



RAI RAFAEL SANTOS SILVA
ATIVIDADE 2 – TESTES DE MUTAÇÃO

COMP0444 - TESTE DE SOFTWARE
GLAUCO DE FIGUEIREDO CARNEIRO

03 de setembro de 2024.

1 Configuração do projeto

Para dar início ao teste de mutação, o primeiro passo foi encontrar um repositório Python com testes unitários criados, já que o objetivo dos testes de mutação é verificar a qualidade dos testes unitários para um dado projeto.

Para isso, foi utilizado o [GitHub](#) com as seguintes configurações de busca conforme a Imagem 1.

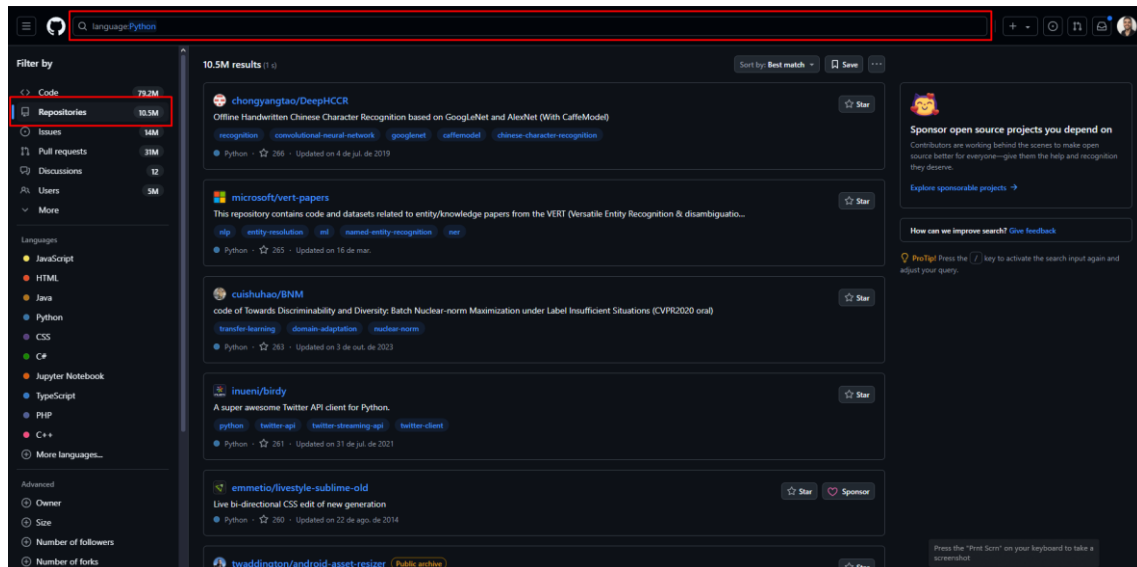


Imagem 1 – Configurações de busca de repositórios

Após buscar listar alguns repositórios, foi encontrado o repositório alvo dos testes de mutação que serão realizados. O repositório em questão foi o “status”, de URL <https://github.com/avinassh/status>.

Ao localizar o repositório, foi realizado o “clone” do repositório para máquina local, conforme a Imagem 2.

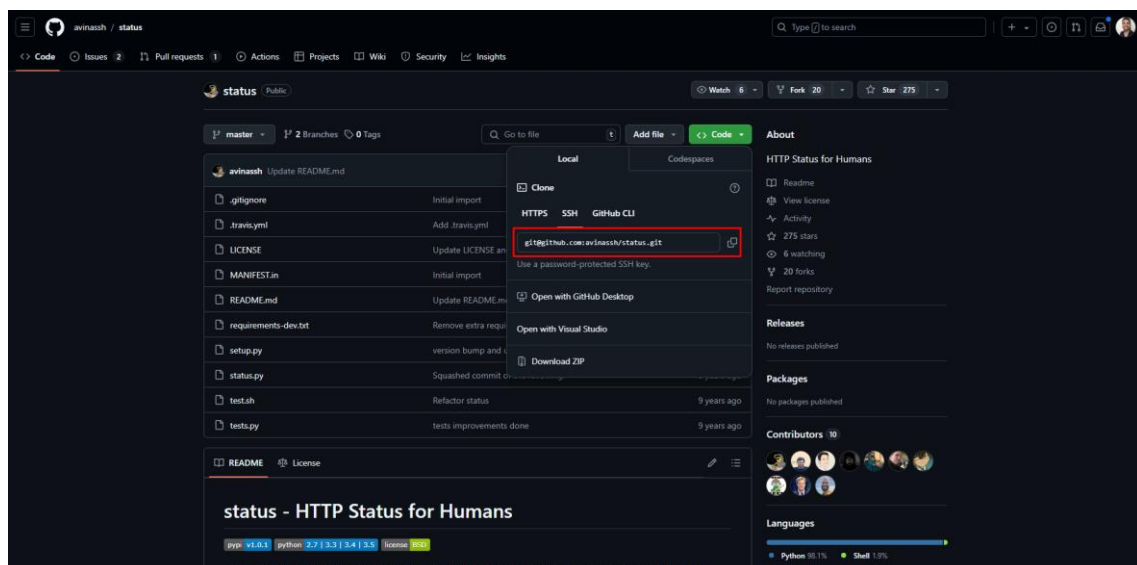


Imagem 2 – “Clone” do repositório no GitHub.

1.1 Instalação das ferramentas

Inicialmente, para realização dos testes é necessário instalar o Python. Para a atividade em questão, foi utilizado o Python na versão 3 (v3.11.9). Para realizar a instalação do Python, basta acessar a seguinte URL: <https://www.python.org/downloads/>. Após o download realizado, basta seguir o passo-a-passo da instalação para realizar a instalação do Python em sua máquina local.

