****

Campus Santa Fe

Implementación de redes de área amplia y servicios distribuidos

Desarrollo de aplicaciones avanzadas de ciencias computacionales

TC3003B.501 y TC3002B.502

Documentación de UX/UI y testing

**Assessment**

Benjamín Alejandro Cruz Cervantes - A01747811

Alberto Iván Tamez González - A01026999

Bernardo Alejandro Limón Montes de Oca - A01736575

Fecha de entrega:

Lunes 09 de Junio de 2025

**ÍNDICE:**

[**1. Interfaz de Usuario (UI) y Experiencia de Usuario (UX) 3**](#)

[1.1 Estructura General **3**](#)

[1.2 Componentes Visuales Clave **3**](#)

[1.3 Flujo de Usuario **3**](#)

[1.4 Diagrama de flujo de navegación (imagen sugerida) **4**](#)

[**2. Testing (Pruebas de usabilidad y funcionalidad) 4**](#)

[2.1 Pruebas de UI/UX **4**](#)

[2.2 Pruebas Internas **5**](#)

[2.3 Pruebas Automatizadas **5**](#)

[Backend (Python) **5**](#_w7nsyhlrzjer)

[Frontend (React) **6**](#_teeduntrwmje)

[**3. Conclusión 6**](#)

**Documento 4: Interfaz de Usuario, Testing y Manual de Usuario**

# 1. Interfaz de Usuario (UI) y Experiencia de Usuario (UX)

## 1.1 Estructura General

La aplicación cuenta con una interfaz construida en *NextJS* con *Tailwind CSS* y *Radix UI*. La navegación está organizada por pestañas:

* **Dashboard**: Página principal con el formulario de carga de archivos y visualización de recomendaciones.
* **Perfil**: Vista donde el usuario puede ver su información personal y métricas de uso.
* **Admin**: Panel exclusivo para administradores con estadísticas de uso y seguridad.

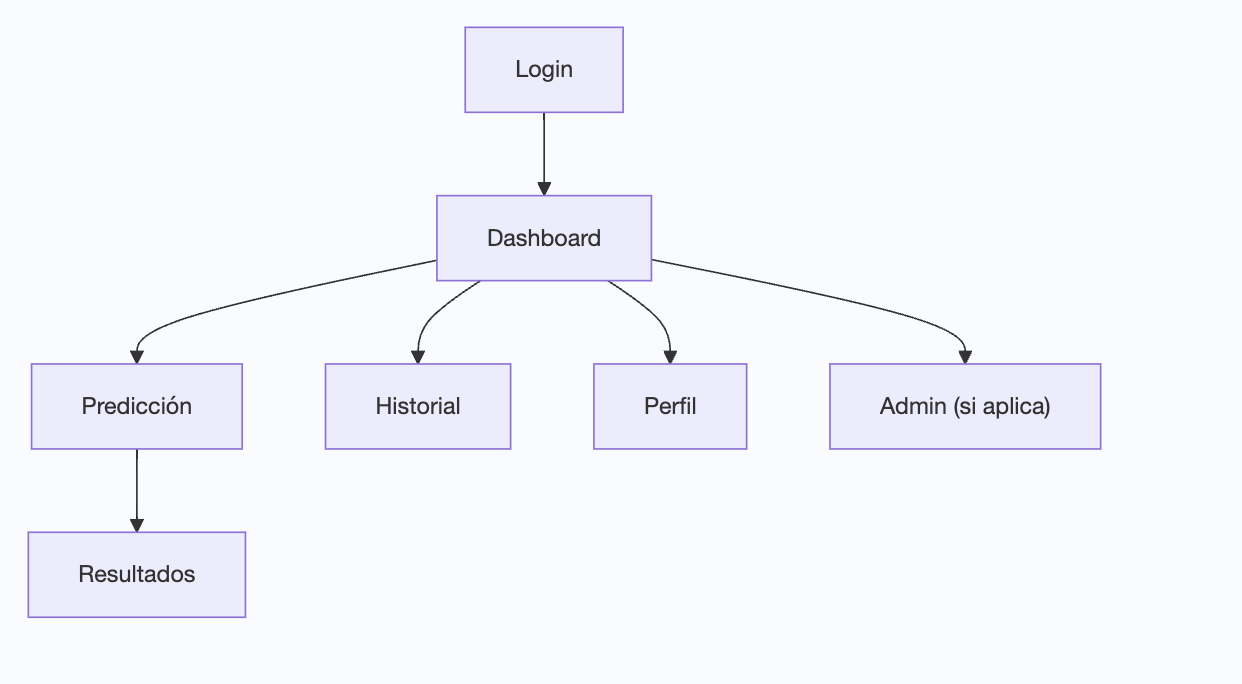
## 1.2 Componentes Visuales Clave

* **Barra de navegación fija**: Con enlaces a cada sección.
* **Formulario de carga**: Zona destacada con “drag and drop” y botón para cargar archivos *CSV*.
* **Tarjetas de resultados**: Cada predicción se muestra en una tarjeta con sus características de entrada y cultivo recomendado.
* **Controles de paginación**: Permite navegar entre lotes de resultados sin sobrecargar el navegador.

