Para efectos del challange de Mercado Libre, respecto al rol de QA Engineer a continuación se redacta un plan de pruebas detallado para el requerimiento solicitado.

PLAN DE PRUEBAS

Introducción

Este documento debe ser leído por todas las partes interesadas del proyecto, incluyendo, si se requiere, a directores técnicos, gerentes, personas de estrategia, arquitectos, desarrolladores, gerentes de proyecto, Scrum Masters, Project Leader o Technical Leader y miembros del equipo de QA.

Objetivos

Este plan documenta el enfoque de pruebas usado por el equipo de pruebas de para asegurarse que todos los entregables tengan los más altos estándares de calidad y esté alineado con los requerimientos y especificaciones funcionales del proyecto, este plan de pruebas está orientado para probar el proyecto Operación Fuego Quasar.

El plan se enfoca en validar los siguientes criterios:

- Los servicios y APIs se comportan adecuadamente como lo establecen las Historias de Usuario o Requerimiento.
- Integración entre servicios y APIs.
- Creación de casos de prueba
- Pruebas automatizadas.
- Pruebas de performance.
- Pruebas Funcionales.

Roles y Responsabilidades

QA Automation Engineer: Es quien se encarga de crear la estrategia inicial y actividades de QA en el proyecto, estima Historias de Usuario, crea toda la documentación requerida para el cliente, comunica riesgos. Trabaja cercanamente con los desarrolladores y arquitecto y DevOps para realizar integración continua para pruebas de regresión automática, pruebas de carga y pruebas funcionales o manuales.

Es quien lidera la planificación, el diseño, la ejecución y la administración de las pruebas manuales y las pruebas automatizadas, siendo pro activo en la identificación y prevención también comunicando bugs o fallas críticas con el mínimo esfuerzo antes de salir a producción.

Se tiene en cuenta las siguientes responsabilidades:

- Dar visibilidad de los casos de prueba.
- Priorizar acciones que permitan automatizar las pruebas.
- Capacitar al Project Leader o Technical Leader en Gherkin y promover su uso.
- Ser parte de todo el ciclo de vida del software, ayudando a establecer las definiciones de READY y DONE
- Fomentar la mejores prácticas de pruebas para el equipo
- Aplicar la estrategia de pruebas de acuerdo con los lineamientos y necesidades del negocio, asegurando el correcto control de calidad sobre el producto.
- Incentivar al equipo de desarrollo a mantener el ambiente de pruebas actualizado con el código y los datos adecuados.
- Ejecutar las pruebas de regresión en cada paso a producción.
- Crear y monitorear los bugs de acuerdo con los resultados de las pruebas, asegurándose que el Project Leader o Technical Leader los priorice para su gestión.
- Participar activamente en todas las ceremonias de Scrum y los refinamientos.

Metodología

El equipo de QA tomará un enfoque ágil para hacer pruebas, este enfoque incluye la integración de equipos multi funcionales que trabajan iterativamente en la iniciativa. Por cada Historia de Usuario, se definen casos de prueba y escenarios que son marcados como manuales o automáticos si es posible, después se corren los casos de prueba manuales o automáticos y se entrega un reporte de resultados al equipo, los casos de prueba y las pruebas se encargan de verificar que todos los criterios de aceptación se cumplan.

El equipo de QA estará en constante comunicación con el equipo de desarrollo en todo el proceso y debe ser la mano derecha del Project Leader o Technical Leader. En caso de que hayan errores triviales o con severidad muy baja, estos pueden ser resueltos en el momento para ser probados inmediatamente. Errores con severidad más alta deben ser registrados en Jira como Story Bugs si son parte de una Historia de Usuario y como Bugs si son errores en producción o parte de una historia ya cerrada.

Nuestro enfoque es BDD (Behavior Driven Development) o desarrollo basado en comportamientos, para esto el equipo de QA se encarga de fomentar reuniones tales como tres amigos donde se deja más claro lo esperado por parte del negocio y todas las partes quedan alineadas y también fomentar el uso del formato Gherkin para la creación de criterios de aceptación y casos de prueba.

Errores, incidencias o Bugs

El siguiente es el proceso de manejo de bugs o errores en Jira, hay dos tipos de bugs, Story Bugs y Bugs, a continuación se resaltan las diferencias:

- **Story Bugs:** Son bugs relacionados directamente con la funcionalidad de la historia, es decir impactan el buen funcionamiento de alguna funcionalidad, estos bugs son creados durante el sprint son creados como subtareas de la Historia de Usuario padre.

