Plan de Pruebas de Software Funcionales

ParkingBox

Fecha: 23/08/2024

Tabla de contenido

Historial de Versiones

Fecha	Versión	Autor	Organización	Descripción
23/08/2024	1.0	Jhon Sebastián Álvarez, Javier Gil Sierra.	Estacionamiento Calle 109	Estas pruebas se hacen para asegurar la funcionalidad del sistema software.

Información del Proyecto

Empresa / Organización	Estacionamiento Calle 109
Proyecto	ParkingBox
Fecha de preparación	23/08/2024
Gerente / Líder de Proyecto	Jhon Sebastián Álvarez, Javier Gil Sierra.
Gerente / Líder de Pruebas de Software	Jhon Sebastián Álvarez, Javier Gil Sierra.

Aprobaciones

Nombre y Apellido	Cargo	Departamento u Organización	Fecha	Firma
Jhon Álvarez Medina	Aprendiz de programador software	Estacionamiento de la calle 109	23/08/2024	
Javier Gil Sierra	Aprendiz de programador software	Estacionamiento de la calle 109	23/08/2024	

Resumen Ejecutivo

Este plan de pruebas es un documento detallado que describe el proceso y los procedimientos para realizar pruebas estáticas de caja negra, enfocadas en la validación funcional de la aplicación utilizando la herramienta Cypress.js para pruebas end-to-end (E2E).

Propósito

El propósito del plan es garantizar que todas las funcionalidades críticas de la aplicación sean verificadas de manera exhaustiva mediante pruebas automatizadas que simulan el comportamiento del usuario en un entorno controlado. Estas pruebas están diseñadas para identificar y reportar cualquier error o inconsistencia en la funcionalidad de la aplicación antes de su despliegue en producción.

Tipo de Plan

Este es un plan detallado, que se centra en aspectos específicos del ciclo de pruebas, definiendo claramente los escenarios, casos de prueba, y las métricas que se utilizarán para evaluar el éxito de las pruebas.

Alcance

El alcance del plan de pruebas abarca todas las funcionalidades clave de la aplicación que están definidas en el plan de proyecto de software. Esto incluye, pero no se limita a, las operaciones críticas como el flujo de inicio de sesión, gestión de usuarios, y procesos de transacción. Las pruebas se centrarán en la interacción de los usuarios con la interfaz y las respuestas del sistema bajo diferentes condiciones.

Restricciones

Las principales restricciones de este plan incluyen limitaciones de recursos, tanto en términos de tiempo como de personal, y restricciones de presupuesto que limitan la cantidad de pruebas que pueden ser automatizadas y el alcance de las pruebas manuales. Además, el entorno de prueba debe ser cuidadosamente gestionado para asegurar que las pruebas no afecten a los sistemas en producción.

Esfuerzo de Pruebas

El esfuerzo de pruebas estará centrado en garantizar que las funcionalidades principales funcionen como se espera en diferentes navegadores y dispositivos, con un enfoque especial en identificar errores críticos que puedan impactar negativamente la experiencia del usuario. El esfuerzo también incluirá la validación de la robustez del sistema frente a entradas inesperadas o maliciosas.

Este plan de pruebas busca proporcionar una cobertura adecuada de las pruebas para asegurar la calidad y estabilidad de la aplicación antes de su despliegue, dentro de los límites y restricciones establecidos.

Alcance de las Pruebas

Elementos de Pruebas

Alcance de Pruebas

Este plan de pruebas detallado se enfoca en la validación funcional de los módulos Inicio de Sesión y Vehículo de la aplicación, utilizando la herramienta Cypress.js para pruebas end-to-end (E2E). Cypress.js permitirá una validación minuciosa de cada componente y operación, asegurando que todas las funcionalidades críticas funcionen como se espera.

Módulos y Componentes a Probar:

Módulo de Inicio de Sesión:

Ruta: **http://localhost:3000/**

Pruebas Específicas

Verificación de la carga correcta de la página principal.

