

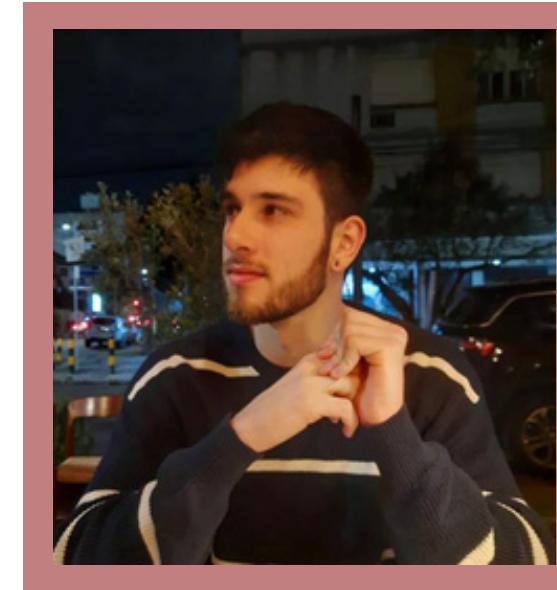


Técnicas de Construção de Programas

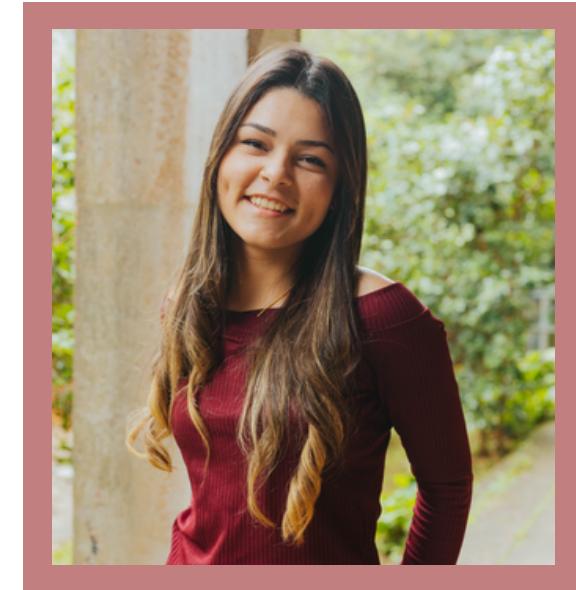
ARIGÓFLIX

Artur Webber – Bruna Rosa – Elano Tavares – João Dorneles

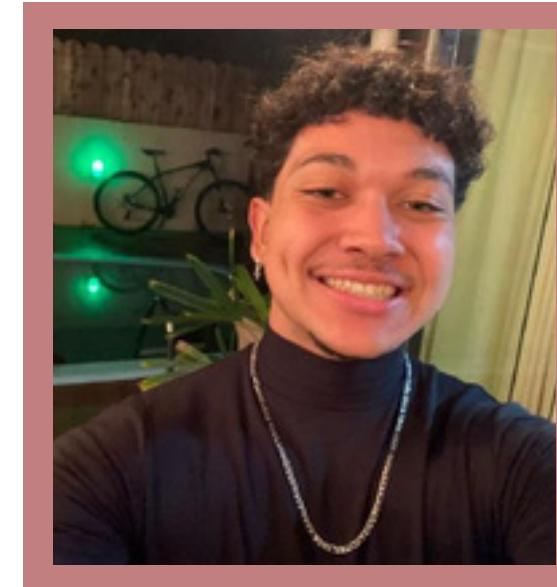
NOSO TIME



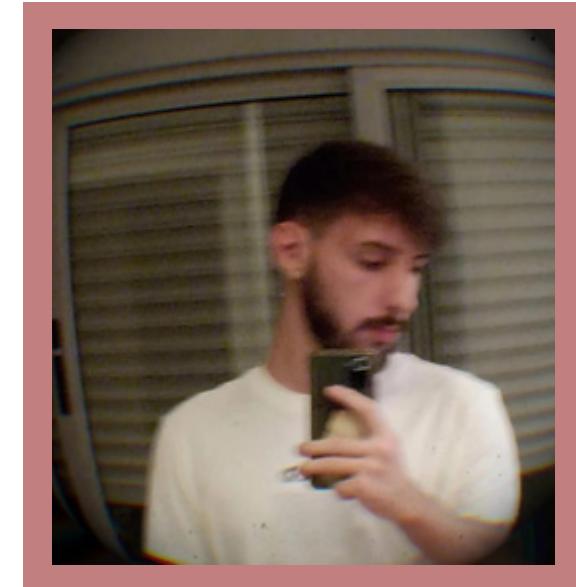
ARTUR WEBBER



BRUNA ROSA



ELANO TAVARES



