Testes unitários como ferramentas de design de código

...

VI encontro PythOnRio - 27 de março de 2016

Paula Grangeiro

Programadora por profissão, desenhista nas horas vagas e colecionadora de gatos.




Sobre mim

Os fatos relatados durante esta apresentação são baseados em experiências pessoais.
Utilize-os com moderação.

Ministério da procrastinação adverte:

Por que pensar em Design de Código?

Atividades Vi IMproved Sáb, 02:34

custom_modelmanager.py + (~/Documentos) - GVIM1

```
 832
 833  def get_report_overall(self, filter_args):
 834      from collections import Counter
 835
>> 836      objEleitor = Eleitor.objects.filter(**filter_args)
 837
 838      graph_data = {}
 839      graph_data['temp_count'] = objEleitor.count()
 840
 841      graph_data['sexo'] = Counter(objEleitor.values_list('sexo', flat=True))
 842      graph_data['sexo'] = {
 843          'null': (
 844              float(graph_data['sexo'][None]) / graph_data['temp_count']
 845          ) * 100,
 846          'F': (
 847              float(graph_data['sexo']['F']) / graph_data['temp_count']
 848          ) * 100,
 849          'M': (
 850              float(graph_data['sexo']['M']) / graph_data['temp_count']
 851          ) * 100,
 852      }
 853
 854      temp_escolaridade = Counter(
 855          objEleitor.exclude(
 856              escolaridade__isnull=True
 857          ).values_list('escolaridade', flat=True)
 858      )
 859      count_escolaridade = len(
>> 860          Eleitor.objects.filter(
 861              **filter_args
 862          ).values('escolaridade').exclude(escolaridade__isnull=True)
 863      )
 864
 865      if count_escolaridade != 0:
 866          graph_data['escolaridade'] = {
 867 +----- 11 linhas: 1: (float(temp_escolaridade[1]) / count_escolaridade) * 100,----------
 878          }
 879      else:
 880 +---- 14 linhas: graph_data['escolaridade'] = {----------
 894
 895      graph_data['aprova_o_governo'] = Counter(
 896          objEleitor.values_list('aprova_o_governo', flat=True)
custom_modelmanager.py                                                    833:3[Syntax: line:1 (26)]
  4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]
[Lista de locais] :SyntasticCheck flake8 (python)                                   4,1          12%
```

Atividades Vi IMproved Sáb, 02:34

custom_modelmanager.py + (~/Documentos) - GVIM1

```
894
895         graph_data['aprova_o_governo'] = Counter(
896             objEleitor.values_list('aprova_o_governo', flat=True)
897         )
898
899         graph_data['aprova_o_governo'] = {
900 +---- 35 linhas: 2: (----------------------------------------------------------------------------------------------------------------------
935         }
936
937         temp_graph_data = Counter(objEleitor.values_list('idade', flat=True))
938
939         idades = dict(
940 +----  9 linhas: dict.fromkeys(----------------------------------------------------------------------------------------------------------------
949         )
950
951         for item in range(15, 25):
952             idades['idade16_24'] += temp_graph_data[item]
953
954         for item in range(25, 35):
955             idades['idade25_34'] += temp_graph_data[item]
956
957         for item in range(35, 45):
958             idades['idade35_44'] += temp_graph_data[item]
959
960         for item in range(45, 60):
961             idades['idade45_59'] += temp_graph_data[item]
962
963         for item in range(60, 100):
964             idades['idade60'] += temp_graph_data[item]
965
966         count_idade = sum(idades.values())
967
968         graph_data['idade'] = {
969 +----  6 linhas: 0: (float(idades['idade16_24']) / count_idade) * 100,----------------------------------------------------------------------
975         }
976
977         temp_prioridade = Counter(objEleitor.values_list(
978             'qual_a_sua_prioridade', flat=True)
979         )
980
981         prioridade = dict(dict.fromkeys(
982             ['agua', 'asfalto', 'educacao', 'emprego', 'saude', 'seguranca'], 0
custom_modelmanager.py                                                    894:0[Syntax: line:1 (26)]
  4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]
[Lista de locais] :SyntasticCheck flake8 (python)                                   4,1          12%
```

