INSTITUTO POLITECNICO NACIONAL

UNIDAD PROFESIONAL

INTERDICIPLINARIA DE INGENERIA Y CIENCIAS SOCIALES

ADMJNISTRATIVAS

Materia: Ingería de pruebas

Plan de Pruebas: Pagina web

Equipo de pruebas:

Rodríguez Mendiola Valentina (Analista)

Martínez Laguna Andrik Geovanny (Programador)

Mejía Ramírez Luis Alejandro (Diseñador)

Jimenez Rogel Sergio (Tester)

Fecha de Elaboración: 04/11/2025

OBJETIVO

El propósito de este plan de pruebas es verificar que la aplicación **Tienda de Vinilos 1.0 - Fase Beta** funcione correctamente y cumpla con los requerimientos definidos tras las correcciones de la fase alfa.

Se busca confirmar que el sistema gestione adecuadamente el registro, inicio de sesión, restricción del carrito y visualización de productos, garantizando:

- Correcta autenticación y registro de usuarios (tanto frontend como backend PHP)
- Control de acceso al carrito solo con sesión iniciada
- Visualización clara y funcional de los productos
- Mensajes adecuados ante errores o acciones inválidas
- Persistencia de datos entre sesiones (LocalStorage + Backend)
- Fluidez general, estabilidad y buena experiencia de uso
- Envío correcto de tickets por correo mediante API Brevo.
- Renderizado de imágenes en carrito y recomendaciones.
- Fluidez general, estabilidad y buena experiencia de usuario validada con usuarios externos.

ALCANCE

El alcance de las pruebas incluirá las siguientes áreas funcionales:

1. Registro de usuario y autenticación (inicio de sesión)

- Verificar que el sistema permita crear una cuenta y acceder con credenciales válidas
- **Backend:** Validar almacenamiento en users. json con contraseñas hasheadas
- **Frontend:** Confirmar sincronización con LocalStorage
- **Integración:** Probar comunicación entre script.js y archivos PHP (register.php, login.php, current_user.php, logout.php)
- Confirmar que los campos obligatorios (nombre, correo, contraseña) funcionen correctamente
- Validar formato de email tanto en frontend como backend
- Prevenir registros duplicados con el mismo correo electrónico
- Validar que solo los usuarios con sesión iniciada puedan agregar productos
- Verificar que cada usuario tenga su carrito individual (clave: mv_cart_${email})

- Probar aislamiento de carritos entre diferentes usuarios
- Comprobar que los productos se muestren correctamente con su nombre, autor, precio y descripción
- Evaluar que cada botón cumpla su función prevista: Agregar, Crear cuenta, Iniciar sesión, Cerrar sesión, Carrito, etc.
- Mensajes del sistema (éxito, error y advertencias).
- Verificar que se muestren correctamente según la acción del usuario.
- Sección de contacto y pie de página.
- Validar que los datos de contacto y la información legal se desplieguen correctamente.
- Manejo de errores y estabilidad.

  Confirmar que el sistema responda adecuadamente ante errores o problemas de conexión.
- Usabilidad general.

  Evaluar la facilidad de uso, claridad de mensajes y comprensión del flujo por parte del usuario final.
- Verificar almacenamiento en LocalStorage (clave: mv_history_${email})
- Validar registro correcto de fecha, ítems, total y recomendación
- Confirmar que la sesión PHP se mantenga tras recargar página
- Verificar envío de tickets de compra mediante API Brevo (sendMail.php).
- Verificar que las imágenes se carguen desde la ruta especificada en products.json.

CRITERIOS DE ACEPTACIÓN

1. Registro y autenticación

- El sistema debe permitir crear una cuenta con nombre, apellido, correo y contraseña válidos
- **Backend debe guardar usuario en users. json con contraseña hasheada**
- No debe permitir registrar más de una cuenta con el mismo correo electrónico
- Los mensajes de confirmación o error deben mostrarse correctamente después del registro
- El inicio de sesión debe requerir credenciales válidas y mostrar el nombre del usuario
- Al cerrar sesión, el carrito debe vaciarse y el botón debe mostrar "Iniciar sesión"

- Los campos nombre y apellido deben aceptar solo letras mayúsculas sin espacios ni símbolos especiales.

