

**Bacharel em Engenharia de Software**

**Disciplina Modelose técnias de engenharia de Software**

# RELATÓRIO VISÃO E ANALISE DE PROJETOS

|  |  |
| --- | --- |
| **Alunos/RA**: |  |
| Caroline Araldi | 172320037 |
| Marcelo Augusto Dos Santos | 172313472 |
| Nicole Karolyne Balão | 172313357 |
| João Victor | 942321478 |

**Curitiba, PR**

**2023**

1. **Sumário**
2. **Introdução...................................................................................................3**
3. **Definição de diagramas.............................................................................4**

**3.1 Diagrama de classes...........................................................................5**

1. **Código em Python.....................................................................................6**
2. **Referências.................................................................................................7**

# 2. Introdução

A Análise Orientada a Objetos (OOA) é uma abordagem no desenvolvimento de sistemas que se baseia no conceito de objetos interagindo entre si para realizar tarefas computacionais.

O ponto de partida na OOA é criar um modelo descritivo, muitas vezes chamado de "proposta técnica", que contém informações detalhadas sobre o problema a ser resolvido, os objetivos do projeto, casos de uso, requisitos funcionais e não funcionais, atividades do sistema e soluções propostas.

Usando padrões da Linguagem de Modelagem Unificada (UML), os requisitos do sistema são especificados, e são construídos artefatos necessários para a documentação adequada.

Ao longo do ciclo de vida do desenvolvimento, são gerados diversos diagramas que representam os objetos de análise, como diagramas de caso de uso, de classe, de objeto, de componente, de implantação, de atividade, de estado, de colaboração e de sequência. Esses diagramas não são apenas representações visuais, mas ferramentas essenciais para compreender e orientar o desenvolvimento do sistema, proporcionando clareza e estrutura ao projeto.

**3. Definição de Diagramas**

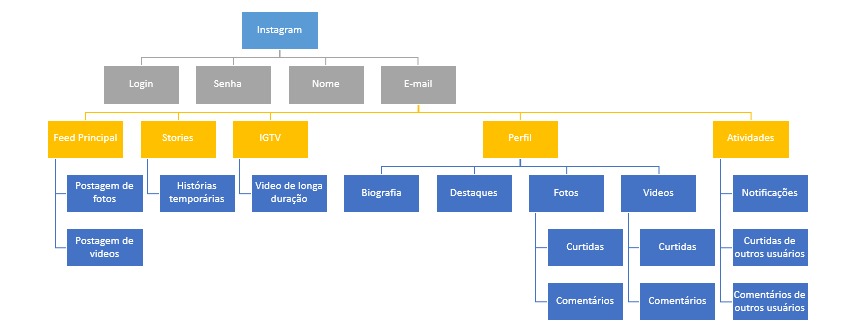
**Diagrama de classe:** Um diagrama de classes é uma representação visual na UML (Unified Modeling Language) que ilustra as classes em um sistema e os relacionamentos entre elas. Cada classe é geralmente representada por um retângulo dividido em três partes: o nome da classe, seus atributos e métodos. As linhas conectando as classes indicam associações, heranças ou dependências. O diagrama de classes proporciona uma visão estruturada e hierárquica do sistema, facilitando a compreensão da organização e interações entre as entidades.

**Diagrama de pacotes:** Um diagrama de pacotes na UML representa a organização e agrupamento lógico de elementos em um sistema, destacando a estrutura de pacotes e suas relações. Cada pacote contém classes, subpacotes ou interfaces, mostrando a modularidade do sistema. As linhas conectando os pacotes indicam dependências ou associações. Esse tipo de diagrama facilita a compreensão da arquitetura geral do sistema, permitindo uma visão modular e organizada da distribuição de componentes

**Diagrama de componentes:** O diagrama de componentes na UML representa a estrutura e a organização dos componentes em um sistema. Os componentes podem ser bibliotecas, módulos ou subsistemas, e suas dependências são mostradas por linhas de ligação. Esse diagrama destaca a relação entre diferentes partes do sistema, enfatizando a interdependência e a comunicação entre os componentes. Ele fornece uma visão clara da arquitetura e facilita a compreensão da distribuição de responsabilidades e funcionalidades no sistema.

**3.1 Diagrama de Classes**

Para realizar o nosso trabalho, escolhemos o diagrama de classes. Assim como foi feito no trabalho anterior, usamos como base o “Instagram”.

****

**4. Código em Python**

Realizamos o codigo na linguagem de Python:

class Usuario:

def \_init\_(self, id\_usuario, nome, email):

self.id\_usuario = id\_usuario

self.nome = nome

self.email = email

self.postagens = []

def criar\_postagem(self, conteudo):

postagem = Postagem(conteudo, self)

self.postagens.append(postagem)

return postagem

class Postagem:

def \_init\_(self, conteudo, autor):

self.conteudo = conteudo

self.autor = autor

self.comentarios = []

def adicionar\_comentario(self, comentario):

self.comentarios.append(comentario)

class Comentario:

def \_init\_(self, conteudo, autor):

self.conteudo = conteudo

self.autor = autor

# Exemplo de Uso

usuario1 = Usuario(1, "Carol", "araldicarol")

postagem1 = usuario1.criar\_postagem("Minha primeira postagem!")

comentario1 = Comentario("Ótimo post!", usuario1)

postagem1.adicionar\_comentario(comentario1)

**5. Referências**

**[1] https://jacksongomesbr.gitbooks.io/programacao-orientada-a-objetos-com-python/content/projeto-orientado-a-objeto.html**

**[2] http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/intro/intro.htm**

**[3] https://www.teclogica.com.br/ooa-object-oriented-analysis-analise-orientada-a-objetos/#:~:text=A%20An%C3%A1lise%20Orientada%20a%20Objetos,descritivo%20contendo%20informa%C3%A7%C3%B5es%20do%20projeto.**