Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Laboratório de Programação

Ano Letivo 2024/25

Trabalhos Laboratoriais nº 2

Elaborado em: 23/09/2024

António Dinis a2021157297

Mariana Magalhães a2022147454

Tânia Martinho a2021153931

Índice

Lista de Figuras

1 Introdução

O objetivo deste trabalho é melhorar e documentar um fragmento de código existente, aplicando boas práticas de programação e criando documentação técnica abrangente.

Este trabalho está dividido em quatro partes principais:

Parte 1: Convenções de Codificação

- Reestruturar o código fornecido aplicando convenções adequadas
- Focar em nomeação, formatação e estruturação consistentes

Parte 2: Documentação de Código e API

- Criar documentação clara usando ferramentas como Doxygen ou Sphinx
- Documentar funções e classes com descrições, parâmetros, retornos e exemplos

Parte 3: Comentários Úteis

- Adicionar comentários estratégicos para explicar lógica complexa
- Manter o equilíbrio entre informação útil e excesso de comentários

Parte 4: Elaboração de Documentação Técnica e Manual do Usuário

- Criar documentação técnica descrevendo arquitetura e fluxo de dados
- Desenvolver manual do usuário com instruções claras de utilização

2 Método / Metodologia

A atividade foi dividida em 5 partes:

- Aplicar convenções de codificação para melhorar a clareza e consistência do código.
- Elaborar documentação de código eficaz, incluindo a documentação de APIs, que permita a outros programadores compreender e utilizar o sistema.
- Utilizar comentários úteis e estratégicos para explicar lógica complexa, sem sobrecarregar o código com informações desnecessárias.
- Criar uma documentação técnica clara a concisa, incluindo um manual de utilizador, para facilitar a utilização e manutenção futura do sistema.
- Refletir sobre a importância das boas práticas de programação e documentação no contexto de desenvolvimento de software.

3 Fichas Laboratoriais

3.1 Parte 1

Nesta primeira parte, vamos focar na melhoria da estrutura e legibilidade do código existente. As convenções de codificação são essenciais para garantir que o código seja consistente, fácil de ler e manter. Ao aplicar estas convenções, não estamos apenas a melhorar o código, mas também estamos a estabelecer um padrão para futuros desenvolvimentos.

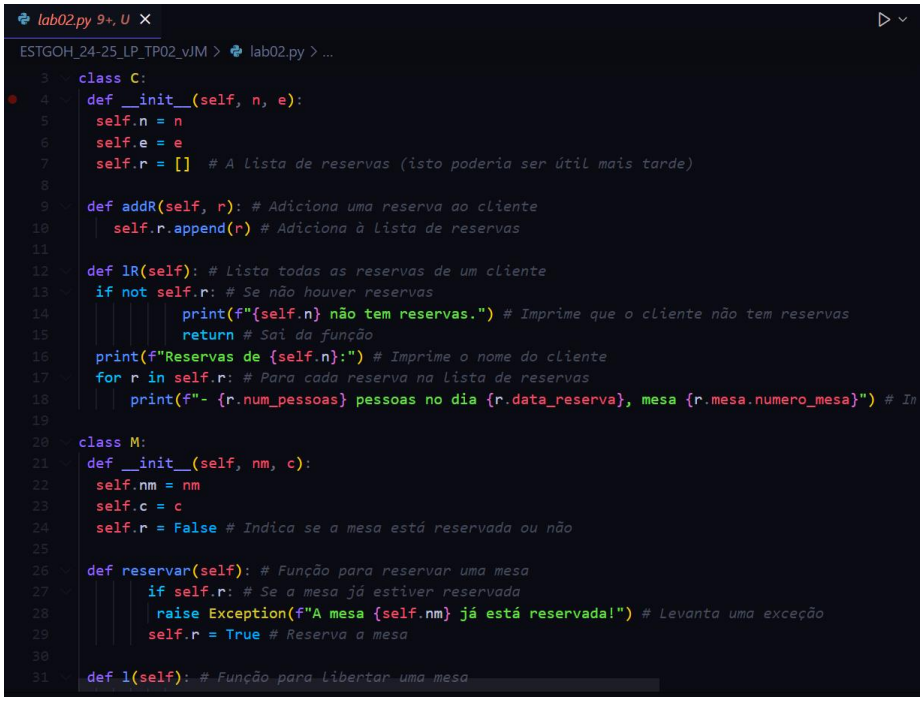

Figura 1 - Código (antes)

Figura 2 - Código (depois)

3.2 Parte 2

Passamos agora para a documentação interna. Uma boa documentação de código e API é crucial para que outros desenvolvedores possam entender e utilizar o código eficientemente e, portanto, nesta parte, vamos criar documentação clara e concisa para cada componente do nosso sistema e para tal vamos utilizar a ferramenta de documentação Doxygen.

