****

**Plan de pruebas**

**PROYECTO SMC Bomberos**



**Integrantes**

**Gonzalo Rubilar**

**Eder Dureaux**

**Kevin Carrasco**

**Sep 25, 2024**

Índice

**1.** Tipos de pruebas planificadas

**1.1** Pruebas Unitarias

**1.2** Pruebas de integración

**1.3** Pruebas de integración

**1.4** Pruebas end-to-end (E2E)

**1.5** Pruebas de Rendimiento

**1.6** Pruebas de Seguridad

**2** Metodologias de Prueba a utilizar

**3** Herramientas a utilizar

**5** Responsables de las pruebas

1. Tipos de pruebas planificadas

1.1 Pruebas Unitarias

**Descripción:**

Validan la correcta funcionalidad de las unidades individuales tanto en el frontend (Ionic) como en el backend (Django).

**Objetivo:**

Asegurar que cada componente funcione de forma independiente.

**Metodologia:**

* Para el front end (Ionic), se probarán componentes individuales de la UI y la lógica de la aplicación.
* Para el backend (Django), se validarán modelos, vistas, y endpoints API.

**Herramientas:**

* **Frontend:** Karma (para ejecutar pruebas con Angular en Ionic).
* **Backend:** PyTest (para componentes de Django).
* **Cobertura:** pytest-cov y Karma con Istanbul.

**Responsable:**

* **Eder Dureaux - Desarrollador Full Stack.**

1.2 Pruebas de integración

**Descripción:**

Se enfocan en validar que el frontend (Ionic) y el backend (Django) interactúen correctamente.

**Objetivo:**

Asegurar la correcta integración entre las APIs del backend y los componentes del frontend.

**Metodología:**

Se realizarán pruebas para validar que las llamadas API desde Ionic hacia Django devuelvan los resultados esperados y se manejen correctamente.

**Herramientas:**

* **Postman:** Para probar la API de Django.
* **jest:** Para probar la integración de Angular/Ionic.
* **Docker:** Para simular entornos de desarrollo y de pruebas consistentes.

**Responsables:**

**Kevin Carrasco - Disenador de UI/Tester**

**Eder Dureaux - Desarrollador Full Stack**

1.3 Pruebas de Aceptación del Usuario (UAT)

**Descripción:**

Pruebas donde los usuarios finales validan el sistema en función de los requisitos originales.

**Objetivo:**

Verificar que el sistema cumpla con las expectativas y los casos de uso definidos.

**Metodología:**

Los usuarios clave realizarán pruebas basadas en escenarios reales usando la app móvil/web desarrollada en Ionic y conectada a Django.

**Herramientas:**

* **Google Forms:** Para recopilar el feedback de los usuarios.
* **JIRA:** Para gestionar incidencias reportadas durante las pruebas.

**Responsable:**

**Kevin Carrasco - Disenador UI/Tester**

1.4 Pruebas End-toEnd (E2E)

**Descripción:**

Pruebas completas que validan todo el flujo del sistema, desde el frontend en Ionic hasta el backend en Django.

**Objetivo:**

Asegurar que todas las funcionalidades trabajen sin fallos a lo largo de todo el ciclo de uso.

**Metodología:**

Automatización de pruebas que simulan la interacción de los usuarios con el sistema completo.

**Herramientas**:

* **Cypress**: Para pruebas automatizadas del frontend en Ionic.
* **Selenium**: Para pruebas automatizadas de todo el sistema (frontend y backend).

**Responsable**:

Kevin Carrasco - Disenador UI/Tester.

1.5 Pruebas de Rendimiento

**Descripción:**

Evaluar cómo el sistema responde bajo diferentes cargas, especialmente en la interacción entre Ionic y Django.

**Objetivo:**

Asegurar que el sistema pueda manejar múltiples usuarios y solicitudes concurrentes sin degradación en el rendimiento.

**Metodología:**

Pruebas de carga y estrés simulando múltiples usuarios accediendo a la app móvil y la API simultáneamente.

**Herramientas:**

* **Apache JMeter:** Para pruebas de carga y estrés en Django (backend).
* **Google Cloud Monitoring:** Para supervisar el rendimiento del sistema en producción.

**Responsable:**

**Eder Dureaux - Desarrollador Full Stack**

1.6 Pruebas de Seguridad

**Descripción:**

Verificar la seguridad tanto en el frontend como en el backend, asegurando la protección de datos y autenticación segura.

**Objetivo:**

Evitar vulnerabilidades de seguridad en la comunicación entre el frontend (Ionic) y el backend (Django).

**Metodología:**

Realización de pruebas de penetración y validación de mecanismos de autenticación y permisos.

**Herramientas:**

* **OWASP ZAP:** Para pruebas de seguridad automatizadas.
* **Burp Suite:** Para realizar pruebas de penetración manuales.

**Responsable:**

**Gonzalo Rubilar - Gestor de Proyectos**

2 Metodologías de Prueba

**Desarrollo Guiado por Pruebas (TDD):**

Para las pruebas unitarias en **Django** y **Ionic**, asegurando que el código sea escrito de acuerdo con los test.

**Pruebas Automatizadas:**

Para pruebas de integración, end-to-end, y rendimiento, aprovechando frameworks como **Cypress** y **Selenium** para un ciclo de pruebas continuo.

**Validación por Casos de Uso:**

En las pruebas de aceptación, basadas en los escenarios definidos en la etapa de requisitos.

**Pruebas de Caja Negra:**

En las pruebas de seguridad, para evaluar cómo se comporta el sistema sin conocimiento del código.

3 Herramientas a utilizar

**Frontend (Ionic):**

* Karma para pruebas unitarias.
* Jest para integración.
* Cypress para pruebas end-to-end.

**Backend (Django):**

* PyTest para pruebas unitarias y de integración.
* Postman para probar la API.
* Apache JMeter para pruebas de carga.

**Pruebas de Seguridad:**

* OWASP ZAP y Burp Suite.

**Monitoreo y Gestión de Incidencias:**

* Google Cloud Monitoring para rendimiento.
* **JIRA** para la gestión de pruebas e incidencias.

4 Responsables de las Pruebas

**Kevin Carrasco:** Diseñador UI/Tester (Pruebas de interfaz, integración y end-to-end en Ionic).

**Eder Dureaux:** Desarrollador Full Stack (Pruebas de backend en Django, integración, y rendimiento).

**Gonzalo Rubilar:** Gestor de proyectos (Supervisión general, UAT, y seguridad).

Este plan de pruebas podría estar sujeto a modificaciones en el futuro.