É necessário instalar também um plugin, caso use o VS Code, para que as instalações das dependências do projeto ocorram de maneira correta dentro do ambiente virtual que será criado. O plugin se chama Python Auto Venv

Em seguida, é necessário abrir a pasta onde seu repositório foi “clonado” com seu Terminal desejado e realizar a configuração do ambiente virtual Python. Para isso, foi utilizado o comando `python -m venv C:\Users\Rai_R\vscode-workspace\status\venv`. Após a execução do comando, será possível perceber a criação da pasta `venv` dentro do seu repositório local.

Após a configuração do ambiente virtual, é necessário realizar a instalação das dependências necessárias para rodar o projeto e a dependência do *mutmut*, que é a ferramenta escolhida para realizar os testes de mutação. Para fazer isso é necessário abrir o repositório clonado com sua IDE desejada e adicionar mais uma dependência no arquivo *requirements-dev.txt*, conforme a Imagem 3.

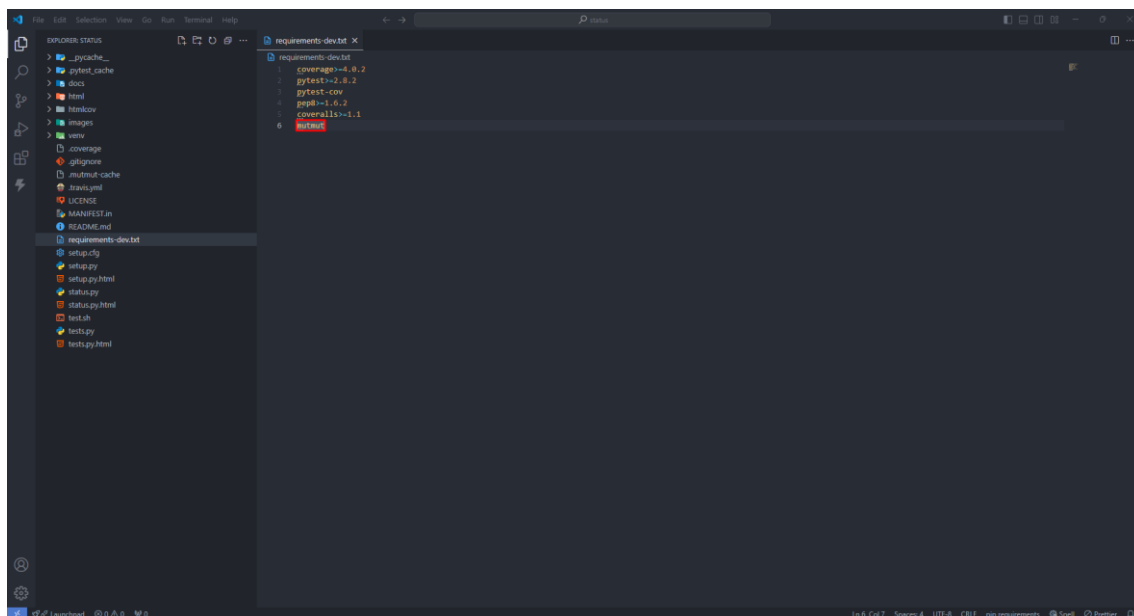


Imagem 3 – Dependência *mutmut* adicionada no *requirements-dev.txt*

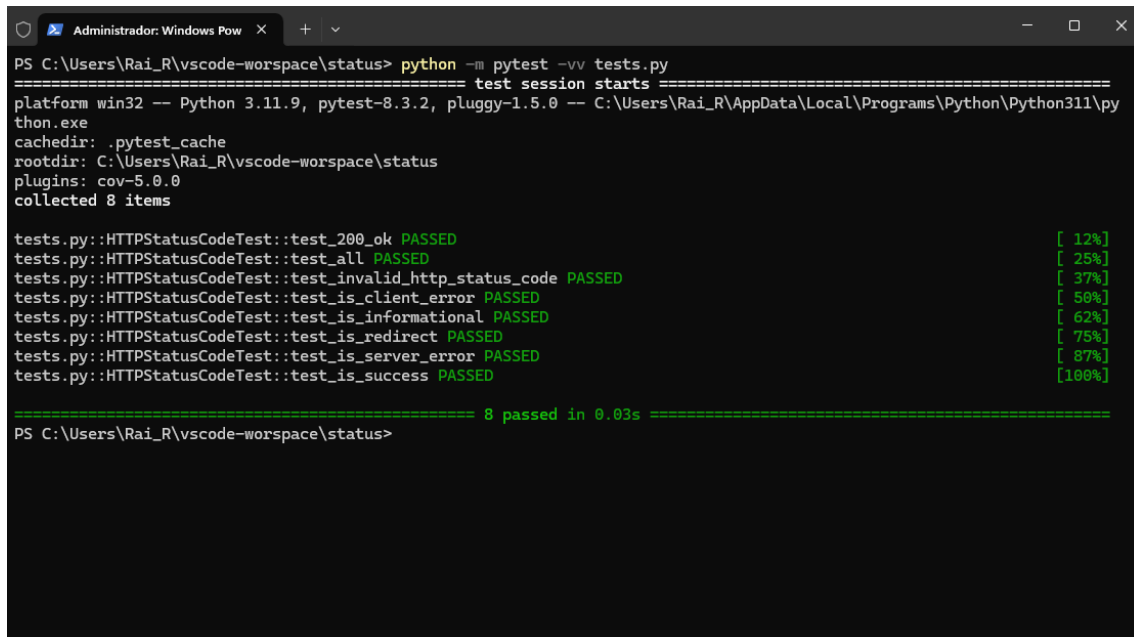
Em seguida basta executar o comando: `pip install -r requirements-dev.txt` e aguardar a instalação de todas as dependências. Dentro do arquivo *requirements-dev.txt* já se encontra presente a dependência do *pytest*, conforme pode ser visto na Imagem 3.