Atividades Vi IMproved Sáb, 02:34

custom_modelmanager.py + (~/Documentos) - GVIM1

```
983         ))
984
985         prioridade['agua'] = (temp_prioridade[u'ÁGUA'] + temp_prioridade[u'AGUA'] + temp_prioridade[u'Água'] + temp_prioridade[u'água'] + temp_prioridade[u'agua'])
986         prioridade['asfalto'] = (temp_prioridade[u'ASFALTO'] + temp_prioridade[u'Asfalto'] + temp_prioridade[u'asfalto'])
987         prioridade['educacao'] = (temp_prioridade[u'EDUCAÇAO'] + temp_prioridade[u'EDUCACAO'] + temp_prioridade[u'Educação'] + temp_prioridade[u'Educacao'] + temp_
    prioridade[u'educação'] + temp_prioridade[u'educacao'])
988         prioridade['emprego'] = (temp_prioridade[u'EMPREGO'] + temp_prioridade[u'Emprego'] + temp_prioridade[u'emprego'])
989         prioridade['saude'] = (temp_prioridade[u'SAÚDE'] + temp_prioridade[u'SAUDE'] + temp_prioridade[u'Saúde'] + temp_prioridade[u'Saude'] + temp_prioridade[u'sa
    úde'] + temp_prioridade[u'saude'])
990         prioridade['seguranca'] = (temp_prioridade[u'SEGURANÇA'] + temp_prioridade[u'SEGURANCA'] + temp_prioridade[u'Segurança'] + temp_prioridade[u'Seguranca'] +
    temp_prioridade[u'segurança'] + temp_prioridade[u'seguranca'])
991
992         count_prioridade = sum(prioridade.values())
993
994         graph_data['prioridade'] = {
995 +----  7 linhas: 0: (float(prioridade['agua']) / count_prioridade) * 100,----------
1002        }
1003
1004
S>1005      temp_contato = {
1006 +----  6 linhas: 0: objEleitor.values_list('numero_1', flat=True).count(),----------
1012        }
1013
1014        count_contato = (
1015 +----  7 linhas: float(objEleitor.values_list('numero_1', flat=True).exclude(numero_1__isnull=True).count() +----------
1022        )
1023
1024        graph_data['telefone'] = {
1025 +----  7 linhas: 0: float(float(temp_contato[0]) / count_contato * 100),----------
1032        }
1033
1034        email_informed = len(objEleitor.values_list('email', flat=True).exclude(email__isnull=False))
1035        email_uninformed = len(objEleitor.values_list('email', flat=True).exclude(email__isnull=True))
1036
1037        graph_data['email'] = {
1038 +----  2 linhas: 0: float((float(email_uninformed) / graph_data['temp_count']) * 100),----------
1040        }
1041
1042        objRenda = objEleitor.values_list('renda_mensal_presumida', flat=True)
1043
1044        renda = dict(dict.fromkeys(['renda_mais_1356', 'renda_mais_2034', 'renda_mais_3390', 'renda_nao_informada'], 0))
1045
custom_modelmanager.py                                                    983:3[Syntax: line:1 (26)]
 4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]
[Lista de locais] :SyntasticCheck flake8 (python)                                  4,1          12%
```

