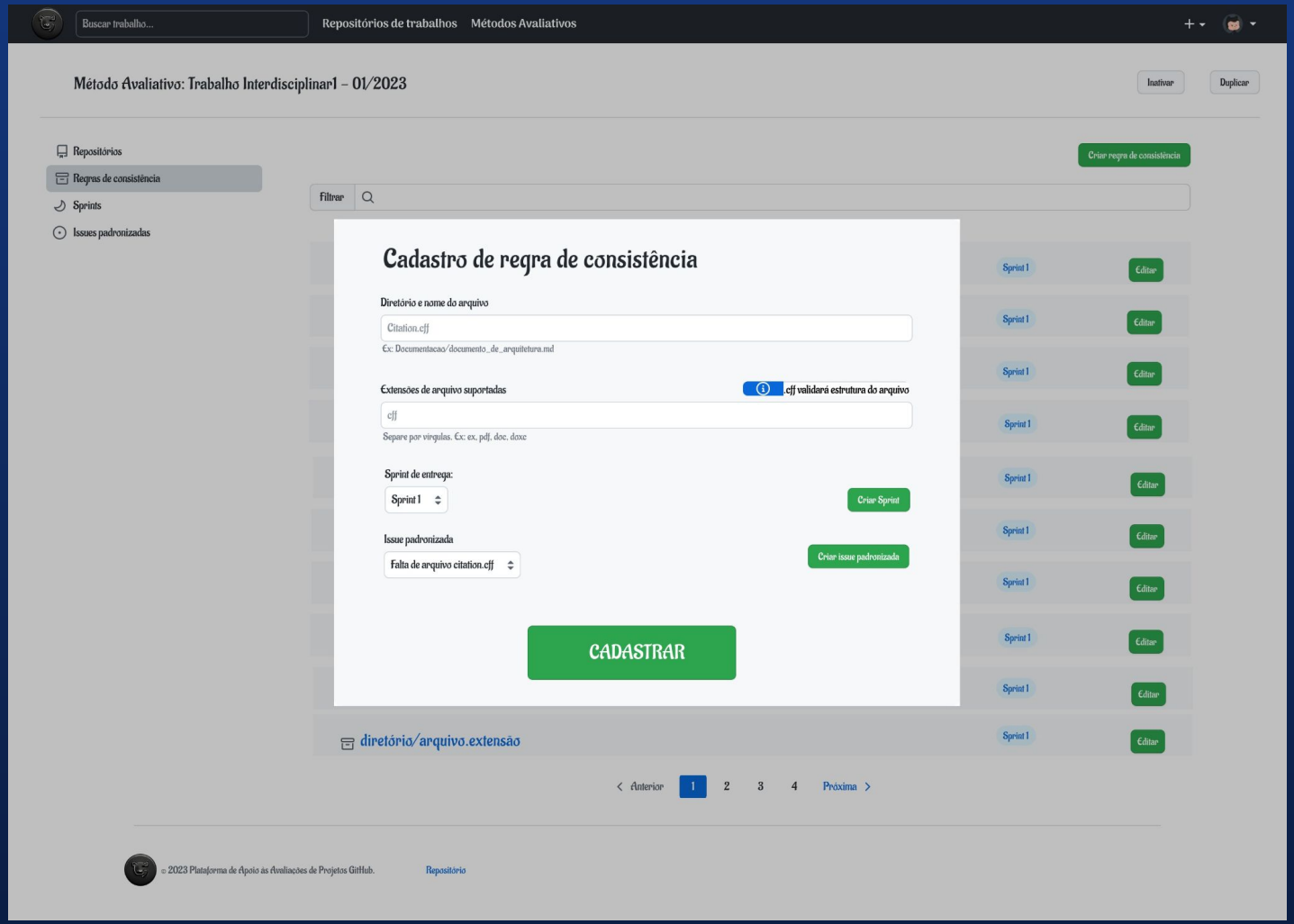
- O principal componente da plataforma é o Web Server.
- Ele consome informações do GitHub, armazena e recupera dados do SGBD, comunica com uma instância do SonarQube e expõe uma API HTTP.
- Essa API é consumida pelo Web App, o cliente da plataforma.
- O JobScheduler é o componente responsável por disparar ações periódicas da plataforma.

