**|**

**Desarrollo de Test Plan Ágil para el Sistema Integrador Urbano de Monitoreo Ambiental**

Jhair Anderson Aguilar Palacios

Pedro Renato Escobar Ortuño

Matias Ricardo Mendoza Peducasse

Omar Velasco Teran

Ingeniería de Sistemas Informáticos, Universidad Privada del Valle

Grupo A, Software Quality Assurance

Ingeniero Rolando Lara Sanchez

Octubre 20, 2025

**Contenido**

[1 Introducción 5](#_Toc211804871)

[2 Integración de la calidad 6](#_Toc211804872)

[2.1 Contexto de proyecto 6](#_Toc211804873)

[2.2 Objetivo general 7](#_Toc211804874)

[2.3 Objetivos específicos 7](#_Toc211804875)

[2.3.1 Roles de proyecto 7](#_Toc211804876)

[2.4 Requerimientos funcionales 8](#_Toc211804877)

[2.5 Requerimientos no funcionales 10](#_Toc211804878)

[2.6 Historias de usuarios 11](#_Toc211804879)

[2.7 Marco ágil 14](#_Toc211804880)

[3 Ciclo de vida del software 16](#_Toc211804881)

[3.1 Ciclo de vida del desarrollo del software (SDLC) 16](#_Toc211804882)

[3.1.1 Entrega 1 – Desarrollo de idea de proyecto 16](#_Toc211804883)

[3.1.2 Entrega 2 – Ampliación y predicción ambiental 18](#_Toc211804884)

[3.2 Ciclo de vida de las pruebas del software (STLC) 20](#_Toc211804885)

[3.2.1 Entrega 1 - Ciclo de pruebas del MVP 20](#_Toc211804886)

[3.2.2 Entrega 2 - Ciclo de pruebas de predicción de datos 21](#_Toc211804887)

[4 Técnicas y estrategia de pruebas 23](#_Toc211804888)

[4.1 Tipos de pruebas funcionales 23](#_Toc211804889)

[4.1.1 Pruebas unitarias 23](#_Toc211804890)

[4.1.2 Pruebas de integración 24](#_Toc211804891)

[4.1.3 Pruebas de sistema 24](#_Toc211804892)

[4.1.4 Pruebas de aceptación 25](#_Toc211804893)

[4.1.5 Pruebas de interfaz de usuario 25](#_Toc211804894)

[4.2 Tipos de pruebas no funcionales 26](#_Toc211804895)

[4.2.1 Pruebas de rendimiento 26](#_Toc211804896)

[4.2.2 Pruebas de estrés 27](#_Toc211804897)

[4.2.3 Pruebas de seguridad 28](#_Toc211804898)

[4.2.4 Pruebas de configuración 29](#_Toc211804899)

[4.2.5 Pruebas de calidad de la información 31](#_Toc211804900)

[4.3 Pruebas de validación 32](#_Toc211804901)

[4.4 Técnicas de diseño de Test cases 36](#_Toc211804902)

[4.4.1 Partición de equivalencias 36](#_Toc211804903)

[4.4.2 Análisis de valores limite 38](#_Toc211804904)

[4.4.3 Tablas de decisión 40](#_Toc211804905)

[5 Actividades de control y seguimiento 42](#_Toc211804906)

[5.1 Inspección 42](#_Toc211804907)

[5.2 Revisión 43](#_Toc211804908)

[5.3 Auditoria 44](#_Toc211804909)

[5.3.1 Lista de verificación de auditorias 46](#_Toc211804910)

[5.3.2 Riesgos y mitigación 48](#_Toc211804911)

[6 Pruebas de desarrollo continuo 50](#_Toc211804912)

[6.1 Test driven development 50](#_Toc211804913)

[6.2 Acceptance test driven development 50](#_Toc211804914)

[6.3 Behaviour driven development 56](#_Toc211804915)

[6.4 Domain driven development 64](#_Toc211804916)

[7 Automatización y Evidencia 72](#_Toc211804917)

[7.1 Versionamiento (Github) 72](#_Toc211804918)

[7.2 Trello 74](#_Toc211804919)

# Introducción

En Santa Cruz de la Sierra, los procesos de urbanización acelerada y el crecimiento industrial han generado un incremento sostenido en la concentración de contaminantes atmosféricos, especialmente en las principales áreas metropolitanas.

La calidad ambiental urbana depende en gran medida de las actividades industriales y de factores urbanos como tráfico y condiciones meteorológicas. Muchas veces la información existe en bases de datos públicas, registros normativos y sistemas corporativos, pero está fragmentada y poco accesible.

Este proyecto propone una plataforma de análisis y predicción que integra fuentes de datos externas, aplica modelos predictivos y ejecuta automatizaciones con n8n para mejorar la transparencia, la planificación y la gestión ambiental.

En el presente, se desarrolla el Plan de Aseguramiento de la Calidad del Software (SQA Plan) y su correspondiente Plan de Pruebas (Test Plan), que definen las estrategias, criterios, técnicas, artefactos y métricas que garantizarán la calidad del sistema propuesto, bajo un enfoque ágil basado en Kanban y conforme a las normas internacionales de gestión de calidad ISO 9001:2015

# Integración de la calidad

## Contexto de proyecto

El proyecto propone una plataforma digital orientada a la gestión y monitoreo ambiental urbano-industrial, que centraliza datos provenientes de fuentes públicas, sensores, o APIs.

El desarrollo de esta plataforma es para transformar la información dispersa en indicadores unificados y analíticos que permitan a autoridades, empresas y ciudadanos comprender el estado actual del entorno urbano e identificar tendencias críticas que afecten la calidad del aire y el bienestar poblacional.

El proyecto incorpora un módulo especializado de predicción ambiental basado en técnicas de aprendizaje automático aplicado a series temporales. Este módulo integra datos meteorológicos y de contaminantes provenientes de APIs públicas y genera pronósticos multihorizonte (1 h, 12 h, 24 h, 72 h, 168 h) para los principales contaminantes (PM2.5, PM10, NO₂, O₃).

El sistema aplica modelos predictivos para anticipar eventos críticos (picos de contaminación, y/o concentraciones anómalas de material particulado) y ejecuta automatizaciones con n8n, disparando alertas, generando reportes y notificando a los actores involucrados.

El proyecto abordará uno de los Objetivos de Desarrollo Sostenible (ODS 11), promoviendo la sostenibilidad urbana. Además de fortalecer la toma de decisiones basada en datos (Data-Driven Governance) y fomentar una cultura de transparencia y corresponsabilidad ambiental a través del acceso abierto a la información.

## Objetivo general

Desarrollar una plataforma digital predictiva y automatizada que integre datos urbanos e industriales, genere reportes y dashboards, y ofrezca alertas en tiempo real, con el fin de mejorar la gestión ambiental y la transparencia en Santa Cruz de la Sierra.

## Objetivos específicos

* Integrar fuentes de datos relevantes (APIs públicas, reportes industriales y registros urbanos) en una base unificada.
* Diseñar e implementar modelos de análisis predictivo para anticipar episodios de contaminación y riesgos normativos.
* Configurar automatizaciones con n8n que conviertan predicciones en acciones concretas (alertas, reportes, tickets de seguimiento).
* Desarrollar dashboards dinámicos para autoridades e industrias, y una app ciudadana con información accesible.
* Validar el sistema mediante pruebas de calidad y entregar reportes automáticos para su uso en la gestión ambiental.

### Roles de proyecto

* **Desarrollador Backend** - Jhair Anderson Aguilar Palacios.
  + **Responsabildad:**  Encargado del desarrollo del backend, creación de APIs REST y manejo de la base de datos PostgreSQL y Mongo. Implementa la lógica de negocio, seguridad y despliegue del sistema, garantizando la integridad de los datos y el correcto funcionamiento de los servicios.
* **Ingeniero de Automatización con N8N** - Pedro Renato Escobar Ortuño.
  + **Responsabilidad:** Diseña y mantiene los workflows automatizados con n8n para recopilar, procesar y enviar datos y reportes. Supervisa la ejecución de jobs programados, alertas ambientales y métricas de automatización, asegurando la continuidad de los procesos automáticos.
* **Análista de Datos** - Matias Ricardo Mendoza Peducasse.
  + **Responsabilidad:** Responsable del análisis y modelado de datos industriales y urbanos. Desarrolla modelos predictivos de calidad del aire, evalúa su rendimiento mediante métricas (AUC, p95) y genera los indicadores ambientales integrados en los dashboards del sistema.
* **Auditor/Frontend** - Omar Velasco Teran.
  + **Responsabilidad:** Desarrolla la interfaz de usuario en React + TailwindCSS y ejecuta las pruebas funcionales y de aceptación. Verifica criterios de calidad del Plan SQA, accesibilidad y rendimiento visual. Además, actúa como auditor interno, evaluando el cumplimiento de la ISO 9001:2015.

## Requerimientos funcionales

* **RF-01. Registro y login por rol**
  + Crear cuenta y autenticarse (industria /ciudadanía/admin) con validación de correo y recuperación de contraseña.
* **RF-02. Perfiles y permisos**
  + Accesos diferenciados: ciudadanía (consulta pública), industria (cumplimiento/alertas propias), autoridad (mapa predictivo, reportes agregados), admin (config general).
* **RF-03. Bitácora de actividad**
  + Registrar inicio de sesión, altas/bajas de reglas, generación/envío de alertas y descargas de reportes.
* **RF-04. Conectores de datos externos**
  + Conectar y traer datos desde APIs (Google API, clima, tránsito) con programaciones (cada N min/h).
* **RF-05. Normalización y catálogo**
  + Unificar unidades (μg/m³, AQI), zonas/barrios y fuentes en un catálogo único.
* **RF-06. Mapa actual/pronóstico por zona**
  + Mapa con capa actual y pronóstico 24 h, filtros por zona, contaminante, rango temporal, leyenda y tooltips.
* **RF-07. Dashboards por rol**
  + Autoridad: panorama urbano (tendencias, hotspots, quejas).
  + Industria: cumplimiento y riesgos propios.
  + Ciudadanía: vista simplificada (índice/ICA y recomendaciones).
* **RF-08. Motor de predicción**
  + Ejecutar modelos (series temporales/ML) que entreguen riesgo esperado (probabilidad o valor pronosticado) por zona/ventana (6–24 h).
  + **RF-08.1 – Generación automática de predicciones**
    - El sistema debe ejecutar el motor predictivo cada hora para generar valores futuros por contaminante y zona.
  + **RF-08.2 – Entrenamiento y versionamiento de modelos**
    - Debe existir un proceso reproducible para entrenar el modelo, registrar métricas, fecha de entrenamiento y versión.
  + **RF-08.3 – Validación contra baseline**
    - Cada modelo debe compararse contra persistencia y promedio histórico, y debe superar umbrales mínimos para ser aceptado en producción.
  + **RF-08.4 – API de predicción en tiempo real**
    - El sistema debe exponer un endpoint de inferencia para la plataforma web/móvil y para n8n.
  + **RF-08.5 – Gestión de degradación del modelo**
    - Si la precisión cae más de 10 % respecto al histórico, debe marcarse como degradado y enviar alerta interna.
* **RF-9. Validación y versión de modelos**
  + Guardar versión del modelo, fecha de entrenamiento y métricas (MAE/precisión descriptiva) para trazabilidad.
* **RF-10. Reglas de alerta**
  + Crear reglas por contaminante/zona/umbral/ventana y elegir canal (email/in-app).
* **RF-11. Evaluación y envío**
  + Automatización programada, evalúa predicciones y dispara alertas, registrar destinatarios/estado (enviada/fallida).
* **RF-12. Anti-spam/cooldown**
  + Evitar repetidos por mismo motivo en una ventana mínima (ejemplo - 2 h), con opción de pausar reglas.
* **RF-13. Registro y clasificación de quejas**
  + Formulario (ruido/humo/olores/tráfico) con clasificación automática (reglas/IA ligera) y asignación a industria/autoridad.
* **RF-14. Seguimiento de quejas**
  + Estado (abierta/en gestión/cerrada), evidencias (fotos/enlaces) y notificaciones al denunciante.
* **RF-15. Reporte PDF/Excel**
  + Generación semanal/mensual automática para industria/autoridad con métricas clave y adjunto por email.