JOÃO DORNELES

O que é o ARIGÓFLIX?

- **Proposta:** Plataforma centralizada para descoberta e avaliação de entretenimento (Filmes, Séries, Jogos e Livros).

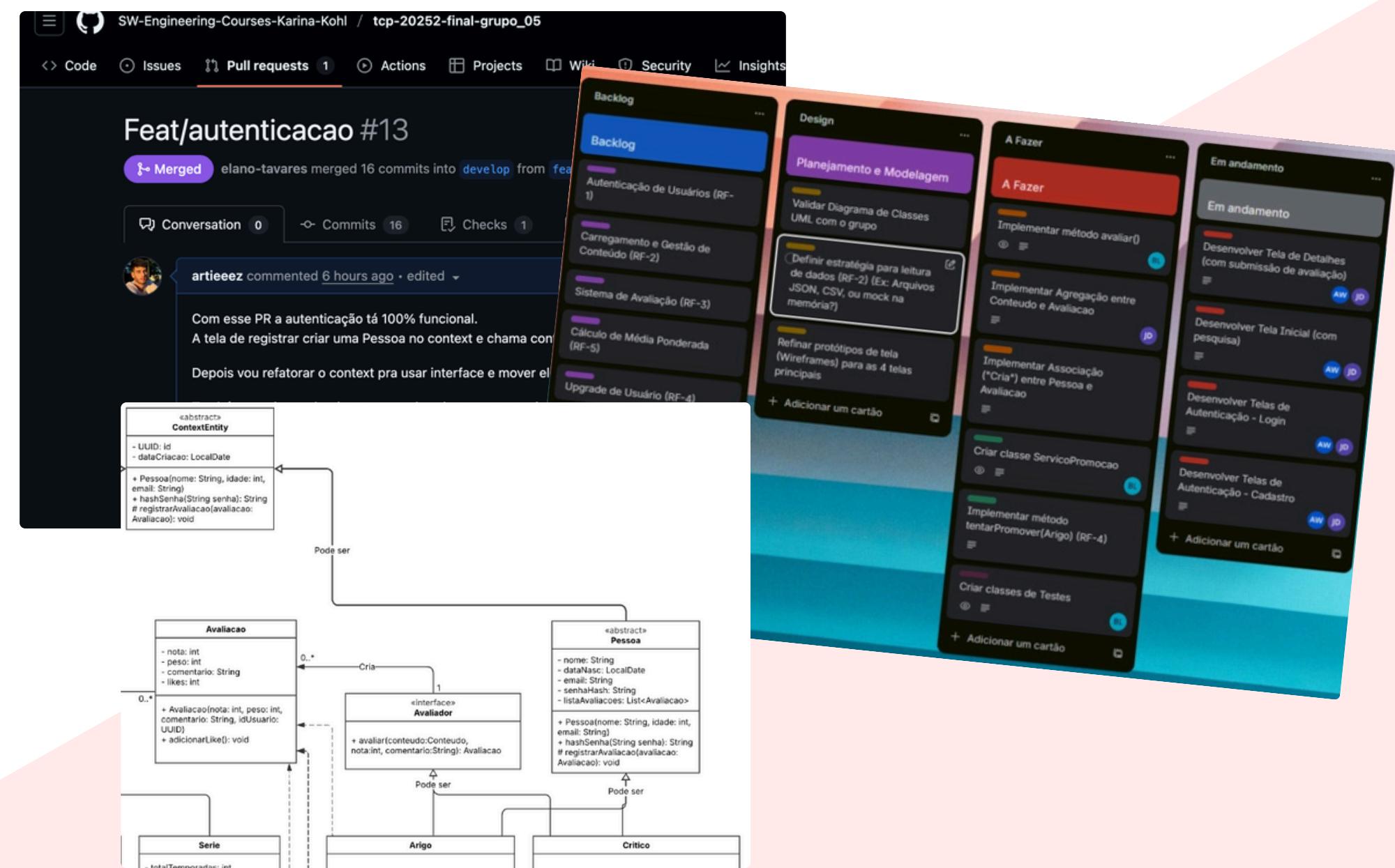
- **O Problema:** Agregadores comuns tratam todas as opiniões com o mesmo peso, misturando reviews casuais com análises técnicas.