- **Bugs:** Son bugs relacionados directamente con una funcionalidad ya probada o finalizada, errores en producción o errores globales que no son parte de ninguna Historia de Usuario, estos pueden ser creados durante el sprint y generalmente se crean por errores en el ambiente de producción.

Información que debe tener cada bug

Al momento de la creación de cualquier error, sea bug o story bug, la siguiente información debe ser incluida:
Summary: Es el título del bug, debe ser diciente para que cualquier persona entienda donde ocurre el bug.
Description: Se debe incluir la información detallada para reproducir el error, esta información incluye:

- Pasos para reproducir el bug: Una serie de pasos requeridos para reproducir el error
- Resultado esperado: Resultado del comportamiento esperado basado en las especificaciones funcionales, propuestas visuales, etc
- Resultado obtenido: Descripción del funcionamiento actual del sistema
- Imagen o video adjunto: Todos los bugs deben tener una imagen o un video que evidencie la existencia del bug.

Environment: Ambiente donde fue encontrado el bug, las opciones son: Local, Development, Staging y Producción

Severity: Impacto que causa el bug en la aplicación, se escoge entre las siguientes opciones:

- Blocker:
    - Funcional: Es un defecto que impide que el usuario realice alguna funcionalidad, ambientes, servicios o bases de datos caídos o alguna funcionalidad, página o servicio que impida que el usuario pueda utilizar el sistema de manera correcta. Ninguna Historia de Usuario debe estar cerrada con un bug de este tipo abierto.
    - Negocio: Es un defecto que afecta áreas con un potencial impacto a alto nivel.
    - Ninguna Historia de Usuario debe estar cerrada con un bug de este tipo abierto.
- High:
    - Funcional: El error resulta de una funcionalidad severamente deteriorada, puede haber una solución pronta al problema, pero el uso de la funcionalidad no es satisfactoria.
    - Negocio: El error afecta un área que puede los ingresos de la marca, pero el nivel de impacto puede no ser alto.

    - Ninguna Historia de Usuario debe estar cerrada con un bug de este tipo abierto.
- Medium:
    - Funcional: El error es una falla en un área no crítica del sistema, o puede ser un error visual muy evidente.
    - Negocio: Afecta un área que puede afectar la marca, diseño o servicios del sistema .
    - Ninguna Historia de Usuario debe estar cerrada con un bug de este tipo abierto.
- Low:
    - Funcional: El error no impacta la funcionalidad global del sistema, generalmente son cambios visuales.
    - Negocio: El error muestra pequeñas diferencias visuales o funcionales a las acordadas.
    - Puede haber Historias de Usuario cerradas con algunos de estos tipos de errores abiertos.

Reportes y/o evidencias

Item	Criterios	Frecuencia
Reporte de pruebas	○ Historias completadas por Sprint ○ Número de errores abiertos ○ Casos de prueba diseñados vs casos de prueba ejecutados	Cada Sprint
Evidencias	○ Pruebas Automatizadas ○ Pruebas Manuales ○ Smoke test en producción	Cada Sprint

Herramientas

Item	Descripción
Serenity	Conjunto de librerías para facilitar la escritura de pruebas de aceptación
Cucumber	Conjunto de librerias que soportan Behaviour-Driven Development (BDD)
Junit	Conjunto de librerias que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java
Intellij	IDE para Automatización
Tidy Gherkin	Herramienta que facilita crear los StepsDefinitions
CuckeTest	Herramienta que facilita creación de features en Gherkin
Java	Lenguaje de programación para automatizar
Jmeter	Herramienta para realizar pruebas de Performance o No Funcionales

Alcance

El equipo de QA, dependiendo del alcance de la iniciativa, tiene dos niveles de pruebas, smoke testing y full testing, smoke se enfoca en el camino feliz y las funcionalidades más importantes del proyecto y full, como su nombre lo indica, se enfoca en hacer todas las pruebas, incluyendo las funcionalidades importantes, las menos importantes, caminos felices y casos aislados.

El alcance se centran en las pruebas que se pueden realizar sobre la herramienta Operación Fuego Quasar en cuanto a su :

- Rendimiento (tiempo de espera en carga información el cual no debe exceder de 1 segundo)
- Pruebas básicas de seguridad
- Pruebas de carga.
- Pruebas de estrés.
- Pruebas de performance.
- Pruebas de manuales de API.
- Pruebas de Automatizadas de API.

Lo que no esta dentro de el alcance es :

- Look and field de la aplicación.
- Pruebas UI.
- Pruebas responsive.
- Pruebas de Intrusión (Implementadas por un tercero).
- Pruebas en otros navegadores.
- Pruebas en dispositivos móviles.