## Requerimientos no funcionales

* **RNF-01. Latencia**
  + P95 ≤ 1.2 seg en consultas de mapa/dashboards (datos en caché), P95 ≤ 2 seg en consultas con recomputo ligero.
* **RNF-02. Carga concurrente**
  + Soportar 100 req/min sostenidos sin errores (> 99%).
* **RNF-03. Disponibilidad**
  + Disponibilidad mensual ≥ 99.5% para consulta pública.
* **RNF-04. Degradación**
  + Si una API externa falla, mostrar últimos datos válidos + aviso de datos no actualizados.
* **RNF-05. Autenticación/autorización**
  + JWT/ sesiones seguras, roles y control de acceso por recurso.
* **RNF-06. OWASP**
  + Implementación de cierta seguridad de OWASP.
* **RNF-07. Accesibilidad**
  + Garantizar que toda la funcionalidad sea accesible y operable
* **RNF-08. Responsive**
  + Optimizado para desktop y móvil (mapa y dashboards utilizables).
* **RNF-9. Compatibilidad**
  + Asegurar el funcionamiento correcto en las últimas versiones de Chrome, Firefox, Safari y Edge.

## Historias de usuarios

* **HU-01 – Registro de Usuario:** Yo como nuevo usuario de la plataforma, quiero registrar una cuenta personal o institucional para poder acceder al sistema según mi rol (ciudadanía, industria o autoridad).
  + El sistema valida formato de correo electrónico y longitud mínima de contraseña (≥ 6 caracteres).
  + Muestra mensajes claros de error cuando los datos son inválidos.
  + Una vez confirmado, el usuario puede iniciar sesión según su rol asignado.
* **HU-02 – Inicio de Sesión:** Yo como usuario registrado, quiero iniciar sesión en la plataforma usando mi correo y contraseña válidos para acceder a las funcionalidades según mi rol.
  + Valida credenciales contra la base de datos (JWT/seguridad).
  + Muestra mensaje de error ante credenciales inválidas.
  + Redirige al dashboard correspondiente.
  + Registra el evento en la bitácora del sistema con fecha y resultado.
* **HU-03 - Configurar alertas:** Yo como encargado ambiental de una industria, quiero definir reglas de alerta (umbral/ventana/canales), para activar mitigaciones a tiempo.
  + Crear/editar/pausar reglas, y validaciones de formulario.
  + Recibo email/in-app cuando la predicción ≥ umbral.
  + Bitácora registra la alerta (fecha, zona, valor, canal).
* **HU-04 - Panel de industria:** Yo como encargado ambiental, quiero ver mi dashboard con cumplimiento y riesgos, para planear acciones y evitar sanciones.
  + KPIs de cumplimiento, tendencias, eventos.
* **HU-05 - Consulta ciudadana:** Yo como ciudadano, quiero consultar la calidad del aire en mi barrio, para tomar decisiones (salir, usar mascarilla).
  + Buscador por zona y contaminante; ver ICA simplificado.
  + Recomendaciones básicas (evitar actividad al aire libre).
* **HU-06 - Quejas ciudadanas:** Yo como ciudadano, quiero reportar humo/olores/ruido con foto, para que la autoridad gestione el caso.
  + Formulario con categoría, ubicación y evidencia.
  + Recibo número de caso y estado (abierta/en gestión/cerrada).
* **HU-07 - Clasificación de quejas:** Yo como analista de la autoridad, quiero que el sistema sugiera clasificación y responsable, para acelerar la atención.
  + Sugerencia automática (reglas/IA ligera) editable.
  + Asignación y notificación a la unidad/industria.
* **HU-08 - Reporte automático:** Yo como autoridad, quiero recibir un PDF semanal con indicadores clave, para presentar resultados y tomar decisiones.
  + Reporte llega el lunes < 08:00 con sección de pronóstico.
  + Cubre top zonas críticas, tendencias y alertas enviadas.
* **HU-09 - Historial y auditoría:** Yo como auditor interno, quiero ver un historial de alertas y cambios de reglas, para asegurar trazabilidad.
  + Filtrar por fecha/usuario/regla exportar CSV.
  + Ver cambios (antes/después) y responsable.
* **HU-10 - Preferencias de notificación:** Yo como usuario, quiero elegir canales (email/in-app) y horarios válidos, para recibir alertas efectivas sin spam.
  + Configuro canales y no molestar.
  + Cooldown, que evita duplicados por 2 h mínimo.
* **HU-11 - Estado de datos:** Yo como analista, quiero ver el estado de conectores (última actualización/errores), para reaccionar rápido ante fallas.
  + Tablero de integraciones con semáforos.
  + Detalle por conector y reintentos.
* **HU-12 – Generación automática de predicciones:** Yo como sistema, quiero generar predicciones cada hora usando los datos más recientes para actualizar la calidad del aire futura.
  + Ejecuta 1h/12h/24h/72h/168h
  + Log completo de ejecución
  + Manejo de fallos de datos externos
* **HU-13 – Visualización de pronósticos:** Yo como usuario, quiero ver los valores predichos por hora para comprender cómo variará la calidad del aire.
* **HU-14 – Evaluación de reglas de alerta (dependencia del modelo):** Yo como sistema de alertas, quiero evaluar predicciones y disparar notificaciones cuando se excedan umbrales.

## Marco ágil

La metodología de desarrollo seleccionada para este proyecto es Kanban, debido que ofrece un enfoque flexible y visual para del flujo de trabajo del proyecto, permitiendo priorizar tareas, limitar el trabajo en progreso (WIP) y optimizar la entrega continua de valor. A diferencia de marcos como Scrum o Extreme Programming (XP), Kanban no requiere una planificación por sprints ni roles estrictos, lo cual lo hace ideal para equipos con estructuras dinámicas o en etapas tempranas en metodologías ágiles.

Kanban se organiza a través de un tablero visual dividido en columnas que representan los estados del ciclo de vida del trabajo, los tableros pueden ir desde columnas simples como pendientes, en progreso y finalizadas, hasta una serie de columnas más complejas como:

* Por hacer.
* En desarrollo.
* En revisión.
* En pruebas.
* Finalizadas.

Cada tarjeta dentro del tablero representa un artefacto del proyecto, como una historia de usuario, un requerimiento funcional o una corrección de defectos, los cuales avanzan progresivamente conforme se completan las tareas y se cumplen los criterios de calidad definidos.

Mediante Kanban y su enfoque ágil, permite la entrega continua de funcionalidades y permite incorporar la mejora constante mediante la retroalimentación de los usuarios y del equipo técnico. Adicionalmente, complementa con el aseguramiento de calidad, ya que cada tarjeta es verificada en etapas intermedias antes de ser marcada como finalizada, cumpliendo con las definiciones de Definition of Ready (DoR) y Definition of Done (DoD).

|  |  |
| --- | --- |
| **Etapa** | **Criterio** |
| DoR | HU documentada y con criterios de aceptación claros. |
| DoD | Código revisado, testeado, con evidencias y commit firmado |

# Ciclo de vida del software

## Ciclo de vida del desarrollo del software (SDLC)

El proceso de desarrollo se estructuró de acuerdo con las fases del SDLC: Planificación, Análisis de Requisitos, Diseño del Sistema, Desarrollo, Pruebas, Despliegue y Mantenimiento. Cada entrega siguió estas fases, generando los artefactos que se detallan a continuación:

### Entrega 1 – Desarrollo de idea de proyecto

La primera entrega abarca la concepción y desarrollo del MVP (Producto Mínimo Viable) del sistema Integrador Urbano. Durante esta etapa se implementaron las funcionalidades iniciales para la gestión de usuarios, conexión con APIs públicas de calidad del aire, generación de reportes semanales en PDF y envío de alertas automatizadas mediante n8n.

|  |  |  |
| --- | --- | --- |
| **Fase** | **Actividades principales** | **Resultado / Artefacto** |
| Planificación | Definición de alcance, cronograma y responsables. | Documento de planificación y backlog inicial. |
| Análisis de requisitos | Identificación de requerimientos funcionales y no funcionales. | Documento de HU priorizadas. |
| Diseño del sistema | Diagramas de arquitectura, flujos de datos y estructura de BD. | Diagramas de casos de uso/UML, modelo entidad-relación. |
| Desarrollo | Implementación de autenticación, consulta API y reportes PDF. | Código fuente, commits versionados, jobs n8n configurados. |
| Pruebas (unitarias/integración) | Validación inicial de módulos y flujo general del sistema. | Registro de pruebas unitarias y checklist de QA interno. |
| Despliegue | Publicación en entorno de pruebas (staging). | Build en staging y documentación de despliegue. |
| Mantenimiento inicial | Corrección de incidencias del entorno MVP. | Registro de incidencias, actualización de manuales. |

* **Artefactos generados**
  + Documento de planificación del proyecto.
  + Documento de requisitos (RF/NF).
  + Diagramas del sistema.
  + Código base y entorno de pruebas configurado.
  + Registro de pruebas y reporte de validación del MVP.