A escolha da utilização do Doxygen, veio pelo simples facto de ter sido a ferramenta à qual nos habituamos melhor.

Aplicações como o Doxygen analisam o código-fonte, identificam e processam os comentários especiais. É através dos comentários extraídos que criam as páginas de documentação estruturadas com a respetiva informação.

```python
## @class Mesa
#  @brief Esta classe representa uma mesa do restaurante.
#  @details Possui os métodos reservar e libertarMesa.
class Mesa:
    ## @brief Construtor da classe Mesa.
    #  @param numero_mesa Número da mesa.
    #  @param capacidade_mesa Capacidade da mesa.
    def __init__(self, numero_mesa, capacidade_mesa):
        ## @var numero_mesa
        #  @brief Número da mesa.
        self.numero_mesa = numero_mesa
        ## @var capacidade_mesa
        #  @brief Capacidade da mesa.
        self.capacidade_mesa = capacidade_mesa
        ## @var reservada
        #  @brief Estado de reserva da mesa.
        self.reservada = False

    ## @brief Método reservar.
    #  @details Reserva a mesa se estiver disponível.
    #  @throws Exception Se a mesa já estiver reservada.
    def reservar(self):
        if self.reservada:
            raise Exception(f"A mesa {self.numero_mesa} já está reservada!")
        self.reservada = True

    ## @brief Método libertarMesa.
    #  @details Define a mesa como não reservada.
    def libertarMesa(self):
        self.reservada = False
```

