República Bolivariana de Venezuela
Ministerio del Poder Popular para la Educación
Universidad Nacional de las Telecomunicaciones e Informática

Unidad Curricular: Proyecto Sociotecnológico II

Planificación de Pruebas Y Despliegue

Tutora:
Yuly Delgado

Autor:
Jean Díaz V-30.831.319

Plan de Pruebas

Estrategia de Pruebas.

La estrategia de pruebas para Worklyst busca asegurar la calidad, escalabilidad y experiencia de usuario en una plataforma colaborativa impulsada por IA. Se prioriza la cobertura funcional, la robustez ante el crecimiento y la facilidad de evolución del sistema, alineando las pruebas con los objetivos de crear, gestionar y colaborar en proyectos de manera flexible y eficiente.

Enfoque y Niveles de Prueba.

El enfoque será ágil y modular, alineado con la arquitectura desacoplada de Worklyst. Se aplicarán pruebas automatizadas y manuales en los siguientes niveles :

1. Pruebas Unitarias:

Cada componente (frontend, backend, microservicio IA) tendrá pruebas unitarias para validar funciones y métodos individuales.

2. Pruebas de Integración:

Se validará la interacción entre módulos (por ejemplo, frontend-backend, backend-IA, backend-base de datos).

3. Pruebas de Sistema:

Se probará la plataforma como un todo, asegurando que los flujos principales (crear proyectos, asignar tareas, chat, drag & drop) funcionen correctamente.

4. Pruebas de Aceptación:

Validación con usuarios clave para asegurar que la aplicación cumple los objetivos y expectativas funcionales.

Tipos de Pruebas.

Se contemplan los siguientes tipos de pruebas para cubrir tanto aspectos funcionales como no funcionales :

• <u>Funcionales:</u>

- Creación y gestión de proyectos y equipos
- Generación y personalización de tableros Kanban
- Asignación inteligente de tareas y roles
- Interacción en chat grupal y con IA
- Drag & drop de tareas

• <u>No funcionales:</u>

- Usabilidad: Claridad visual, accesibilidad y coherencia emocional

- Rendimiento: Respuesta bajo carga, especialmente en vistas Dashboard y Proyecto
- Seguridad: Autenticación, autorización y protección de datos
- Compatibilidad: Diferentes navegadores y dispositivos
- Escalabilidad: Pruebas de carga y stress para validar el crecimiento horizontal y microservicios

Entornos de Prueba.

Se definirán varios entornos para asegurar la confiabilidad y reproducibilidad de los resultados :

• Desarrollo:
Entorno local para pruebas rápidas y debugging.
- Integración:
Entorno intermedio donde se despliegan ramas de desarrollo para pruebas de integración y automatizadas.
- Calidad (QA):
Réplica del entorno de producción, con datos simulados, para pruebas de sistema, aceptación y carga.
- Producción:
Monitoreo (monitoreo de errores).

Cada entorno debe contar con configuraciones similares a producción (servidores, base de datos Supabase, microservicios IA, etc.) para garantizar la validez de las pruebas .

Responsables de la Calidad.

La responsabilidad de la calidad se divide en dos equipos principales:

• Equipo Backend:
- Pruebas unitarias y de integración de API, lógica de negocio, microservicio IA y base de datos.
- Pruebas de rendimiento y seguridad en endpoints y servicios.

• Equipo Frontend:
- Pruebas unitarias y de integración de componentes visuales y flujos de usuario.
- Pruebas de usabilidad, accesibilidad y compatibilidad.
- Validación de la experiencia conversacional y de interacción (chat, drag & drop, modal IA).

Ambos equipos colaboran en pruebas de sistema, aceptación y escalabilidad, asegurando la cobertura de extremo a extremo .

Casos de Pruebas

Casos de prueba basados en los requisitos.

1.Crear proyectos personalizados.