2 Execução dos testes

A partir deste ponto, já é possível iniciar a realização dos testes. O passo-a-passo será demonstrado nos tópicos a seguir.

2.1 Verificação inicial do conjunto de testes

Para realizar a execução dos testes unitários criados, basta executar o comando `python -m pytest -vv tests.py`. Após a execução, será apresentado o resultado dos testes conforme a Imagem 4.



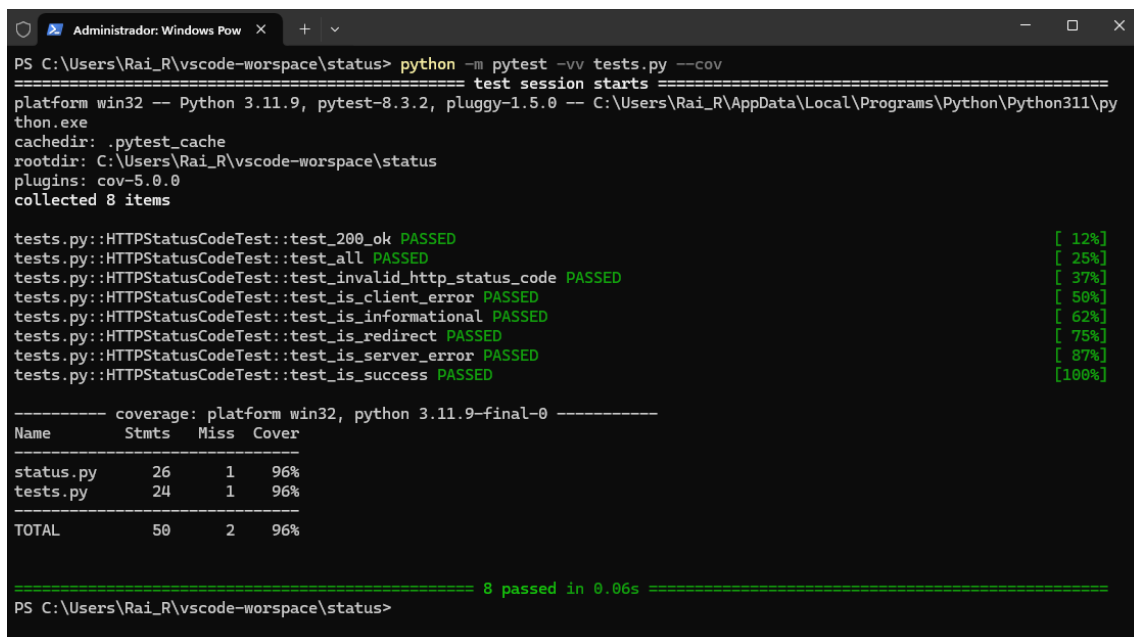
```
PS C:\Users\Rai_R\vscode-worspace\status> python -m pytest -vv tests.py
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.3.2, pluggy-1.5.0 -- C:\Users\Rai_R\AppData\Local\Programs\Python\Python311\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Rai_R\vscode-worspace\status
plugins: cov-5.0.0
collected 8 items

tests.py::HTTPStatusCodeTest::test_200_ok PASSED [ 12%]
tests.py::HTTPStatusCodeTest::test_all PASSED [ 25%]
tests.py::HTTPStatusCodeTest::test_invalid_http_status_code PASSED [ 37%]
tests.py::HTTPStatusCodeTest::test_is_client_error PASSED [ 50%]
tests.py::HTTPStatusCodeTest::test_is_informational PASSED [ 62%]
tests.py::HTTPStatusCodeTest::test_is_redirect PASSED [ 75%]
tests.py::HTTPStatusCodeTest::test_is_server_error PASSED [ 87%]
tests.py::HTTPStatusCodeTest::test_is_success PASSED [100%]

===== 8 passed in 0.03s =====
PS C:\Users\Rai_R\vscode-worspace\status>
```

Imagem 4 – Execução dos testes unitários

Também é possível realizar a verificação da cobertura dos testes unitários no código. Para isso, basta executar o comando `python -m pytest -vv tests.py --cov`. O resultado apresentado após a execução será conforme a Imagem 5. É possível gerar um relatório da cobertura dos testes utilizando o comando `python -m pytest -vv tests.py --cov --cov-report html`. O resultado apresentado será conforme as imagens 6, 7 e 8. Para acessar o relatório gerado, basta buscar a pasta `htmlcov` que foi gerada dentro do seu repositório local e abrir o arquivo `index.html`.



```
PS C:\Users\Rai_R\vscode-worspace\status> python -m pytest -vv tests.py --cov
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.3.2, pluggy-1.5.0 -- C:\Users\Rai_R\AppData\Local\Programs\Python\Python311\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Rai_R\vscode-worspace\status
plugins: cov-5.0.0
collected 8 items

tests.py::HTTPStatusCodeTest::test_200_ok PASSED [ 12%]
tests.py::HTTPStatusCodeTest::test_all PASSED [ 25%]
tests.py::HTTPStatusCodeTest::test_invalid_http_status_code PASSED [ 37%]
tests.py::HTTPStatusCodeTest::test_is_client_error PASSED [ 50%]
tests.py::HTTPStatusCodeTest::test_is_informational PASSED [ 62%]
tests.py::HTTPStatusCodeTest::test_is_redirect PASSED [ 75%]
tests.py::HTTPStatusCodeTest::test_is_server_error PASSED [ 87%]
tests.py::HTTPStatusCodeTest::test_is_success PASSED [100%]

----- coverage: platform win32, python 3.11.9-final-0 -----
Name           Stmts  Miss  Cover
-----
status.py       26      1   96%
tests.py        24      1   96%
TOTAL           50      2   96%

===== 8 passed in 0.06s =====
PS C:\Users\Rai_R\vscode-worspace\status>
```