Atividades Vi IMproved Sáb, 02:34

custom_modelmanager.py + (~/Documentos) - GVIM1

```
1004
S>1005        temp_contato = {
1006 +----  6 linhas: 0: objEleitor.values_list('numero_1', flat=True).count(),----------
1012         }
1013
1014         count_contato = (
1015 +----  7 linhas: float(objEleitor.values_list('numero_1', flat=True).exclude(numero_1__isnull=True).count() +----------
1022         )
1023
1024         graph_data['telefone'] = {
1025 +----  7 linhas: 0: float(float(temp_contato[0]) / count_contato * 100),----------
1032         }
1033
1034         email_informed = len(objEleitor.values_list('email', flat=True).exclude(email__isnull=False))
1035         email_uninformed = len(objEleitor.values_list('email', flat=True).exclude(email__isnull=True))
1036
1037         graph_data['email'] = {
1038 +----  2 linhas: 0: float((float(email_uninformed) / graph_data['temp_count']) * 100),----------
1040         }
1041
1042         objRenda = objEleitor.values_list('renda_mensal_presumida', flat=True)
1043
1044         renda = dict(dict.fromkeys(['renda_mais_1356', 'renda_mais_2034', 'renda_mais_3390', 'renda_nao_informada'], 0))
1045
1046         for item in range(len(objRenda)):
1047 +----  8 linhas: if objRenda[item] > 1356.00 and objRenda[item] < 2033.99:----------
1055
1056         count_renda = sum(renda.values())
1057
1058         graph_data['renda'] = {
1059 +----  5 linhas: 0: (float(renda['renda_nao_informada']) / count_renda) * 100,----------
1064         }
1065
1066         return graph_data
~
custom_modelmanager.py                                                    1004:0[Syntax: line:1 (26)]
  4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]
[Lista de locais] :SyntasticCheck flake8 (python)                                   4,1          12%
```


Código Mogwai

Testes unitários

Garantia de entrada/saída no teste unitário

```python
def foo(param):
    if param:
        return 'It has a param!'
    return 'Nope!'
```

```python
from unittest import TestCase
from foo import foo


class FooTestCase(TestCase):

    def test_if_function_return_correct_value(self):
        self.assertEqual('It has a param!', foo(1))
        self.assertEqual('Nope!', foo(''))
```

- Unittest
- Py.test

Atividades Vi IMproved Sáb, 02:52

custom_modelmanager_test.py (~/Documentos) - GVIM

```
>>  1 class CustomTestCase(TestCase):
    2     def setUp(self):
    3         print('Initializing Graph test case')
>>  4         usuarioTeste = coreModels.Usuario.objects.create(
    5             username='teste',
    6             nome='nome',
    7         )
    8
>>  9         campanhaTeste = coreModels.Campaign.objects.create(
   10             name='CampanhaTeste',
   11             description='Description',
>> 12             start_date=datetime.datetime(2014, 01, 01),
>> 13             end_date=datetime.datetime(2014, 01, 01),
   14             cordinator=usuarioTeste,
   15             manager=usuarioTeste,
   16         )
   17
>> 18         perfilTeste = coreModels.Profile.objects.create(
   19             name='perfilTeste',
   20             description='Description',
   21             contact_type=1,
   22             goal=100,
   23             profile_type=1,
   24             target_filter={}
   25         )
   26
>> 27         taskTeste = coreModels.Task.objects.create(
   28             name='Task Teste',
   29             goal=100,
   30             description='',
   31             contact_type=1,
   32             task_type=1,
>> 33             start_date=datetime.datetime(2014, 01, 01),
>> 34             end_date=datetime.datetime(2014, 01, 01),
   35             operator=usuarioTeste,
   36             cordinator=usuarioTeste,
   37             campaign=campanhaTeste,
   38             profile=perfilTeste,
   39             status=1,
   40             visible_fields={},
   41             fields={},
```

```
custom_modelmanager_test.py                                   41:6[Syntax: line:1 (112)]
  4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]
  5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]
[Lista de locais] :SyntasticCheck flake8 (python)                    4,1          2%
```