•Caso de Prueba 1.1: Crear un nuevo proyecto con nombre y descripción
- Precondiciones: Usuario autenticado en la plataforma
- Pasos:
  1. Acceder a la opción “Crear proyecto”
  2. Ingresar nombre y descripción
  3. Guardar el proyecto
- Datos de entrada:
  - Nombre: “Proyecto Alpha”
  - Descripción: “Desarrollo de nueva funcionalidad”
  - Resultado esperado:
  - El proyecto aparece en la lista de proyectos del usuario con los datos ingresados

2. Formar equipos y añadir miembros.

Caso de Prueba 2.1: Añadir miembros a un proyecto existente
- Precondiciones: Proyecto creado, usuario con permisos de edición
- Pasos:
  1. Acceder al proyecto
  2. Seleccionar “Añadir miembro”
  3. Ingresar correo electrónico del nuevo miembro
  4. Confirmar adición
- Datos de entrada:
  - Correo: usuario@ejemplo.com
  - Resultado esperado:
  - El nuevo miembro aparece en la lista de miembros del proyecto y recibe notificación de invitación

3. Generar tableros tipo Kanban o con otras plantillas visuales

Caso de Prueba 3.1: Crear un tablero Kanban para un proyecto
- Precondiciones: Proyecto existente
- Pasos:
  1. Acceder al proyecto
  2. Seleccionar “Crear tablero”
  3. Elegir plantilla “Kanban”
  4. Confirmar creación
- Datos de entrada:
  - Plantilla: Kanban
  - Resultado esperado:
  - El tablero Kanban se muestra con columnas predeterminadas (Por hacer, En progreso, Hecho)

4. Interactuar mediante chat grupal y chat con IA

Caso de Prueba 4.1: Enviar mensaje en el chat grupal del proyecto
- Precondiciones: Usuario es miembro del proyecto
- Pasos:
  1. Acceder al chat grupal del proyecto
  2. Escribir mensaje
  3. Enviar
- Datos de entrada:
  - Mensaje: “Hola equipo”
  - Resultado esperado:
  - El mensaje aparece en el chat grupal visible para todos los miembros

Caso de Prueba 4.2: Consultar al chat con IA sobre el estado del proyecto
- Precondiciones: Proyecto con tareas creadas
- Pasos:
  1. Acceder al chat con IA
  2. Escribir: “¿Cuántas tareas están en progreso?”
  3. Enviar
- Datos de entrada:
  - Pregunta: “¿Cuántas tareas están en progreso?”
  - Resultado esperado:
  - La IA responde con el número de tareas en progreso

5. Asignar tareas y roles de forma inteligente.

Caso de Prueba 5.1: Asignación automática de tareas por IA
- Precondiciones: Proyecto con miembros y tareas sin asignar
- Pasos:
  1. Solicitar a la IA: “Asigna las tareas automáticamente”
- Datos de entrada:
  - Comando: “Asigna las tareas automáticamente”
  - Resultado esperado:
  - Las tareas se asignan a los miembros según sus habilidades y disponibilidad

6. Utilizar drag & drop para organizar tareas.

Caso de Prueba 6.1: Mover tarea de una columna a otra usando drag & drop
- Precondiciones: Tablero Kanban con tareas
- Pasos:
  1. Seleccionar una tarea en la columna “Por hacer”
  2. Arrastrar y soltar en la columna “En progreso”
- Datos de entrada:
  - Tarea seleccionada
  - Resultado esperado:
  - La tarea aparece en la columna “En progreso” y se actualiza su estado

Plan de Despliegue y Configuración:

Pasos para pasar el sistema a los entornos de staging y producción.

1. Preparación de Infraestructura.

- Seleccionar los entornos: Definir subdominios o dominios separados para staging (ej. `staging.worklyst.com`) y producción (`worklyst.com`).
- Verificar recursos: Asegúrar contar con suficiente espacio en el hosting y recursos para ambos entornos .

2. Configuración del Entorno de Staging.

1. Clonación del Proyecto:
   - Duplicar el código fuente del frontend, backend y microservicio de IA en el entorno de staging.
   - Crear una base de datos independiente para staging, clonando la estructura y, si es necesario, datos de prueba desde producción .

2. Configuración de Variables de Entorno:
   - Ajustar las variables de entorno para apuntar a los servicios y bases de datos de staging (URLs, claves API, endpoints de IA, etc.).

3. Desplegar de Componentes:
   - Desplegar el frontend (Next.js) y backend (Node.js/Express) en servidores o servicios cloud configurados para staging.
   - Desplegar el microservicio de IA en su entorno correspondiente, asegurando que apunte a la base de datos de staging.
   - Configurar Supabase (PostgreSQL) para el entorno de staging.

