Plan de pruebas

Proyecto: Tienda de Mascotas Pet\_0001

## Historial de versiones

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Historial de versiones** | | |
| Fecha | Autor | Descripción |
| 18/07/2022 | Miguel Hugo | Creación del documento Plan de pruebas y casos de pruebas |
| 20/07/2022 | Miguel Hugo | Se finalizó con la creación de los casos de pruebas de acuerdo a las historias de usuario tomando los valores de los criterios de aceptación |
| 21/07/2022 | Miguel Hugo | Se finalizó con la documentación del plan de prueba de acuerdo con los datos de los casos de prueba y las historias de usuario |
|  |  |  |

## Índice

[Historial de versiones 1](#_Toc109312220)

[Índice 1](#_Toc109312221)

[Introducción 2](#_Toc109312222)

[Recursos 2](#_Toc109312223)

[Alcance 3](#_Toc109312224)

[Fuera del alcance 4](#_Toc109312225)

[Infraestructura y suposiciones 4](#_Toc109312226)

[Riesgos 5](#_Toc109312227)

## Introducción

Este documento de plan de pruebas hace parte integral de la metodología de pruebas, dicho documento es una herramienta general que describe el marco de trabajo para que sea efectivo y adecuado a lo largo de toda la vida del proyecto. El plan de pruebas es un conjunto de casos de pruebas que se encarga de probar una funcionalidad completa de un producto o software en concreto.

Nos enfocaremos en Módulo relacionado a la tienda de mascotas, el módulo que se implementará es el Pet, junto con ello creando los casos de prueba. Garantizando de que nuestro sistema Backend esté funcionando correctamente.

Tienda de mascotas La empresa pet store quiere construir el backend de un e-comerce para ofertar la adopción de mascotas:

Pet - Todo acerca de las mascotas:

* Crear una mascota
* Añadir imágenes de la mascota
* Actualizar información de una mascota
* Filtrar por mascotas (estado - id)
* Eliminar una mascota

## Recursos

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Tester | % Participación | Seniority |
| Miguel Hugo | 100% | Junior |

## Alcance

En este apartado se verá el alcance en relación a las historias de usuario que conforma la épica: Se realizará casos de prueba tanto inválidos como inválidos de cada historia de usuario.

En la primera historia de usuario que está relacionado con “subir imagen” se cubrirá los criterios de aceptación, tales como: El petId sólo debe ser un número entero, Si la petición es correcta se debe mostrar status code 200, si petId no es válido se debe mostrara status code 400 Pet not found, Sí algún valor no es válido se debe mostrar status code 405 invalid input. Utilizaremos el método POST

En la segunda historia de usuario llamado “Agregar una mascota en la tienda” se cubrirá los criterios de aceptación, tales como: Al crear una nueva mascota debe arrojar status code 201 con la información de la mascota, con sus campos obligatorios (name, status, category (id name) ), Si algún campo es inválido el status code debe ser 405 invalid input. Utilizaremos el método POST

En la tercera historia de usuario llamado “Actualizar información de una mascota” se cubrirá los criterios de aceptación, tales como: El petId solo debe ser un número entero, Si la petición es correcta se debe mostrar status code 200 con la información de la mascota actualizada, Sí petId no es válido se debe mostrar status code 400 Pet not found, Si algún valor no es válido se debe mostrar status code 405 invalid input, con sus campos obligatorios (id, category, in\_ategory, name\_category, name\_mascota, photoURL, id\_tag, name\_tag, status) Utilizaremos el método PUT.

En la cuarta historia de usuario “Actualizar una mascota en la tienda con datos de formulario” se cubrirá los criterios de aceptación, tales como: El petId solo debe ser un número entero, Sí la petición es correcta se debe mostrar status code 200 con la información de la mascota actualizada, Si petId no es válido se debe mostrar status code 405 invalid input. Llenando los campos: petId, mane y status. Utilizaremos el método POST.

En la historia de usuario cuatro y cinco que trata sobre “Filtrar mascotas por status” y “Filtrar mascotas por ID” respectivamente, se cubrirá los criterios de aceptación, tales como: Se debe buscar el status (available, pending, sold), Sí el estado es incorrecto se debe mostrar status code 400 invalid status value. Si la petición es correcta se debe mostrar status code 200 con un listado de las mascotas que coinciden con el estado. En la otra H.U queda tal que: El petId solo debe ser un número entero, Si la petición es correcta se debe mostrar status code 200 con la mascota buscada, Si petId no es válido se debe mostrar status code 400 Pet not found. En ambos casos se utilizará el método GET.