### Entrega 2 – Ampliación y predicción ambiental

La segunda entrega amplía el sistema con módulos predictivos de contaminación, clasificación automática de alertas y mejoras en rendimiento, seguridad y notificaciones personalizadas.

|  |  |  |
| --- | --- | --- |
| **Fase** | **Actividades principales** | **Resultado / Artefacto** |
| Planificación | Ajuste de tareas e identificación de nuevas HU. | Plan de iteración 2 y nuevas historias. |
| Análisis de requisitos | Definición de requerimientos para modelos predictivos y analítica. | Documento técnico de modelo predictivo. |
| Diseño del sistema | Diseño del módulo de predicción y API de inferencia. | Diagrama de arquitectura actualizada. |
| Desarrollo | Implementación del modelo, endpoints y optimización del sistema. | Código del modelo, servicios REST, scripts de entrenamiento. |
| Pruebas (unitarias/integración) | Pruebas de rendimiento, validación de modelo (AUC ≥ 0.80), regresión. | Reporte de validación del modelo y métricas de QA. |
| Despliegue | Integración continua y publicación de la nueva versión. | Build v2.0 liberado y documentación CI/CD. |
| Mantenimiento inicial | Seguimiento de métricas, retrabajo y mejora continua. | Reporte de postproducción y lecciones aprendidas. |

* **Artefactos generados**
  + Documento técnico de modelo predictivo.
  + Scripts y código del módulo de analítica.
  + Reporte de rendimiento y validación (AUC, p95).
  + Documentación de CI/CD.
  + Acta de liberación y métricas post-despliegue.

## Ciclo de vida de las pruebas del software (STLC)

El proceso de pruebas se ejecutó en paralelo con el desarrollo, siguiendo las fases del Software Testing Life Cycle (STLC): Análisis de requisitos de prueba, Planificación de pruebas, Diseño de casos, Configuración de entorno, Ejecución y Cierre. Cada entrega generó artefactos específicos, asegurando la trazabilidad con los requerimientos y las métricas de calidad definidas.

### Entrega 1 - Ciclo de pruebas del MVP

|  |  |  |
| --- | --- | --- |
| **Fase** | **Actividades principales** | **Resultado / Artefacto** |
| Análisis de requisitos de pruebas | Revisión de HU y definición de alcance de prueba. | Matriz de trazabilidad requerimientos. |
| Planificación de pruebas | Elaboración del plan de pruebas inicial (Test Plan v1). | Documento de planificación QA. |
| Diseño de casos de pruebas | Creación de casos aplicando partición, límite y decisión. | Tabla de test cases y datasets. |
| Configuración del entorno de prueba | Configuración del entorno de staging y credenciales. | Checklist de entorno validado. |
| Ejecución de las pruebas | Pruebas unitarias, integración, API y aceptación. | Evidencias de ejecución y defectos registrados. |
| Cierre de pruebas | Consolidación de resultados y métricas. | Informe de cierre de pruebas v1. |

* **Artefactos generados**
  + Test Plan v1.
  + Matriz de trazabilidad.
  + Tabla de test cases.
  + Informe de cierre de pruebas (MVP).

### Entrega 2 - Ciclo de pruebas de predicción de datos

|  |  |  |
| --- | --- | --- |
| **Fase** | **Actividades principales** | **Resultado / Artefacto** |
| Análisis de requisitos de pruebas | Definición de requisitos de validación del modelo predictivo. | Documento de pruebas ML para predicción de datos. |
| Planificación de pruebas | Actualización del plan de pruebas (Test Plan v2). | Plan de pruebas extendido. |
| Diseño de casos de pruebas | Casos de rendimiento, regresión y validación de modelo. | Nuevos casos de prueba ML y rendimiento. |
| Configuración del entorno de prueba | Configuración de datasets y jobs automatizados. | Dataset anonimizado y pipeline configurado. |
| Ejecución de las pruebas | Pruebas de integridad de datos, pruebas unitarias de generación de features, pruebas de regresión al reentrenar el modelo | Reporte de resultados y métricas. |
| Cierre de pruebas | Análisis de resultados, documentación final y aprobación QA. | Informe de cierre de pruebas v2 y lecciones aprendidas. |

* **Artefactos generados**
  + Test Plan v2.
  + Dataset de validación.
  + Reporte de métricas (AUC, p95).
  + Informe final de QA con evidencias.

# Técnicas y estrategia de pruebas

## Tipos de pruebas funcionales

### Pruebas unitarias

Verificar que las funciones y componentes individuales del sistema se comporten correctamente de forma aislada, sin depender de otros módulos. En el proyecto creamos el archivo Login.test.tsx, ubicado en src/\_\_tests\_\_/

const validateEmail = (email: string) => {

  const re = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

  return re.test(email);

};

* Donde declaramos una constante para poder realizar la validación de los caracteres permitidos en nuestro login.

import { render, screen, fireEvent } from '@testing-library/react';

import Login from '../pages/Login';

test('El botón "Iniciar sesión" se habilita con correo y contraseña válidos', () => {

  render(<Login />);

  fireEvent.change(screen.getByLabelText(/Correo electrónico/i), {target:{value:'velascoteranomar@gmail.com'}});

  fireEvent.change(screen.getByLabelText(/Contraseña/i), {target:{value:'123456'}});

  expect(screen.getByRole('button', {name:/Iniciar sesión/i})).not.toBeDisabled();

});

* Esta es la prueba unitaria para validar correo y habilitar botón de envío, en donde prueba la unidad funcional LoginForm, verificando que las validaciones implementadas en la función validateEmail() y el control del botón de envío respondan según los criterios de aceptación definidos.

### Pruebas de integración

Validar la interacción entre módulos o componentes, asegurando que las funciones trabajen correctamente en conjunto, como la comunicación entre la interfaz y el servicio de autenticación.

Para este ejemplo, se utilizo el endpoint de autenticación de Login, en donde:

import { render, fireEvent, getByText } from '@testing-library/react';

import Login from '../pages/Login';

test('Envía payload correcto y recibe respuesta 200 simulada', async () => {

  global.fetch = vi.fn(() =>

    Promise.resolve({

      ok: true,

      json: () => Promise.resolve({ token: 'abc123' }),

    })

  ) as any;

  const { getByLabelText, getByText } = render(<Login />);

  fireEvent.change(getByLabelText(/Correo/i), { target: { value: 'velascoteranomar@gmail.com' } });

  fireEvent.change(getByLabelText(/Contraseña/i), { target: { value: '123456' } });

  fireEvent.click(getByText(/Iniciar sesión/i));

  expect(global.fetch).toHaveBeenCalledWith('/api/login', expect.any(Object));

});

* La prueba valida la respuesta del UI con la API, comprobando que se envía la estructura correcta del payload y se maneja la respuesta simulada.

### Pruebas de sistema

Evalua el comportamiento end-to-end del sistema en su conjunto, asegurando que todos los componentes y servicios funcionen correctamente para el flujo completo. Para nuestro proyecto en desarrollo, decidimos explayarlo mediante el uso de:

* Smoke test de navegación y visibilidad de elementos (la maqueta no “rompe” rutas ni estilos). En este caso, al usar react manejamos la navegación mediante Routes o Rutas. Al implementar backend se mostrara evidencia de logs.