- **Nossa Solução:** Um sistema de Média Ponderada que distingue dois perfis de usuários:
 1. Arigó: Usuário comum (peso padrão).
 2. Crítico: Usuário especializado (peso maior na nota final).

FERRAMENTAS

Foram usadas três ferramentas:

1. Github para colaborar e revisar pull requests;
2. Um quadro kanban no Trello para organizar as tarefas;
3. LucidChart para produção do UML;



Organização

Branches:

- main
- develop
- feat/feature-name

Fluxo de trabalho:

- Criar branch feat
- Manter atualizada com develop
- PR para develop ao terminar a implementação
- Revisão por par e aceitação do PR

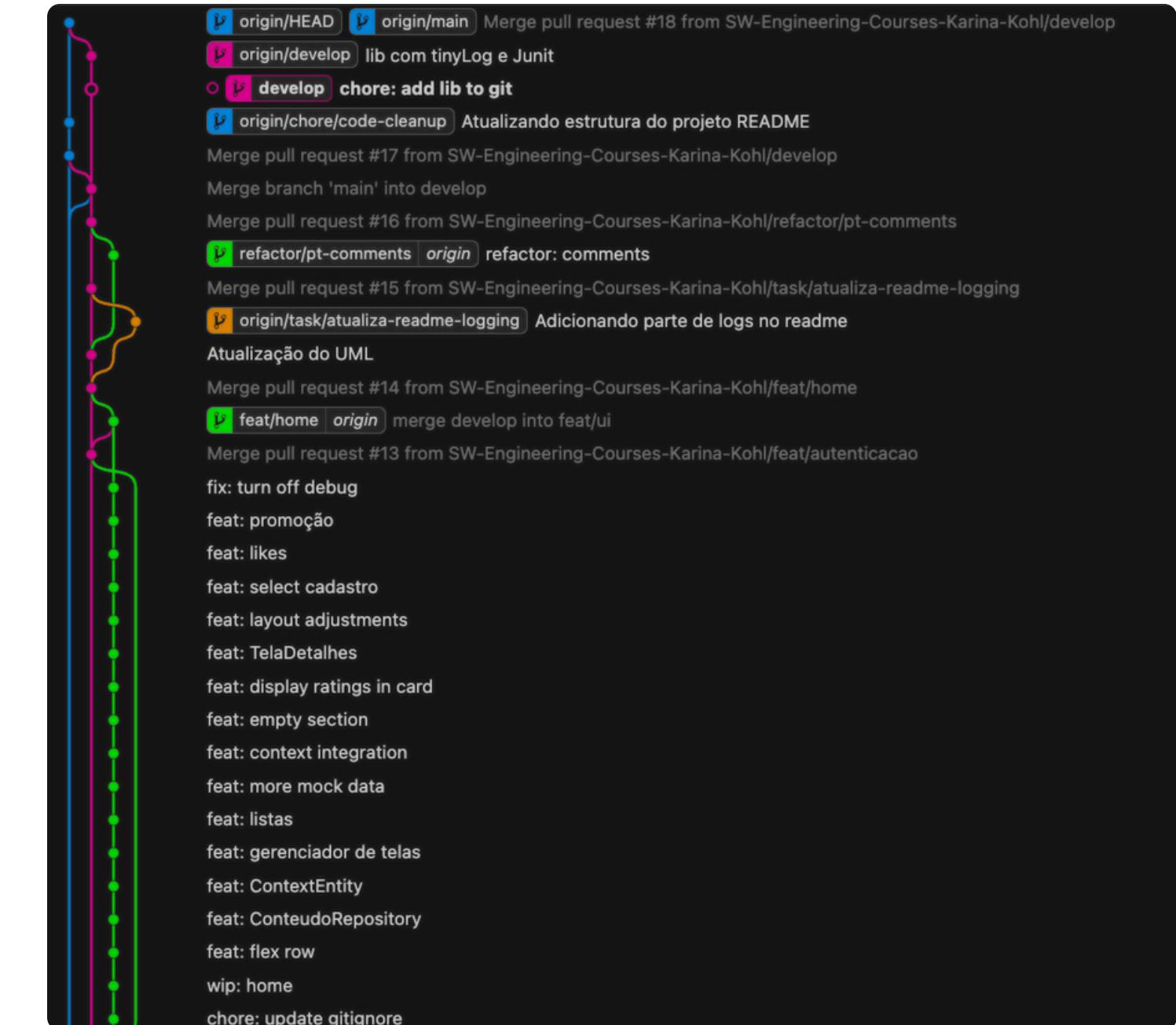
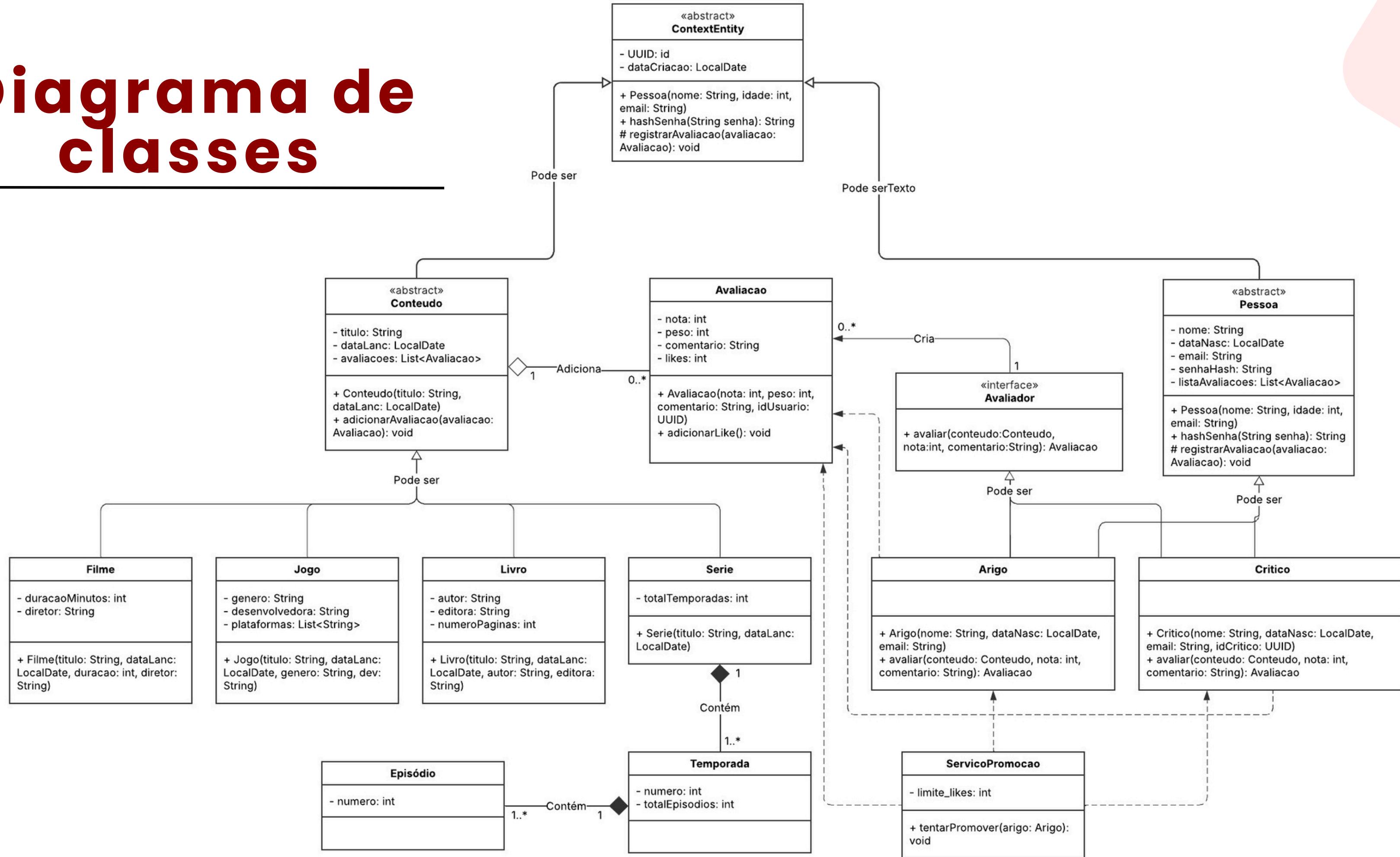


