Nombre: Juan Hernández

Matricula: 2023-0613

Carrera: Desarrollo de Software

Tema: Proyecto Final

Materia: Programación III

Maestro: Kelyn Tejada Belliard

Fecha de Entrega: 10/8/2024

Introducción

Este proyecto final de Programación 3 tiene como objetivo aplicar los conocimientos adquiridos en las materias anteriores y profundizar en la implementación de sistemas utilizando la metodología Agile-Scrum. Esta metodología es ampliamente reconocida por su enfoque iterativo e incremental en el desarrollo de software, lo que permite una mayor flexibilidad y adaptación a los cambios durante el ciclo de vida del proyecto. El proyecto se estructurará en dos componentes principales: la creación e implementación de un sistema modelo y el desarrollo de un plan de pruebas exhaustivo.

Componentes del Proyecto

Desarrollo del Sistema Modelo:

Utilizando un sistema previamente desarrollado en materias anteriores o creando uno nuevo, se buscará alinear el trabajo de desarrollo con los principios y prácticas de la metodología Agile-Scrum. Esto incluye la definición de historias de usuario (HU), la planificación y ejecución de sprints, y la entrega de incrementos funcionales al final de cada sprint. El sistema elegido para este proyecto deberá demostrar la aplicación efectiva de Scrum, mostrando la colaboración constante entre el equipo de desarrollo y los interesados, así como la capacidad de adaptarse a los cambios y mejoras sugeridas durante el proceso.

Plan de Pruebas:

Un componente esencial del proyecto será la creación de un plan de pruebas detallado, que abarque al menos 20 casos de pruebas obligatorios para cualquier tipo de sistema. Estos casos de pruebas estarán directamente relacionados con las historias de usuario definidas para el proyecto. Entre los casos de pruebas se incluirán funcionalidades críticas como el inicio de sesión de usuarios, la creación de usuarios y roles, y otras funcionalidades específicas del sistema desarrollado. El objetivo del plan de pruebas es garantizar que el sistema sea robusto, seguro y cumpla con los requisitos definidos, asegurando así su calidad y funcionalidad.

Elabore una estrategia de trabajo donde pueda planificar los siguientes puntos:

1. Nombre del proyecto de Software: Bookshop. Este es una aplicación con la que podremos registrar libros, registrar autores y registrar las editoriales que publican estos libros.

2. Tecnologías a aplicar: Node, Express, HBS, MySQL.

3. Objetivo del Proyecto: Lograr que una biblioteca pueda tener el control de los libros que tiene, las editoriales que maneja y los autores. A parte, también que los autores y editoriales sepan que sus libros están disponibles en este lugar.

4. Alcance del proyecto: El alcance de este proyecto se define en torno a la creación, desarrollo y validación de un sistema de software utilizando la metodología Agile-Scrum. A continuación, se detallan los principales aspectos que se cubrirán:

5. Definición y Desarrollo del Sistema: Selección del Sistema: Utilización de un sistema previamente desarrollado en materias anteriores o la creación de un nuevo sistema basado en las necesidades identificadas.
   Historias de Usuario (HU): Redacción de historias de usuario que describan las funcionalidades clave del sistema desde la perspectiva del usuario final.
   Planificación de Sprints: División del desarrollo en sprints, cada uno con un conjunto de tareas y metas específicas que resultarán en un incremento funcional del sistema.
   Implementación del Sistema: Desarrollo del sistema basado en las historias de usuario y conforme a los principios de la metodología Scrum, con entregas incrementales y adaptaciones basadas en el feedback recibido.

6. Cronograma del proyecto:

FASE	**ACTIVIDADES CLAVE**	**ENTREGABLES**	**DURACION ESTIMADA**
1. INVESTIGACION Y PLANIFICACION			
Investigación de mercado: Identificar las necesidades específicas de las bibliotecas y los usuarios.	Informe de investigación de mercado	Informe detallado con hallazgos clave y segmentación de usuarios.	2-3 semanas
Análisis de la competencia: Evaluar las apps existentes y sus características.	Análisis comparativo	Matriz comparativa de funcionalidades y UX/UI.	1 semana
Definición de requisitos funcionales: Especificar las funcionalidades clave de la app.	Documento de requisitos funcionales	Documento de requisitos funcionales (FRS) detallado.	2-3 semanas