2. Validación de formularios

- Todos los campos obligatorios deben validarse antes de enviar el formulario
- El formato de email debe validarse en frontend y backend
- Los mensajes de error deben ser claros y específicos (no genéricos)
- La contraseña debe tener al menos 6 caracteres
- No se debe permitir continuar si los datos no cumplen los requisitos mínimos

3. Carrito de compras

- Solo los usuarios con sesión iniciada pueden agregar productos al carrito
- Si el usuario no ha iniciado sesión, debe mostrarse un mensaje informando que debe autenticarse.
- Los productos agregados deben reflejarse correctamente en el carrito (nombre, precio, cantidad, imagen)
- **Cada usuario debe tener carrito independiente (clave: mv_cart_${email})**
- El total debe calcularse con precisión decimal (sin errores de punto flotante)
- Las imágenes de productos deben mostrarse correctamente en el carrito

4. Visualización de productos

- Los productos deben mostrarse con su título, autor, precio y descripción completa
- Los botones "Añadir" deben estar visibles y funcionales
- Recomendar productos
- Las imágenes deben cargarse desde la ruta en products.json

5. Sistema de recomendaciones

- El sistema debe recomendar productos basándose en el género más comprado
- El modal de recomendaciones debe mostrar imagen, título y razón de la recomendación
- El botón "Ver producto recomendado" debe hacer scroll al producto en la página

6. Envío de correos

- El ticket de compra debe enviarse al correo del usuario mediante sendMail.php y API Brevo
- El correo debe incluir: nombre, apellido, lista de items, total y recomendación
- Debe mostrarse mensaje de éxito cuando el correo se envía correctamente
- Debe mostrarse mensaje de error específico cuando falla el envío

6 . Mensajes del sistema

- Los mensajes deben indicar claramente si una acción fue exitosa, fallida o requiere corrección
- No deben mostrarse mensajes contradictorios

7 . Contacto y pie de página

- La sección de contacto debe mostrar correctamente el teléfono, correo y dirección
- El pie de página debe incluir los datos de contacto
- **Los caracteres especiales deben mostrarse correctamente (UTF-8)**

8. Usabilidad

- La interfaz debe ser comprensible para el usuario promedio sin necesidad de ayuda adicional
- Los flujos de navegación (inicio, registro, carrito, contacto) deben ser intuitivos

9. Manejo de errores

- El sistema debe responder adecuadamente a errores de conexión PHP
- No deben generarse cuentas duplicadas ni dejar el sistema en estado inconsistente

RECURSOS

1. Personal de prueba

Las pruebas serán realizadas por el equipo de desarrollo y validación, encargado de verificar el correcto funcionamiento del sistema antes de su entrega final.

Responsabilidades:

- Ejecutar los casos de prueba
- Registrar incidencias o defectos detectados
- Validar correcciones y confirmar el cumplimiento de los criterios de aceptación
- Coordinar pruebas con usuarios externos (Prueba Beta)

2. Entorno de prueba

Las pruebas se ejecutarán en el siguiente entorno:

- **Sistema operativo:** Windows 10/11 o Linux
- **Navegadores:** Google Chrome (principal), Microsoft Edge (secundario)
    - Pendiente: Safari, Firefox
- **Servidor web:** Apache o Nginx con soporte PHP 7.4+
- **Base de datos:** Archivo JSON (users. json) en directorio /api/ (por el momento)
- **Almacenamiento:**
    - Frontend: LocalStorage del navegador
    - Backend: users.json (persistente en servidor)
- **Herramientas de prueba:**
    - Editor de código (VS Code) para revisar logs
    - Selenium
- **Tipo de entorno:** Local (desarrollo) y entorno simulado de producción

TIPOS DE PRUEBAS

→ **Pruebas de Unidad**

Verificar individualmente cada componente funcional del sistema:

Frontend:

- Botones: "Iniciar sesión", "Crear cuenta", "Agregar al carrito", "Cerrar sesión"
- Formularios de registro y login
- Cálculo de totales con precisión decimal
- Renderizado de imágenes en productos, carrito y recomendaciones
- Sistema de recomendación por género
- Envío de correos mediante sendMail.php y API Brevo
- Validación de duplicados en register.php

→ **Pruebas de Integración**

Validar la interacción entre los diferentes módulos:

- Aislamiento de carritos entre usuarios diferentes
- Relación entre inicio de sesión → carrito → productos → checkout → correo
- Persistencia de sesión tras recargar página
- Integración entre frontend y API Brevo para envío de tickets

→ **Pruebas de Sistema**

Evaluar la tienda como un todo para garantizar el correcto funcionamiento bajo distintos escenarios:

- Navegación completa por toda la aplicación
- Conexión, navegación y compras simuladas
- Validar usabilidad y experiencia de usuario
- Verificar estabilidad y manejo de errores
- Compatibilidad con navegadores Chrome 119+ y Edge 119+

→ **Pruebas de Caja Blanca (Cobertura de Código)**

Validarán que todas las rutas lógicas y condiciones se ejecuten correctamente.