Diagrama de classes



Estrutura do Projeto

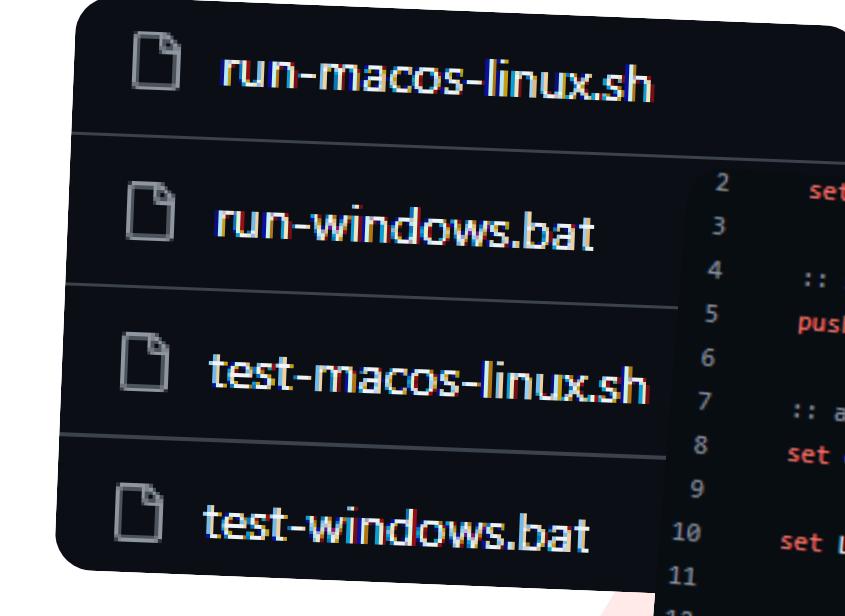
```
TCP-20252-final-grupo-05
|
+-- config           <-- Arquivos de configuração (tinylog.properties)
+-- lib              <-- Dependências externas (Tinylog, JUnit)
+-- scripts          <-- Scripts de automação (build/test para Windows e Linux/macOS)
+
+-- src
    |
    +-- main   <-- Pacote principal
        |
        +-- Main.java      <-- Classe principal (Ponto de entrada)
        |
        +-- models          <-- (Entidades do domínio)
            |-- ContextEntity.java (abstrata base)
            |-- Pessoa.java     (abstrata)
            |-- Arigo.java
            |-- Critico.java
            |-- Avaliador.java (interface)
            |-- Conteudo.java  (abstrata)
            |-- Filme.java
            |-- Serie.java
            |-- Livro.java
            |-- Jogo.java
            |-- Temporada.java
            |-- Episodio.java
            |-- Avaliacao.java
        |
        +-- service         <-- (Lógica de negócio e persistência em memória)
            |-- Context.java    (container dos repositórios e estado global)
            |-- BaseRepository.java
            |-- ConteudoRepository.java
            |-- CarregadorDeDados.java
            |-- ServicoPromocao.java
            |
            +-- autenticacao   <-- (Subpacote de segurança)
                |-- Autenticacao.java (interface)
                |-- ServicoAutenticacao.java
                |-- Exceptions... (CredenciaisInvalidas, EmailJaCadastrado)
        |
        +-- ui (ou view)    <-- (Interface Gráfica Swing)
            |-- GerenciadorTelas.java (Controlador de navegação)
            |-- TelaLogin.java
            |-- TelaCadastro.java
            |-- TelaInicial.java
            |-- TelaDetalhes.java
            |-- TelaFilmeLista.java
            |-- TelaSerieLista.java
            |-- TelaLivroLista.java
            |-- TelaJogoLista.java
        |
        +-- resources
            +-- data           <-- (Arquivos CSV: filmes, series, usuarios, etc.)
        |
        +-- test             <-- (Testes Unitários JUnit para models e services)
```