Atividades Vi IMproved Sáb, 02:52

custom_modelmanager_test.py (~/Documentos) - GVIM

```
40            visible_fields={},
41            fields={},
42        )
>> 43        eleitorTesteM = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste', sexo="M")
>> 44        eleitorTesteF = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste', sexo="F")
>> 45        eleitorTesteNull = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste')
46
47    def test_gender_graph_data_two_days_same_distribution_m_and_f(self):
48        #Required data
>> 49        eleitorTesteM = coreModels.Eleitor.objects.filter(sexo="M")[0]
>> 50        eleitorTesteF = coreModels.Eleitor.objects.filter(sexo="F")[0]
>> 51        taskTeste = coreModels.Task.objects.all()[0]
52
>> 53        test1m = coreModels.TaskResult.objects.create(
S> 54                                                elector = eleitorTesteM,
S> 55                                                task = taskTeste,
S> 56                                                done_at = datetime.date(2014, 01, 01),
S> 57                                                status = 1
58        )
59        #unfornutally done at is auto_now_add so we need to do this litle hack
>> 60        test1m.done_at = datetime.date(2014, 01, 01)
61        test1m.save()
62
>> 63        test1f = coreModels.TaskResult.objects.create(
S> 64                                                elector = eleitorTesteF,
S> 65                                                task = taskTeste,
S> 66                                                done_at = datetime.date(2014, 01, 01),
S> 67                                                status = 1
68        )
69        #unfornutally done at is auto_now_add so we need to do this litle hack
>> 70        test1f.done_at = datetime.date(2014, 01, 01)
71        test1f.save()
72
>> 73        start_date = datetime.datetime(2014, 01, 01, 0, 0, 0)
>> 74        end_date = datetime.datetime(2014, 01, 02, 23, 59, 59)
75        lookup_field = 'elector__sexo'
76
>> 77        result = coreViews.generateGraphData(start_date, end_date, lookup_field)
78
S> 79        self.assertEqual(result, [[["2014-01-01",50.0],["2014-01-02",50.0]],[["2014-01-01",50.0],["2014-01-02",50.0]],[["2014-01-01",0.0],["2014-01-02",0.0]]])
80
```

```
custom_modelmanager_test.py                                                    41:6[Syntax: line:1 (112)]
  4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]
  5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]
[Lista de locais] :SyntasticCheck flake8 (python)                                  4,1          2%
```

Atividades Vi IMproved Sáb, 02:52

custom_modelmanager_test.py (~/Documentos) - GVIM

```
S> 79      self.assertEqual(result, [[["2014-01-01",50.0],["2014-01-02",50.0]],[["2014-01-01",50.0],["2014-01-02",50.0]],[["2014-01-01",0.0],["2014-01-02",0.0]]])
   80
   81     def test_gender_graph_two_days_same_distribution_m_and_f_and_null(self):
   82         #Required data
>> 83         eleitorTesteM = coreModels.Eleitor.objects.filter(sexo="M")[0]
>> 84         eleitorTesteF = coreModels.Eleitor.objects.filter(sexo="F")[0]
>> 85         eleitorTesteNull = coreModels.Eleitor.objects.filter(sexo__isnull=True)[0]
   86
>> 87         taskTeste = coreModels.Task.objects.all()[0]
   88
   89         #Elector day 1 Masc
>> 90         test1m = coreModels.TaskResult.objects.create(
S> 91                                         elector = eleitorTesteM,
S> 92                                         task = taskTeste,
S> 93                                         done_at = datetime.date(2014, 01, 01),
S> 94                                         status = 1
   95         )
   96         #unfornutally done at is auto_now_add so we need to do this litle hack
>> 97         test1m.done_at = datetime.date(2014, 01, 01)
   98         test1m.save()
   99
  100         #Elector day 1 Fem
>>101         test1f = coreModels.TaskResult.objects.create(
S>102                                         elector = eleitorTesteF,
S>103                                         task = taskTeste,
S>104                                         done_at = datetime.date(2014, 01, 01),
S>105                                         status = 1
  106         )
  107         #unfornutally done at is auto_now_add so we need to do this litle hack
>>108         test1f.done_at = datetime.date(2014, 01, 01)
  109         test1f.save()
  110
  111         #Elector day 1 Null
>>112         test1Null = coreModels.TaskResult.objects.create(
S>113                                         elector = eleitorTesteNull,
S>114                                         task = taskTeste,
S>115                                         done_at = datetime.date(2014, 01, 01),
S>116                                         status = 1
  117         )
  118         #unfornutally done at is auto_now_add so we need to do this litle hack
>>119         test1Null.done_at = datetime.date(2014, 01, 01)
custom_modelmanager_test.py                                                    79:6[Syntax: line:1 (112)]
  4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]
  5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]
[Lista de locais] :SyntasticCheck flake8 (python)                                  4,1          2%
whitespace after '[' [E201]
```

