Desarrollo de Sistemas de información para la empresa Control y Seguridad SA

PLAN DE PRUEBAS Y ACEPTACION DE USUARIOS

Versión: 0001

NORMA IEEE

Fecha: 29/02/2024

VERSION 01.

SENA	**CENTRO DE SERVICIOS FINANCIEROS** COORDICACION DE TECNOLOGIAS DE LA INFORMACION.		
Proyecto	INVENTORIES AND QUICK ORGANIZATION		
Entregable	Manual de Usuario		
Autor	TEAM SCRUM (Nº). Tecnólogo en Análisis y Desarrollo de Software		
Versión/Edición	0001	**Fecha Versión**	29/02/2024
Aprobado por	NOMBRE INSTRUCTOR EVALUADOR	**Fecha Aprobación**	DD/MM/AAAA
ADSO FICHA Nº	2558723	**Nº Total de Páginas**	6

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0001	Versión inicial	<Nombre Apellido1 Apellido2>	29/02/2024
0002	N/A	N/A	N/A
0003	N/A	N/A	N/A

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos (TEAM SCRUM)
Johan Rodriguez Vega
Karoll Stephany Celis Sotaquira
Alison Valoyes Parra
Diana Carolina Riaño Pedraza

ÍNDICE

PLAN DE PRUEBAS Y ACEPTACION DE USUARIOS

INTRODUCCIÓN

Las pruebas de estrés son fundamentales en el desarrollo de proyectos en Python, ya que permiten evaluar el comportamiento del sistema bajo condiciones extremas de carga. Estas pruebas son esenciales para identificar posibles cuellos de botella, errores de rendimiento y limitaciones en el software.

OBJETO

El objetivo de la prueba de estrés es evaluar la capacidad del sistema para manejar la creación de 100 usuarios de forma simultánea, simulando una carga intensiva en el proceso de registro. Esta prueba busca determinar la resistencia y el rendimiento del sistema bajo condiciones extremas de concurrencia, identificando posibles cuellos de botella, errores de rendimiento y limitaciones en el proceso de creación de usuarios.

APLICACIÓN PARA EVALUAR

Se ha seleccionado el sistema IQO sobre el cual se realizarán pruebas de estrés. Este sistema es de fácil acceso, dispone de diferentes módulos que permiten realizar diferentes tipos de pruebas como pruebas unitarias que permiten validar la funcionalidad del sistema y de estrés comprobando el rendimiento del mismo.

ALCANCE

Evaluar la capacidad del sistema para manejar situaciones extremas de carga, específicamente la creación de 100 usuarios simultáneos. Esta prueba busca medir la resistencia y el rendimiento del software bajo condiciones de alta demanda, identificando posibles problemas de estabilidad, cuellos de botella y limitaciones en el proceso de registro de usuarios. Además, se pretende verificar la eficacia del sistema al gestionar errores en condiciones extremas y determinar su límite de funcionamiento antes de presentar fallas.

PLAN DE PRUEBAS

MATRIZ DE EJECUCIÓN

En esta matriz se muestran en detalle las fechas y horas planeadas, ejecutadas y el porcentaje de avance de las pruebas realizadas a la aplicación.

Inventories And Quick Organization							
		FECHA ESTIMADA		TIEMPO ESTIMADO	FECHA REAL	TIEMPO	PORCENTAJE
MÓDULO	FASE	INICIAL	FINAL	HORAS	INICIAL FINAL	REAL EN HORAS	AVANCE
USUARIOS	EJECUCIÓN	15/03/24	15/03/24	2h 30 min	18/03/2024	3h	100%