Se analizarán los 5 archivos PHP principales (register.php, login.php, current_user.php, logout.php, sendMail.php), identificando todas las rutas lógicas y validando su comportamiento ante entradas válidas, inválidas.

→ **Pruebas de Aceptación del Usuario (UAT)**

Validar la usabilidad del sistema con usuarios finales:

- Facilidad para registrarse e iniciar sesión
- Comprensión de la navegación y funcionalidades
- Retroalimentación sobre aspectos visuales y de usabilidad
- Identificación de problemas no evidentes durante pruebas internas

→ **Pruebas de Seguridad:**

Confirmar que la aplicación maneje adecuadamente entradas inválidas y errores del servidor (por ejemplo, fallos de conexión o duplicidad de registro), sin mostrar datos sensibles ni cerrar inesperadamente.

→ **Pruebas de Regresión**

Validar que las 7 correcciones implementadas desde la fase alfa no hayan introducido nuevos defectos ni afectado funcionalidades previamente operativas.

Correcciones de Alpha a validar:

- Separación de campos nombre/apellido en registro
- Carga de productos desde products.json con imágenes
- Implementación de envío de correos mediante API Brevo
- Validaciones en tiempo real para campos de texto
- Persistencia de sesión tras recarga de página
- Aislamiento de carritos entre usuarios diferentes
- Prevención de registros duplicados

Alcance: Se re-ejecutarán casos de prueba unitarios, de integración y de sistema que validaron funcionalidades modificadas, verificando que las correcciones no rompieron flujos existentes ni generaron inconsistencias.

Casos de Prueba

Registro e inicio de sesión:

1. Registro duplicado: intentar registrar el mismo correo con distinto nombre → debe mostrar mensaje de error.
2. Registro sin apellido → sistema debe mostrar mensaje de error indicando que el apellido es obligatorio.
3. Error de conexión: simular falla de red al registrar → se muestra mensaje adecuado indicando la situación.
4. Inicio de sesión correcto: ingresar credenciales válidas → acceso exitoso.
5. Inicio de sesión incorrecto: ingresar correo válido con contraseña errónea → mensaje de error claro.
6. Registro con nombre o apellido con símbolos/espacios → sistema debe rechazar entrada y mostrar formato correcto.
7. Registro con contraseña menor a 6 caracteres → sistema debe mostrar mensaje de validación.

Carrito y productos:

1. Agregar sin sesión: intentar añadir producto sin iniciar sesión → mostrar advertencia.
2. Flujo correcto: registro → inicio de sesión → agregar producto → ver carrito → confirmar visualización correcta.
3. Productos: verificar que los productos se muestren con nombre, autor, precio y descripción completos.
4. Carrito vacío: verificar que, al no tener productos, se muestre mensaje informativo (“Tu carrito está vacío”).
5. Imágenes en carrito: verificar que las imágenes de productos agregados se muestren correctamente.

Sistema de correo:

1. Checkout con carrito lleno → verificar que se muestre confirmación de envío de ticket.

2. Verificar que el correo llegue con toda la información (nombre, apellido, items, total, recomendación).
3. Simular fallo en API Brevo → verificar mensaje de error adecuado.

Recomendaciones:

1. Completar compra → verificar que modal de recomendaciones muestre imagen y datos correctos.
2. Click en "Ver producto recomendado" → verificar scroll y highlight del producto.

Mensajes y visualización:

1. Mensajes del sistema: validar que los mensajes de éxito y error sean claros y coherentes.
2. Pie de página: comprobar que la información de contacto se visualice correctamente.
3. Usabilidad: confirmar que el usuario pueda navegar fácilmente entre secciones sin confusión.