En la H.U “Eliminar una mascota” se cubrirá los criterios de aceptación, tales como: El petId solo debe ser un número entero, Si la petición es correcta se debe mostrar status code 200 eliminando la mascota, Si petId no es válido se debe mostrar status code 400 Pet not found. Se utilizará el método DELETE.

Se realizará pruebas automatizadas puesto que se están utilizando varios métodos. Con ello, se realizarán los casos de prueba de acuerdo a los criterios de aceptación que están en las historias de usuario.

## Fuera del alcance

Pruebas de sistema ya que no se tiene el sistema como tal y con ello no se puede probar la integración del sistema, pruebas de integración, de regresión. Estará fuera de alcance la gran mayoría de las pruebas ya que estamos en una etapa temprana del proyecto.

No sabemos por dónde va a pasar el servicio, en qué servidores va a pasar, lo que está fuera del alcance es el recorrido del proyecto por dónde va a pasar.

## Infraestructura y suposiciones

Tenemos que tener acceso al ambiente de pruebas, que es la configuración de todo lo que vamos a implementar en todo el proyecto tales como: Navegador, Sistema Operativo, Servidores, etc.

Las herramientas que vamos a utilizar para ejecutar las pruebas son: Postman que lo utilizaremos como colecciones de APIs en relación al Backend, necesitaremos muchos datos para lograr hacer las pruebas automatizadas, usaremos el GitHub que nos permitirá alojar proyectos utilizando el sistema control de versiones Git.

Con toda esta infraestructura adicionaremos nuestras variables de entorno tales como: Gradle, Maven que son gestores de dependencias, Java el cual es un lenguaje de programación de alto nivel orientado a objetos que no ayuda a la hora de estructurar clases.

Usar la herramienta Trello, para ir checando las historias de usuario, llevando un control y seguimiento de cada Historia de Usuario.

Los métodos que vamos a utilizar de acuerdo al protocolo HTTP son: GET (solicitar datos), DELET (Eliminar un recurso), POST (Enviar datos para la creación), PUT (Reemplazar por completo un recurso), PATCH (Reemplazar fragmentos de un recurso).

El Status Code es el que escribe el estado de la petición hecha al servidor que se utilizará en esta documentación de plan de pruebas son los siguientes: 2XX (200, 2NN), 4XX (400, 4NN). El 2XX es el status que fue recibida, procesada y completada y el 4XX es un error del cliente, si el cliente envió un dato erróneo.

Ambiente de Q.A se encargará de hacer los casos de pruebas y el documento de plan de pruebas. Los testers tendrán en cuenta las H.U, caso de prueba para hacer las pruebas en etapas más avanzadas del proyecto.

Hablando de los desarrolladores legamos a suponer que las pruebas unitarias deben ser probadas y que cubran el 98% del código, esto debe estar correctamente hecho para poder avanzar en las pruebas.

Otras personas del equipo de Q.A se encargará de las pruebas de performance, mantenibilidad, los atributos de calidad, entre otros.

## Riesgos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Factor de riesgo | Riesgo | Probabilidad (Alto - Medio - Bajo) | Impacto (Alto - Moderado - Bajo) | Severidad (Probabilidad por impacto) | Plan de mitigación |
| R1 | Rotaciones en el equipo | Si hay una alta rotación en el equipo, se incurrirá en pérdida de productividad por el on boarding y adaptación | Alto | Moderado | Alto | Negociar las horas del equipo hasta su finalización |
| R2 | Cambio en las historias de usuario | Si el cliente o la persona que solicitó el software decide agregar o cambiar la forma en la que utilizará dicho sistema | Alto | Medio | Alto | Analizar las H.U y ver cuánto se ha modificado, hablar con el cliente lo que se podría hacer. Llegando a un acuerdo |
| R3 | El sistema no pueda soportar muchos registros de mascotas | Si se registra varias mascotas en el sistema esto puede colapsar porque no soporta muchas cantidades de datos | Alto | Alto | Alto | Hacer pruebas de performance antes de finalizar para ver cuánto resiste en datos |
| R4 | Módulos mal documentados | A la hora de redactar dichos módulos, se implementó mal la forma de expresarlo en la documentación | Medio | Medio | Medio | Tener unas bases de pruebas, buscar otros casos de pruebas, plan de pruebas que los proyectos pasados. |