Diseño de la arquitectura de la app: Establecer la estructura técnica de la app.	Diagrama de arquitectura	Diagrama de arquitectura UML y documentación técnica.	1 semana
2. **DESARROLLO**			
Desarrollo del frontend: Crear la interfaz de usuario de la app.	Prototipo interactivo	Prototipo interactivo de alta fidelidad y código fuente.	4-6 semanas
Desarrollo del backend: Desarrollar la lógica de la app y la base de datos.	API RESTful	API RESTful documentada y base de datos estructurada.	6-8 semanas
Integración de funcionalidades clave: Incorporar las funcionalidades de búsqueda, préstamo, devolución, etc.	App funcional con características básicas	Versión alfa de la app funcional con todas las características principales.	4-6 semanas
3. Pruebas y Refinamiento			
Pruebas unitarias: Verificar el correcto funcionamiento de cada módulo.	Informes de pruebas unitarias	Informes de cobertura de código y resultados de pruebas.	2-3 semanas
Pruebas de integración: Asegurar que los diferentes módulos interactúan correctamente.	Informes de pruebas de integración	Informes de pruebas de integración y resolución de errores.	2 semanas
Pruebas de usuario: Obtener feedback de usuarios reales para mejorar la usabilidad.	Informe de pruebas de usuario	Informe de usabilidad con recomendaciones de mejora.	2 semanas

4. Lanzamiento y Mantenimiento			
Preparación para el lanzamiento: Configuración de la tienda de apps y marketing inicial.	Plan de lanzamiento	Materiales de marketing (descripciones, capturas de pantalla) y plan de lanzamiento	2 semanas
Lanzamiento de la app: Publicación en las tiendas de apps.	App disponible en las tiendas	App publicada en las tiendas de aplicaciones (App Store, Google Play).	1 semana
Mantenimiento y actualizaciones: Corrección de errores y adición de nuevas funcionalidades.	Plan de mantenimiento	Plan de mantenimiento y versiones actualizadas de la app.	Continuo

7. Definir claramente lo que el sistema en un primer Release va a poder hacer: Se va a poder registrar y guardar en la base de datos los libros, los autores y editoriales, también poder hacer todas las funciones ApiREST en cada uno de estos elementos. Aparte, podremos también filtrar los libros en la pantalla principal mediante nombres en un buscador .

8. Definir requerimientos del sistema para el primer Release:

Requerimientos Mínimos Viables (MVP) para el Primer Lanzamiento:

Catálogo de libros:
Agregar nuevos libros (título, autor, ISBN, editorial, año de publicación).
Buscar libros por título, autor o ISBN.
Ver detalles de un libro (sinopsis, ejemplares disponibles, etc.).

Gestión de usuarios:
Registro de nuevos usuarios (bibliotecarios y usuarios finales).
Autenticación segura (contraseñas encriptadas).
Permisos de acceso diferenciados (bibliotecarios con permisos de administración, usuarios finales solo para consulta).

Préstamos y devoluciones:
Realizar préstamos de libros a usuarios registrados.
Registrar fechas de préstamo y devolución.
Notificaciones a usuarios sobre fechas de devolución próximas.

Gestión de ejemplares:
Registrar múltiples ejemplares de un mismo libro.
Indicar el estado de cada ejemplar (disponible, prestado, perdido, etc.).

Informes básicos:
Listado de libros más solicitados.
Estadísticas de préstamos por usuario.
Informe de ejemplares perdidos o dañados.

Requerimientos Adicionales a Considerar (para futuras versiones):

Reservas de libros: Permitir a los usuarios reservar libros que estén prestados.
Sugerencias de lectura: Algoritmo de recomendación basado en el historial de préstamos.

Integración con otras plataformas: Catálogos de bibliotecas públicas, Goodreads, etc.
Módulo de gestión de usuarios más avanzado: Roles de usuarios, grupos, permisos personalizados.

Notificaciones push: Avisos sobre novedades, reservas, etc.
Módulo de estadísticas más completo: Análisis de uso de la app, comportamiento de los usuarios.

Consideraciones Técnicas:

Plataformas: Definir las plataformas móviles a las que se dirigirá la app (iOS, Android o ambas).

Tecnología: Seleccionar las tecnologías de desarrollo adecuadas (lenguajes de programación, frameworks, base de datos).

Diseño de la interfaz: Crear una interfaz de usuario intuitiva y fácil de usar.

Seguridad: Implementar medidas de seguridad para proteger los datos de los usuarios.
Escalabilidad: Diseñar la arquitectura de la app para que pueda soportar un aumento en el número de usuarios y libros.

Fechas para las ceremonias

1. **Sprint Planning:**

- **Fecha:** 10-8-24.
- **Duración:** 2 horas.
- **Objetivo:** Definir el objetivo del Sprint, seleccionar los Items del Product Backlog a desarrollar y crear un plan de trabajo.

2. **Daily Scrum:**

- **Fecha:** A partir de mañana y durante los próximos 10 días hábiles.
- **Hora:** Por ejemplo, a las 10:00 AM.
- **Duración:** 15 minutos.
- **Objetivo:** Sincronizar al equipo, revisar el progreso desde la última reunión, identificar impedimentos y planificar el trabajo del día.

3. **Sprint Review:**

- **Fecha:** El último día del Sprint (dentro de 10 días hábiles).
- **Duración:** 1 hora.