Interacción con el control de "Inicio de Sesión".

Autenticación de usuarios con credenciales válidas.

Manejo de errores al ingresar credenciales inválidas.

Verificación de la redirección correcta post autenticación.

Validación de mensajes de error y éxito.

Módulo de Vehículo

CRUD Completo (Crear, Leer, Actualizar, Eliminar):

Crear Vehículo

Pruebas de creación de un nuevo vehículo con todos los campos requeridos.

Validación de la correcta inserción en la base de datos.

Obtener Vehículos

Verificación de la carga correcta de la lista de vehículos.

Filtrado y búsqueda de vehículos específicos.

Visualización de los detalles de un vehículo.

Editar Vehículo

Edición de la información de un vehículo existente.

Validación de la actualización de datos en la base de datos.

Eliminar Vehículo

Pruebas de eliminación de un vehículo.

Verificación de la eliminación correcta en la base de datos.

Funcionalidades Adicionales

Cotizar Vehículo

Simulación del proceso de cotización de un vehículo.

Cálculo del tiempo de estacionamiento y costos.

Validación del cambio de estado del vehículo a 'Salida'.

Reingresar Vehículo

Verificación de la funcionalidad de reingreso de un vehículo.

Actualización del estado del vehículo y otros datos relacionados.

Nivel de Detalle

Este testing es de nivel detallado, ya que Cypress.js permite observar y documentar el comportamiento específico de cada operación en tiempo real, mostrando resultados exhaustivos que indican claramente si las pruebas han sido aprobadas o si han fallado, junto con detalles de los errores detectados. Esto asegura una cobertura completa de todas las funcionalidades críticas dentro de los módulos de Inicio de Sesión y Vehículo.

Nuevas Funcionalidades a Probar

1. Inicio de Sesión

Acceso a la Aplicación: El usuario podrá iniciar sesión en la aplicación desde la página principal (http://localhost:3000/) ingresando sus credenciales. La prueba verifica que el usuario pueda acceder de manera segura y eficiente, y que reciba mensajes claros en caso de error o éxito al iniciar sesión.

2. Gestión de Vehículos

Crear Vehículo: Los usuarios podrán agregar un nuevo vehículo ingresando la información requerida. La prueba asegura que el proceso de registro es sencillo y que la información se guarda correctamente.

Ver Vehículos

Los usuarios podrán ver una lista de todos los vehículos registrados, con la posibilidad de buscar y filtrar por diferentes criterios. Se prueba que la visualización de los datos es clara y accesible.

Editar Vehículo

Los usuarios podrán modificar la información de un vehículo existente. La prueba verifica que los cambios se guardan correctamente y que la interfaz es intuitiva.

Eliminar Vehículo

Los usuarios podrán eliminar vehículos de la lista. La prueba asegura que el proceso es directo y que se reciben confirmaciones adecuadas.

Cotizar Vehículo

Los usuarios podrán calcular el costo del estacionamiento para un vehículo específico. La prueba v salida que el cálculo del tiempo y costo es preciso y fácil de usar.

Reingresar Vehículo

Los usuarios podrán actualizar el estado de un vehículo para permitir su reingreso. Se prueba que esta función es fluida y que la información del vehículo se actualiza correctamente.

Estas pruebas garantizan que las interacciones diarias del usuario con los módulos de Inicio de Sesión y Vehículos sean eficientes, seguras y satisfactorias.

Pruebas de Regresión

1. Barra de Navegación

Acceso a Funcionalidades: La barra de navegación permite a los usuarios acceder a diferentes secciones de la aplicación, incluyendo la página de inicio de sesión y la gestión de vehículos. Es esencial probar que, tras los cambios en el módulo de inicio de sesión y el módulo de vehículos, la barra de navegación sigue funcionando correctamente y redirige a las páginas correctas.

