

Métodos Formais de Engenharia de Software



AlbumFlow

Uma plataforma para fotógrafos

Equipe do projeto:



Felipe Hidequel



Leonardo Inácio



Maria Alice

Sobre o projeto

Nossa proposta visa modernizar o processo de entrega e seleção de fotos realizadas por fotógrafos profissionais, criando uma plataforma online que substitua métodos tradicionais e descentralizados, oferecendo uma solução mais prática, eficiente e profissional para fotógrafos e seus clientes.



Mudanças finais

- Requisitos funcionais mais específicos e reduzidos. Por exemplo:
 - [RF01] Criar Álbum;
 - [RF02] Deletar Álbum;
 - [RF03] Listar Álbuns;
 - [RF04] Visualizar Álbum
 - [RF05] Criar Foto;
 - [RF06] Excluir Foto;
 - [RF07] Visualizar informações de Usuário;
 - [RF08] Realizar Login

Mudanças finais

- Alteração nas nomenclaturas dos atributos, classes e tabelas do diagrama de classe e modelo de dados;

Modelo de Dados

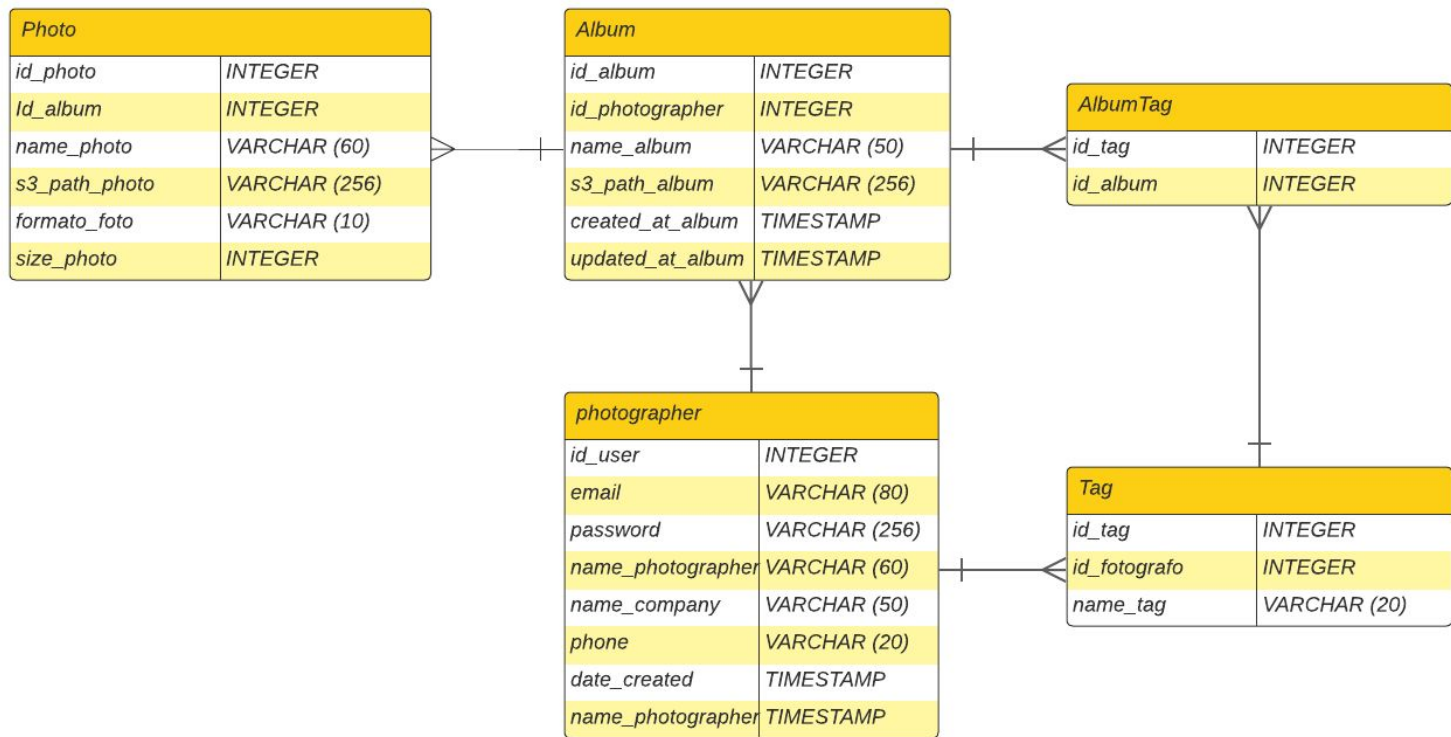
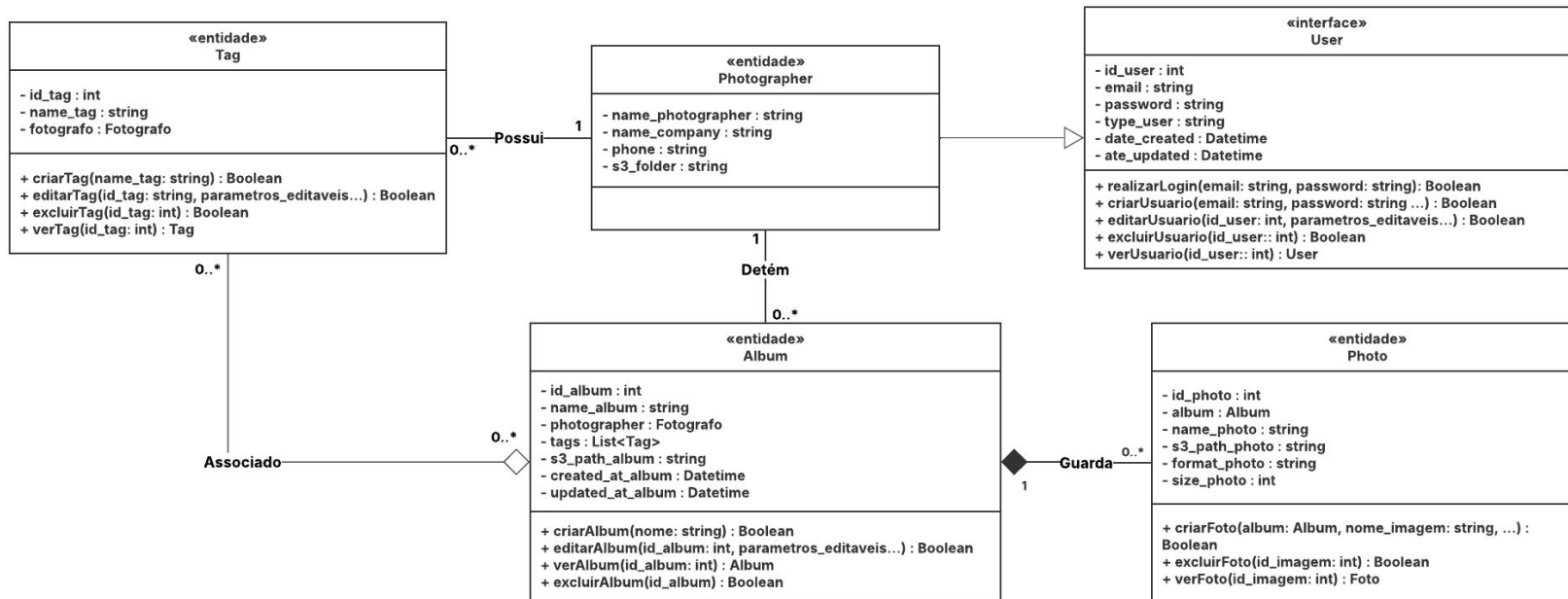



Diagrama de Classe



Telas desenvolvidas



The logo for Album Flow Photography features a yellow camera shutter icon on the left. To its right, the word "ALBUM" is in small white capital letters, "FLOW" is in large white capital letters, and "Photography" is in smaller yellow lowercase letters.