## 1.3 Flujo de Usuario

1. El usuario se autentica vía *WorkOS*.
2. Es redirigido al Dashboard.
3. Carga un archivo *CSV* (o usa el de ejemplo).
4. Presiona “Obtener recomendaciones”.
5. Visualiza las tarjetas de resultado.
6. Puede acceder al historial desde el menú.
7. Si es admin, accede al dashboard con gráficas.

## 1.4 Diagrama de flujo de navegación (imagen sugerida)



# 2. Testing (Pruebas de usabilidad y funcionalidad)

## 2.1 Pruebas de UI/UX

* **Usuarios reales (estudiantes)**: Validaron el flujo completo de autenticación, carga y visualización.
* **Metodología**: Se realizaron sesiones de observación directa y entrevistas breves después del uso.
* **Hallazgos**:
  + El drag-and-drop fue intuitivo y evitó errores en la carga.
  + La paginación evitó cuelgues con archivos de más de 2000 filas, gracias a la carga parcial.
  + Se agradeció el CSV de ejemplo, especialmente por parte de usuarios nuevos.
  + Algunos usuarios solicitaron una vista previa del archivo antes de enviarlo.
  + La interfaz fue evaluada como clara, aunque se propuso que los mensajes de error fueran más visibles.

## 2.2 Pruebas Internas

* **Casos extremos considerados**:
  + Carga de archivos CSV vacíos, corruptos o con delimitadores inválidos.
  + Cambios bruscos de tamaño de ventana: se verificó que la interfaz se mantuviera responsive.
  + Intentos de acceso sin login redirigidos correctamente a WorkOS.
  + Simulación de múltiples usuarios accediendo al mismo tiempo.
  + Evaluación de tiempos de respuesta ante archivos con más de 2000 registros.
* **Resultados**:
  + El sistema manejó correctamente errores de archivo y sesiones no válidas.
  + No se detectaron vulnerabilidades en los formularios o campos manipulables desde el cliente.
  + Las métricas en base de datos (logs) permitieron comprobar que los intentos inválidos quedaban registrados.

## 2.3 Pruebas Automatizadas

#### **2.3.1 Backend (Python)**

* **Tecnología:** PyTest.
* **Cobertura:**
  + **Configuración y Fixtures:** Se emplea un archivo conftest.py que centraliza la configuración. Este utiliza **inyección de dependencias** para proveer a cada prueba una sesión de base de datos aislada y un usuario autenticado simulado (mock), permitiendo un control total sobre el contexto de la prueba.
  + **Pruebas de API y Autorización:** Las pruebas se centran en los endpoints de la API. Se verifica tanto el comportamiento esperado como las reglas de autorización. Por ejemplo, se comprueba que un usuario no-administrador solo pueda leer sus propios logs, mientras que un administrador pueda acceder a todos los registros.
  + **Validación de Base de Datos:** Las pruebas no solo validan la respuesta de la API, sino que también consultan la base de datos para asegurar que los datos se escribieron correctamente, garantizando la integridad de las operaciones.
  + **Pruebas de Integración con Microservicios:** Se incluyen scripts dedicados para verificar la conectividad y funcionalidad del microservicio de IA externo. Estas pruebas confirman que el servicio está en línea y puede procesar peticiones, incluyendo la carga de archivos CSV.

#### **2.3.2 Frontend (React)**

* **Tecnología:** Vitest (o Jest) y React Testing Library.
* **Cobertura y Estrategia:**
  + **Pruebas de Componentes en Aislamiento:** Se enfoca en verificar que los componentes de la interfaz de usuario se rendericen y comporten como se espera, utilizando datos de prueba simulados.
  + **Componentes Clave Probados:**
    - CropRecommendationCard: Se prueba que la tarjeta muestre correctamente todos los datos de una predicción, incluyendo valores de suelo/clima y el cultivo recomendado con su nivel de confianza.
    - CardPagination: Se verifica que el componente de paginación se renderice correctamente, muestre el número total de páginas y responda a los eventos de cambio de página.
  + **Pruebas de Lógica Pura:** Se realizan pruebas unitarias sobre las funciones de utilidad, como paginateData para la paginación y getCropLabelFrom\* para el mapeo de datos, garantizando que la manipulación de la información sea predecible y correcta.
  + **Simulación de Datos:** Se emplea una función (generateTestData) para generar datos de prueba realistas, simulando la respuesta que se obtendría del backend. Esto permite probar el frontend de manera independiente.

# 3. Conclusión

La aplicación fue diseñada con un enfoque de simplicidad, accesibilidad y robustez técnica. Durante su desarrollo se priorizó una experiencia de usuario clara y coherente, lo cual fue validado por medio de pruebas reales con estudiantes y revisiones internas bajo distintos escenarios de carga y comportamiento.

La combinación entre una interfaz limpia, una arquitectura modular y mecanismos de validación robustos en frontend y backend permitió construir una plataforma confiable para el propósito de predicción agrícola. El sistema no solo resiste entradas inválidas, sino que guía al usuario de forma intuitiva durante todo el flujo, desde el inicio de sesión hasta la obtención de resultados.

Asimismo, el manual de usuario fue elaborado para asegurar que cualquier persona con acceso institucional pueda comprender y utilizar todas las funcionalidades sin necesidad de asistencia técnica directa. Este documento también refleja un nivel de documentación y pruebas que se espera en entornos profesionales.

Se concluye que la solución cumple con estándares académicos y prácticos, siendo una herramienta completa tanto desde el punto de vista funcional como en términos de mantenibilidad y escalabilidad.