2. Sistema de Notificaciones

Mensajes de Éxito y Error: Las notificaciones que informan al usuario sobre el éxito o fracaso de acciones como iniciar sesión, crear, editar, eliminar, cotizar, o reingresar un vehículo deben funcionar correctamente. Es crucial asegurar que estos mensajes son claros, precisos, y se muestran en los momentos adecuados.

3. Gestión de Sesiones

Persistencia de la Sesión: Después de iniciar sesión, el sistema debe mantener la sesión activa y permitir que los usuarios naveguen entre diferentes funcionalidades sin interrupciones. Se debe verificar que la sesión se mantiene estable durante todas las interacciones con el módulo de vehículos.

4. Paginación y Filtrado de Listas

Visualización de Vehículos: La funcionalidad de paginación y filtrado en las listas de vehículos debe seguir funcionando correctamente. Los usuarios deben poder navegar y buscar vehículos sin problemas, incluso después de los cambios realizados en el módulo de gestión de vehículos.
Estas pruebas garantizan que las funcionalidades indirectamente afectadas por los cambios continúen proporcionando una experiencia de usuario coherente y sin interrupciones.

Funcionalidades a No Probar

1. Módulo de Categorías (CRUD)

Funcionalidades Excluidas

Creación, edición, eliminación, y visualización de categorías.

Razón de Exclusión

Este módulo no ha sido modificado recientemente y se ha probado exhaustivamente en iteraciones anteriores. Se considera estable y su funcionalidad no está relacionada con los cambios actuales en los módulos de inicio de sesión y vehículos.

Riesgos

Existe un riesgo bajo de que errores imprevistos puedan surgir si hay dependencias ocultas entre este módulo y otros componentes, aunque es poco probable.

2. Módulo de Área de Parqueo (CRUD)

Funcionalidades Excluidas

Creación, edición, eliminación, y visualización de áreas de parqueo.

Razón de Exclusión

Similar al módulo de categorías, este módulo no ha sido afectado por los cambios actuales. Las pruebas se centran en módulos con funcionalidades que interactúan directamente con los usuarios o los vehículos.

Riesgos

Existe un riesgo mínimo de que errores no detectados puedan afectar la experiencia del usuario, pero dado que este módulo está bien delimitado, el riesgo es considerado bajo.

3. Módulo de Usuarios (CRUD)

Funcionalidades Excluidas

Registro, edición, eliminación, y gestión de usuarios.

Razón de Exclusión

No se realizarán pruebas en este módulo ya que no ha sido objeto de cambios recientes y se considera que está operando de manera estable. Las pruebas se enfocan en áreas de mayor impacto, como el inicio de sesión y la gestión de vehículos.

Riesgos

Existe un riesgo leve de problemas no detectados, especialmente si hay interacciones complejas entre este módulo y las áreas probadas. Sin embargo, se considera que estos riesgos son manejables dada la estabilidad del módulo.
Este enfoque asegura que los recursos de pruebas se concentren en las áreas de mayor impacto y relevancia, aceptando un riesgo calculado en módulos considerados estables y no directamente afectados por los cambios recientes.

Enfoque de Pruebas (Estrategia)

1. Tipos de Pruebas a Realizar

Pruebas Funcionales: Se enfocan en verificar que cada funcionalidad de la aplicación se comporta según lo esperado. Cypress.js se utilizará para realizar pruebas end-to-end (E2E) de las funcionalidades clave, asegurando que la aplicación funcione correctamente desde la perspectiva del usuario final.

Pruebas Estáticas: Las pruebas no ejecutan código en tiempo de ejecución, sino que revisan el código y la configuración para asegurarse de que todo esté configurado correctamente.

Pruebas de Caja Negra: Estas pruebas validan la funcionalidad sin tener en cuenta la estructura interna del código. Se centra en las entradas y salidas de la aplicación.