Login

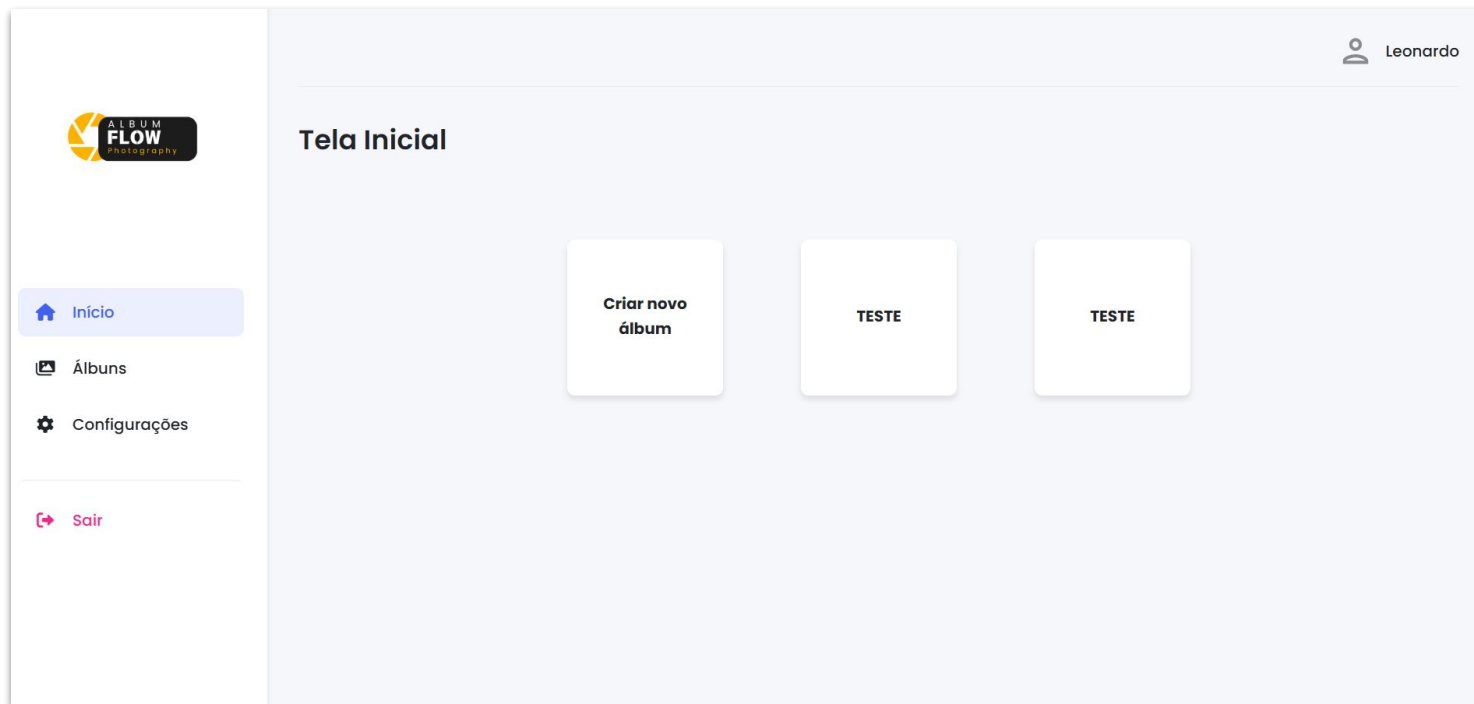
Usuário

Senha


Entrar





[Esqueci minha senha](#) • [Criar conta](#)

Telas desenvolvidas



Telas desenvolvidas



-  Início
-  Álbuns
-  Configurações
-  Sair

Criando um álbum

Preencha as informações abaixo:

Campos marcados com * são obrigatórios

* Nome do Álbum

Máximo de 100 caracteres

Tags (opcional)

Arraste suas fotos aqui ou clique para selecionar. Formatos aceitos: JPG, PNG, GIF. Máx. 10MB por arquivo.