Imagem 5 – Execução da cobertura dos testes

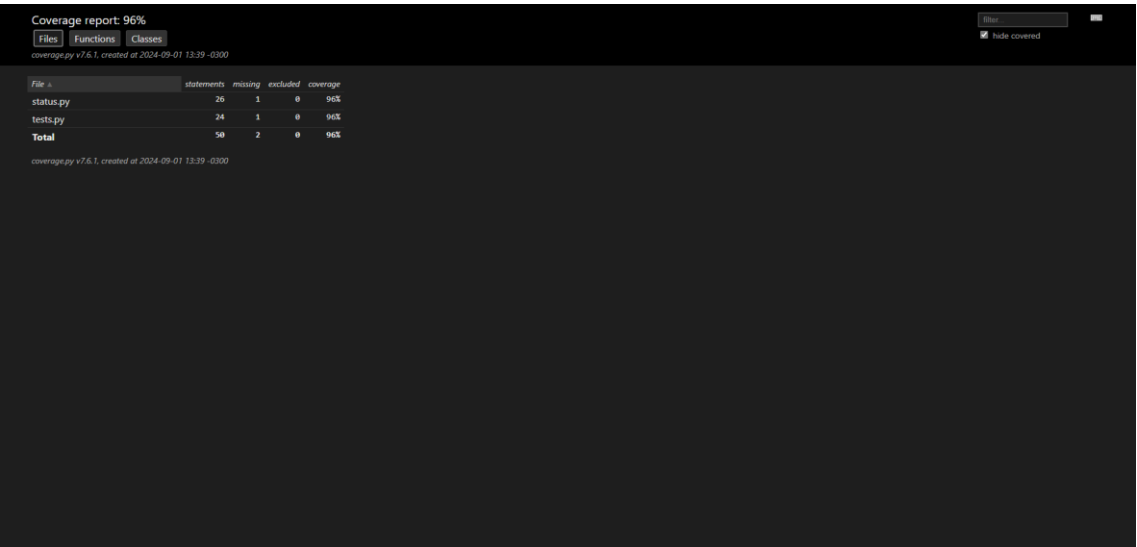


Imagem 6 – Tela inicial do relatório de cobertura de testes

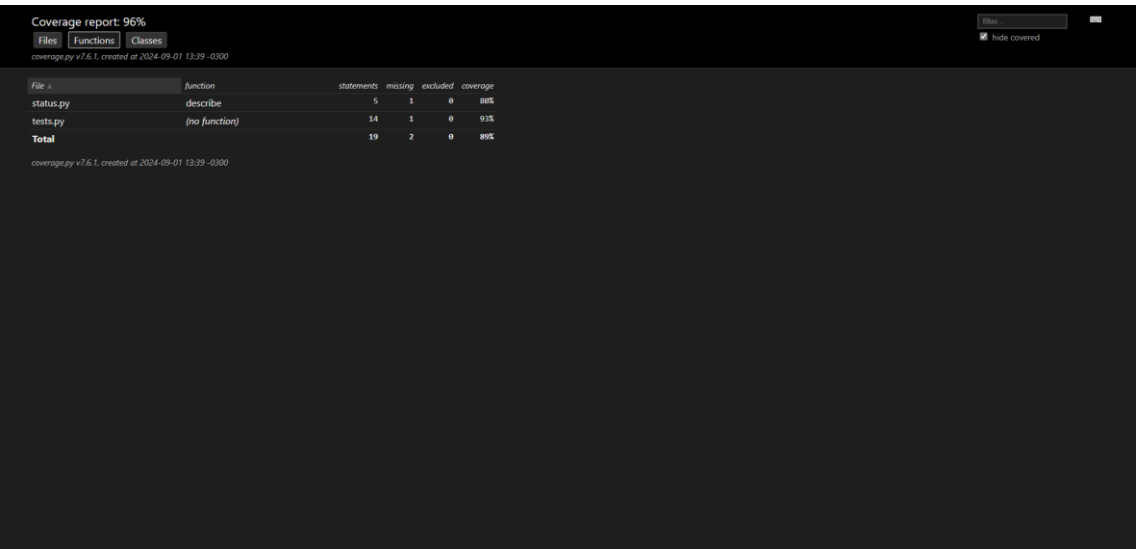


Imagem 7 – Tela das funções cobertas pelos testes



Imagem 8 – Funções cobertas pelos testes unitários

2.2 Execução dos testes de mutação

Para realização dos testes de mutação, inicialmente é necessário configurar a ferramenta de testes de mutação. Para isso é necessário criar o arquivo *setup.cfg* na raiz do projeto local. O arquivo de configuração em questão terá aparência conforme a Imagem 9.