1. Requerimientos Especiales

Configuración del Entorno: Cypress.js requiere que el entorno de desarrollo esté configurado correctamente para realizar pruebas. La configuración debe asegurar que la aplicación se ejecute en la dirección http://localhost:3000, que es la base para todas las pruebas E2E.

Dependencias: Cypress.js debe instalarse como una dependencia de desarrollo utilizando npm, y configurarse adecuadamente para integrarse con el proyecto Node.js y React.

3. Proceso de Instalación y Configuración de Cypress.js

Abrir una Terminal y Ubicarse en la Carpeta Raíz:

Desde la terminal, navega a la carpeta raíz de tu proyecto donde se encuentra el archivo package.json.

Instalar Cypress.js:

Ejecuta el siguiente comando en la terminal para instalar Cypress.js como una dependencia de desarrollo:

npm install cypress --save-dev

Esto descargará e instalará Cypress.js, y lo agregará a la sección de dependencias de desarrollo en tu archivo package.json.
Configurar Cypress.js:

Después de la instalación, Cypress generará automáticamente un archivo de configuración predeterminado. Dirígete al archivo cypress.config.js para configurarlo.
Modifica el archivo cypress.config.js con la siguiente configuración:

```
const { defineConfig } = require('cypress')

module.exports = defineConfig({
  e2e: {
    baseUrl: 'http://localhost:3000',
    setupNodeEvents(on, config) {
      // Puedes agregar eventos aquí
    },
  },
})
```

Esta configuración establece la URL base para las pruebas E2E y permite la configuración adicional de eventos si es necesario.
Escribir Pruebas E2E:

Dirígete al archivo cypress/e2e y crea un archivo de prueba con el siguiente contenido:

```
describe('My First Test', () => {
  it('Visits the React app', () => {
    cy.visit('http://localhost:3000')
    cy.contains('PARKINGBOX') // Reemplaza con un texto que exista en tu app
  })
})
```

Después para inicializar el cypress.js ingresar este comando: npx cypress open

Esta prueba visita la aplicación React en http://localhost:3000 y verifica que un texto específico, como 'PARKINGBOX', esté presente en la página.

4. Configuraciones a Probar

Entorno Local: Las pruebas se realizarán en el entorno local (http://localhost:3000), que emula el entorno de desarrollo de la aplicación.
Configuración del Servidor: Asegúrate de que el servidor local esté en ejecución antes de iniciar las pruebas, para que Cypress.js pueda interactuar con la aplicación correctamente.

5. Nivel de Pruebas de Regresión

Pruebas de Regresión Detalladas: Se realizarán pruebas de regresión cada vez que se implementen cambios en el código, para asegurar que las funcionalidades existentes no se vean afectadas por nuevas implementaciones o modificaciones.

6. Conclusión

Cypress.js proporciona una solución integral para realizar pruebas funcionales, estáticas y de caja negra en aplicaciones Node.js y React. La instalación y configuración son sencillas, y una vez configurado, permite realizar pruebas detalladas que aseguran la estabilidad y funcionalidad de la aplicación desde la perspectiva del usuario.

Criterios de Aceptación o Rechazo

Criterios de Aceptación o Rechazo: Las pruebas realizadas deben alcanzar un 100% de aceptación para ser consideradas exitosas. Esto implica que todos los casos de prueba cubren los componentes del módulo de inicio de sesión, así como el módulo de gestión de vehículos con su funcionalidad CRUD completa (crear, editar, obtener, eliminar). Además, las pruebas deben verificar la correcta funcionalidad de los botones de acción específicos del módulo, como "Cotizar" y "Reingresar". Si todas estas funcionalidades funcionan según lo esperado, el plan de pruebas se considerará completado y aceptado.