- **Objetivo:** Demostrar el incremento del producto al Product Owner y a los stakeholders, recopilar feedback y obtener la aceptación del Sprint.

4. Sprint Retrospective:

- **Fecha:** Inmediatamente después del Sprint Review (el mismo día).
- **Duración:** 30 minutos.
- **Objetivo:** Reflexionar sobre el Sprint pasado, identificar lo que fue bien, lo que se puede mejorar y definir acciones para el próximo Sprint.

Plan de Pruebas

1. Lista de requerimientos funcionales y no funcionales de acorde a las historias de usuarios. (mínimo de 10 HU).

   • HU01: Como usuario, quiero registrarme en la plataforma para poder acceder a todas las funcionalidades.
   Requerimiento Funcional: El sistema debe permitir a los usuarios registrarse con un nombre de usuario único, correo electrónico y contraseña válida.
   Requerimiento No Funcional: El proceso de registro debe completarse en menos de 5 segundos, y el sistema debe verificar la validez del correo electrónico en tiempo real.
   • HU02: Como usuario, quiero iniciar sesión en la plataforma para poder realizar compras y gestionar mi cuenta.
   Requerimiento Funcional: El sistema debe permitir a los usuarios iniciar sesión con un nombre de usuario y contraseña válidos.

Requerimiento No Funcional: El tiempo de respuesta al iniciar sesión no debe exceder los 2 segundos.

- HU03: Como usuario, quiero agregar productos al carrito de compras para poder adquirirlos más tarde.

Requerimiento Funcional: El sistema debe permitir a los usuarios agregar uno o más productos al carrito de compras.

Requerimiento No Funcional: El sistema debe reflejar el cambio en el carrito de compras inmediatamente sin necesidad de recargar la página.

- HU04: Como usuario, quiero eliminar productos del carrito para ajustar mi selección antes de realizar una compra.

Requerimiento Funcional: El sistema debe permitir a los usuarios eliminar productos del carrito de compras.

Requerimiento No Funcional: El carrito debe actualizarse instantáneamente, mostrando el nuevo precio total y la cantidad de productos.

- HU05: Como usuario, quiero ver el resumen de mi carrito de compras para verificar los productos seleccionados y su precio total.

Requerimiento Funcional: El sistema debe mostrar un resumen detallado del carrito, incluyendo los productos añadidos, cantidades y precio total.

Requerimiento No Funcional: El resumen del carrito debe cargarse en menos de 3 segundos.

- HU06: Como usuario, quiero poder realizar un pedido con datos de pago válidos para completar mi compra.

Requerimiento Funcional: El sistema debe permitir a los usuarios realizar un pedido introduciendo datos de pago válidos.

Requerimiento No Funcional: El proceso de validación y confirmación del pedido debe completarse en menos de 5 segundos.

- HU07: Como usuario, quiero recibir una confirmación de pedido por correo electrónico para asegurarme de que mi compra fue exitosa.

Requerimiento Funcional: El sistema debe enviar un correo electrónico de confirmación con los detalles del pedido después de que se haya completado la compra.

Requerimiento No Funcional: El correo de confirmación debe enviarse en un máximo de 1 minuto tras la confirmación del pedido.

- HU08: Como usuario, quiero poder iniciar sesión utilizando caracteres especiales en mi nombre de usuario para mayor seguridad.

Requerimiento Funcional: El sistema debe permitir el inicio de sesión con nombres de usuario que incluyan caracteres especiales.
Requerimiento No Funcional: La validación de los caracteres especiales debe realizarse en tiempo real y sin afectar la experiencia del usuario.

- HU09: Como usuario, quiero poder realizar pedidos con tarjetas de crédito válidas para completar mis compras de forma segura.
  Requerimiento Funcional: El sistema debe aceptar y procesar tarjetas de crédito válidas para realizar pedidos.
  Requerimiento No Funcional: La validación de la tarjeta debe realizarse en menos de 2 segundos.
- HU10: Como usuario, quiero poder enviar mensajes al servicio de atención al cliente desde la página de contacto para resolver cualquier duda o problema.
  Requerimiento Funcional: El sistema debe permitir a los usuarios enviar mensajes a través de la página de contacto.
  Requerimiento No Funcional: El sistema debe confirmar el envío del mensaje en menos de 3 segundos y mostrar un mensaje de éxito.