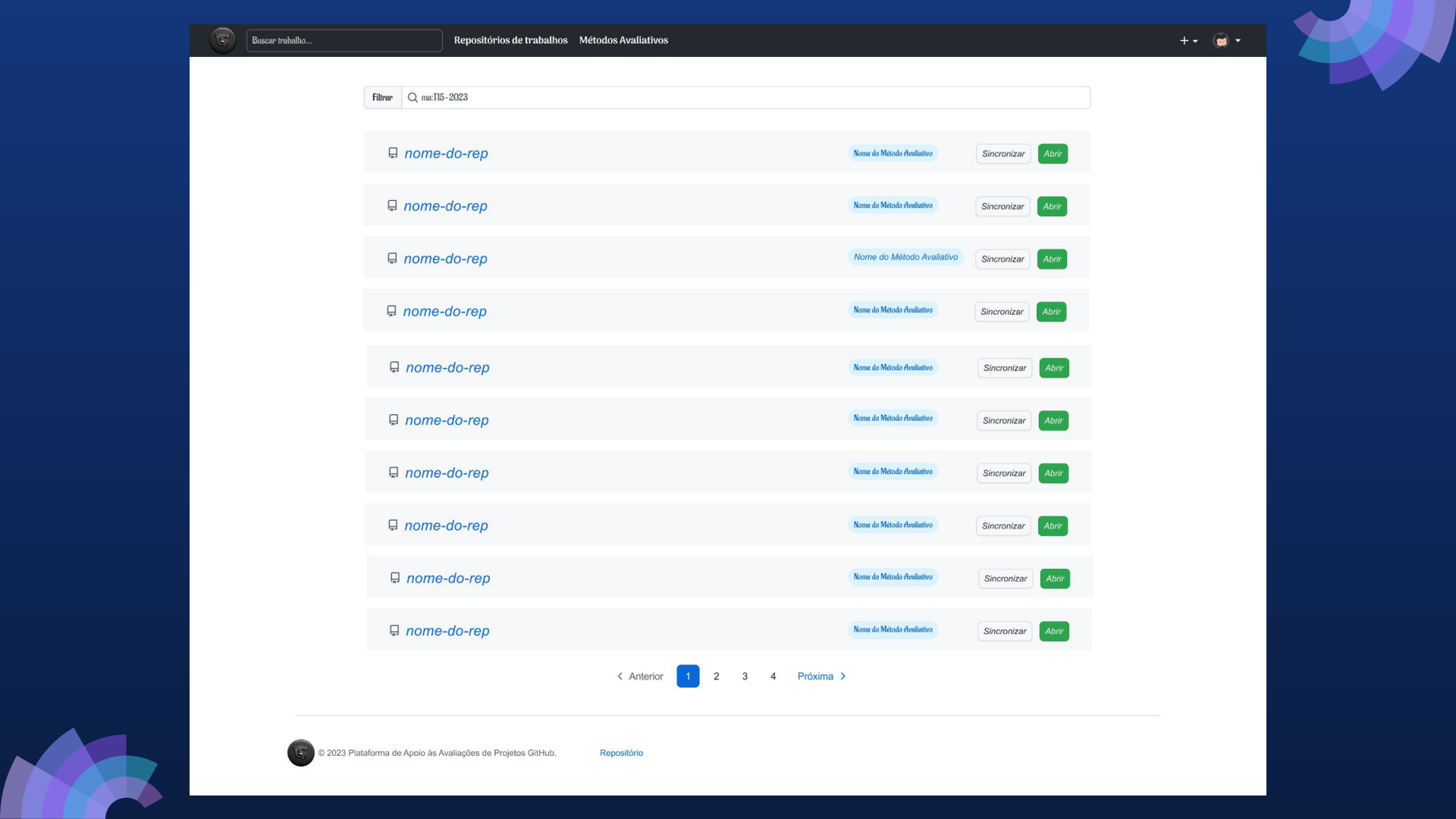
09

Projeto de Interfaces com o Usuário


Buscar trabalho...
Repositórios de trabalhos
Métodos Avaliativos
Filtrar
TrabalhoInterdisciplinar1 - 01/2023
TrabalhoInterdisciplinar2 - 01/2023
TrabalhoInterdisciplinar3 - 01/2023
TrabalhoInterdisciplinar4 - 01/2023
TrabalhoInterdisciplinar5 - 01/2023
TrabalhoInterdisciplinar6 - 01/2023
TrabalhoInterdisciplinar1 - 02/2023
TrabalhoInterdisciplinar2 - 02/2023
TrabalhoInterdisciplinar3 - 02/2023
TrabalhoInterdisciplinar4 - 02/2023
TrabalhoInterdisciplinar5 - 02/2023
Duplicar
Editar
Anterior
1
2
3
4
Próxima
© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub.
Repositório


Buscar trabalho...
Repositórios de trabalhos
Métodos Avaliativos
Método Avaliativo: Trabalho Interdisciplinar1 - 01/2023
Inativar
Duplicar
Repositórios
Regras de consistência
Sprints
Issues padronizadas
Adicionar repositório
Filtrar
nome-do-rep
Sincronizar
Abrir
Anterior
1
2
3
4
Próxima
© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub.
Repositório


Buscar trabalho...
Repositórios de trabalhos
Métodos Avaliativos
Método Avaliativo: Trabalho Interdisciplinar1 – 01/2023
Inativar
Duplicar
Repositórios
Regras de consistência
Sprints
Issues padronizadas
Criar regra de consistência
Filtrar
Cadastro de regra de consistência
Diretório e nome do arquivo
Citation.cff
Ex: Documentacao/documento_de_arquitetura.md
Extensões de arquivo suportadas
.cff validará estrutura do arquivo
cff
Separe por vírgulas. Ex: ex, pdf, doc, docx
Sprint de entrega:
Sprint 1
Criar Sprint
Issue padronizada
Falta de arquivo citation.cff
Criar issue padronizada
CADASTRAR
Sprint 1
Editar
diretório/arquivo.extensão
Anterior
1
2
3
4
Próxima
© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub.
Repositório