Nenhum arquivo escolhido

Você pode selecionar múltiplas fotos

Testes e métricas de qualidade

Os testes implementados validam o fluxo completo da API relacionada ao modelo do projeto, incluindo: a serialização dos dados, os endpoints REST e as operações CRUD básicas, com verificações tanto para casos de sucesso quanto para tratamentos de erro.

```
def test_photographer_serializer_valid_data(self):
    serializer = PhotographerSerializer(instance=self.photographer)
    self.assertEqual(self.expected_data, serializer.data)

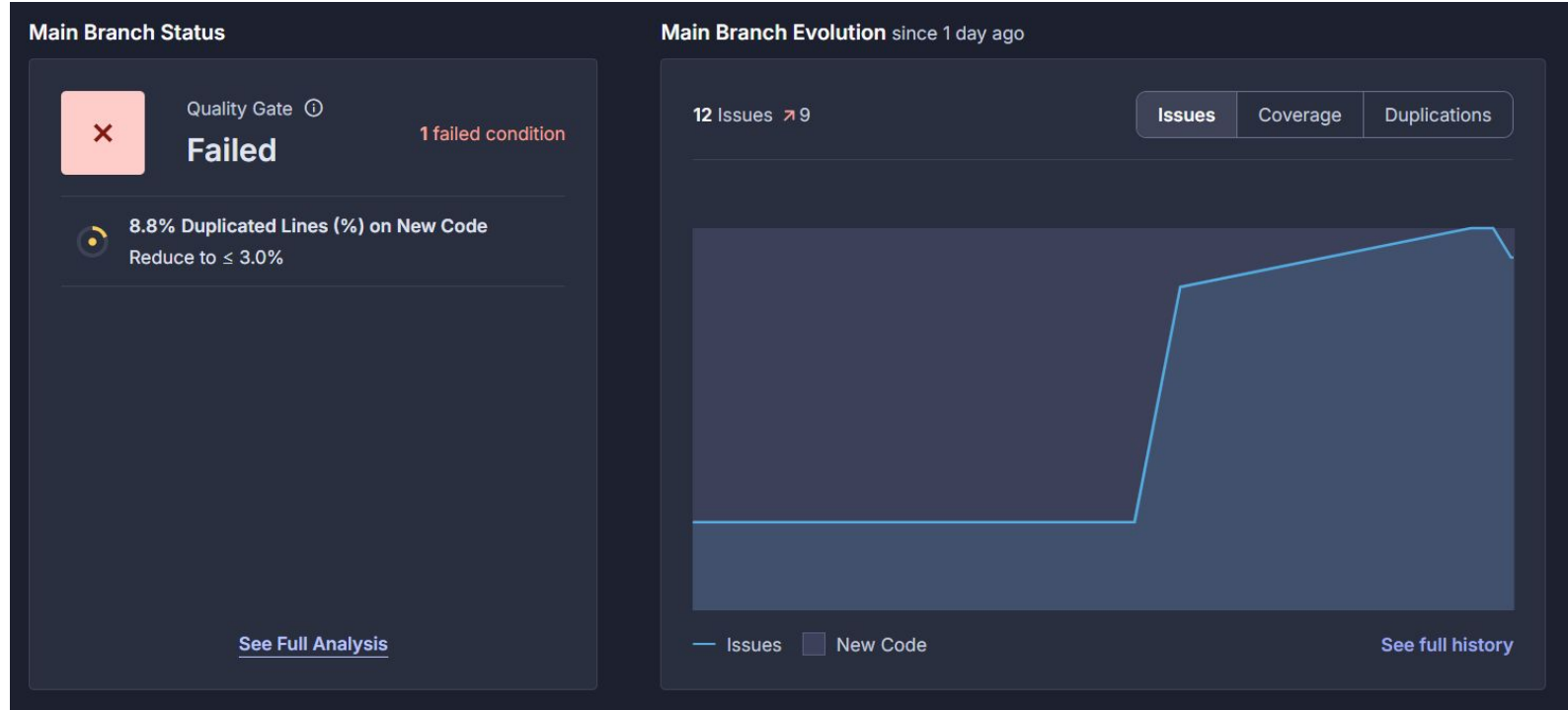
def test_photographer_get_via_api(self):
    url = f"/api/users/{self.user.id}/"
    response = self.client.get(url, format="json")
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertEqual(response.data["name_photographer"], "Hidequel")
    self.assertEqual(response.data["name_company"], "fotos massa")
```