Criterios de Suspensión

Las pruebas pueden ser suspendidas si no se siguen estrictamente los pasos indicados en el documento de ejecución de pruebas funcionales. Cualquier desviación de estos procedimientos o la omisión de un proceso obligatorio pueden resultar en pruebas no exitosas. Por ejemplo, si un paso crítico en el flujo de trabajo del módulo de inicio de sesión o en el proceso de gestión de vehículos no se ejecuta correctamente, las pruebas deben ser detenidas hasta que se corrija el problema.

Criterios de Reanudación

Las pruebas se reanudarán una vez que el usuario encargado de ejecutarlas siga rigurosamente los márgenes establecidos para cada prueba. Es fundamental no saltar pasos de suma importancia, ya que estos son esenciales para garantizar la integridad de las pruebas. La reanudación solo será posible cuando se haya verificado que todos los pasos se cumplen conforme a lo especificado en el documento de pruebas funcionales, asegurando así la precisión de los resultados.

Entregables

Como parte de la ejecución del plan de pruebas, se entregarán dos documentos clave. El primero es un Plan Maestro de Pruebas, que detalla las pruebas propuestas, los objetivos y el alcance del proceso de pruebas. El segundo documento es el de Ejecución de Pruebas Funcionales, donde se registran los resultados de las pruebas realizadas, los casos de prueba ejecutados, los defectos encontrados y la evidencia que respalda la funcionalidad probada. Estos documentos serán fundamentales para la validación final del software y servirán como referencia para futuras mejoras.

Recursos

Requerimientos de Entornos – Hardware

Para llevar a cabo las pruebas de software, se requiere un equipo de computación con especificaciones que permitan ejecutar programas actuales con fluidez. Aunque un computador con recursos básicos puede ser suficiente, es preferible disponer de equipos con mejor calidad, incluyendo procesadores rápidos, suficiente memoria RAM (mínimo 8 GB), y almacenamiento adecuado (mínimo 256 GB SSD). Además,

los equipos deben estar conectados a una red estable que permita el acceso sin interrupciones a los servidores de aplicación y bases de datos necesarias para las pruebas.

Requerimientos de Entornos – Software

El entorno de pruebas requiere la instalación de Visual Studio Code como editor de código principal. Es fundamental seguir los procedimientos de instalación detallados en el manual técnico para asegurar un entorno de desarrollo correcto. Además, los testers deben tener acceso a sistemas en un entorno de pruebas, incluyendo bases de datos (Xammp o workbenth), y deben estar familiarizados con lenguajes y tecnologías como JavaScript, Node.js y React.js, que son esenciales para la realización de las pruebas.

Herramientas de Pruebas Requeridas

Las pruebas se realizarán utilizando Cypress.js, una herramienta de automatización de pruebas E2E (End-to-End). Es crucial que el personal asignado a las pruebas esté capacitado para configurar e instalar Cypress, comprendiendo que las pruebas realizadas son de caja negra y estáticas, lo que asegura que las pruebas se ejecuten de manera automática y sin interferencias manuales, reduciendo así posibles errores y malentendidos en los resultados.

Personal

El equipo de pruebas debe estar compuesto por al menos un encargado de testing, que se responsabilizará de ejecutar las pruebas, y un analista de pruebas, que se encargará de interpretar los resultados y documentar cualquier incidente o fallo. Además, es esencial contar con un líder de pruebas que coordine las actividades, asegurándose de que todas las pruebas se hayan completado correctamente y de que el equipo siga los procedimientos establecidos en el plan de pruebas.

Entrenamiento

Antes de iniciar las pruebas, es necesario que todo el personal involucrado reciba entrenamiento adecuado en las herramientas y lenguajes que constituyen el sistema, como Cypress.js, JavaScript, Node.js y React.js. Este entrenamiento garantizará que el equipo esté preparado para interactuar con el sistema, realizar pruebas efectivas y aplicar correcciones si es necesario, minimizando así la curva de aprendizaje y asegurando una ejecución fluida del plan de pruebas.