Figura 3 - Classe mesa

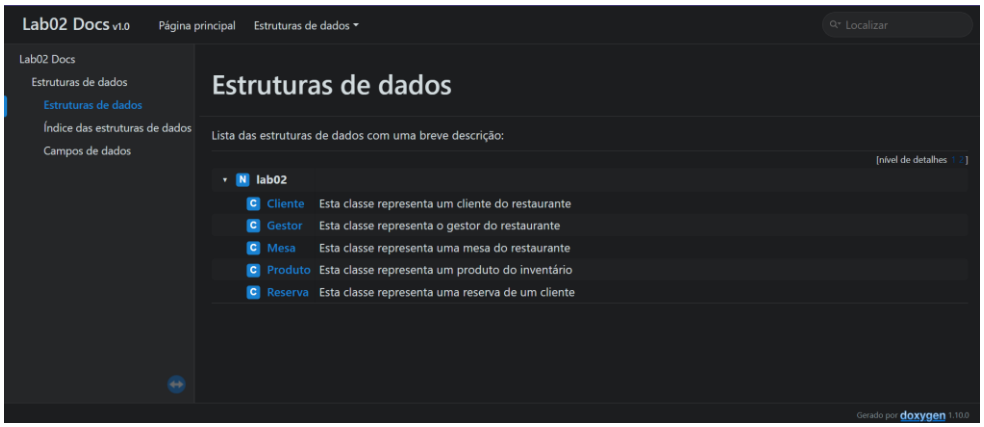

Figura 4 - Documentação gerada pelo Doxygen

3.3 Parte 3

Para além da documentação formal, comentários estratégicos no código são essenciais para explicar lógicas complexas e decisões de implementação e, portanto, vamos adicionar comentários que realmente agreguem valor ao entendimento do código, sem cair na armadilha de comentários óbvios ou excessivos.

```python
class Cliente:
    ## @brief Construtor da classe Cliente.
    #  @param nome Nome do cliente.
    #  @param email Email do cliente.
    def __init__(self, nome, email):
        ## @var nome
        #  @brief Nome do cliente.
        self.nome = nome
        ## @var email
        #  @brief Email do cliente.
        self.email = email
        ## @var reservas
        #  @brief Lista de reservas do cliente.
        self.reservas = []

    ## @brief Método adicionarReserva.
    #  @details Anexa uma reserva ao cliente.
    #  @param reserva Objeto do tipo Reserva.
    def adicionarReserva(self, reserva):
        self.reservas.append(reserva)

    ## @brief Método listarReservasCliente.
    #  @details Lista as reservas de um dado cliente.
    def listarReservasCliente(self):
        if not self.reservas:
            print(f"{self.nome} não tem reservas.")
            return
        print(f"Reservas de {self.nome}:")
        for reserva in self.reservas:
            print(f"- {reserva.num_pessoas} pessoas no dia {reserva.data_reserva}, mesa {reserva.mesa.numero_mesa}")
```

Figura 5 - Classe cliente

Por exemplo, nesta classe temos um array de reservas, e o comentário “Lista de reservas do cliente” ajuda a distinguir claramente que se trata das reservas do cliente, evitando confusões com as reservas de mesas.

3.4 Parte 4

Nesta parte, vamos criar dois tipos importantes de documentação externa:

- documentação técnica: fornece uma visão abrangente da arquitetura e do funcionamento interno do sistema
- manual do utilizador: oferece instruções claras sobre como usar a aplicação.

4 Discussão - Parte 5

Nesta quinta parte do nosso trabalho propuseram-nos refletir sobre o impacto das boas práticas de programação e documentação na manutenção e longevidade do código. Estas práticas são cruciais para o sucesso de qualquer projeto, não apenas para melhorar a qualidade do código, mas também para facilitar o trabalho em equipa e consequentemente garantir a manutenção do sistema ao longo do tempo.

As boas práticas de codificação, como a estruturação, legibilidade e manutenção são fundamentais para garantir que o software possa ser compreendido, mantido e evoluído por diferentes programadores ao longo do tempo, criando um código robusto e escalável. A estruturação reduz a possibilidade de erros e facilitando a sua modificação e extensão. O código deve ser claro e eficiente abordando legibilidade.

A documentação, por sua vez, serve como um guia valioso para desenvolvedores novatos e experientes, permitindo um trabalho em equipa eficiente e apoiando o desenvolvimento continuo reduzindo erros. A utilização de ferramentas de documentação pode auxiliar neste processo automatizando e assegurando que a documentação está sempre atualizada. No nosso caso utilizámos o Doxygen, uma vez que tivemos uma melhor adaptação. Esta ferramenta permite a criação de documentação detalhada em múltiplos formatos, incluindo HTML, PDF e LaTeX, a partir de comentários no código-fonte.

No entanto, implementar estas práticas apresenta alguns desafios. Um dos principais são os próprios desenvolvedores, que muitas vezes veem a documentação como menos gratificante que o próprio código. Além disso, manter a documentação atualizada e consistente pode ser um desafio, especialmente em projetos grandes e dinâmicos.

Apesar desses obstáculos, as vantagens superam os desafios. Boas práticas de codificação promovem padrões e convenções dentro do código base, resultando em código mais organizado e eficiente. A documentação, por sua vez, facilita o trabalho entre membros da equipa, suportando a transferência de conhecimento aumentando assim a produtividade geral.

5 Conclusão

6 Referências

- "Doxygen: Documenting the Code." *Doxygen.nl*, 2024, www.doxygen.nl/manual/docblocks.html#pythonblocks.
- "Getting Started — Sphinx Documentation." *Www.sphinx-Doc.org*, www.sphinx-doc.org/en/master/usage/quickstart.html.
- "Installing Sphinx — Sphinx Documentation." *Sphinx-Doc.org*, 2024, www.sphinx-doc.org/en/master/usage/installation.html. Accessed 23 Sept. 2024.
- "Multiple Author Names | Sphinx." *Sphinxsearch.com*, 2024, sphinxsearch.com/forum/view.html?id=5604. Accessed 23 Sept. 2024.
- "NixOS Search." *Nixos.org*, 2024, search.nixos.org/packages?channel=unstable&from=0&size=50&sort=relevance&type=packages&query=sphinx. Accessed 23 Sept. 2024.
- "Doxygen Awesome." *Github.io*, 2024, jothepro.github.io/doxygen-awesome-css/index.html. Accessed 23 Sept. 2024.