2. Definir cuales son los criterios de aceptación de las pruebas.

Proceso de Estimación de Historias (HU01)
Funcionalidad: Los participantes pueden estimar las historias de usuario utilizando tarjetas numeradas para reflejar sus puntos de vista.
Validación: El sistema debe recopilar todas las estimaciones y mostrar el promedio, así como las estimaciones individuales.
Interactividad: Los participantes deben poder cambiar sus estimaciones si se consideran necesarias tras la discusión.
Rendimiento: Las estimaciones deben reflejarse en la plataforma en menos de 1 segundo.
Crear y Organizar el Backlog (HU02)
Funcionalidad: El equipo puede crear y organizar un backlog de sprint con historias de usuario y épicas.
Validación: El backlog debe mostrar claramente las prioridades, los puntos de historia asignados y la asignación a los miembros del equipo.
Usabilidad: La interfaz debe ser intuitiva y permitir la reorganización del backlog mediante arrastrar y soltar.
Rendimiento: Los cambios en el backlog deben actualizarse en menos de 2 segundos.

Planificación del Sprint (HU03)
Funcionalidad: Los participantes pueden planificar el sprint, seleccionando historias del backlog y asignando tareas específicas.
Validación: El sistema debe verificar que la capacidad del equipo esté correctamente distribuida entre las tareas.
Interactividad: La planificación del sprint debe incluir la opción de agregar o quitar tareas según la discusión del equipo.
Rendimiento: La planificación debe completarse en menos de 5 minutos para sprints de tamaño estándar.
Revisión y Ajuste del Sprint (HU04)
Funcionalidad: El equipo puede revisar el progreso del sprint en tiempo real y ajustar tareas según sea necesario.
Validación: El sistema debe mostrar el progreso actualizado de cada tarea y permitir ajustes de estimación o reasignación.
Interactividad: Los ajustes deben ser colaborativos, permitiendo que todos los miembros del equipo contribuyan.
Rendimiento: Las actualizaciones deben reflejarse inmediatamente en la vista del sprint.
Presentación del Incremento Funcional (HU05)
Funcionalidad: Al final del sprint, el equipo debe poder presentar un incremento funcional que muestre el trabajo realizado.
Validación: El sistema debe verificar que todas las historias completadas estén incluidas en el incremento.
Interactividad: La presentación del incremento debe ser clara y comprensible, con una visualización detallada de las mejoras.
Rendimiento: La presentación debe cargarse en menos de 3 segundos y estar disponible para revisión posterior.

3. Definir cuales son los criterios de rechazo en las pruebas

Proceso de Estimación de Historias (HU01)

Funcionalidad: Los participantes no pueden estimar las historias de usuario utilizando tarjetas numeradas.
Validación: El sistema no recopila todas las estimaciones o no muestra el promedio correctamente.

Interactividad: Los participantes no pueden cambiar sus estimaciones tras la discusión.
Rendimiento: Las estimaciones tardan más de 1 segundo en reflejarse en la plataforma.
Crear y Organizar el Backlog (HU02)

Funcionalidad: El equipo no puede crear o organizar un backlog de sprint con historias de usuario y épicas.
Validación: El backlog no muestra las prioridades correctamente o las asignaciones están mal distribuidas.
Usabilidad: La interfaz no permite la reorganización del backlog mediante arrastrar y soltar.
Rendimiento: Los cambios en el backlog tardan más de 2 segundos en actualizarse.
Planificación del Sprint (HU03)

Funcionalidad: Los participantes no pueden planificar el sprint seleccionando historias del backlog.
Validación: El sistema no verifica correctamente la capacidad del equipo, resultando en una distribución ineficaz.
Interactividad: La planificación del sprint no permite agregar o quitar tareas durante la discusión.
Rendimiento: La planificación del sprint tarda más de 5 minutos en completarse.
Revisión y Ajuste del Sprint (HU04)

Funcionalidad: El equipo no puede revisar el progreso del sprint en tiempo real o ajustar tareas según sea necesario.
Validación: El sistema no muestra el progreso actualizado de cada tarea o no permite ajustes de estimación.
Interactividad: Los ajustes no son colaborativos, impidiendo la contribución de todos los miembros del equipo.
Rendimiento: Las actualizaciones tardan más de lo esperado en reflejarse en la vista del sprint.
Presentación del Incremento Funcional (HU05)

Funcionalidad: El equipo no puede presentar un incremento funcional al final del sprint.
Validación: El sistema no verifica que todas las historias completadas estén incluidas en el incremento.
Interactividad: La presentación del incremento no es clara o comprensible, dificultando la visualización de las mejoras.

Rendimiento: La presentación tarda más de 3 segundos en cargarse o no está disponible para revisión posterior.

4. Comentar sobre las herramientas de pruebas que estarían usando y justificar respuesta.

Playwright

Descripción: Utilizare esta herramienta ya que me permite poder trabajar con una sintaxis que se me hace cómoda (vanilla js). A parte de esto esta me permite generar reportes de una manera más fácil y rápida. A parte de esto tiene una librería que permite muy rápida de generar videos de cada una de las pruebas.

Justificación: Es la herramienta principal para automatizar pruebas de la interfaz de usuario, permitiéndote escribir scripts que interactúan con tu aplicación como lo haría un usuario real.

