Testes unitários

O que são?

Testes automáticos realizados para verificar se **as unidades individuais de código** (geralmente funções ou métodos) estão a funcionar corretamente de acordo com o comportamento esperado.

A ideia é garantir que as unidades de código possam ser testadas de forma independente e que erros numa parte do código sejam detetados rapidamente, sem afetar outras áreas.

Vantagens dos testes unitários

1. **Detecção precoce de erros;**
2. **Refatoração segura;**
3. **Documentação do comportamento de uma função;**
4. **Redução de bugs em produção;**
5. **Facilidade na manutenção do código;**
6. **Independência de execução;**

Como são estruturados os testes unitários?

1. **Arrange**:

   Preparar os dados necessários, criar instâncias de objetos ou chamar funções auxiliares para configurar o estado inicial.

2. **Act**:

   Ação que queremos testar. Geralmente envolve chamar o método ou a função que está a ser testada, com os dados configurados anteriormente.

3. **Assert**:

   Verificação do comportamento ou resultado esperado. Comparamos o resultado da ação com o valor esperado. Se o resultado for o esperado, o teste passa. Caso contrário, o teste falha.

Como escrever testes unitários?

1. **Escreva testes pequenos e focados**:

   Cada teste unitário deve ser focado numa única unidade de código (por exemplo, uma função ou um método). Isso ajuda a identificar rapidamente o que está errado quando um teste falha.

2. **Garanta a independência**:

   Testes unitários devem ser independentes uns dos outros. Isso significa que, ao rodar um teste, o resultado não deve depender da execução de testes anteriores. Cada teste deve ser autossuficiente.

Possíveis perguntas sobre testes unitários

1. **Qual a importância de testar apenas uma unidade de código num teste unitário?**
   - Garantia de que se está a validar um comportamento específico sem interferência de outras partes do sistema, o que facilita a identificação de erros e torna os testes mais confiáveis.