Clean Code

```
15
16     public class CarregadorDeDados {
17
18         private static final String SEPARADOR = ";";
19         private static final DateTimeFormatter FORMATO_DATA = DateTimeFormatter.ofPattern("yyyy-MM-dd");
20
21         // --- Constantes para Caminhos de Arquivos ---
22         private static final String CAMINHO_USUARIOS = "data/usuarios.csv";
23         private static final String CAMINHO_FILMES = "data/filmes.csv";
24         private static final String CAMINHO_LIVROS = "data/livros.csv";
25         private static final String CAMINHO_JOGOS = "data/jogos.csv";
26         private static final String CAMINHO_SERIES = "data/series.csv";
27         private static final String CAMINHO_EPISODIOS = "data/episodios.csv";
28
29         // --- Constantes para Avaliações ---
30         private static final int PESO_AVALIACAO = 1;
31         private static final int NOTA_MINIMA = 1;
32         private static final int NOTA_MAXIMA = 5;
33
34         /**
35          * Construtor com ID opcional. Se o ID for null, gera um UUID aleatório.
36          */
37         @Override
38         public Avaliacao avaliar(Conteudo conteudo, int nota, String comentario) {
39             if (nota < 1 || nota > 5) {
40                 if (nota < NOTA_MINIMA || nota > NOTA_MAXIMA) {
41                     Logger.warn("Usuário {} tentou avaliar conteúdo {} com nota inválida: {}.", this.getId(),
42                             conteudo != null ? conteudo.getTitulo() : "<conteúdo nulo>", nota);
43                     throw new IllegalArgumentException("A nota deve ser entre 1 e 5");
44                 }
45             }
46             Avaliacao novaAvaliacao = new Avaliacao(nota, PESO_AVALIACAO, comentario, this.getId());
47             return novaAvaliacao;
48         }
49
50     }
```

```
15
16     public class Arigo extends Pessoa implements Avaliador {
17
18         private static final int PESO_AVALIACAO = 1;
19         private static final int NOTA_MINIMA = 1;
20         private static final int NOTA_MAXIMA = 5;
21
22         /**
23          * Construtor com ID opcional. Se o ID for null, gera um UUID aleatório.
24          */
25         @Override
26         public Avaliacao avaliar(Conteudo conteudo, int nota, String comentario) {
27             if (nota < 1 || nota > 5) {
28                 if (nota < NOTA_MINIMA || nota > NOTA_MAXIMA) {
29                     Logger.warn("Usuário {} tentou avaliar conteúdo {} com nota inválida: {}.", this.getId(),
30                             conteudo != null ? conteudo.getTitulo() : "<conteúdo nulo>", nota);
31                     throw new IllegalArgumentException("A nota deve ser entre 1 e 5");
32                 }
33             }
34             Avaliacao novaAvaliacao = new Avaliacao(nota, PESO_AVALIACAO, comentario, this.getId());
35             return novaAvaliacao;
36         }
37
38     }
```