5. Estimar el tiempo que duraría la ejecución de pruebas. (elaborar cronograma de trabajo).

ACTIVIDAD	**DURACION ESTIMADA**	**DESCRIPCION**
1. CONFIGURACION DEL ENTORNO	1HORA	Instalación y configuración de herramientas de pruebas.
2. CREACION DE CASOS DE PRUEBAS	2HORAS	Redacción y revisión de los casos de prueba.
3. EJECUCION DE CASOS DE PRUEBA	1.5HORAS	Ejecución de 20 casos de prueba (5 minutos por caso).
4. GENERACION DE INFORMES	2HORAS	Compilación y revisión de informes de resultados.
5. REVISION Y AJUSTES	1HORA	Revisión de resultados y ajustes necesarios.
6. REPORTE FINAL Y REVISION	1HORA	Preparación del reporte final y revisión con el equipo.

6. Elaborar plantillas para cada caso de pruebas

1. Plantilla para Inicio de Sesión (HU01)

Caso de Prueba: Inicio de Sesión con Credenciales Válidas

ID: TC_HU01_01

Descripción: Verificar que el usuario puede iniciar sesión correctamente con un nombre de usuario y contraseña válidos.

Precondiciones: El usuario debe estar registrado en el sistema.

Pasos:

Navegar a la página de inicio de sesión.

Ingresar el nombre de usuario válido.

Ingresar la contraseña válida.

Hacer clic en el botón "Iniciar Sesión".

Resultados Esperados: El usuario es redirigido a la página principal de la aplicación.

Criterios de Aceptación:

El usuario inicia sesión correctamente.

El tiempo de respuesta no supera los 2 segundos.

Criterios de Rechazo:

El usuario no puede iniciar sesión con credenciales válidas.

El sistema tarda más de 2 segundos en completar el inicio de sesión.

2. Plantilla para Recuperación de Contraseña (HU02)

Caso de Prueba: Solicitar Enlace de Recuperación de Contraseña

ID: TC_HU02_01

Descripción: Verificar que el usuario puede solicitar un enlace de recuperación de contraseña.

Precondiciones: El usuario debe tener un correo electrónico registrado en el sistema.

Pasos:

Navegar a la página de inicio de sesión.

Hacer clic en "Olvidé mi contraseña".

Ingresar el correo electrónico registrado.

Hacer clic en "Enviar Enlace de Recuperación".

Resultados Esperados: El sistema envía un correo electrónico con un enlace de recuperación.

Criterios de Aceptación:

El enlace de recuperación es enviado correctamente en menos de 1 minuto.

El enlace expira después de 24 horas.

Criterios de Rechazo:

El correo electrónico con el enlace de recuperación no se envía.

El enlace no expira después de 24 horas.

El proceso tarda más de 1 minuto en completarse.

3. Plantilla para Visualización de Tareas Pendientes (HU03)

Caso de Prueba: Visualización Correcta de Tareas Pendientes

ID: TC_HU03_01

Descripción: Verificar que la lista de tareas pendientes se muestra correctamente después del inicio de sesión.

Precondiciones: El usuario debe tener tareas pendientes asignadas.

Pasos:

Iniciar sesión en la plataforma.

Navegar a la sección de tareas pendientes.

Resultados Esperados: Solo se muestran las tareas que están pendientes (sin completar).

Criterios de Aceptación:

La lista de tareas se carga en menos de 3 segundos.

Solo se muestran tareas pendientes.

Criterios de Rechazo:

Se muestran tareas que ya están completadas.

La lista de tareas tarda más de 3 segundos en cargarse.

La interfaz no es clara.

4. Plantilla para Compartir en el Foro de Tareas (HU04)

Caso de Prueba: Creación de Nuevo Debate en el Foro

ID: TC_HU04_01

Descripción: Verificar que el usuario puede crear un nuevo debate en el foro.

Precondiciones: El usuario debe estar autenticado en la plataforma.

Pasos:

Navegar al foro de tareas.

Hacer clic en "Nuevo Debate".

Completar todos los campos obligatorios.

Hacer clic en "Crear Debate".

Resultados Esperados: El nuevo debate se añade a la lista de temas en el foro.

Criterios de Aceptación:

El debate se crea correctamente en menos de 2 segundos.

Todos los campos obligatorios son validados.

Criterios de Rechazo:

El debate no se añade a la lista de temas.

El sistema permite la creación del debate sin completar los campos obligatorios.

El debate tarda más de 2 segundos en crearse.

5. Plantilla para Marcado de Tareas Completadas (HU05)

Caso de Prueba: Marcar una Tarea como Completada

ID: TC_HU05_01

Descripción: Verificar que el usuario puede marcar una tarea como completada.

Precondiciones: El usuario debe tener tareas pendientes asignadas.

Pasos:

Navegar a la sección de tareas pendientes.