PRUEBA DE ESTRÉS

Al navegar por la aplicación, se está realizando una prueba de estrés, con el fin de identificar inconsistencias o fallas que impidan el funcionamiento básico del software respecto al registro de varios usuarios a la vez.

```python
from django.test import TestCase
from registroUsuario.models import CustomUser, City, Role, Contact, Client_Type, Time_Work, Availability_Status, Identification_type, Position, Position_Specialty, Employee,Client

class StressTestCase(TestCase):
    def setUp(self):
        # Limpiar la base de datos antes de cada prueba
        CustomUser.objects.all().delete()
        City.objects.all().delete()
        Role.objects.all().delete()
        Contact.objects.all().delete()
        Client_Type.objects.all().delete()
        Time_Work.objects.all().delete()
        Availability_Status.objects.all().delete()
        Identification_type.objects.all().delete()
        Position.objects.all().delete()
        Position_Specialty.objects.all().delete()
        Employee.objects.all().delete()
    def test_create_users(self):
        # Crear datos necesarios para la creación de usuarios
        city = City.objects.create(city_name="Ciudad Ejemplo")
        role = Role.objects.create(name_role="Rol Ejemplo", role_description="Descripción del Rol")
        contact = Contact.objects.create(address="Dirección Ejemplo", city=city)
        client_type = Client_Type.objects.create(client_type_name="Tipo de Cliente Ejemplo", client_type_description="Descripción del Tipo de Cliente")
        time_work = Time_Work.objects.create(work_time_type="Tipo de Tiempo de Trabajo Ejemplo", work_time_description="Descripción del Tipo de Tiempo de Trabajo")
        availability_status = Availability_Status.objects.create(availability_type="Tipo de Disponibilidad Ejemplo", description_type_availability="Descripción del Tipo de Disponibilidad")
        identification_type = Identification_type.objects.create(identification_type="Tipo de Identificación Ejemplo", description_type_identification="Descripción del Tipo de Identificación")
        position = Position.objects.create(name_position="Cargo Ejemplo")
        position_specialty = Position_Specialty.objects.create(name_specialty_position="Especialidad del Cargo Ejemplo", description_specialty_position="Descripción de la Especialidad del Cargo")
# Crear 100 usuarios de prueba
        for i in range(100):
            user = CustomUser.objects.create(
                username=f"usuario{i}",
                second_name="Segundo Nombre",
                second_last_name="Apellido",
                password="password",  # Se almacenará de manera segura automáticamente
                password2="password",  # Se almacenará de manera segura automáticamente
                email=f"usuario{i}@example.com",
                phone="1234567890",
                birth_date="1990-01-01",
                identification_type=identification_type,
                role=role,
                contact=contact
            )
            Client.objects.create(client_type=client_type, user=user)
            Employee.objects.create(work_time=time_work, user=user, position=position, availability_status=availability_status)
        # Verificar que se crearon 100 usuarios
        self.assertEqual(CustomUser.objects.count(), 100)
```

```
test/test_estres.py::StressTestCase::test_create_users PASSED
```

Tipo de Prueba: Estrès
Nivel de Prueba: Pruebas de Aceptación

INGRESO AL MÓDULO DE USUARIOS

DESCRIPCIÓN, se puede registrar el usuario y no presenta bloqueo o inconsistencia. El módulo funciona de acuerdo con lo esperado.

DISEÑO DE ALTO NIVEL	SI CUMPLE	NO CUMPLE
1. El usuario puede registrarse en el sistema	x	
2. Los campos de registro estàn validados	x	
3. Los usuarios se guardan correctamente en el sistema	x	

CRITERIOS DE ACEPTACIÓN

El proceso de pruebas se enfocò en la prueba de estrés realizada con el módulo de usuarios para validar el registro de varios usuarios al mismo tiempo y también se realizaron pruebas unitarias verificando que los crud de todos los módulos funcionan correctamente.

- Se ha ejecutado el 100% de los casos de prueba (unitarias y de estrés) diseñados para este proyecto y su resultado ha sido exitoso.
- Los errores encontrados en las pruebas unitarias estuvo en las tablas que contenían llaves foráneas, estos fueron detectados en la ejecución de éstas, siendo validados y solucionados.