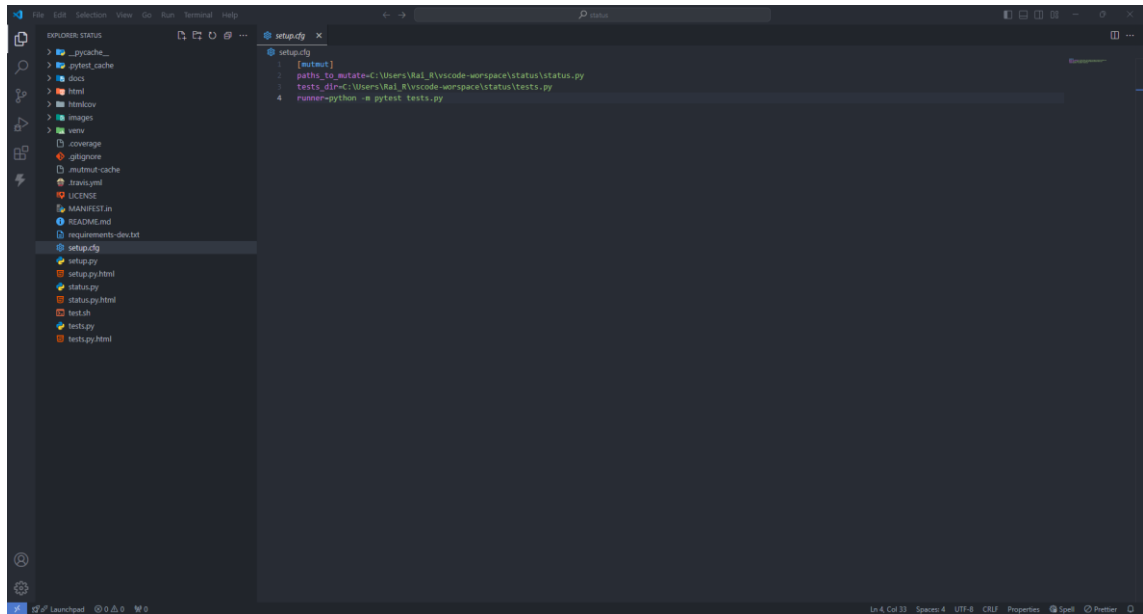


Imagem 9 – Configuração do mutmut

Após configuração do setup, basta executar o comando *python -m mutmut run* para realizar os testes de mutação e então aguardar o término dos testes. A partir deste ponto será apresentada uma tela conforme a Imagem 10.

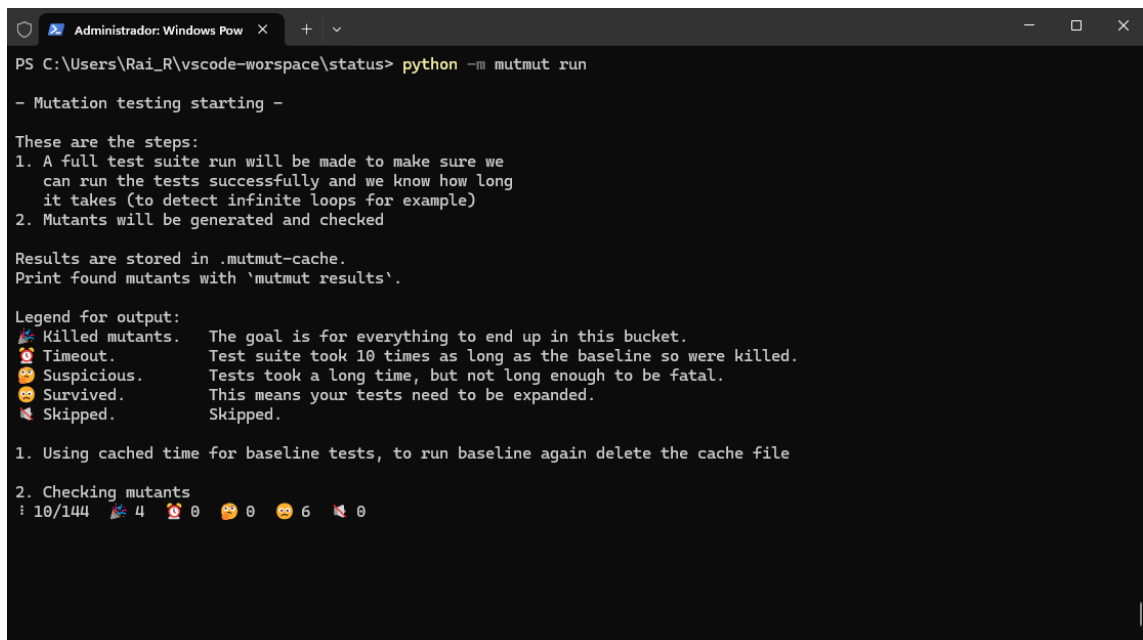
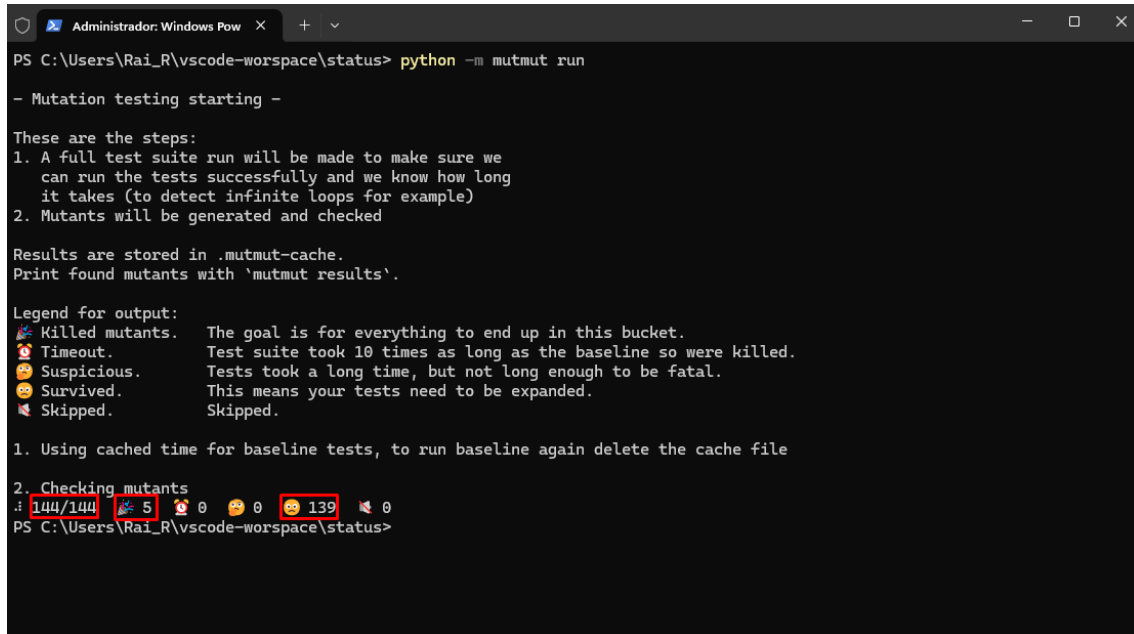