Buscar trabalho...
Repositórios de trabalhos
Métodos Avaliativos
Filtrar
ma:T15-2023
nome-do-rep
Nome do Método Avaliativo
Sincronizar
Abrir
Anterior
1
2
3
4
Próxima
© 2023 Plataforma de Apoio às Avaliações de Projetos GitHub.
Repositório

Buscar trabalho... Repositórios de trabalhos Métodos Avaliativos

nome-do-repositório

Nome do Método Avaliativo | Sincronizar

Métricas do repositório | Qualidade de código | Histórico de commits | Consistência | Configurações

Sincronização automática

Informação

- Contribuições de Commits
- Tipos de arquivos contribuidos
- Contribuições de Linhas de código
- Contribuições de Arquivos
- Qualidade da descrição dos commits
- Qualidade de código
- Issues fechadas

Branch

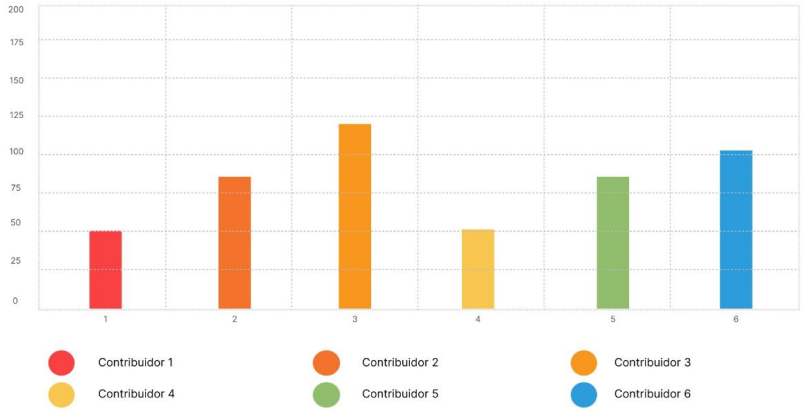
- master
- main
- a11y-improvements
- button-bug-fixes
- radio-input-components
- release-1.0.0
- text-input-implementation
- visual-design-tweaks

10 de fevereiro – 24 fevereiro, 2023

Sprint

Selecione uma sprint

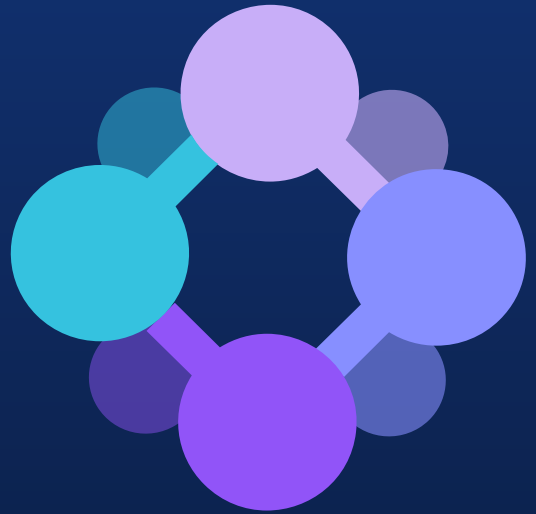
Commits por contribuidor



 Repositório

10

Cronograma do TCC II



Cronograma 2/2023

01/08

Sprint 1

Implementar casos de uso: UC2, UC15, UC16, UC31 e UC32.

16/08

Sprint 2

Implementar casos de uso: UC1, UC3, UC4, UC5, UC15, UC16, UC17, UC18, UC19, UC21, UC27, UC31, UC33, UC34, UC35, UC37 e UC38.

01/09

Sprint 3

Implementar casos de uso: UC1, UC3, UC4, UC5, UC6, UC7, UC8, UC9, UC21, UC22, UC25, UC33, UC34, UC35, UC37, UC36 e UC38.

16/09

Sprint 4

Implementar casos de uso: UC6, UC7, UC9, UC10, UC12, UC22, UC24, UC26, UC37, UC39, UC40, UC41 e UC42.

01/10

Sprint 5

Implementar casos de uso: UC23, UC25, UC27, UC28, UC29 e UC30.

16/10

Sprint 6

Implementar caso de uso: UC11.

01/11

Sprint 7

Correções e aprimoramentos com relação ao feedback da primeira versão estável da plataforma;

Pagamento de dívidas técnicas não intencionais.

16/11

Sprint 8

Documentar instruções de implantação e utilização;

Implantar sistema;

Escrita do Post-mortem.

30/11

FIM

OBRIGADO

Alunos:

Guilherme Gabriel Silva Pereira
Lucas Ângelo Oliveira Martins Rocha

Orientadores:

Cleiton Silva Tavares
José Laerte Pires Xavier Junior