Seleccionar una tarea pendiente.

Marcar la tarea como completada.

Resultados Esperados: La tarea marcada como completada se mueve automáticamente a la sección de tareas completadas.

Criterios de Aceptación:

La tarea se mueve a la sección de tareas completadas instantáneamente.

El sistema confirma visualmente que la tarea ha sido completada.

Criterios de Rechazo:

La tarea no se mueve a la sección de tareas completadas.

La actualización del estado de la tarea no se refleja instantáneamente.

El sistema no confirma visualmente la tarea completada.

7. Elaborar una plantilla con los equipos de pruebas y sus responsabilidades.

Proyecto: E-COMMERCE

Fecha: 10/8/24

1. Equipo de Pruebas Funcionales

Responsabilidades:

Diseñar, ejecutar y documentar casos de prueba funcionales.

Verificar que cada función del software cumpla con los requisitos especificados.

Reportar y hacer seguimiento a defectos o errores detectados durante las pruebas.

Realizar pruebas de regresión para asegurar que nuevas actualizaciones no afecten funcionalidades existentes.

Miembros:

Líder de Pruebas Funcionales: Juan Hernandez

Analistas de Pruebas: Juan Hernandez

2. Equipo de Pruebas de Integración

Responsabilidades:

Asegurar que los diferentes módulos del sistema interactúan correctamente entre sí.

Diseñar y ejecutar casos de prueba que validen la integración entre módulos.

Identificar y resolver problemas de comunicación o incompatibilidades entre los componentes del sistema.

Verificar la correcta integración de sistemas externos o APIs.

Miembros:

Líder de Pruebas de Integración: Juan Hernandez

Ingenieros de Pruebas: Juan Hernandez

3. Equipo de Pruebas de Rendimiento
Responsabilidades:

Realizar pruebas de carga, estrés y escalabilidad para evaluar el rendimiento del sistema bajo diferentes condiciones.
Identificar cuellos de botella en el rendimiento y recomendar mejoras.
Monitorear y documentar tiempos de respuesta, uso de recursos y estabilidad del sistema bajo carga.
Validar que el sistema cumpla con los requisitos de rendimiento definidos.
Miembros:

Líder de Pruebas de Rendimiento: Juan Hernandez
Especialistas en Rendimiento: Juan Hernandez
4. Equipo de Pruebas de Seguridad
Responsabilidades:

Identificar y evaluar vulnerabilidades en el sistema.
Realizar pruebas de penetración para simular ataques potenciales.
Asegurar que se implementen las medidas de seguridad adecuadas en el sistema.
Verificar que los datos sensibles estén protegidos y que se cumplan las políticas de seguridad.
Miembros:

Líder de Pruebas de Seguridad: Juan Hernandez
Analistas de Seguridad: Juan Hernandez
5. Equipo de Pruebas de Usabilidad
Responsabilidades:

Evaluar la interfaz de usuario para asegurar que sea intuitiva y fácil de usar.
Realizar pruebas con usuarios finales para obtener feedback sobre la experiencia de usuario.
Identificar problemas de usabilidad y proponer mejoras.
Asegurar que el diseño cumpla con las pautas de accesibilidad.
Miembros:

Líder de Pruebas de Usabilidad: Juan Hernandez
Especialistas en UX/UI: Juan Hernandez

6. Equipo de Pruebas de Aceptación del Usuario (UAT)
Responsabilidades:

Coordinar y ejecutar pruebas con usuarios finales o representantes del cliente.
Validar que el sistema cumpla con los requisitos del negocio y las expectativas del usuario.
Documentar los resultados de las pruebas de aceptación y recolectar firmas de aprobación.
Identificar y reportar cualquier discrepancia entre el sistema y las necesidades del negocio.
Miembros:

Líder de Pruebas UAT: Juan Hernandez
Representantes del Cliente/Usuarios Finales: Juan Hernandez

7. Equipo de Automatización de Pruebas
Responsabilidades:

Desarrollar y mantener scripts de pruebas automatizadas.
Ejecutar pruebas automatizadas para validar la funcionalidad, regresión y rendimiento del sistema.
Asegurar que las pruebas automatizadas se integren en el proceso de CI/CD.
Monitorear y actualizar las pruebas automatizadas según sea necesario.
Miembros:

Líder de Automatización de Pruebas: Juan Hernandez
Ingenieros de Automatización: Juan Hernandez
Observaciones y Comentarios Adicionales
Comunicación: Durante el proceso de pruebas, se observó que la comunicación entre los equipos fue eficiente, lo que facilitó la identificación y resolución rápida de problemas.

Coordinación: Las reuniones semanales entre los líderes de equipo permitieron mantener un control adecuado sobre el progreso de las pruebas, lo que ayudó a ajustar las estrategias a tiempo y evitó posibles retrasos.