Imagem 10 – Execução dos testes de mutação

2.3 Resultados iniciais

Ao término da execução dos testes de mutação iniciais, o resultado foi que 5 mutantes foram mortos e 139 mutantes sobreviveram, conforme a Imagem 11. Também é possível realizar a geração de um relatório em HTML para uma visualização mais clara dos mutantes que

sobreviveram, para isso basta executar o comando `python -m mutmut html`. Após a execução do comando, para executar o relatório, basta procurar a pasta `html` que foi gerada e procurar o arquivo `index.html`. Ao abrir o arquivo com seu navegador desejado, será apresentada a tela inicial com todos os arquivos conforme a Imagem 12. Para visualizar os mutantes de cada arquivo, basta clicar no arquivo em questão que os resultados serão dispostos conforme a Imagem 13.



```
PS C:\Users\Rai_R\vscode-worspace\status> python -m mutmut run

- Mutation testing starting -

These are the steps:
1. A full test suite run will be made to make sure we
   can run the tests successfully and we know how long
   it takes (to detect infinite loops for example)
2. Mutants will be generated and checked

Results are stored in .mutmut-cache.
Print found mutants with 'mutmut results'.

Legend for output:
🔪 Killed mutants. The goal is for everything to end up in this bucket.
⌚ Timeout. Test suite took 10 times as long as the baseline so were killed.
😟 Suspicious. Tests took a long time, but not long enough to be fatal.
😌 Survived. This means your tests need to be expanded.
🚫 Skipped. Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
.: 144/144 🔪 5 ⌚ 0 😟 0 😌 139 🚫 0
PS C:\Users\Rai_R\vscode-worspace\status>
```

Imagem 11 – Resultado do teste de mutação inicial

Mutation testing report

Killed 5 out of 144 mutants		Total Skipped Killed % killed Survived			
File					
C:\Users\Rai_R\vscode-worspace\status\status.py		144	0	5	3.47 139

Imagem 12 – Arquivos gerados no relatório de testes de mutação pelo mutmut

Killed 5 out of 144 mutants

Survived mutation testing. These mutants show holes in your test suite

```
--- C:\Users\Raif_R\vscode-workspace\status\status.py
```

Imagem 13 – Detalhes de cada mutante no arquivo *status.py*.

3 Alteração proposta

A partir deste momento, serão realizadas alterações nos testes unitários para que seja possível visualizar alguns dos mutantes listados anteriormente serem eliminados.

3.1 Tentativa de melhoria nos casos de teste

Após o relatório gerado conforme a Imagem 13, foi visto que a ferramenta *mutmut* realizou alteração na função *is_informational*. Trocando os sinais de \geq por $=$ e vice-versa. Dessa forma, será realizada uma tentativa de melhoria ajustando o teste unitário da função *is_informational* para cobrir os testes nos pontos críticos destas verificações de maior e maior que. Sendo assim, o teste alterado será definido conforme a Imagem 14, onde foram adicionadas duas novas linhas no teste unitário referente a função *is_informational*.