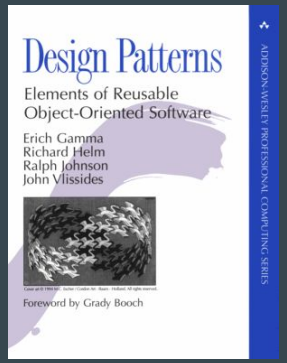
Atividades Vi IMproved Sáb, 02:52

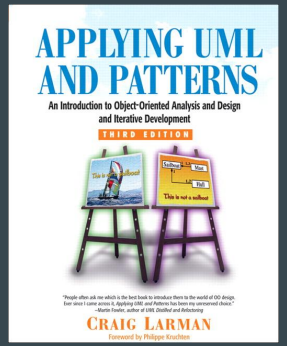
custom_modelmanager_test.py (~/Documentos) - GVIM

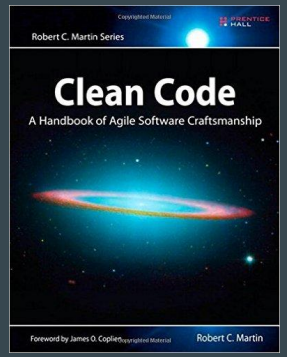
```
S> 94                                                status = 1
   95          )
   96          #unfornutally done at is auto_now_add so we need to do this litle hack
>> 97          test1m.done_at = datetime.date(2014, 01, 01)
   98          test1m.save()
   99
  100          #Elector day 1 Fem
>>101          test1f = coreModels.TaskResult.objects.create(
S>102                                                elector = eleitorTesteF,
S>103                                                task = taskTeste,
S>104                                                done_at = datetime.date(2014, 01, 01),
S>105                                                status = 1
  106          )
  107          #unfornutally done at is auto_now_add so we need to do this litle hack
>>108          test1f.done_at = datetime.date(2014, 01, 01)
  109          test1f.save()
  110
  111          #Elector day 1 Null
>>112          test1Null = coreModels.TaskResult.objects.create(
S>113                                                elector = eleitorTesteNull,
S>114                                                task = taskTeste,
S>115                                                done_at = datetime.date(2014, 01, 01),
S>116                                                status = 1
  117          )
  118          #unfornutally done at is auto_now_add so we need to do this litle hack
>>119          test1Null.done_at = datetime.date(2014, 01, 01)
  120          test1Null.save()
  121
>>122          start_date = datetime.datetime(2014, 01, 01, 0, 0 ,0)
>>123          end_date = datetime.datetime(2014, 01, 02, 23, 59, 59)
  124          lookup_field = 'elector__sexo'
  125
>>126          result = coreViews.generateGraphData(start_date, end_date, lookup_field)
  127          print result
S>128          self.assertEqual(result, [[["2014-01-01",33.33], ["2014-01-02",33.33]], [["2014-01-01",33.33], ["2014-01-02",33.33]], [["2014-01-01",33.33], ["2014-01-02",3
     3.33]]])
S>129
~
~
~
custom_modelmanager_test.py                                                    109:6[Syntax: line:1 (112)]
  4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]
  5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]
[Lista de locais] :SyntasticCheck flake8 (python)                                  4,1          2%
```

Padrões de Projeto

Design Patterns







- Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos
- Utilizando UML e Padrões
- Código limpo
- Curso Python Patterns - Luciano Ramalho