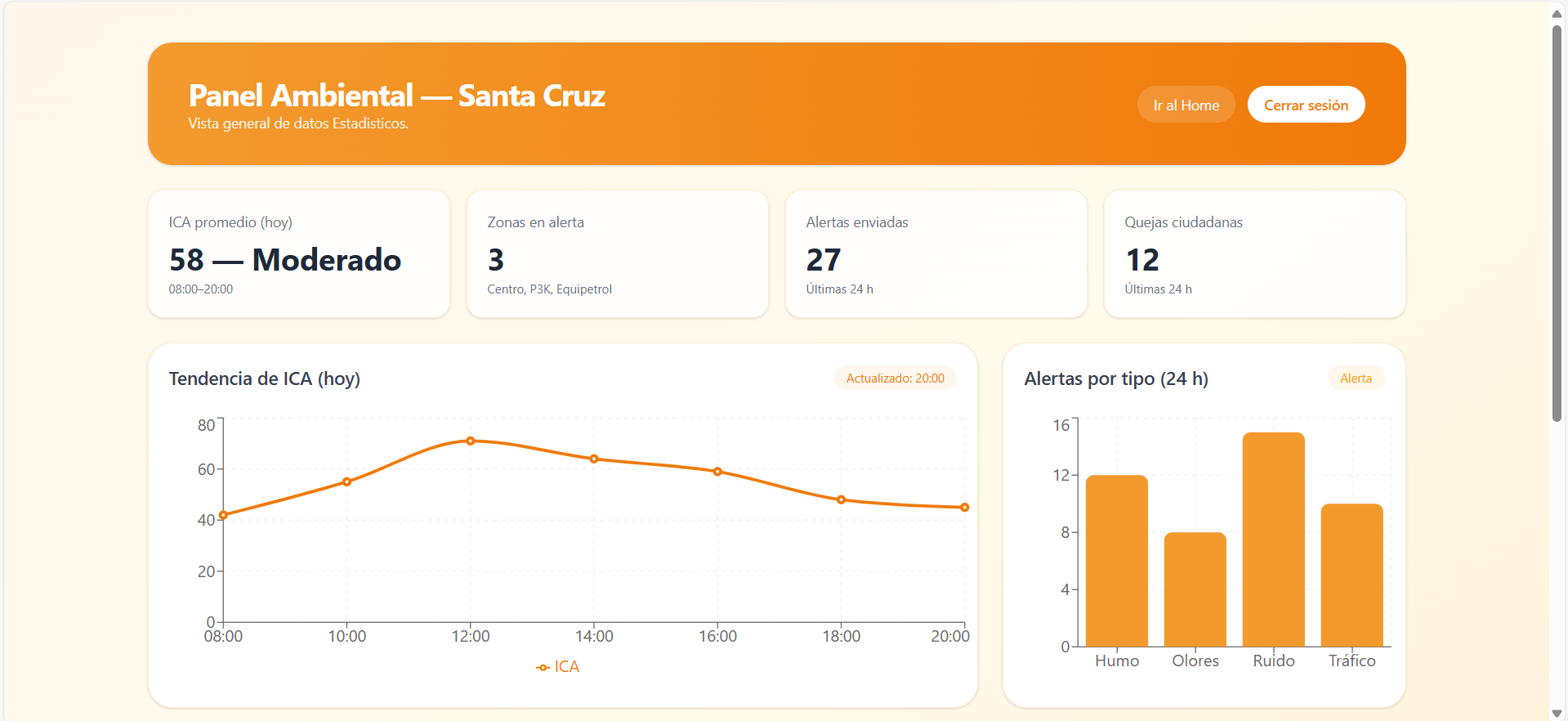
### Pruebas de aceptación

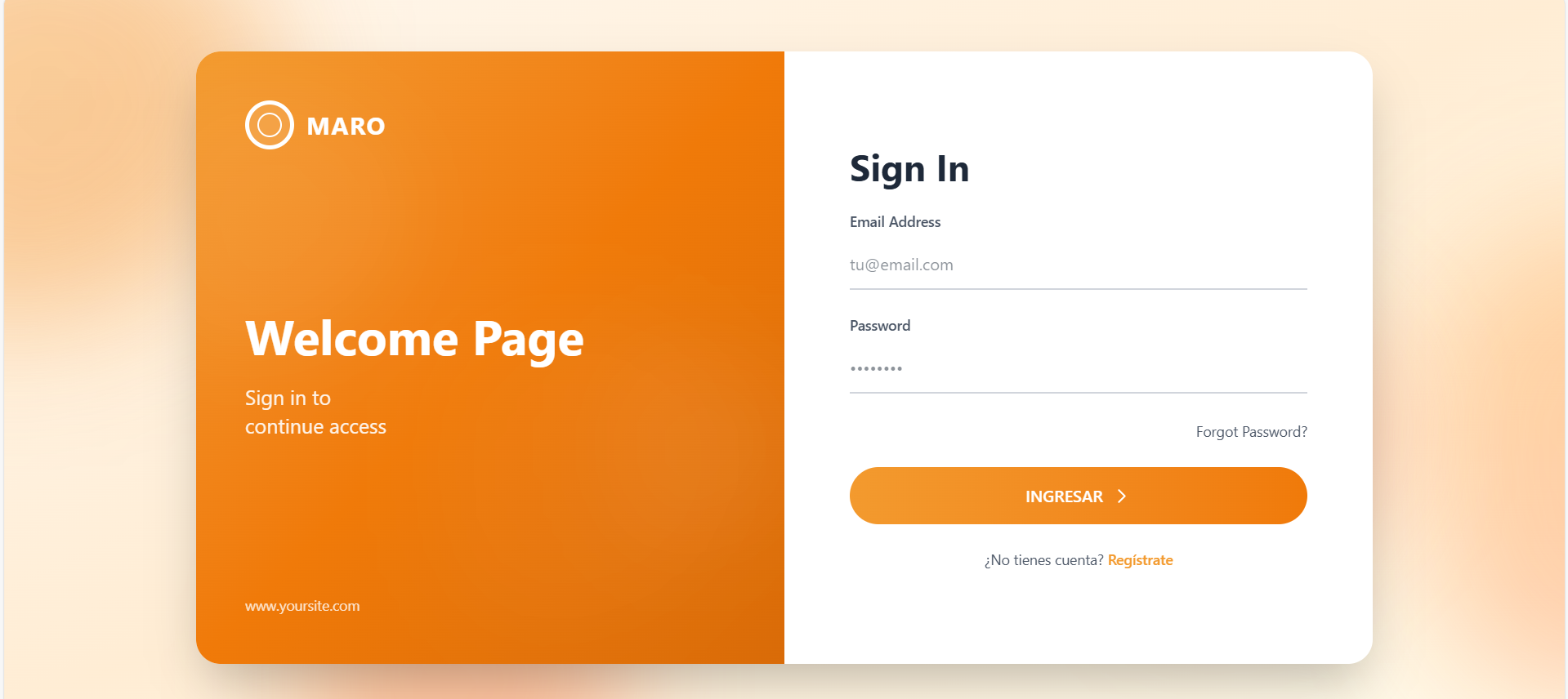
Confirmar que el sistema cumple los criterios de aceptación definidos por el usuario o cliente, garantizando que las funcionalidades entregadas satisfacen las necesidades planteadas. Para complementar este punto mediante las pruebas BDD, con su formato de Given–When–Then, se asegurará que cumpla con los requisitos funcionales y no funcionales, a su vez de revisar una inspección de calidad en la implementación completa de las historias de usuarios. Tomando un ejemplo de escenario:

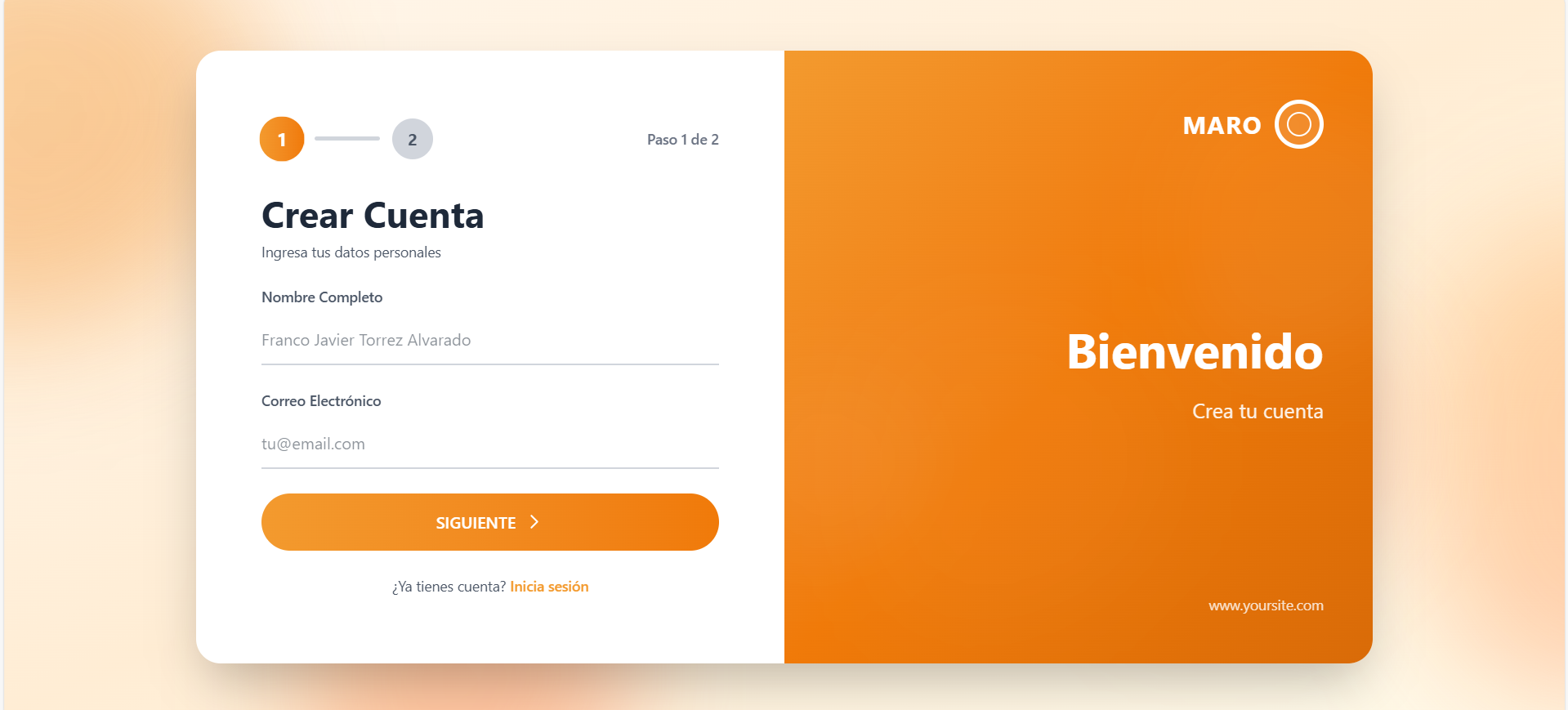
* Escenario: Registro exitoso de un nuevo usuario Dado que el usuario completa el formulario con datos válidos Cuando presiona el botón Crear cuenta, Entonces el sistema muestra el mensaje “Registro exitoso” Y redirige automáticamente a la pantalla de inicio de sesión.

### Pruebas de interfaz de usuario

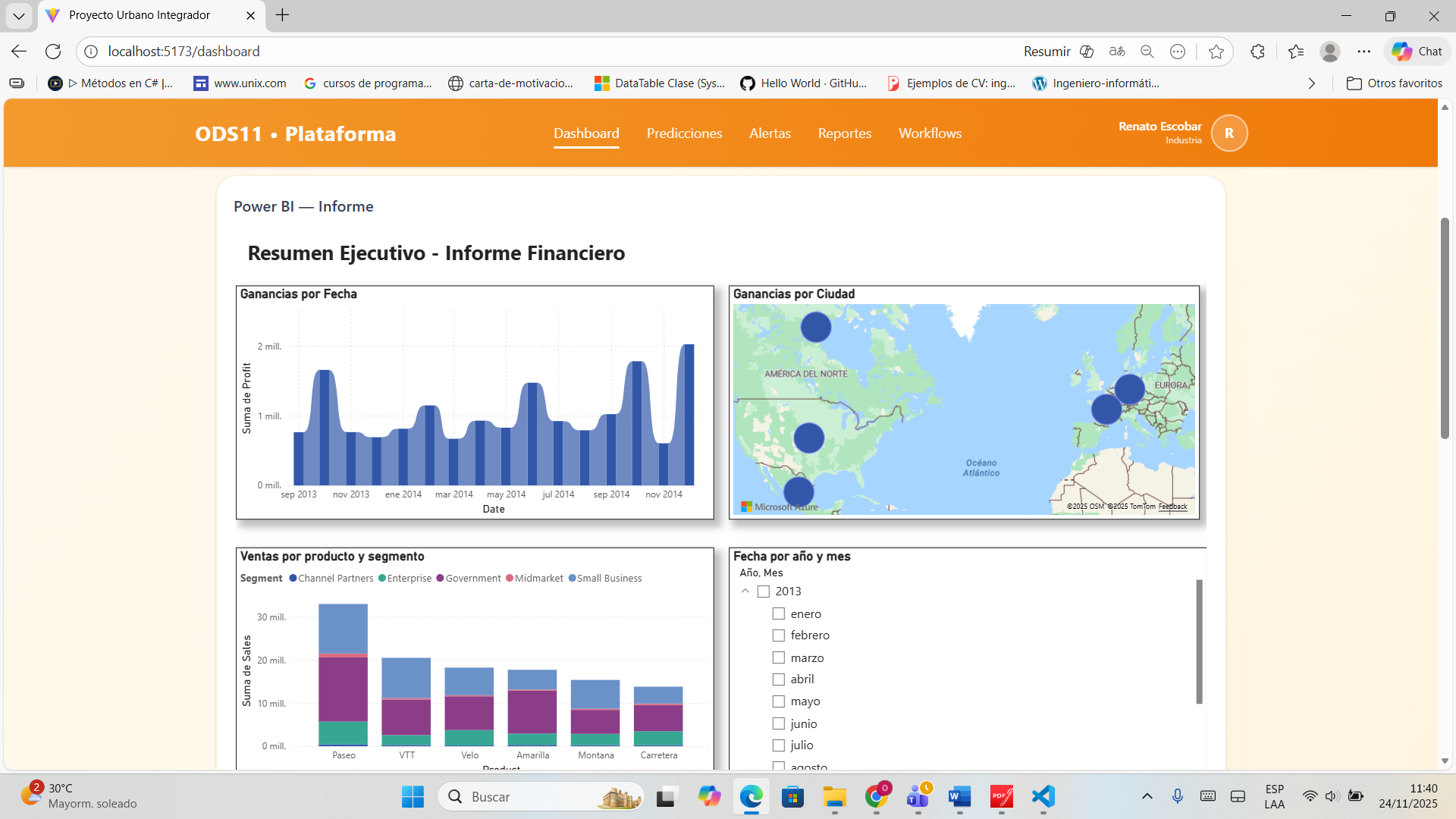
En este punto se evaluará la usabilidad, consistencia visual y accesibilidad de la interfaz desarrollada con TailwindCSS, asegurando su correcta visualización en distintos dispositivos.