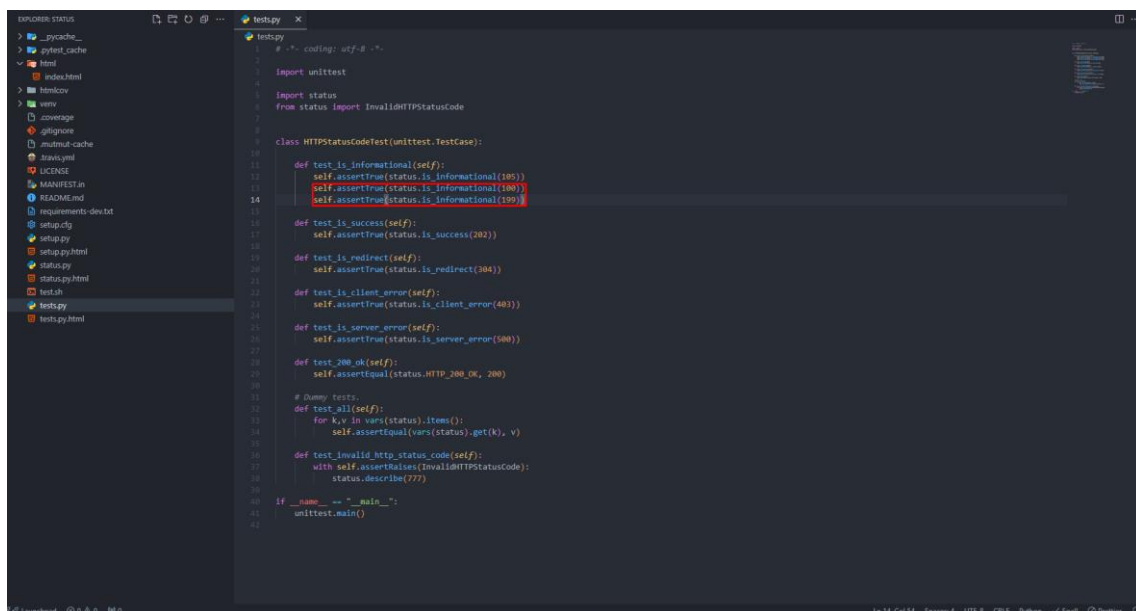
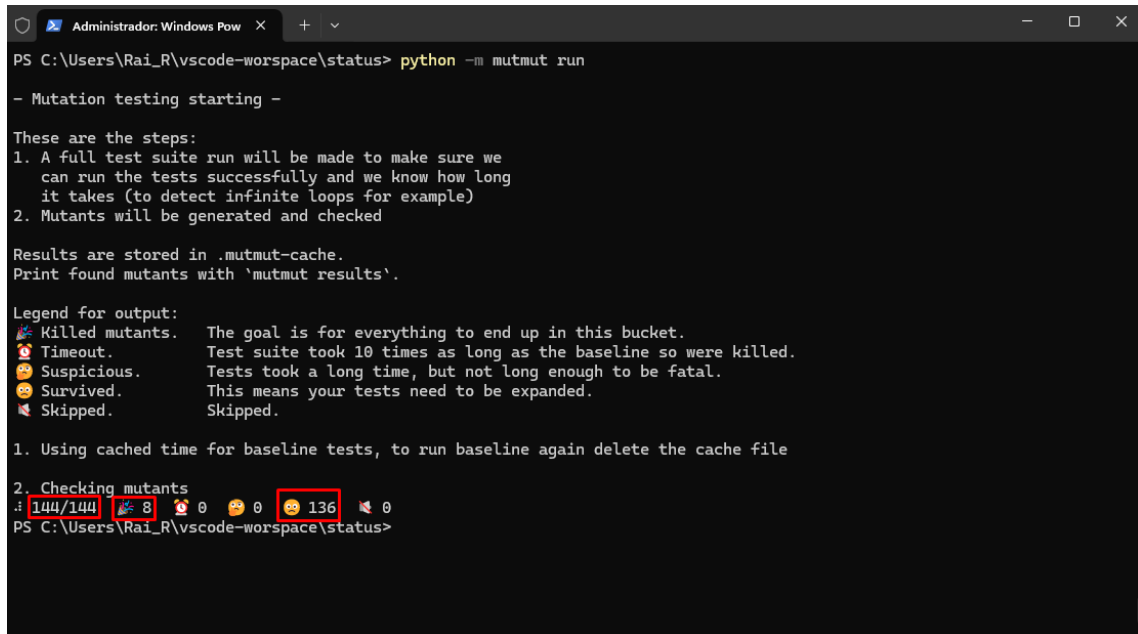


Imagem 14 – Alteração no teste unitário

3.2 Realização de novos testes de mutação

Após o ajuste realizado no teste unitário da Imagem 14, foi realizado um novo teste de mutação, repetindo o teste realizado no tópico do 2.2 deste documento. O resultado obtido está representado na Imagem 15. Na Imagem 15, percebemos que 8 mutantes foram eliminados, em comparação com 5 mutantes eliminados visto no teste de mutação inicial. Foi visto também que 136 mutantes sobreviveram em comparação com 139 mutante sobreviventes do primeiro teste.



```
PS C:\Users\Rai_R\vscode-worspace\status> python -m mutmut run

- Mutation testing starting -

These are the steps:
1. A full test suite run will be made to make sure we
   can run the tests successfully and we know how long
   it takes (to detect infinite loops for example)
2. Mutants will be generated and checked

Results are stored in .mutmut-cache.
Print found mutants with 'mutmut results'.

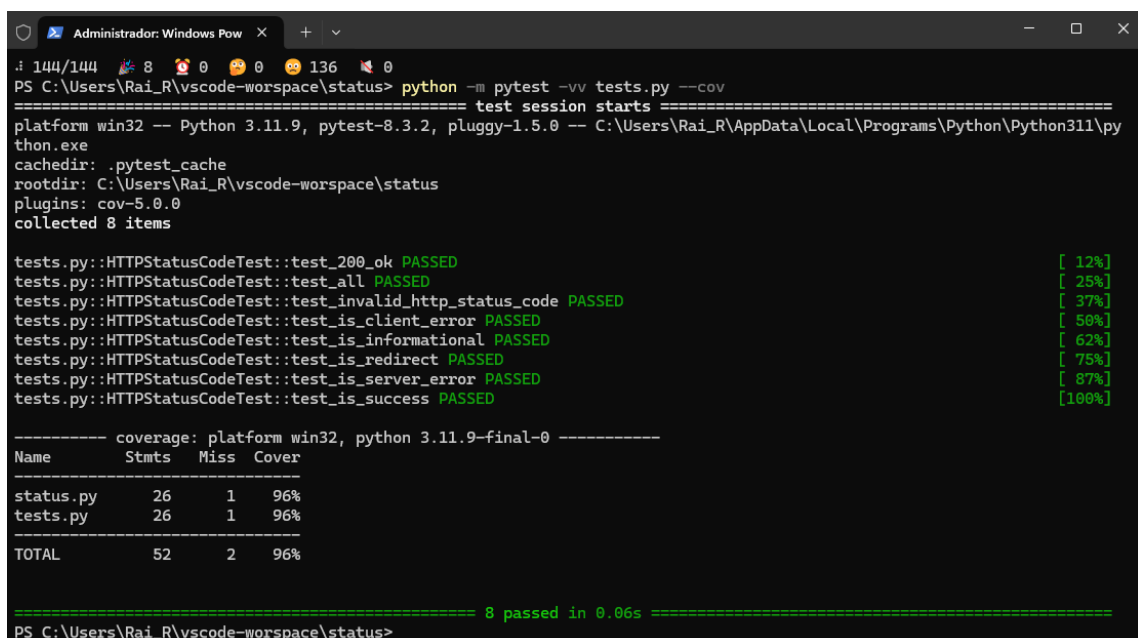
Legend for output:
🔪 Killed mutants. The goal is for everything to end up in this bucket.
⌚ Timeout. Test suite took 10 times as long as the baseline so were killed.
😟 Suspicious. Tests took a long time, but not long enough to be fatal.
👉 Survived. This means your tests need to be expanded.
🚫 Skipped. Skipped.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
.: 144/144 🔪 8 ⌚ 0 😟 0 👉 136 🚫 0
PS C:\Users\Rai_R\vscode-worspace\status>
```