Matriz de Casos de Pruebas Unitarias

ID	Nombre del Caso	Datos de Prueba	Acción / Procedimiento	Resultado Esperado	Tipo
CP-U001	Registro de usuario	(nombre, email, password)	Enviar datos a *register.php*	Usuario creado en *users.json* con hash	**Funcional**
CP-U002	Validación email duplicado	email existente	Verificar en *users.json*	Mensaje: "Ya existe cuenta con ese correo"	Validación
CP-U003	Validación campos vacíos registro	("", email, password)	Validar formulario	Mensaje: "Completa todos los campos"	Límite
CP-U004	Validación formato email	email inválido	Validar con *FILTER_VALIDATE_EMAIL*	Mensaje: "Email inválido"	Validación
CP-U005	Login con credenciales válidas	(email, password correctos)	Verificar con *password_verify()*	Sesión iniciada, *currentUser* definido	**Funcional**
CP-U006	Login con credenciales inválidas	(email, password incorrectos)	Verificar con *password_verify()*	Mensaje: "Credenciales incorrectas"	Error
CP-U007	Cálculo de total carrito	items con qty y price	*cartTotal()*	Total sin errores de punto flotante	**Funcional**
CP-U008	Recomendación por género	items con géneros	*generateRecommendation()*	Producto del género más comprado	**Funcional**

ID	Nombre del Caso	Datos de Prueba	Acción / Procedimiento	Resultado Esperado	Tipo
CP-U009	Restricción de carrito sin sesión	Click en carrito	Verificar en *users.json*	Mensaje: “Necesario iniciar sesion”	**Funcional**
CP-U010	Visualización de productos	Al entrar a la pagina	Verificar visualmente los productos	Deben verse los productos	**Funcional**
CP-U011	Visualización de imágenes en carrito	Agregar producto y abrir carrito	Renderizar imagen desde products.json	Imagen visible	**Funcional**
CP-U012	Visualización de imágenes en recomendaciones	Confirmar compra	Pagar y enviar ticket, abrir modal recomendaciones	Imagen visible	**Funcional**
CP-U013	Envío de ticket por correo	Confirmar compra	Finalizar compra	Mensaje de éxito; si falla, mostrar código y mensaje de error	**Funcional**

Matriz de Casos de Prueba de Integración

ID	Nombre del Caso	Precondición	Procedimiento / Pasos	Resultado Esperado	Prioridad	Tipo
CP-I001	Flujo completo Registro → Login → Compra	Sistema iniciado	1. Registrar usuario "usuario1@gmail.com" 2. Login 3. Agregar "vinyl-01" 4. Verificar carrito 5. Checkout	Flujo completo sin errores, ticket generado	CRÍTICA	Funcional
CP-I002	Llamadas a register.php	Sistema iniciado	1. Completar registro 2. Click "Crear cuenta" 3. Verificar *Network*	Petición POST válida con respuesta JSON	ALTA	Integración
CP-I003	Validación respuestas JSON backend	Backend activo	1. Realizar registro 2. Capturar response en console	JSON bien formado {ok, user}	ALTA	Integración
CP-I004	Sincronización PHP-JavaScript	Sesión PHP activa	1. Login en PHP 2. Verificar $_SESSION['user'] y localStorage	Ambos estados sincronizados	CRÍTICA	Integración
CP-I005	Aislamiento de carritos entre usuarios	Dos usuarios	1. Usuario1 agrega producto 2. Logout 3. Usuario2 login 4. Verificar vacío	Carritos independientes	ALTA	Integración
CP-I006	Persistencia de sesión	Usuario logueado	Recargar página (F5)	Sesión mantiene usuario activo	CRÍTICA	Funcional / Integración
CP-I007	Envío de ticket	Usuario logueado con carrito no vacío	1. Checkout 2. Enviar ticket 3. Validar respuesta y logs	Ticket enviado correctamente o error	CRÍTICA	Funcional / Integración