Documentación: Se destacó la importancia de documentar todos los hallazgos críticos. Esta práctica permitió que los problemas se compartieran de manera oportuna con los equipos relevantes, evitando impactos mayores en el cronograma del proyecto.

Capacitación: Los nuevos miembros del equipo recibieron sesiones de capacitación adicionales, lo que resultó en una integración más rápida y en una mejor comprensión de las herramientas de automatización y las metodologías de prueba aplicadas en el proyecto.

8. Elaborar un plan de automatización de pruebas.

1. Introducción

Este documento describe el plan de automatización de pruebas para el proyecto E-COMMERCE. El objetivo es mejorar la eficiencia de las pruebas, reducir los tiempos de ejecución y asegurar la calidad continua del software a lo largo de su ciclo de vida.

2. Objetivos

Reducir Tiempo de Pruebas: Disminuir el tiempo necesario para la ejecución de pruebas repetitivas y regresiones.

Mejorar Cobertura de Pruebas: Aumentar la cobertura de pruebas automatizando casos que son críticos para el funcionamiento del sistema.

Asegurar Consistencia: Asegurar que las pruebas se ejecuten de manera consistente y confiable en diferentes entornos.

Integración Continua: Facilitar la integración de pruebas automatizadas en el pipeline de CI/CD para detectar defectos tempranamente.

3. Alcance

Áreas a Automatizar:

Pruebas de Regresión: Casos de prueba que verifican funcionalidades existentes.

Pruebas de Interfaz de Usuario (UI): Flujos de usuario críticos.

Pruebas de API: Validación de endpoints y su integración.

Pruebas de Carga y Rendimiento: Simulación de múltiples usuarios concurrentes.

Herramientas de Automatización:

UI Testing: Selenium WebDriver, Cypress.

API Testing: Postman, RestAssured.

CI/CD Pipeline: Jenkins, GitLab CI.

Gestión de Pruebas: TestRail, Jira.

4. Estrategia de Automatización

Selección de Casos de Prueba: Priorizar la automatización de casos de prueba de regresión, pruebas de fumigación, y aquellas que son propensas a errores humanos.

Desarrollo de Scripts: Los scripts de prueba serán desarrollados en [Lenguaje de Programación], siguiendo las mejores prácticas de codificación y reutilización de componentes.

Ejecución de Pruebas: Las pruebas automatizadas se ejecutarán en ambientes de prueba estandarizados y estarán integradas en el pipeline de CI/CD.

Mantenimiento de Scripts: Establecer un proceso regular para actualizar y mantener los scripts de prueba para adaptarse a los cambios en la aplicación.

5. Recursos y Roles

Equipo de Automatización:

Ingeniero de Automatización de Pruebas: Desarrollar y mantener scripts de prueba automatizados.

Administrador de CI/CD: Integrar pruebas automatizadas en el pipeline de CI/CD.

Analista de Pruebas: Seleccionar casos de prueba a automatizar y revisar los resultados de las pruebas.

Herramientas y Entornos:

Infraestructura: Máquinas virtuales o contenedores dedicados para la ejecución de pruebas.

Repositorios de Código: Almacenar scripts de prueba en [GitHub/GitLab/Bitbucket].

Documentación: Guía de usuario y documentación técnica para la creación y ejecución de scripts de prueba.

6. Cronograma

Semana 1-2: Revisión de los casos de prueba y selección de los que serán automatizados.

Semana 3-6: Desarrollo y validación inicial de scripts de automatización.

Semana 7-8: Integración de pruebas automatizadas en el pipeline de CI/CD.

Semana 9 en adelante: Ejecución continua y mantenimiento de scripts de prueba automatizados.

7. Métricas de Éxito

Cobertura de Pruebas: % de casos de prueba cubiertos por automatización.

Tiempo de Ejecución: Reducción en el tiempo total de ejecución de pruebas.

Detección Temprana de Defectos: Número de defectos identificados por las pruebas automatizadas.

Frecuencia de Ejecución: Número de ejecuciones de pruebas automatizadas por ciclo de desarrollo.

8. Riesgos y Mitigaciones

Cambio Frecuente en la Aplicación: Asegurar una comunicación constante con los desarrolladores para adaptar los scripts a cambios en la UI y funcionalidades.

Falsos Positivos/Negativos: Revisión y ajuste de los scripts para mejorar la precisión de los resultados.

Limitaciones de Herramientas: Evaluación continua de herramientas para garantizar que sean las adecuadas para las necesidades del proyecto.

9. Revisión y Mantenimiento

El plan de automatización será revisado trimestralmente para ajustar las estrategias según los cambios en el proyecto y las lecciones aprendidas. Los scripts de prueba serán actualizados regularmente para mantener su eficacia.