Automação/Execução



```
run-macos-linux.sh
run-windows.bat
test-macos-linux.sh
test-windows.bat
```

```
2  setlocal enabledelayedexpansion
3
4  :: sobe para a raiz do projeto
5  pushd %~dp0..
6
7  :: adicionamos "bin" ao classpath no lugar do "."
8  set CLASSPATH=lib\junit-platform-console-standalone-1.10.2.jar;lib\tinylog-api-2.7.0.jar;lib\ti
9
10 set LOG_CONFIG=config\tinylog.properties
11
12 echo.
13 echo [1/3] Preparando Ambiente...
14
15 :: Cria pasta bin para os .class (para não sujar a raiz)
16 if not exist "bin" mkdir bin
17
18 :: Copia os CSVs da pasta resources\data para a pasta bin
19
20 :: sobe para a raiz
21 pushd %~dp0..
22
23 :: cria a pasta bin se não existir
24 if not exist "bin" mkdir bin
25
26 :: classpath para compilação
27 set CLASSPATH=lib\tinylog-api-2.7.0.jar;lib\tinylog-impl-2.7.0.jar;src\main;src\resources;;
28
29 set LOG_CONFIG=config\tinylog.properties
30
31 echo.
32 echo [1/3] Compilando para pasta temporaria 'bin'...
33 javac -cp "%CLASSPATH%" -d bin src\main\*.java src\main\ui\*.java src\main\models\*.java src\main\service\*.java src\main\service\autenticacao\*.java
34
35 if %errorlevel% neq 0 (
36     echo [ERRO] Falha na compilacao.
37     rmdir /s /q bin
38     pause
39     popd
40     exit /b 1
41 )
42
43 echo.
44 echo [2/3] Executando Aplicacao...
45
46 :: precisa copiar os CSVs aqui tambem para a aplicação principal usa-los
47 if exist "src\resources\data\*.csv" copy /Y "src\resources\data\*.csv" .
48
49 :: CLASSPATH para execução: precisa incluir bin\ onde estão os .class compilados
50 set RUNTIME_CLASSPATH=lib\tinylog-api-2.7.0.jar;lib\tinylog-impl-2.7.0.jar;bin;src\resources;;
51
52 java -Dtinylog.configuration="%LOG_CONFIG%" -cp "%RUNTIME_CLASSPATH%" main.Main
53
54 echo.
55 echo [3/3] Limpando arquivos temporarios...
56 if exist "bin" rmdir /s /q bin
57 del *.csv >nul 2>&1
```

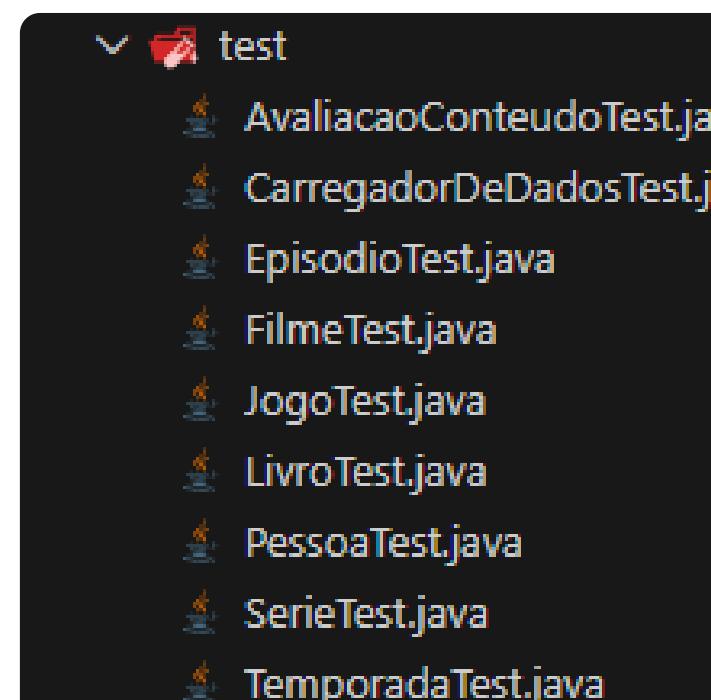
Testes

Ferramenta:

- JUnit

Benefícios:

- Garantia da qualidade do software e a conformidade com os requisitos



```
83 | ┌── JUnit Jupiter ✓
84 | | └── SerieTest ✓
85 | |   | ┌── construtorLancaQuandoCamposNulos() ✓
86 | |   | ┌── calculaMediaPonderadaSerieComTemporadaEpisodio() ✓
87 | |   | └── adicionarTemporadaNulaLanca() ✓
88 | └── EpisodioTest ✓
89 |   | ┌── construtorLancaQuandoTituloNulo() ✓
90 |   | ┌── construtorLancaQuandoDuracaoInvalida() ✓
91 |   | └── gettersRetornamValores() ✓
92 | └── AvaliacaoTest ✓
93 |   | ┌── deveIncrementarLikes() ✓
94 |   | ┌── deveCriarAvaliacaoComDadosValidos() ✓
95 |   | └── naoDevePermitirPesoZeroOuNegativo() ✓
96 | └── LivroTest ✓
97 |   | ┌── construtorLancaQuandoCamposNulos() ✓
98 |   | └── calculaMediaPonderadaLivro() ✓
99 | └── ServicoAutenticacaoTest ✓
100 |   | ┌── naoDevePermitirEmailDuplicado() ✓
101 |   | ┌── deveRegistrarCriticoQuandoSolicitado() ✓
102 |   | ┌── autenticacaoFalhaComSenhaErrada() ✓
103 |   | └── autenticacaoComSucesso() ✓
104 |   | └── deveRegistrarNovoUsuarioComSucesso() ✓
105 | └── ConteudoRepositoryTest ✓
106 |   | ┌── buscaRetornaTodosSeTermoVazio() ✓
107 |   | ┌── buscaIgnoraMaiusculasEMinusculas() ✓
108 |   | ┌── buscaParcialEncontraResultados() ✓
109 |   | ┌── buscaPorTermoExato() ✓
110 |   | └── buscaSemResultadosRetornaListaVazia() ✓
111 |   └── PessoaTest ✓
112 |     | ┌── verificaAvaliacaoFoiAdicionada() ✓
113 |     | ┌── naoPermitirNotaMaiorQueCinco() ✓
114 |     | └── naoPermitirNotaMenorQueUm() ✓
115 |   └── CarregadorDeDadosTest ✓
116 |     | ┌── encontrarSeriePeloTitulo() ✓
117 |     | ┌── carregarCriticoDoArquivo() ✓
118 |     | ┌── deveCarregarDadosDeEpisodios() ✓
119 |     | └── carregarArigoDoArquivo() ✓
120 |     | └── deveCriarTemporadasCorretamente() ✓
```

Logs

Biblioteca Utilizada:

- TinyLog

Motivo:

- É leve, muito simples de implementar e oferecia o que precisávamos

Objetivo:

- O logging foi adicionado para tornar o comportamento da aplicação mais observável
 1. O código utiliza chamadas como **Logger.info(...)**, **Logger.warn(...)** e **Logger.error(e, ...)** em classes de domínio, serviços e telas.

Logs

Exemplos de utilização no código:

- **INFO** (Service ServicoPromocao.java):

```
Logger.info("Iniciando tentativa de promoção para o usuário com ID {}.", arigo.getId());
```

- **DEBUG** (UI TelaLogin.java):

```
Logger.debug("Navegando da tela de login para a tela de cadastro.");
```

- **WARN** (Model Arigo.java)

```
Logger.warn("Usuário {} tentou avaliar conteúdo {} com nota inválida: {}.", this.getId(),  
conteudo.getTitulo(), nota);
```

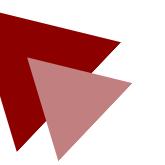
- **ERROR** (Model Conteudo.java):

```
Logger.error("Título inválido ao criar conteúdo: '{}'.", titulo);
```

Dificuldades



- | | |
|--|---|
| <ul style="list-style-type: none">• O GitHub: A principal dificuldade, que aconteceu logo nas primeiras etapas do desenvolvimento, costumes e vícios de outras experiências levaram a alguns erros fatais. | <ul style="list-style-type: none">• Formalização e comunicação: Outra dificuldade perceptível foi a falta de costume em efetuar uma boa documentação. |
| <ul style="list-style-type: none">• Adaptação com a linguagem Java: Inicialmente o desenvolvimento se concretizou de forma mais lenta devido a falta de experiência com a linguagem. | |



**OBRIGADO PELA
ATENÇÃO!**