Testes e métricas de qualidade

Com o uso do SonarQube instalado no repositório e monitorando o código presente na Branch main, pode-se observar e obter algumas métricas em diferentes versões. Por exemplo

- ★ **Nenhum novo bug é introduzido** → Classificação de confiabilidade: A
- ★ **Nenhuma nova vulnerabilidade é introduzida** → Classificação de segurança: A
- ★ **O código novo possui dívida técnica limitada** → Classificação de manutenibilidade: A
- ★ **O código novo tem cobertura de testes suficiente** → Cobertura é maior ou igual a 80,0%
- ★ **O código novo possui duplicações limitadas** → Linhas duplicadas (%) são menores ou iguais a 3,0%

Testes e métricas de qualidade



Especificação Formal

Seguindo o plano de especificação formal, apresenta na unidade 2 da disciplina, foi posto em prática a construção da especificação formal de todos os requisitos do sistema. A especificação formal final pode ser consulta [aqui](#).

Especificação Formal

[FOTOGRAFO, TAG, DATA, FOTO]

Album

id_album : \mathbb{N}

uuid_album : UIID

fotografo : FOTOGRAFO

nome_album : seq₁ Character

link : seq₁ Character

data_criacao : DATA

data_atualizacao : DATA

listaFotos : \mathbb{P} FOTO

tags : \mathbb{P} TAG

AlbumVazio

\exists Album

listaFotos' = \emptyset

Especificação Formal

[DOMINIO: seq₁ Caracter]

CriarAlbum

Δ Album

nome_album? : seq₁ Caracter

tags? : \mathbb{P} TAG

listaFotos? : \mathbb{P} FOTO

uuid_novo! : UUID

link_novo! : seq₁ Caracter

retorno! : {OK, NomeDuplicado}

$\forall a : \text{Album} \bullet a.\text{nome_album} \neq \text{nome_album?}$

$\Rightarrow \text{retorno!} = \{\text{OK}\} \wedge \text{uuid_novo!} \notin \{a.\text{uuid_album} \mid a : \text{Album}\}$

$\exists a : \text{Album} \bullet a.\text{nome_album} = \text{nome_album?}$

$\Rightarrow \text{retorno!} = \{\text{NomeDuplicado}\}$

$\text{retorno!} = \text{OK} \Rightarrow \text{link_novo!} = \text{DOMINIO} \frown \text{" / " } \frown \text{uuid_novo!}$

ListarAlbums

\exists Album

fotografo?: FOTOGRAFO

retorno! : {NenhumAlbum} \cup \mathbb{P} Album

$\{a : \text{Album} \mid a.\text{fotografo} = \text{fotografo?}\} = \emptyset$

$\Rightarrow \text{retorno!} = \text{NenhumAlbum}$

$\{a : \text{Album} \mid a.\text{fotografo} = \text{fotografo?}\} \neq \emptyset$

$\Rightarrow \text{retorno!} = \{a : \text{Album} \mid a.\text{fotografo} = \text{fotografo?}\}$

Especificação Formal

[TIPO, EMAIL]

Fotografo

$id_usuario : \mathbb{N}$
 $email : EMAIL$
 $senha : seq1 \text{ Caracter}$
 $tipo_usuario : TIPO$
 $data_criacao : DATA$
 $data_atualizacao : DATA$
 $nome_fotografo : seq1 \text{ Caracter}$
 $nome_empresa : seq1 \text{ Caracter}$
 $telefone : \mathbb{N}$

VisualizarInformacoesDeUsuario

$\exists f : \text{Fotografo}$

$\forall f : \text{Fotografo} \bullet f.id_usuario = id_usuario$

Partiu pra DEMO do AlbumFlow!

<https://github.com/LeonardoIGD/Projeto-AlbumFlow>

Muito obrigado pela atenção!

<https://github.com/LeonardoIGD/Projeto-AlbumFlow>