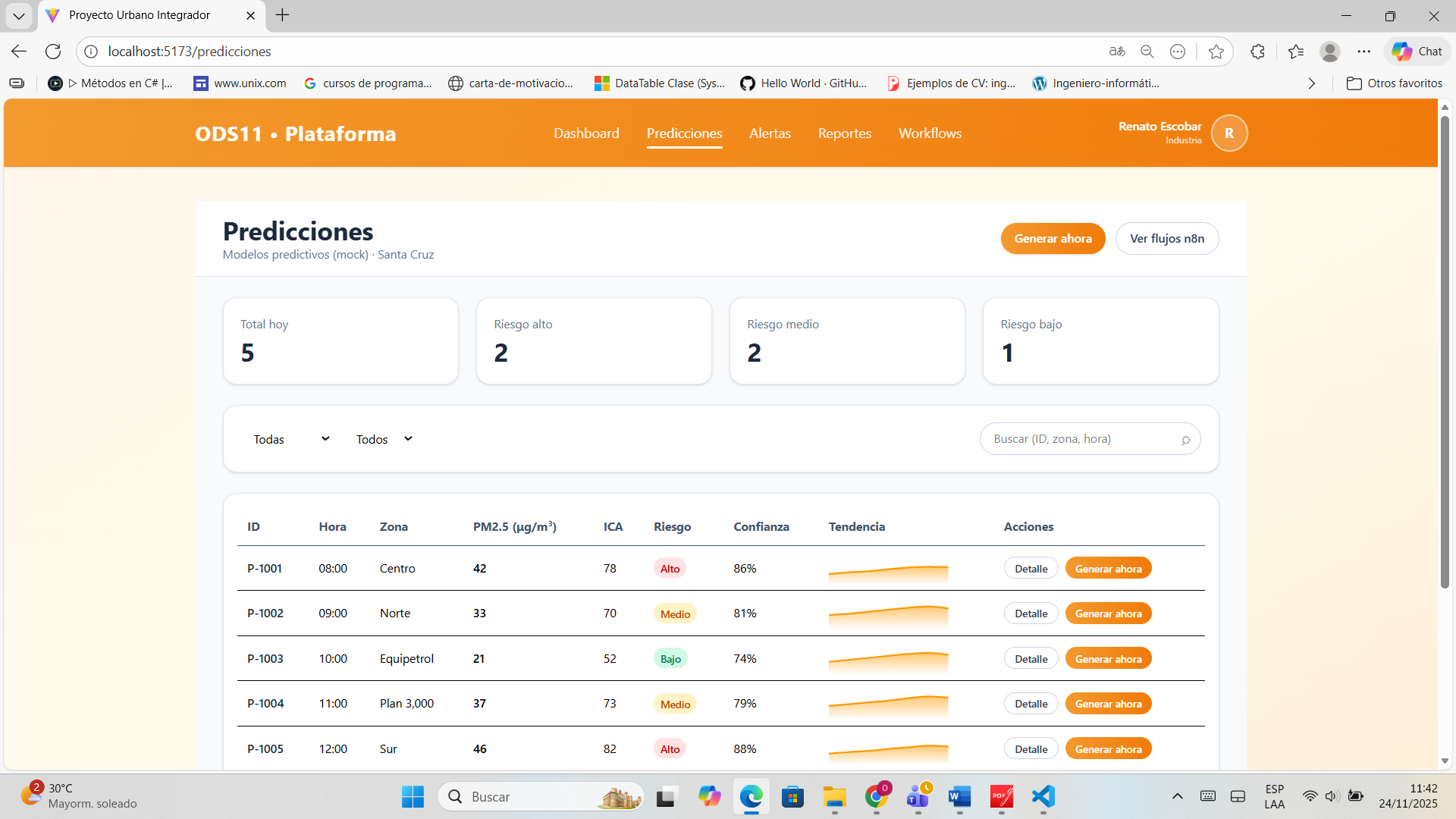


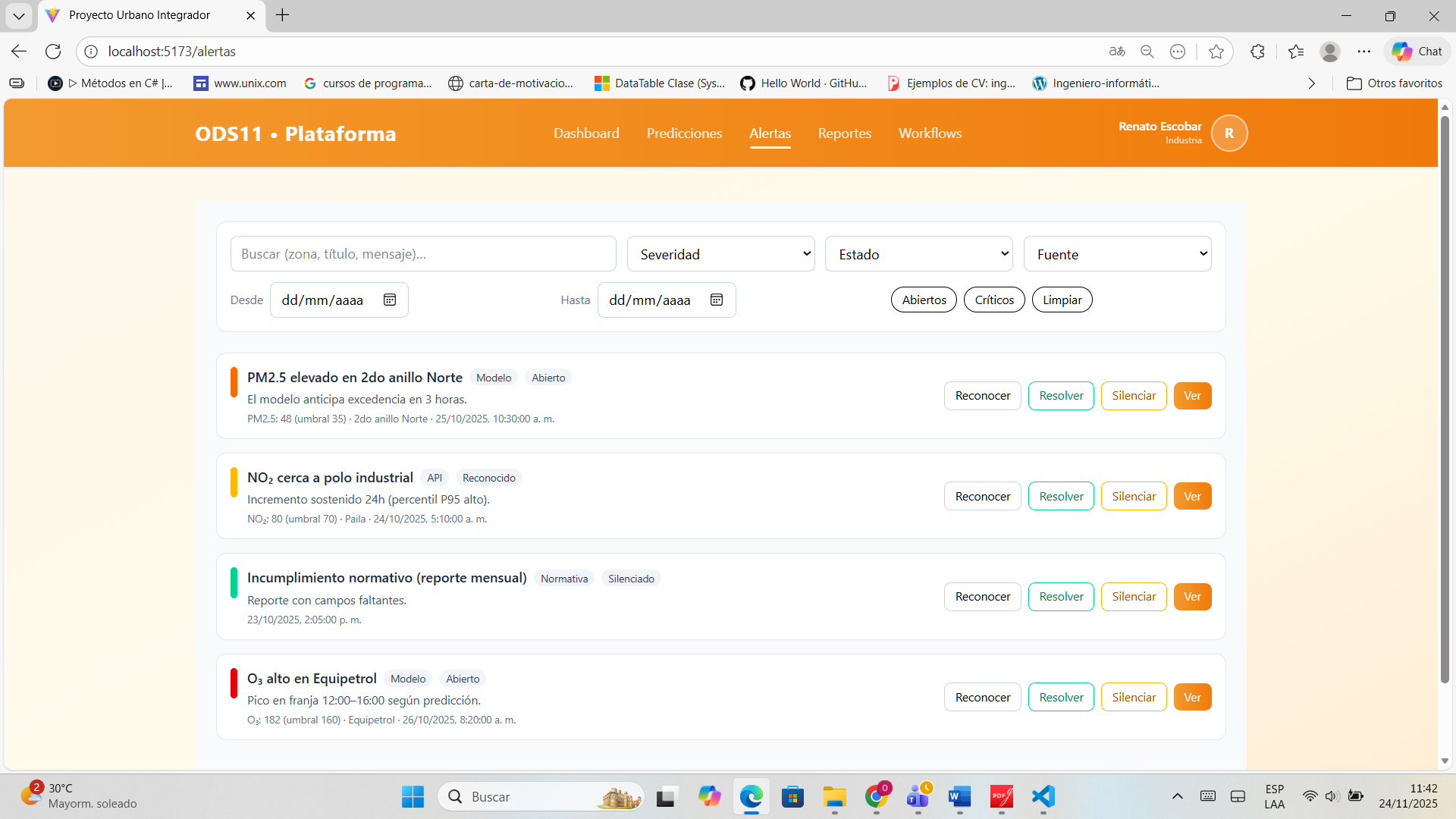


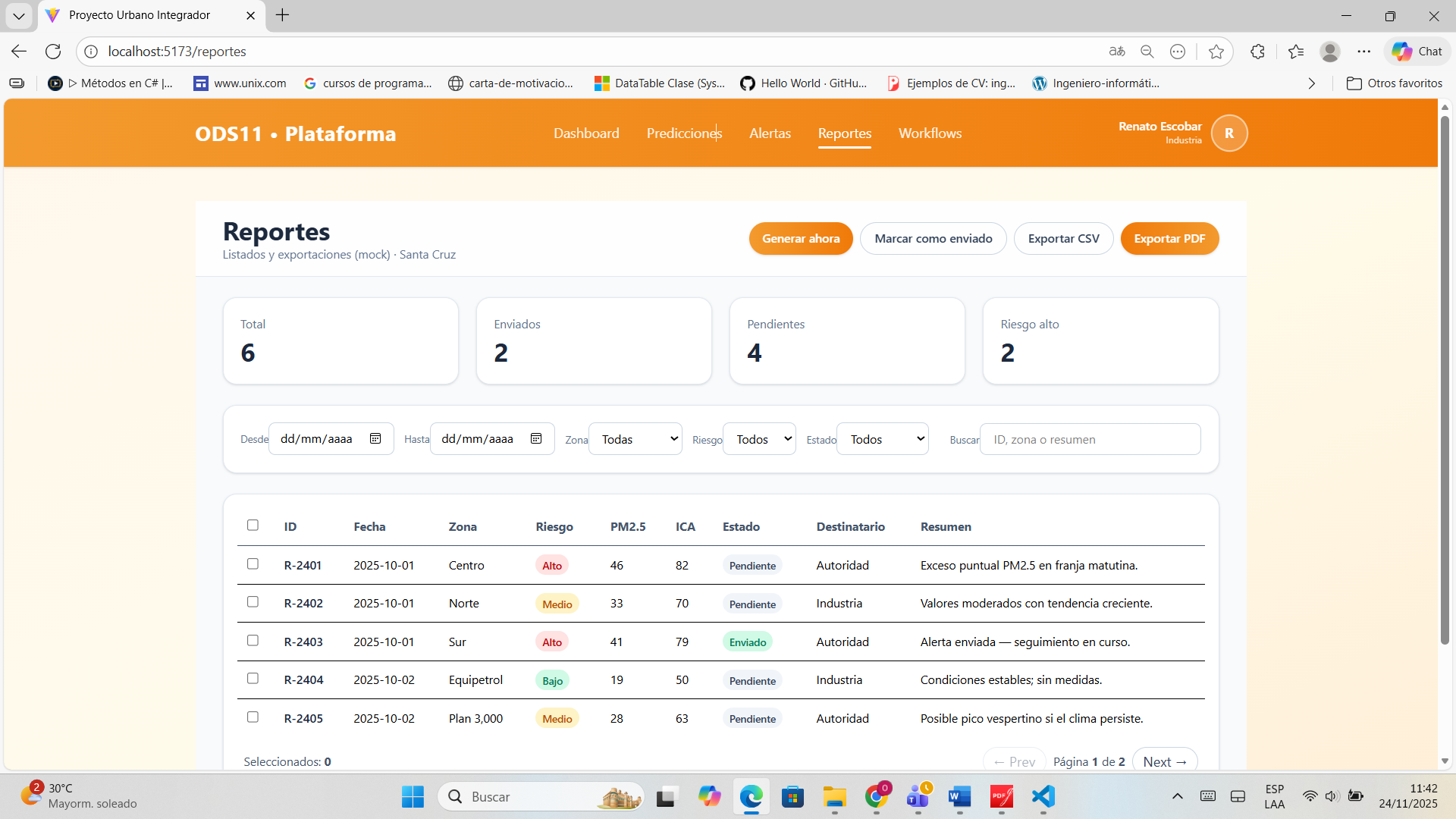


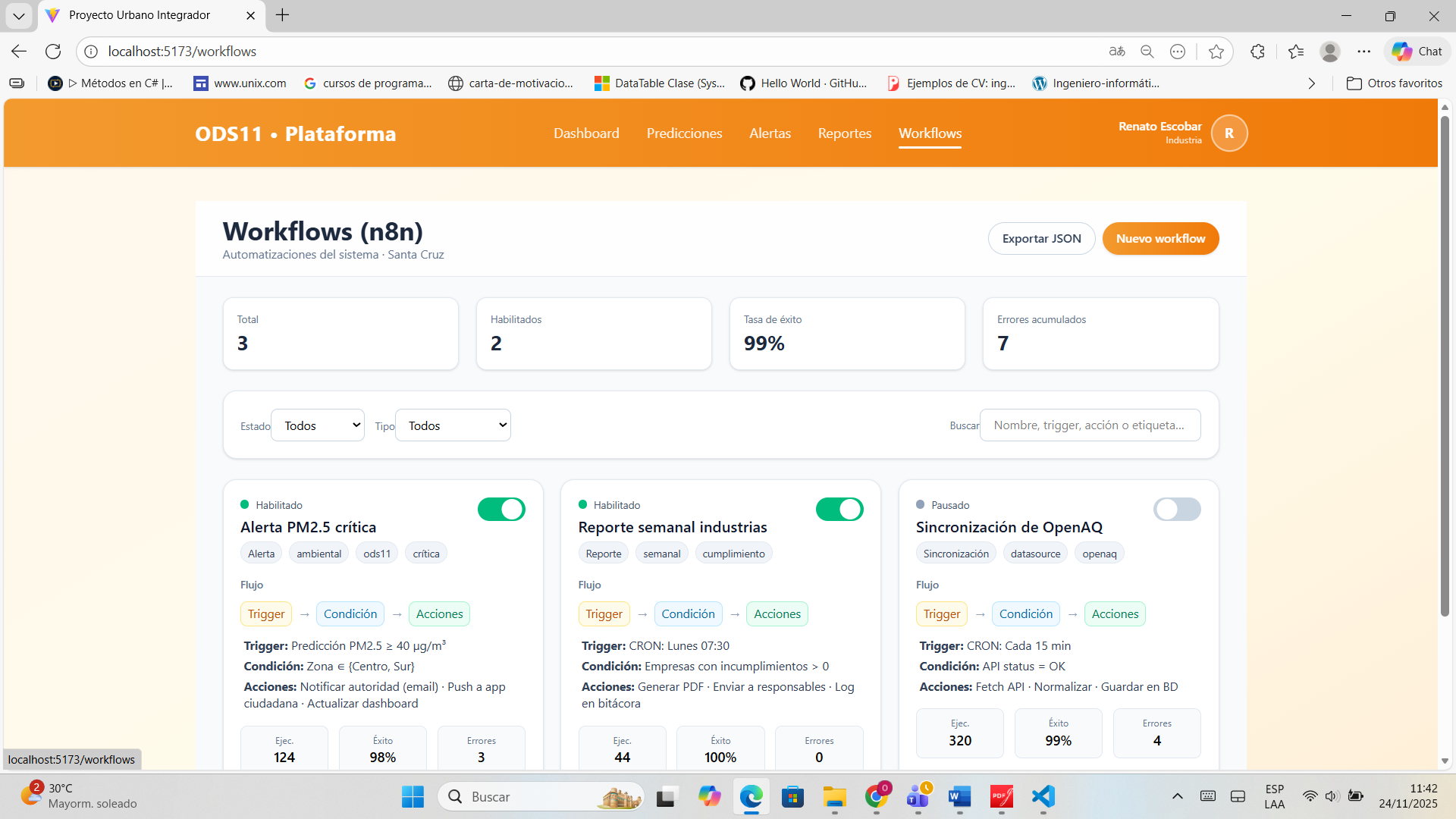
#### Entregable 1











Entregable 2

## Tipos de pruebas no funcionales

### Pruebas de rendimiento

Se evaluará la capacidad del sistema para responder de manera eficiente bajo diferentes niveles de carga, garantizando tiempos de respuesta adecuados, estabilidad operativa y cumplimiento de los requisitos no funcionales de latencia y disponibilidad definidos en el plan.

**Objetivos específicos**

* Validar que el sistema mantenga un P95 ≤ 1.2 segundos en consultas de mapas y dashboards, y ≤ 2 segundos en operaciones con recomputo.
* Asegurar que el sistema soporte al menos 100 solicitudes por minuto sostenidas sin pérdida de integridad ni errores (> 99 % de éxito).
* Medir el desempeño de los servicios de backend, cache y API externas bajo escenarios reales de uso.
* Detectar posibles cuellos de botella en las consultas a bases de datos, workflows de n8n o integraciones de datos externos (por ejemplo, APIs de clima o contaminación).

**Criterios de aceptación**

* El tiempo de respuesta P95 ≤ 1.2 s para consultas en caché y ≤ 2 s para consultas en tiempo real.
* Disponibilidad mínima 99.5 % durante la prueba.
* Error rate < 1 % en todas las peticiones concurrentes.
* Ningún proceso crítico (alertas, pronóstico, dashboard) debe presentar fallos de bloqueo o timeout.

### Pruebas de estrés

Las pruebas de estrés se realizan con el propósito de evaluar la resiliencia, estabilidad y comportamiento del sistema bajo condiciones extremas de carga o uso intensivo, determinando el punto en el que los componentes dejan de responder correctamente o presentan degradación significativa del servicio.

**Objetivos específicos**

* + Determinar la **capacidad máxima** del sistema antes de experimentar fallos críticos o tiempos de respuesta inaceptables.
  + Evaluar el comportamiento del sistema cuando se supera el umbral de carga de **200 solicitudes por minuto**, observando degradación progresiva de los servicios.
  + Identificar componentes o servicios que no escalen correctamente, especialmente en la comunicación entre la **API de backend**, el **motor de alertas**, y las **integraciones externas (n8n, Google Air API, climatología)**.
  + Verificar la capacidad del sistema para **recuperarse automáticamente** después de un evento de sobrecarga.

