Análise de projeto SOLID

Projeto: Smart Menu

Autor: Adaias Souza Por: Talisson Freitas Avila

O que é o SOLID?

O acrônimo SOLID representa os cinco princípios que facilitam o processo de desenvolvimento — o que facilita a manutenção e a expansão do software.

Estes princípios são fundamentais na programação orientada a objetos e podem ser aplicados em qualquer linguagem que adote este paradigma.

Os 5 princípios são:

- S — *Single Responsibility Principle* (Princípio da responsabilidade única)
- O — *Open-Closed Principle* (Princípio Aberto-Fechado)
- L — *Liskov Substitution Principle* (Princípio da substituição de Liskov)
- I — *Interface Segregation Principle* (Princípio da Segregação da Interface)
- D — *Dependency Inversion Principle* (Princípio da inversão da dependência)

S – *Single Responsability Principle* (ou princípio da responsabilidade única)

Uma classe deve ter um, e somente um, motivo para mudar.

Usando o arquivo ‘views.py’ dentro da pasta ‘usuarios’, observe a figura abaixo (Figura 1):

```python
def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
    else:
        username = request.POST.get('username')
        email = request.POST.get('email')
        senha = request.POST.get('senha')

        user = User.objects.filter(username=username).first()

        if user:
            return HttpResponse('Já existe um usuário com esse username')

        user = User.objects.create_user(username=username, email=email, password=senha)
        user.save()

        return HttpResponse('usuario cadastrado!')
```

Figura 1 – Função ‘cadastro’ do arquivo ‘views.py’.

O método ‘cadastro’ tem mais de uma função, já que ele verifica a requisição, checa se já existe um usuário na base de dados, cria o usuário e o salvo caso não tenha e retorna com a resposta de ‘usuario cadastrado!’ caso não tenha um usuário com mesmo *username* e ‘Já existe um usuário com esse username’ caso já tenha. Para

essa e outras funções, o ideal é quebrar em mais funções para que cada função faça somente uma coisa, como na figura abaixo (Figura 2).

```python
def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
    else:
        username = get_user_username(request)
        email = get_user_email(request)
        senha = get_user_password(request)

        user = get_first_user(username)
        if not user:
            create_user(username, email, senha)
            return HttpResponse('usuario cadastrado!')
        return HttpResponse('Já existe um usuário com esse username')
```

Figura 2 – Função ‘cadastro’ do arquivo ‘views.py’ após refatoração.

O – *Open-Closed Principle* (ou princípio do Aberto-Fechado)

Objetos ou entidades devem estar abertos para extensão, mas fechados para modificação.

Para o OCP, não foram encontrados nenhuma quebra do princípio. Mesmo com os condicionais na Figura 1, tal modelo de utilização é padrão do framework django quando utilizado *views* como funções.

L – *Liskov Substitution Principle* (ou princípio da substituição de Liskov)

Uma classe derivada deve ser substituível por sua classe base**.**

Para o LSP, não foram encontrados nenhuma quebra do princípio.

I – *Interface Segregation Principle* (ou princípio da segregação de interface)

Uma classe não deve ser forçada a implementar interfaces e métodos que não irá utilizar.

Para o ISP, não foram encontrados nenhuma quebra do princípio.

D– *Dependecy Inversion Principle* (ou princípio da inversão de dependência)

Dependa de abstrações e não de implementações.

Para o DIP, não foram encontrados nenhuma quebra do princípio.