Imagem 15 – Novo resultado de teste de mutação

Após realização do teste de mutação, foi gerado um novo relatório de cobertura de testes, conforme a Imagem 16. Para gerar o relatório, basta executar o comando apresentado na sessão 2.1 deste documento. O novo relatório de cobertura mostra que a cobertura dos testes não foi alterada.



```
PS C:\Users\Rai_R\vscode-worspace\status> python -m pytest -vv tests.py --cov
===== test session starts =====
platform win32 -- Python 3.11.9, pytest-8.3.2, pluggy-1.5.0 -- C:\Users\Rai_R\AppData\Local\Programs\Python\Python311\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Rai_R\vscode-worspace\status
plugins: cov-5.0.0
collected 8 items

tests.py::HTTPStatusCodeTest::test_200_ok PASSED [ 12%]
tests.py::HTTPStatusCodeTest::test_all PASSED [ 25%]
tests.py::HTTPStatusCodeTest::test_invalid_http_status_code PASSED [ 37%]
tests.py::HTTPStatusCodeTest::test_is_client_error PASSED [ 50%]
tests.py::HTTPStatusCodeTest::test_is_informational PASSED [ 62%]
tests.py::HTTPStatusCodeTest::test_is_redirect PASSED [ 75%]
tests.py::HTTPStatusCodeTest::test_is_server_error PASSED [ 87%]
tests.py::HTTPStatusCodeTest::test_is_success PASSED [100%]

----- coverage: platform win32, python 3.11.9-final-0 -----
Name      Stmts  Miss  Cover
-----
status.py   26     1    96%
tests.py    26     1    96%
TOTAL       52     2    96%

===== 8 passed in 0.06s =====
PS C:\Users\Rai_R\vscode-worspace\status>
```

Imagem 16 – Novo relatório de cobertura de testes

Também foi gerado um novo relatório do resultado do teste de mutação para verificar os mutantes que foram eliminados. Para gerar este relatório basta repetir o comando apresentado na sessão 2.3 deste documento. O resultado do relatório está representado na Imagem 17. Para acessar o relatório basta procurar novamente a pasta *html* no projeto local e abrir o arquivo *index.html*. Ao clicar no arquivo *status.py* conforme a Imagem 12, é apresentado o resultado conforme a Imagem 17.

C:\Users\Rai_R\vscode-worspace\status\status.py

Killed 8 out of 144 mutants

Survived

Survived mutation testing. These mutants show holes in your test suite.

Mutant 4

```
--- C:\Users\Rai_R\vscode-worspace\status\status.py
+++ C:\Users\Rai_R\vscode-worspace\status\status.py
@@ -14,7 +16,7 @@

```

```
def is_informational(code):
-   return 100 <= code <= 199
+   return 100 <= code <= 200

```

```
def is_success(code):

```

Mutant 5

```
--- C:\Users\Rai_R\vscode-worspace\status\status.py
+++ C:\Users\Rai_R\vscode-worspace\status\status.py
@@ -20,7 +20,7 @@

```

```
def is_success(code):
-   return 200 <= code <= 299
+   return 200 <= code <= 299

```

```
def is_redirect(code):

```

Mutant 6

```
--- C:\Users\Rai_R\vscode-worspace\status\status.py
+++ C:\Users\Rai_R\vscode-worspace\status\status.py
@@ -20,7 +20,7 @@

```

```
def is_success(code):
-   return 200 <= code <= 299
+   return 200 <= code <= 299

```

```
def is_redirect(code):

```

Mutant 7

```
--- C:\Users\Rai_R\vscode-worspace\status\status.py

```

Imagem 17 – Novo resultado do relatório de teste de mutação

Veja que na Imagem 17, os mutantes 1, 2 e 3 que foram apresentados na Imagem 13 foram eliminados após o ajuste realizado no teste unitário *test_is_informational* conforme a Imagem 14.

4 Considerações finais

É possível continuar tentando eliminar mais mutantes sobreviventes, basta seguir os passos apresentados pelos tópicos 3.1 e 3.2 deste documento. Para este tutorial foi realizado somente o ajuste no teste unitário *test_is_informational* representado na Imagem 14.

O código fonte gerado a partir deste tutorial estará disponível no repositório:
https://github.com/rairfs/Teste_Software_Mutantes_2024_Silva_Rai.

Para execução deste tutorial, foi utilizado como referência o vídeo [Introdução ao teste de mutação em Python com a ferramenta mutmut](#), que se encontra hospedado no YouTube.