UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

Modelo Plano de Testes

1. Identificação

a. Nome do projeto: Tindoar

b. Equipe: Vitor Albuquerque de Paula, Gustavo Henrique de Oliveira Okuda da Silva, Afonso Celso Penze Nunes da Cunha, Carlos Roberto dos Santos Júnior, Laura Pereira de Gouveia, Mário Lucas Massaia de Oliveira, Mateus Rocha de Medeiros

c. Data criação do documento: 04/06/2019

2. Introdução

a. **Escopo:** Os testes realizados, serão todos referentes aos usuários e os itens que serão doados. Nos usuários, será testado a criação e diversos casos possíveis na hora do login do usuário. Nos itens, será testado o cadastro do item no sistema, o pedido de algum item, e a aceitação ou rejeição de um pedido. A aceitação de tais testes será feita diretamente pela API Django, que possui um suíte de testes integrado, suíte esse que realiza os testes de maneira automatizada e retorna o resultado de tais testes.

3. Planejamento para realização dos testes

a. Cronograma de atividades: Inicialmente será realizado um estudo em testes unitários na API do Django, logo após a realização dos estudos, será iniciado a criação dos testes dos casos de uso referentes aos usuários, quando terminada essa fase, será criado os testes referentes às doações de itens

b. Responsáveis pelos testes: Carlos Roberto dos Santos Júnior

4. Projeto de casos de teste

a. Casos de uso considerados: login de um usuário que está cadastrado, login de um usuário não cadastrado, login de um usuário cadastrado com senha diferente da cadastrada, login de um usuário cadastrado com o nome de usuário digitado de maneira diferente da cadastrada e com senha correta, criação de um usuário, criação de um usuário que já existe, cadastrar item para doação, fazer pedido de item a ser doado, recusar o pedido de algum item, aceitar o pedido de algum item.

b. Geração de casos de teste: Os testes foram divididos e são gerados usando Particionamento em Classes de Equivalência e Análise do Valor Limite, as partições são:

- Partição Inválida: Conjunto de testes onde os dados não são válidos, e como esperado a saída é um erro, como exemplos de testes temos os casos de teste: login de um usuário não cadastrado, login de um usuário cadastrado com senha diferente da cadastrada, login de um usuário cadastrado com o nome de usuário digitado de maneira diferente da cadastrada e com senha correta, criação de um usuário que já existe;
- Partição Válida: Conjunto de testes onde os dados são válidos, e como esperado a saída não acusa erros, exemplos de casos teste são: login de um usuário que está cadastrado, criação de um usuário, cadastrar item para doação, fazer pedido de item a ser doado, recusar o pedido de algum item, aceitar o pedido de algum item.

Os testes são executados e a saída é conferida pelo suíte de testes do Django.

5. Códigos

```
from django.test import TestCase
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.models import User

class Usuarios(TestCase):
    def setup(self):
        self.user = User.objects.create_user(
            username='jacob', email='jacob@…',
password='top_secret')

    def test_login_usuario(self):
        self.setup()
        user = authenticate(username='jacob',
password='top_secret')
        self.assertIsNotNone(user)

    def test_usuario_nao_existe(self):
        user = authenticate(username='15', password='gfdg')
        self.assertIsNone(user)

    def test_login_user_correto_senha_incorreta(self):
        self.setup()
        user = authenticate(username='jacob', password='top')
        self.assertIsNone(user)

    def test_login_user_incorreto_senha_correta(self):
        self.setup()
        user = authenticate(username='jaco',
password='top_secret')
        self.assertIsNone(user)

    def test_criacao_usuario_existente(self):
        self.setup()
```

```
            with self.assertRaises(Exception):
                self.user = User.objects.create_user(
                    username='jacob', email='jacob@…',
    password='top_secret')

    class Doacoes(TestCase):
        def setup(self):
            pass

        def test_cadastrar_item(self):
            pass

        def test_lista_itens_cadastrados(self):
            pass

        def test_fazer_pedido(self):
            pass

        def test_recusar_pedido(self):
            pass

        def test_aceitar_pedido(self):
            pass
```

6. Resultados

```
PS C:\Documents\GitHub\eng_soft_django\tindoar> python manage.py
test mostrar/
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..........
----------------------------------------------------------------------
Ran 10 tests in 3.892s
OK
Destroying test database for alias 'default'...
PS C:\Documents\GitHub\eng_soft_django\tindoar>
```

7. Conclusão

a. Recursos que serão utilizados: Na realização dos testes serão utilizados ferramentas de testes automatizados que estão incluídos dentro do Django, ferramenta essa utilizada para o desenvolvimento de nosso projeto, mais especificamente as ferramentas de teste unitário.