**Criterios de aceptación**

* + El sistema debe **mantener disponibilidad mínima del 95 %** durante el período de estrés.
  + No se deben registrar **fallos críticos de seguridad o pérdida de datos** bajo ninguna condición.
  + El tiempo de recuperación tras la finalización de la prueba no debe superar los **60 segundos**.
  + Los errores controlados deben generar alertas en el sistema de monitoreo o logs del backend.

### Pruebas de seguridad

Las pruebas de seguridad tienen como propósito garantizar la protección de los datos, la integridad del sistema y la confidencialidad de la información gestionada por la plataforma, frente a posibles vulnerabilidades o ataques internos y externos.

**Objetivos específicos**

* + Verificar la correcta implementación de **autenticación segura (JWT)** y control de acceso por roles (industria, autoridad, ciudadano, administrador).
  + Evaluar la resistencia del sistema ante ataques comunes como **inyecciones SQL/NoSQL**, **Cross-Site Scripting (XSS)**, **Cross-Site Request Forgery (CSRF)** y **fuerza bruta en login**.
  + Comprobar que la información transmitida entre el frontend y el backend se realice mediante **protocolos HTTPS cifrados (TLS 1.2 o superior)**.
  + Validar la gestión adecuada de contraseñas, almacenamiento en hash y políticas de expiración de tokens.
  + Confirmar que los flujos críticos (registro, alertas, dashboards y reportes) estén protegidos contra accesos no autorizados o manipulación de datos.

**Criterios de aceptación**

* + Ninguna vulnerabilidad crítica (nivel alto) debe permanecer sin mitigación.
  + Todas las contraseñas y tokens deben estar **encriptados y protegidos** frente a lectura directa o reutilización.
  + Las pruebas de inyección, XSS o CSRF no deben permitir el acceso o modificación de datos sensibles.
  + El sistema debe cumplir con las recomendaciones **OWASP Top 10 (2021)** y las políticas internas de seguridad del proyecto.
  + Se debe garantizar la **integridad y confidencialidad de los datos** durante toda la sesión de usuario.

### Pruebas de configuración

Las pruebas de configuración tienen como finalidad asegurar que la plataforma funcione correctamente bajo diferentes entornos, versiones de sistema y configuraciones de infraestructura, garantizando la compatibilidad, estabilidad y portabilidad del software.