🧠 Matriz de Casos de Prueba de Sistema

ID	Nombre del Caso	Precondición	Procedimiento / Pasos	Resultado Esperado	Prioridad	Tipo
CP-S001	Estabilidad general sin cierres	Sistema iniciado	Navegar, abrir modales, operar 30 min	Sin *crashes*	CRÍTICA	Estabilidad
CP-S002	Tiempo de respuesta operaciones locales	Sistema funcionando	Agregar 10 productos, medir UI	Todas las operaciones < 500 ms	ALTA	Rendimiento
CP-S003	Uso prolongado sin degradación	Sistema iniciado	Operar 30 min, monitorear memoria	Sin degradación perceptible	ALTA	Rendimiento
CP-S004	Compatibilidad Chrome 119+	Chrome actualizado	Ejecutar todas las funciones	Sin errores críticos en consola	ALTA	Compatibilidad
CP-S005	Manejo de errores de red (simulado)	DevTools abierto	Activar *Offline*, registrar	Mensaje: “Error de conexión al registrar”	CRÍTICA	Error Handling
CP-S006	Validación entradas inválidas	Sistema iniciado	Ingresar email sin @, campos vacíos, SQL injection	Sistema rechaza sin fallar	CRÍTICA	Seguridad
CP-S007	Usabilidad usuario nuevo	Usuario nuevo	Probar sin ayuda	Completa tareas básicas	MEDIA	Usabilidad

🧩 MATRIZ DE PRUEBAS DE REGRESIÓN

ID	Nombre del Caso	Funcionalidad Corregida	Procedimiento	Resultado Esperado	Prioridad	Resultado
CP-R001	Validación separación nombre/apellido	Separación de campos en registro	Registrar usuario con nombre y apellido, verificar users.json	Campos **name** y **lastname** separados correctamente	**CRÍTICA**	⏳ Pendiente
CP-R002	Carga de productos desde JSON	Implementación de products.json	Abrir aplicación, verificar 4 productos con imágenes	Todos los productos visibles con imágenes cargadas	**CRÍTICA**	⏳ Pendiente
CP-R003	Integración API Brevo	Envío de correos	Completar checkout, verificar envío de ticket	Correo enviado o mensaje de error claro	**CRÍTICA**	⏳ Pendiente
CP-R004	Validaciones tiempo real	Validación de caracteres	Ingresar símbolos/números en nombre, verificar rechazo	Sistema rechaza caracteres no permitidos	**CRÍTICA**	⏳ Pendiente
CP-R005	Persistencia de sesión	Sesión tras recarga	Login, presionar F5, verificar usuario activo	Usuario mantiene sesión y carrito	**ALTA**	⏳ Pendiente
CP-R006	Aislamiento de carritos	Carritos independientes	Usuario1 agrega producto, logout, Usuario2 login	Carritos separados por usuario	**ALTA**	⏳ Pendiente
CP-R007	Prevención duplicados	Validación email único	Registrar email existente	Mensaje: "Ya existe cuenta con ese correo"	**CRÍTICA**	⏳ Pendiente

D MATRIZ DE COBERTURA DE PRUEBAS CAJA BLANCA

Archivo	Rutas Totales	Rutas Cubiertas	Cobertura %	Estado
register.php	8	7	**87.5%**	🟢 Buena
login.php	10	7	**70%**	🟡 Mejorar
current_user.php	2	2	**100%**	🟢 Excelente
logout.php	1	1	**100%**	🟢 Excelente
sendMail.php	6	3	**50%**	🔴 Crítica
PROMEDIO TOTAL	27	20	**74%**	🟡 Mejorar

Nota: Cobertura actual **74%** – Objetivo ≥ **90%**

⚠️ **(Las pruebas de caja negra están implícitas en otras matrices)**

MATRIZ DE CASOS DE PRUEBA DE CAJA BLANCA

ID	Nombre del Caso	Archivo Objetivo	Datos de Prueba	Resultado Esperado	Prioridad
CP-CB001	JSON malformado en registro	register.php	Body: {email:"test" (incompleto)	HTTP 400 + "Solicitud inválida"	ALTA
CP-CB002	JSON malformado en login	login.php	Body: {email:"test" (incompleto)	HTTP 400 + "Solicitud inválida"	ALTA
CP-CB003	Login con campos vacíos	login.php	{email: "", password: ""}	HTTP 400 + "Completa correo y contraseña"	MEDIA
CP-CB004	Login sin archivo users.json	login.php	Renombrar users.json temporalmente	HTTP 401 + "Credenciales incorrectas"	BAJA
CP-CB005	sendMail con API Key inválida	sendMail.php	API Key = "INVALID_KEY_TEST"	ok: false + mensaje error Brevo	CRÍTICA