Indicação de estudo

Testes unitários como ferramenta de design

```python
class Foo(object):

    def calculate_charged_price(self, custom_tax):
        price = Price.objects.get(active=True)

        final_value = self.get_taxes()
        if final_value:
            final_value = final_value * price
        else:
            if custom_tax:
                final_value = custom_tax * price
            else:
                final_value = price

        return final_value
```

```python

class FooTestCase(TestCase):

    def setUp(self):
+---  3 linhas: '''----------------------------------------------------------

    def test_calculate_charged_price_correctly(self):
        final_value = self.foo.calculate_charged_price()
        self.assertEqual(1.2, final_value) #without custom tax for product 1

        final_value = self.foo.calculate_charged_price(2)
        self.assertEqual(4, final_value) #with custom tax for product 1

        Tax.objects.delete()
        final_value = self.foo.calculate_charged_price()
        self.assertEqual(1, final_value) # without custom tax, without tax
```

```python

class FooTestCase(TestCase):

    def setUp(self):
+---  3 linhas: '''----------------------------------------------------------

    def test_calculate_charged_price_without_custom_tax(self):
        final_value = self.foo.calculate_charged_price()
        self.assertEqual(1.2, final_value)

    def test_calculate_charged_price_with_custom_tax(self):
        final_value = self.foo.calculate_charged_price(2)
        self.assertEqual(4, final_value)

    def test_calculate_charged_price_without_custom_tax_and_tax(self):
        Tax.objects.delete()
        final_value = self.foo.calculate_charged_price()
        self.assertEqual(1, final_value)

    def test_get_returns_correct_active_price(self):
        # Test if returns active price
```

Teste unitariamente o seu código

```
21 class FooTestCase(TestCase):
22
23     def setUp(self):
24 +---   3 linhas: '''-----------------------------------------------------------------
27
28     def test_calculate_charged_price_without_custom_tax(self):
29         final_value = self.foo.calculate_charged_price()
30         self.assertEqual(1.2, final_value)
31
32     def test_get_price_raises_custom_exception(self):.......
33         Price.objects.delete()
34         self.assertRaises(Price.DoesNotExist, self.foo.calculate_charged_price, 1)
35
36
37     def test_calculate_charged_price_with_custom_tax(self):
38         final_value = self.foo.calculate_charged_price(2)
39         self.assertEqual(4, final_value)
40
41     def test_calculate_charged_price_without_custom_tax_and_tax(self):
42         Tax.objects.delete()
43         final_value = self.foo.calculate_charged_price()
44         self.assertEqual(1, final_value)
45
46     def test_get_returns_correct_active_price(self):
47         # Test if returns active price
48
```

Teste fluxos de sucesso e erro separadamente

```
20
21 class FooTestCase(TestCase):
22
23     def setUp(self):
24 +---  3 linhas: '''-------------------------------------------------------------------------
27
28     def test_calculate_charged_price_without_custom_tax(self):
29         final_value = self.foo.calculate_charged_price()
30         self.assertEqual(1.2, final_value)
31
32     def test_get_price_raises_does_not_exist_if_has_no_price(self):
33         Price.objects.delete()
34         self.assertRaises(Price.DoesNotExist, self.foo.calculate_charged_price, 1)
35
36     def test_calculate_charged_price_with_custom_tax(self):
37         final_value = self.foo.calculate_charged_price(2)
38         self.assertEqual(4, final_value)
39
40     def test_calculate_charged_price_raise_custom_exception_if_custom_tax_is_not_a_number(self):
41         self.assertRaises(InvalidCustomTaxException, self.foo.calculate_charged_price, 't')
42
43
44     def test_calculate_charged_price_without_custom_tax_and_tax(self):
45         Tax.objects.delete()
46         final_value = self.foo.calculate_charged_price()
47         self.assertEqual(1, final_value)
48
49     def test_get_returns_correct_active_price(self):
50         # Test if returns active price
51
52
```

Nomeie os testes de acordo com o fluxo testado

```python
class PatientReport(object):

    def create_report(self, data):

        openned_file = open('/tmp/tmp_file.txt', 'w+')
        for item in data:
            openned_file.write('{};{};{};{}\n'.format(*item))
        openned_file.close()