**Objetivos específicos**

* + Validar la correcta instalación, configuración y despliegue del sistema en los entornos de **staging** y **producción**.
  + Verificar la **compatibilidad** con las versiones objetivo de navegadores, sistemas operativos y dependencias técnicas (Node.js, Vite, TailwindCSS, Supabase y n8n).
  + Comprobar la correcta gestión de **variables de entorno**, rutas de API y claves de acceso.
  + Asegurar que los cambios en la configuración no afecten el rendimiento ni la seguridad del sistema.
  + Confirmar que los procesos de integración y entrega continua (**CI/CD**) desplieguen las versiones estables sin interrupciones ni pérdidas de datos.

**Criterios de aceptación**

* + El sistema debe **instalarse y configurarse correctamente** en menos de 5 minutos, sin errores en la compilación o conexión.
  + Todas las **dependencias** deben encontrarse actualizadas y libres de conflictos de versión.
  + El despliegue en los entornos *staging* y *producción* debe completarse sin fallos en el pipeline.
  + Las variables de entorno deben cargarse correctamente, garantizando la conexión a los servicios externos.
  + La aplicación debe ejecutarse correctamente en los **principales navegadores y resoluciones**.

### Pruebas de calidad de la información

Las pruebas de calidad de la información tienen como objetivo garantizar la exactitud, consistencia, integridad y confiabilidad de los datos procesados y presentados por la plataforma, asegurando que los resultados mostrados (mapas, pronósticos, alertas y reportes) sean correctos, actualizados y trazables a su fuente de origen.

**Objetivos específicos**

* + Validar que los datos importados desde fuentes externas (por ejemplo, *Google Air API*, *OpenWeather*, o sensores locales) se almacenen y transformen sin pérdida de precisión.
  + Verificar la consistencia entre los valores presentados en las distintas capas del sistema: API, dashboard y reportes descargables.
  + Asegurar que los cálculos derivados (índices de contaminación, categorías de riesgo, umbrales de alerta) sean correctos según las fórmulas definidas por la autoridad ambiental.
  + Confirmar la sincronización y actualización periódica de los datos en memoria caché y base de datos.
  + Detectar y registrar valores anómalos, faltantes o inconsistentes, aplicando controles de validación y saneamiento de datos.

**Criterios de aceptación**

* + El 100 % de los registros debe contener datos válidos y completos (sin valores nulos o incorrectos en campos críticos).
  + La variación máxima aceptada entre datos originales y procesados no debe superar el **±2 %**.
  + Las fórmulas de cálculo de índices (ICA, AQI) deben generar resultados coherentes con los rangos oficiales definidos.
  + La frecuencia de actualización de datos debe cumplirse sin retrasos mayores a 5 %.
  + Cualquier error o inconsistencia debe registrarse en logs y notificar al administrador del sistema.

## Pruebas de validación

Las pruebas de validación tienen como propósito confirmar que el sistema cumple con los requisitos funcionales, no funcionales y de calidad definidos en el plan del proyecto, asegurando que el software desarrollado satisface las necesidades reales del usuario final y los objetivos de negocio establecidos.

A diferencia de las pruebas de verificación, que se enfocan en comprobar que el sistema se construya correctamente, la validación garantiza que se haya construido el sistema correcto, es decir, que las funcionalidades entregadas respondan al propósito para el cual fueron diseñadas.

* **Objetivos específicos**
  + Validar el cumplimiento de los requisitos funcionales (RF-01 a RF-15) y no funcionales (RNF-01 a RNF-09) definidos en las secciones previas del plan.
  + Confirmar que las historias de usuario y sus criterios de aceptación se cumplan según los escenarios definidos en ATDD y BDD.
  + Evaluar la usabilidad, accesibilidad y rendimiento integral del sistema en su entorno final.
  + Garantizar que el sistema cumple con los estándares de calidad y las políticas de la norma ISO 9001:2015, aplicables al aseguramiento del software.
  + Obtener la aprobación formal del cliente o del supervisor académico mediante evidencia funcional y técnica documentada.
* **Metodología**

Las pruebas de validación se ejecutan sobre el sistema completamente integrado en un entorno de *staging* que replica las condiciones de producción. El proceso incluye las siguientes etapas:

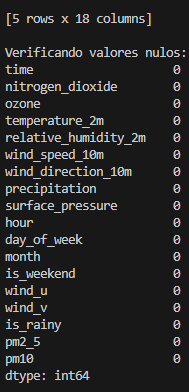
1. **Preparación de entorno de validación:** instalación del build estable, configuración de base de datos, APIs y workflows de n8n.
2. **Revisión de trazabilidad:** verificación de que cada requisito, historia de usuario y caso de prueba tengan correspondencia con una evidencia funcional.
3. **Ejecución de escenarios de aceptación (BDD):** simulación de flujos reales de uso (login, mapa, alertas, dashboards y reportes).
4. **Validación de resultados:** contraste de los resultados obtenidos con los valores esperados en los criterios de aceptación.
5. **Revisión con el cliente o supervisor:** demostración funcional del sistema y levantamiento de observaciones.

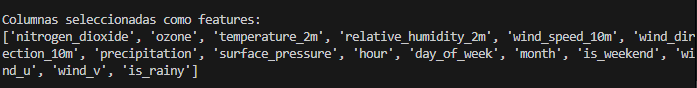
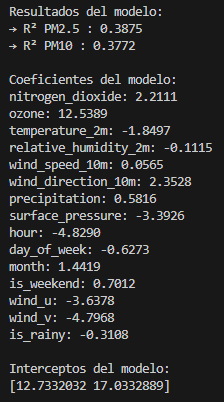
* **Criterios de aceptación**
* El 100 % de los requisitos críticos deben contar con evidencia de validación exitosa.
* Todos los defectos detectados durante la fase de validación deben resolverse o documentarse con plan de acción antes de la entrega final.
* El sistema debe superar las pruebas de aceptación definidas en BDD sin errores funcionales o de integración.
* Los reportes de validación deben incluir evidencias (capturas, logs, videos o registros de ejecución) almacenadas en el repositorio del proyecto.
* **Resultado esperado**

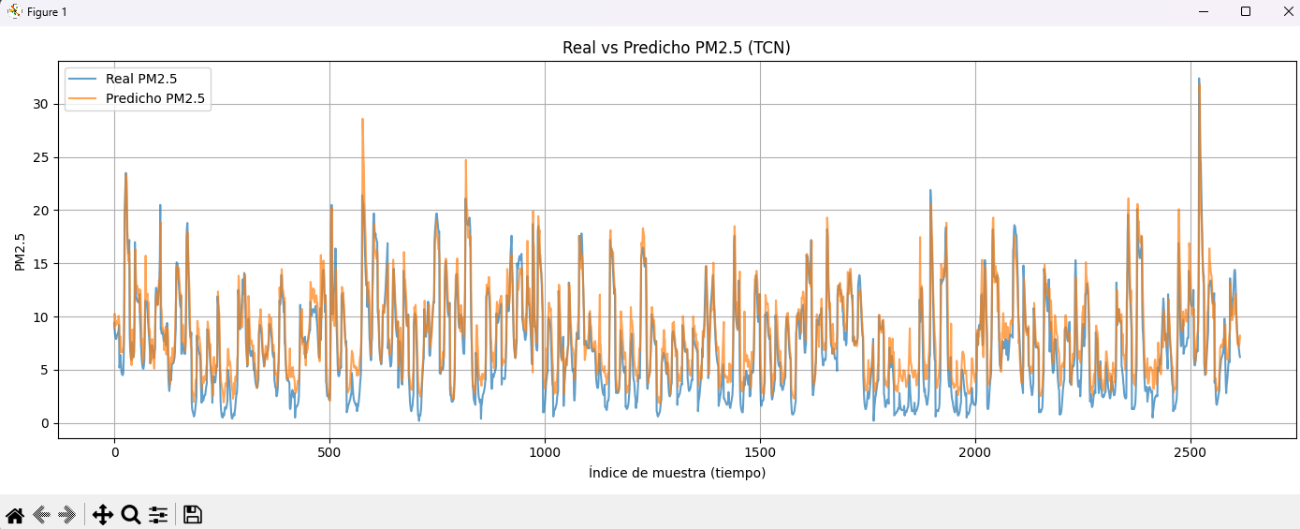
Al finalizar esta etapa, se espera disponer de un informe de validación integral, que consolide los resultados de todas las pruebas ejecutadas y sirva como base para la aprobación formal del sistema. Este informe incluirá el registro de evidencias, observaciones del QA y la conformidad final del cliente o responsable académico.

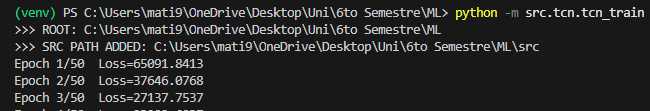
## Pruebas de ML

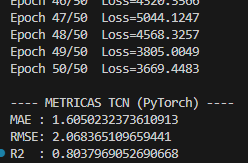
* Pruebas de regresión lineal: Validar la correcta extracción, limpieza, transformación y preparación de los endpoints de variables meteorológicas y de calidad del aire para entrenar un modelo de regresión lineal múltiple. Verificar que el resultado del primer modelo cumpla con el criterio mínimo de desempeño definido para la fase inicial.

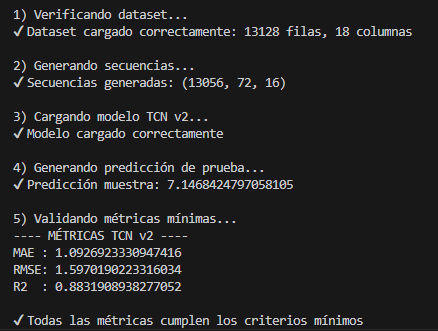




* **Pruebas TCN:** Validar que el modelo TCN procese secuencias temporales correctamente, genere un modelo entrenado y obtenga un rendimiento superior al obtenido con regresión lineal.