ID	Nombre del Caso	Archivo Objetivo	Datos de Prueba	Resultado Esperado	Prioridad
CP-CB006	sendMail con campos vacíos	sendMail.php	{email: "", name: "", items: []}	HTTP 400 + "Faltan datos"	MEDIA

🚀 PLAN DE ACCIÓN PARA MEJORAR COBERTURA

Prioridad	Archivo	Cobertura Actual	Cobertura Objetivo	Casos Faltantes	Impacto
🔴 **Crítica**	sendMail.php	50%	90%	CP-CB005, CP-CB006	Funcionalidad de correos crítica
🟡 **Alta**	login.php	70%	95%	CP-CB002, CP-CB003, CP-CB004	Autenticación es alta
🟢 **Media**	register.php	87.5%	100%	CP-CB001	Ya tiene buena cobertura

Prueba Beta (Usuarios)

Objetivo:

Evaluar la experiencia de usuario final mediante pruebas con personas externas al equipo de desarrollo, para identificar áreas de mejora, validar usabilidad y detectar problemas no evidentes durante las pruebas internas.

→ Métricas Cuantitativas

Para evaluar objetivamente la experiencia de usuario, se implementarán métricas cuantitativas estándar de la industria que permitan medir usabilidad, satisfacción y eficiencia del sistema de forma medible y comparable.

ID	Tarea	Objetivo	Método de Medición
T1	Registro completo	≥ 85%	Observar si el usuario crea cuenta sin ayuda
T2	Login tras registro	≥ 90%	Verificar si accede correctamente después de registrarse
T3	Agregar producto al carrito	≥ 95%	Confirmar si agrega al menos 1 producto exitosamente
T4	Completar checkout	≥ 80%	Validar si finaliza compra y recibe confirmación

Procedimiento:

1. Enviar el enlace de la aplicación a usuarios de distintos perfiles.

2. Solicitar retroalimentación sobre:

- Usabilidad y navegación.
- Claridad de mensajes y botones.
- Visualización de productos, imágenes y recomendaciones.
- Funcionamiento del carrito y registro.

3. Registrar los correos y comentarios de los participantes.
4. Analizar la retroalimentación y documentar hallazgos y acciones a realizar.

Participantes:

1. **Amanda Rodriguez**

Correo: amanda.rod.850@gmail.com

2. **Emilio Badillo**

Correo: oemiliobadillo@gmail.com

3. Enrique Antonio Arellanes León

Correo: enrique2709arellanes@gmail.com

4. Samantha Estrada López

Correo: estradalopezandreasamantha@gmail.com

5. José Martinez

Correo: jm1608473@gmail.com

6. Luis Ignacio Gallardo Díaz

Correo: ls.gallardodiaz@gmail.com

7. Angel Maldonado Cruz

Correo: ipn.angelmc9@gmail.com

8. Carlos David Rodríguez Flores

Correo: carloshe@gmail.com

9. Ana Danae Sierra García

Correo: danae.siergar29@gmail.com

10. Yolanda López

Correo: gm251103@gmail.com

11. Ramon cruz

Correo:(rcruzma@ipn.mx)

Comentarios principales:

- App clara, sencilla y fácil de navegar. ✓
- Botones y mensajes visibles y comprensibles. ✓
- Algunas imágenes tardan en cargar. ⚠
- Correos de confirmación no recibidos. ⚠
- Sugerencias: filtros de búsqueda como barra de busqueda y más descripciones de productos Al igual que agregar mas producto. 💡
  -

Acciones propuestas:

1. Optimizar carga de imágenes.
2. Revisar envío de correos de confirmación.
3. Evaluar agregar filtros y más información en productos.
4. Implementar mensajes de error para imágenes que no cargan.

Cronograma

Etapa	Semana
Pruebas Unitarias	Semana 1 (combinadas con Integración inicial)
Pruebas de Integración	Semana 2
Pruebas de Sistema	Semana 3
Prueba Beta (retroalimentación externa)	Semana 3-4 (antes de UAT y pruebas de regresión)
Pruebas de Aceptación (UAT)	Semana 4 (en paralelo pruebas selenium)
Pruebas de Usabilidad y Seguridad	Semana 4 (en paralelo con UAT y Usabilidad)

Informes

Se generarán informes de estado, destacando:

- Casos de prueba ejecutados.
- Defectos encontrados.
- Avance del proceso de validación.