4. Configuración de Accesos y Seguridad:
   - Limitar el acceso al entorno de staging (por ejemplo, mediante autenticación básica o IP allowlist).
   - Desactivar la indexación en buscadores para staging .

5. Pruebas y Validación:
   - Realizar las pruebas funcionales y de integración en staging antes de cualquier despliegue a producción .

3. Despliegue al Entorno de Producción

1. Verificación Previa:

- Asegúrar que la versión a desplegar haya pasado todas las pruebas en staging y esté aprobada por el equipo .

2. Despliegue Controlado:
- Realizar el despliegue del frontend, backend y microservicio de IA en los servidores de producción.
- Apuntar las variables de entorno a los servicios y base de datos de producción.
- Realizar migraciones de base de datos si es necesario, asegurando respaldo previo.

3. Configuración de Seguridad y Monitorización:
- Habilitar HTTPS, configurar firewalls y monitorización de logs.
- Asegurar que solo el equipo de operaciones pueda realizar despliegues o rollbacks en producción .

4. Verificación Post-Despliegue:
- Realizar pruebas de humo para validar que los servicios principales funcionan correctamente.
- Monitorear el sistema para detectar posibles incidencias.

4. Flujo de Actualización

- Staging: Todas las nuevas versiones y cambios se prueban primero en staging.
- Producción: Solo versiones estables y validadas pasan a producción, siguiendo un proceso de aprobación y checklist de validación .

Requisitos de infraestructura y entornos

1. Definición de Entornos de Prueba

- Entorno de Desarrollo:
  Espacio donde los desarrolladores compilan y prueban localmente sus cambios. Debe ser ágil y flexible, con requisitos mínimos de infraestructura para facilitar iteraciones rápidas.

- Entorno de Staging (Preproducción):
  Réplica lo más fiel posible del entorno de producción, incluyendo configuración, base de datos y servicios. Aquí se realizan pruebas funcionales, de integración y de usuario final para validar que el sistema cumple con los requisitos antes de pasar a producción .

- Entorno de Producción:
  Donde el sistema está disponible para los usuarios finales. Debe contar con alta disponibilidad, seguridad y monitorización.

2. Requisitos de Infraestructura para Pruebas

- Capacidad para replicar escenarios de producción:
  El entorno de staging debe poder simular cargas, configuraciones y datos similares a producción para detectar problemas reales .

- Aislamiento:
  Cada entorno debe estar aislado para evitar interferencias entre pruebas y garantizar resultados confiables .

- Automatización de configuración:
  Uso de herramientas de Infraestructura como Código (IaC) como Terraform, Ansible o Kubernetes para automatizar la creación y configuración de entornos, reduciendo errores y tiempos de despliegue .

- Recursos escalables:
  Infraestructura que permita ajustar recursos (CPU, memoria, almacenamiento) según las necesidades de las pruebas, optimizando costos .

- Seguridad y control de accesos:
  Implementar controles para limitar accesos a entornos de prueba, especialmente staging, para proteger datos y evitar modificaciones no autorizadas .

- Monitorización y logging:
  Sistemas para registrar eventos, errores y métricas durante las pruebas, facilitando la detección y análisis de fallos.

3. Configuración de Software y Herramientas

- Base de datos independiente:
  Cada entorno debe tener su propia base de datos para evitar contaminación de datos y permitir pruebas con datos específicos.

- Versionamiento y despliegue controlado:
  Integración con pipelines CI/CD que permitan desplegar versiones específicas en cada entorno, garantizando trazabilidad y reversión si es necesario .

- Herramientas de prueba:
  Incorporar frameworks para pruebas unitarias, de integración, de carga y de aceptación, que se ejecuten automáticamente en los entornos correspondientes .

4. Procedimientos y Gestión:

- Solicitud y documentación de requisitos:
  Establecer procedimientos claros para que los equipos definan y soliciten los requisitos de los entornos de prueba, incluyendo configuraciones, herramientas y datos necesarios .

- Validación y cualificación de infraestructura:
  Realizar comprobaciones para asegurar que la infraestructura instalada cumple con el diseño y requisitos, incluyendo pruebas de backup, restore y seguridad .

- Planificación de pruebas:
  Definir etapas y tipos de pruebas a realizar en cada entorno, asegurando que se cubren desde pruebas unitarias hasta pruebas de integración y de usuario final .