Testes Unitários

Bootcamp Python
/código[S]
stone

Maicon Francisco de Carvalho
HOW Bootcamps

Testes Unitários

Tem por objetivo testar a menor parte testável do sistema, geralmente um método.

Principais benefícios

Redução do número de bugs: Há uma confiança muito maior de que o código faz o que deve fazer.

Flexibilidade: Para escrever testes temos que separar o código em pequenos pedaços para que sejam testáveis, ou seja, nosso código estará menos acoplado.

Incentivo a refatoração: Ao escrever testes para uma funcionalidade, gerar uma maior confiança para melhorá-lo

Boas práticas ao escrever testes

Os códigos de testes são tão importantes quanto o código de produção.

Ele não é um componente secundário. Ele requer raciocínio, planejamento e cuidado.

É preciso mantê-lo tão limpo quanto o código de produção.

Uma confirmação por teste

Cada função de teste em um teste unitário deve ter um e apenas uma instrução de confirmação (assert).

Os testes devem chegar a uma única conclusão que é fácil e rápida de entender.

Um único conceito por teste

Cada função de testes deve ter um conceito único.

Não é uma boa prática criar funções de testes que saiam testando várias coisas uma após a outra.

Stateless

Não se pode guardar estados, ou seja, a cada teste todos os recursos que foram utilizados (instâncias, mocks e tudo mais) devem ser destruídos completamente e novos devem ser criados.

Isolamento

O teste de unidade não pode conter dependências externas (bancos de dados, apis e etc).

FIRST

Testes limpos seguem outras cinco regras que formam o acrônimo FIRST:

- **F**ast (Rapidez);
- **I**ndependent (Independência);
- **R**epeatable (Repetitividade);
- **S**elf-validating (Autovalidação);
- **T**imely (Pontualidade);

Fast (Rapidez)

Os testes devem ser rápidos. Devem executar com rapidez.

Quando os testes rodam devagar, você não irá desejar executá-los com frequência e, consequentemente, não encontrará problemas cedo o bastante para consertá-lo facilmente. E você não se sentirá livre para limpar o código, que acabará se degradando.

Independent (Independência)

Os testes não devem depender uns dos outros. Um teste não deve configurar as condições para o próximo.

Você deve ser capaz de executar cada teste de forma independente e na ordem que desejar.

Quando eles dependem uns dos outros, se o primeiro falhar causará um efeito dominó de falhas, dificultando o diagnóstico e ocultando os defeitos abaixo dele.

Repeatable (Repetitividade)

Deve-se poder repetir os testes em qualquer ambiente. Você deve ser capaz de efetuar testes no ambiente de produção, no de QA e no seu notebook.

Caso seus testes não possam ser repetidos em qualquer ambiente, então você sempre terá uma desculpa para o motivo das falhas. E também perceberá que não consegue rodar os testes fora o ambiente adequado.

Self-validating (Autovalidação)

Os testes devem ter uma saída booleana.

Obtenham ou não êxito, você não deve ler um arquivo de registros para saber o resultado. Você não deve ter de comparar manualmente dois arquivos texto para ver se os testes foram bem sucedidos.

Se os testes não possuírem autovalidação, então uma falha pode se tornar subjetiva, e executar os testes pode exigir uma longa validação manual.

Timely (Pontualidade)

Os testes precisam ser escritos em tempo hábil.

Devem-se criar os testes unitários imediatamente antes do código de produção no qual serão aplicados.

Se criá-los depois, o teste do código de produção poderá ficar mais difícil, ou ser negligenciado pela alta complexidade de escrita.

python

Obrigado!

stone + howtech