10. Conclusión

La automatización de pruebas en [Nombre del Proyecto] es crucial para asegurar la calidad del software y acelerar los ciclos de lanzamiento. Con este plan, esperamos reducir significativamente el esfuerzo manual y aumentar la fiabilidad de nuestras pruebas.

Conclusiones

Eficiencia Mejorada: La implementación del plan de automatización de pruebas contribuirá significativamente a la reducción del tiempo de prueba. La automatización permitirá ejecutar pruebas repetitivas y de regresión de manera más rápida y consistente, mejorando la eficiencia general del proceso de aseguramiento de calidad.

Cobertura Ampliada: Al automatizar las pruebas de regresión, pruebas de interfaz de usuario críticas y pruebas de API, se logrará una cobertura de prueba más amplia. Esto garantiza que las funcionalidades clave y las integraciones del sistema sean verificadas exhaustivamente en cada ciclo de desarrollo.

Detección Temprana de Defectos: Integrar pruebas automatizadas en el pipeline de CI/CD permitirá la detección temprana de defectos, reduciendo el riesgo de que errores críticos lleguen a producción. Esto resultará en una mayor calidad del software y una experiencia de usuario final más robusta.

Consistencia y Fiabilidad: La automatización asegura que las pruebas se ejecuten de manera consistente en cada ejecución, eliminando el riesgo de errores humanos y proporcionando resultados confiables. Esto es esencial para mantener la integridad del proceso de prueba a lo largo del ciclo de vida del desarrollo.

Mantenimiento y Flexibilidad: El plan contempla la actualización y el mantenimiento continuo de los scripts de prueba para adaptarse a cambios en la aplicación. Esta flexibilidad permitirá que el sistema de pruebas automatizadas se mantenga relevante y efectivo a medida que la aplicación evoluciona.

Optimización de Recursos: La automatización de pruebas permitirá a los equipos de prueba enfocarse en áreas más estratégicas y de mayor valor, como pruebas exploratorias y de usabilidad, en lugar de en tareas repetitivas. Esto optimiza el uso de los recursos del equipo y mejora la calidad del producto final.

Reducción de Riesgos: Se han identificado y planificado mitigaciones para riesgos potenciales, como cambios frecuentes en la aplicación y limitaciones de herramientas. La gestión proactiva de estos riesgos contribuirá a un proceso de automatización más estable y efectivo.

Métricas y Éxito: Las métricas definidas, como la cobertura de pruebas y el tiempo de ejecución, serán claves para evaluar el éxito de la automatización. Estas métricas proporcionarán una visión clara del impacto de la automatización en el proceso de desarrollo y en la calidad del software.

Bibliografía

Beck, K., et al. (2001). Manifesto for Agile Software Development. Agile Alliance. Recuperado de https://agilemanifesto.org/

Documento fundamental que define los principios del desarrollo ágil, incluyendo la importancia de pruebas continuas y la colaboración entre equipos de desarrollo y calidad.
Crispin, L., & Gregory, J. (2009). Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley.

Libro que ofrece una visión completa sobre las prácticas de pruebas ágiles, con enfoque en la integración de pruebas automatizadas en el ciclo de vida del desarrollo ágil.
Fowler, M. (2006). Continuous Integration. Recuperado de https://martinfowler.com/articles/continuousIntegration.html

Artículo que explica el concepto de integración continua y su relación con la automatización de pruebas para mejorar la calidad del software.
Hemmati, H., & Memon, A. M. (2009). Automated Software Testing: Introduction and a Research Agenda. IEEE Software, 26(1), 24-30.

Artículo que proporciona una introducción a las pruebas automatizadas y discute áreas de investigación y tendencias actuales en el campo.
Jorgensen, P. C. (2013). Software Testing: A Craftsman's Approach. CRC Press.

Un texto que cubre una variedad de técnicas y métodos de pruebas, incluyendo la automatización, con un enfoque en el diseño y la ejecución de pruebas efectivas.
Kaner, C., Bach, J., & Pettichord, B. (2001). Testing Computer Software. Wiley.

Una guía clásica sobre las técnicas de prueba de software, incluyendo secciones dedicadas a la automatización y cómo implementar pruebas efectivas en diferentes tipos de proyectos.
Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.

Aunque centrado en la calidad del código, este libro también proporciona principios que ayudan a escribir código que es más fácil de probar y automatizar.
Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill.

Un libro de texto integral sobre ingeniería de software que cubre aspectos de gestión de pruebas, incluyendo estrategias para la automatización.
Sommerville, I. (2016). Software Engineering. Pearson.

Proporciona una visión general de los procesos de ingeniería de software, incluyendo la automatización de pruebas como una técnica clave para mejorar la calidad del software.
Voas, J., & McGraw, G. (1998). Software Testing and Analysis: Process, Principles, and Techniques. Wiley.

Este libro ofrece una visión profunda de los procesos de prueba y análisis de software, con técnicas aplicables a la automatización de pruebas.