Planificación y Organización

Procedimientos para las Pruebas

El proceso de pruebas comenzó con la selección de una herramienta adecuada para realizar pruebas en un entorno de JavaScript, optándose por Cypress.js debido a su capacidad para realizar pruebas automáticas E2E (End-to-End). Después de investigar y seguir los procedimientos de instalación, Cypress.js fue implementado en el proyecto, permitiendo configurar el entorno de pruebas correctamente. Con la herramienta lista, se elaboró un Plan Maestro de Pruebas, detallando los pasos a seguir. Las pruebas fueron ejecutadas en los módulos clave del sistema, específicamente en el de inicio de sesión y el de ingreso de vehículos, culminando con la documentación de los resultados en el documento de Ejecución de Pruebas Funcionales.

Premisas

En la Matriz de Responsabilidades se destacan los roles y tareas asignadas a los miembros del equipo de pruebas. Los aprendices de programación Jhon Álvarez Medina y Javier Gil Sierra fueron los responsables de documentar y desarrollar las pruebas de funcionalidad del sistema. Su instructor, Albeiro Ramos, actuó como aprobador del trabajo realizado y fue la persona consultada durante el proceso. La

empresa Estacionamiento Calle 109 fue informada de los avances y resultados obtenidos, asegurando que todos los interesados estuvieran al tanto del progreso del proyecto.

Cronograma

La de pruebas se diseñó tomando en cuenta las actividades previas, como la documentación y el desarrollo del sistema, realizadas en periodos anteriores. Las pruebas de software fueron implementadas en un lapso de cinco trimestres, considerando hitos clave como la instalación de Cypress.js, la configuración del entorno de pruebas, y la ejecución de las pruebas en los módulos seleccionados. Cada fase del cronograma fue programada de manera que permitiera cumplir con los tiempos estimados y garantizar la calidad del proceso de pruebas (Link).

Premisas

Una de las premisas fundamentales en este plan fue el uso de Cypress.js debido a la compatibilidad del sistema con JavaScript. Esta elección no solo guio la selección de la herramienta de pruebas, sino que también influyó en la documentación y configuración del entorno de pruebas. Se asumió que la herramienta funcionaría de manera eficiente dentro del marco de tiempo establecido y que los recursos estarían disponibles según lo planificado, asegurando así el desarrollo adecuado de las pruebas.

Dependencias y Riesgos

Las pruebas con Cypress.js dependen de varias condiciones, como la disponibilidad de la herramienta al iniciar y mantener su configuración estable durante la ejecución. Entre los riesgos se identifican restricciones de tiempo, ya que si la herramienta es suspendida o el monitor se pone en reposo, Cypress.js deberá ser reiniciado, lo que

podría retrasar las pruebas. También se consideró el riesgo de premisas no ciertas, como la falsa creencia de que las pruebas son de caja blanca, cuando en realidad son de caja negra y enfocadas en el comportamiento del sistema sin conocer su estructura interna.

Referencias

Para el desarrollo de este plan de pruebas, se consultaron varios recursos en línea. Se incluyen referencias a la documentación oficial de Cypress.js, sitios especializados en el desarrollo de planes maestros de pruebas y guías sobre la configuración y uso de herramientas de automatización de pruebas. Estos documentos proporcionaron un marco teórico y práctico que respaldó las decisiones tomadas durante el proceso de pruebas y la elaboración del plan.

Glosario

El glosario incluye definiciones de términos técnicos clave utilizados en este documento. Por ejemplo, Node.js se refiere a un entorno de ejecución de JavaScript del lado del servidor; Cypress.js es una herramienta de automatización de pruebas E2E; y React.js es una biblioteca de JavaScript para construir interfaces de usuario. También se incluyen términos relacionados con las pruebas, como pruebas funcionales, que son aquellas que evalúan la funcionalidad de un sistema, y pruebas de caja negra, que son pruebas donde el tester no tiene conocimiento del código o la estructura interna del sistema.