```

```python

from unittest import TestCase
from mock import mock_open, patch
from foo import PatientReport


class PatientTestCase(TestCase):

    def setUp(self):
        self.data = [(1, 2, 3, 4)]
        self.report = PatientReport()

    @patch('__builtin__.open', mock_open())
    def test_if_creates_file_with_correct_permissions(self, mocked_open):
        self.report.create_report(self.data)
        mocked_open.assert_called_once_with('/tmp/xpto.csv', 'w')

    @patch('__builtin__.open', mock_open())
    def test_if_format_data_and_write_in_file_correctly(self, mocked_open):
        self.report.create_report(self.data)
        mocked_open().write.assert_called_once_with('1; 2; 3; 4\n')

    @patch('__builtin__.open', mock_open())
    def test_if_closes_file_correctly(self, mocked_open):
        self.report.create_report(self.data)
        self.assertTrue(mocked_open.close.called)
```

Testes devem ser isolados!

```python
def test_view_returns_200(self):
    response = self.client.get(self.url)

    self.assert200(response)
    self.assertEqual(PRESENTER_STUB['paciente_id'], response.context['paciente_id'])

    for key in response.context['aso']:
        self.assertEqual(PRESENTER_STUB['aso'][key], response.context['aso'][key])

```

```python

def test_view_returns_200(self):
    response = self.client.get(self.url)
    self.assert200(response)

def test_view_has_correct_context
    self.assertEqual(PRESENTER_STUB['paciente_id'], response.context['paciente_id'])
    for key in response.context['aso']:
        self.assertEqual(PRESENTER_STUB['aso'][key], response.context['aso'][key])

```

Testes devem ser pequenos!

```python
class ScheduleReport(object):

    def create_report(self, recipients):
        scheduling = Schedule.objects.actives()

        openned_file = open('/tmp/tmp_file.txt', 'w+')
        for schedule in scheduling:
            openned_file.write('{};{};{};{}\n'.format(*schedule))
        openned_file.close()

        s3_file = Key(self.bucket, name='tmp_file.txt')
        s3_file.set_contents_from_filename('/tmp/tmp_file.txt')
        s3_file.close()

        url = 'http://s3.test.com/tmp_file.txt'
        email = EmailMessage(
            'Your report is ready!',
            render_to_string('email_template.html', {'url': url}),
            settings.DEFAULT_FROM_EMAIL,
            recipients
        )
        email.send()
```

```python
class SchedulingReportTestCase(TestCase):

    def setUp(self):
+---  2 linhas: self.data = [(1, 2, 3, 4)]----------------------------------------

    @patch('__builtin__.open', mock_open())
    def test_if_creates_file_with_correct_permissions(self, mocked_open):
+---  2 linhas: self.report.create_report(self.data)------------------------------

    @patch('__builtin__.open', mock_open())
    def test_if_format_data_and_write_in_file_correctly(self, mocked_open):
+---  2 linhas: self.report.create_report(self.data)------------------------------

    @patch('__builtin__.open', mock_open())
    def test_if_closes_file_correctly(self, mocked_open):
+---  2 linhas: self.report.create_report(self.data)------------------------------

    def test_if_file_was_send_to_s3_correctly(self):
+---  2 linhas: '''---------------------------------------------------------------

    def test_if_s3_raises_exception_on_connection_faluire(self):
+---  2 linhas: '''---------------------------------------------------------------

    def test_if_email_was_formatted_correctly(self):
+---  2 linhas: '''---------------------------------------------------------------

    def test_if_email_was_send_correctly(self):
        '''
        '''
```

TestCases também devem ser pequenos!

TDD

Obrigada!

http://www.paulagrangeiro.com.br

https://twitter.com/paulagrangeiro

http://fb.me/paula.grangeiro

https://github.com/pgrangeiro

pgrangeiro.dev@gmail.com