## Técnicas de diseño de Test cases

### Partición de equivalencias

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Requisito** | **Técnica aplicada** | **Entradas / condiciones** | **Pasos** | **Resultado esperado** | **Criterio de aceptación / Métrica** |
| TC-PE-01 | RF-01. Registro y login por rol | Partición de equivalencias | Conjuntos válidos/ inválidos de: {email, contraseña (≥6), | Abrir registro, completar con datos válidos y enviar. | Exitoso con datos válidos o rechazo con mensaje. | 100% aceptación válidos. 100% rechazo inválidos. |
| TC-PE-02 | RF-04. Conectores de datos externos | Partición de equivalencias | API key {válida / inválida / expirada} | Llamar llave válida, y repetir con llave inválida. | 200 + datos para válida; 401/403 para inválida. | 0 errores para valida, 100% rechazo claves inválidas/expiradas. |
| TC-PE-03 | RF-07. Dashboards por rol | Partición de equivalencias | Rol = {invitado/usuario convencional, admin} | Autenticarse con cada rol, y navegar dashboard. | Menu/Widgets visibles según rol. | 100% coincidencia visibilidad-rol, 0 accesos no autorizados. |
| TC-PE-04 | RNF-09. Compatibilidad | Partición de equivalencias | Navegador como Chrome, Edge, brave, etc. | Abrir login o registro en cada navegador. | UI consistente (layout, fuentes, controles). | Compatibilidad ≥ 95% en navegadores objetivo, sin fallos críticos visuales. |

### Análisis de valores limite

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Requisito** | **Técnica aplicada** | **Entradas / condiciones** | **Pasos** | **Resultado esperado** | **Criterio de aceptación / Métrica** |
| TC-AVL-01 | RF-06. Mapa actual/pronóstico | Valores límite | Umbrales como: 0, 12, 12.1, 35.4, 35.5, 55.4, 55.5 μg/m³ | Ingresar valores de borde, y consultar clasificación/leyenda. | La categoría (ICA) cambia justo en los límites; leyenda y color correctos. | Margen de error ≤ 2% en límites, clasificación correcta 100% en puntos de corte |
| TC-AVL-02 | RF-15. Reporte PDF | Valores límite | Ventana semanal: Dom 23:59 / Lun 00:00 / Lun 08:00 | Programar job semanal. (2) Verificar generación y envío al cambio de día. | PDF generado y entregado antes de Lun 08:00; adjunto no vacío. | Envío puntual ≤ 08:00; 100% reportes completos; 0 adjuntos vacíos |
| TC-AVL-03 | RNF-01. Latencia | Valores límite | Carga: 50, 100, 200 req/min (bordes de capacidad) | Ejecutar prueba de carga, y medir p95. | Servicio estable sin errores. | p95 ≤ 800 ms (50), ≤ 1.2 s (100), ≤ 2 s (200); error rate <1% |
| TC-AVL-04 | RF-12. Anti-spam/cooldown | Valores límite | Repetición de eventos a 119 s / 120 s / 121 s | Disparar alertas repetidas cerca del límite. | Rechaza duplicados <120 s, permite a ≥120 s, registra evento | 0 duplicados en la ventana; ≥98% entregas válidas. |

### Tablas de decisión

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Requisito** | **Técnica aplicada** | **Entradas / condiciones** | **Pasos** | **Resultado esperado** | **Criterio de aceptación / Métrica** |
| TC-TD-01 | RF-13. Registro y clasificación de quejas | Tablas de decisión | Ruido, humo, tráfico, con su gravedad de alta, media, baja. | Ver asignaciones de quejas. | Ruta correcta (industria/autoridad/unidad) y prioridad acorde. | 100% coincidencia con reglas, con 0 contradicciones. |
| TC-TD-02 | RF-10/11. Reglas de alerta + envío | Tablas de decisión | Predicción {≥ umbral / < umbral} × Alerta {activa / inactiva} × Canal {email | Evaluar combinaciones, y ejecutar workflow n8n. | Envía solo si (≥ umbral ∧ activa); canal según regla; registra estado. | 100% cumplimiento de lógica, MTTA ≤ 2 min. |
| TC-TD-03 | RF-01. Login por rol (seguridad básica) | Tablas de decisión | Usuario {válido/ inválido} × Estado {activo/ bloqueado} × intento | Ver respuestas (código y mensajes), al iniciar sesión. | Acceso solo para válido+activo; bloqueado= rechazo. | 100% rechazo de intentos maliciosos; 0 críticos OWASP básicos. |
| TC-TD-04 | RF-07. Dashboards por rol | Tablas de decisión | Rol × widget {permitir/denegar} | Evaluar matriz de permisos, para acceder a cada recurso | Accesos permitidos/denegados según registros. | 100% coincidencia con matriz, 0 accesos indebidos. |

# Actividades de control y seguimiento

## Inspección

Detectar tempranamente defectos en artefactos clave mediante un proceso formal y estructurado (similar a Fagan), planeación, preparación individual, reunión de inspección, corrección y seguimiento.

* **Inicio de Sesión**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actividad** | **Responsable** | **Comentarios** | **Estado** |
| Revisión de diseño de interfaz. | Frontend | Verificar diseño de campos de entrada. | ✅ |
| Revisión de flujo de login. | Frontend/QA | Verificar si los mensajes de error son claros. | Flecha circular contorno |
| Validación de datos de entrada. | Frontend/Backend | Asegurar validación en frontend y backend. | Flecha circular contorno |
| Verificación de seguridad. | Backend | Comprobar cifrado de contraseñas. | Flecha circular contorno |

## Revisión

Verificar que artefactos y cambios cumplen especificaciones y estándares antes de pasar a pruebas. En tu trabajo previo lo aterrizaste con tablas por HU (validación de RF, UI/UX, lógica y unit tests).

|  |  |  |  |
| --- | --- | --- | --- |
| **Actividad** | **Responsable** | **Comentarios** | **Estado** |
| Validación de requisitos funcionales. | QA/Frontend | Revisar que el login cumpla con todos los requisitos funcionales. | ✅ |
| Evaluación de diseño de UI/UX. | Frontend/QA | Verificar la usabilidad de los campos de entrada. | ✅ |
| Revisión de código y lógica de autenticación. | Backend | Asegurarse de que la lógica de autenticación está correcta. | Flecha circular contorno |
| Verificación de seguridad. | Backend | Verificar cobertura de pruebas unitarias para la función de login. | Flecha circular contorno |

## Auditoria

Verificar la conformidad del SGC del proyecto con los requisitos de la norma ISO 9001:2015, los procedimientos internos de desarrollo y pruebas, y los criterios establecidos en el Plan SQA. La auditoría interna permite evaluar la eficacia del sistema, detectar desviaciones, proponer acciones correctivas y demostrar mejora continua.

Incluye la revisión de procesos de planificación, desarrollo, pruebas, control documental, gestión de riesgos y resultados de los indicadores de calidad (p95 de latencia, MTTA, cobertura de casos, densidad de defectos).

* Entradas (criterios de auditoría).
  + Manual y política de calidad.
  + Plan SQA y Test Plan.
  + Requisitos RF/RNF y HU.
  + Resultados de pruebas y reportes de defectos.
  + Métricas e indicadores.
* Salidas (productos de auditoría).
  + Informe de auditoría con hallazgos y no conformidades.
  + Plan de acciones correctivas y preventivas (AC/AP).
  + Registro de seguimiento y verificación de cierre.

|  |  |  |
| --- | --- | --- |
| **Etapa** | **Actividades** | **Artefactos** |
| Planeación | Definir objetivos, alcance, criterios y frecuencia. | Plan de auditoría interna (ISO 9001 9.2.2). |
| Preparación | Revisión documental; selección de muestra y checklist ISO 9001. | Lista de verificación ISO 9001 adaptada a SQA. |
| Ejecución | Entrevistas, revisión de artefactos, registro de hallazgos y NC. | Notas de auditoría, registro de evidencias (fotos, logs). |
| Informe | Clasificar hallazgos (mayor, menor, observación), emitir reporte a la dirección. | Informe de auditoría (ISO 9001 9.2.2). |
| Acciones correctivas | Asignar AC/AP con responsables y plazos; registrar en Kanban. | Plan de AC/AP (ISO 9001 10.2). |
| Seguimiento (follow up) | Verificar implementación y eficacia de AC, y emitir acta de cierre. | Registro de seguimiento (ISO 9001, 9.2.2). |

### Lista de verificación de auditorias

|  |  |  |  |
| --- | --- | --- | --- |
| **Cláusula ISO 9001** | **Pregunta de verificación** | **Evidencia esperada** | **Estado** |
| 7.5 Información documentada | ¿Están controladas las versiones de artefactos (SQA, Test Plan, HU)? | Registro de control de documentos o Git. | ✅ |
| 8.5.1 Control de producción y servicio | ¿Los builds en staging siguen el procedimiento planificado? | Logs de CI/CD, pipeline documentado. | Flecha circular contorno |
| 9.1 Seguimiento y medición | ¿Se registran métricas de p95, pass/fail? | Dashboard QA, reportes. | Flecha circular contorno |
| 9.2 Auditoría interna | ¿Existe planificación y evidencias de auditorías previas? | Plan y actas de auditoría internas. | Flecha circular contorno |
| 9.3 Revisión por la dirección | ¿Se realizan revisiones por la dirección cada release? | Actas de revisión, decisiones y acciones. | Flecha circular contorno |
| 10.2 Acción correctiva | ¿Se registran NC y acciones con fechas de cierre y verificación? | Plan AC/AP, evidencia de cierre. | Flecha circular contorno |

### Riesgos y mitigación

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Riesgo** | **Probabilidad** | **Impacto** | **Mitigación** | **Contingencia** |
| R-01 | Fallo de API externa clave. | Media | Alto | Alertas de últimos datos válidos. | Cambiar a fuente alterna. |
| R-02 | Retraso en casos de prueba. | Media | Medio | Dividir por módulos. | Reducir alcance a HU críticas (smoke). |
| R-03 | Vulnerabilidades básicas en login. | Baja | Alto | Revisión de seguridad y pruebas. | Bloqueo de credenciales. |
| R-04 | Reporte semanal no se genera a tiempo. | Media | Medio | Job con reintentos, monitoreo y alerta. | Envío manual de contingencia |
| R-05 | Overfitting en horizontes largos (72h/168h) | Medio | Alto | Regularización, early stopping, reducción de features | Fallback automático a modelo baseline |
| R-06 | Data leakage temporal | Media | Crítico | Strict time split, revisar dataset al cargar | Invalidar el dataset utilizado, regenerarlo desde datos crudos |
| R-07 | Fallo de APIs meteorológicas | Alta | Alto | Fallback a último valor válido | Suspender generación de predicción, activar predicción basada en persistencia |

# Pruebas de desarrollo continuo

## Test driven development

El enfoque de Test-Driven Development (TDD) se aplicó en los componentes críticos del sistema, principalmente en los módulos de autenticación y validación de formularios del proyecto.

Esta práctica consiste en escribir primero los casos de prueba unitarios antes del desarrollo del código funcional, permitiendo que las pruebas guíen el diseño y la implementación.

**Red**: se define un caso de prueba para una nueva funcionalidad, en este caso, validar el formato del correo electrónico o la longitud de una contraseña. Al ejecutarlo, la prueba falla porque la función aún no existe.

**Green**: se implementa el código mínimo necesario para que la prueba pase correctamente, asegurando que la funcionalidad cumpla con los criterios de aceptación definidos.

**Refactorizar**: se mejora el código sin alterar su comportamiento, garantizando legibilidad, eficiencia y mantenimiento.

## Acceptance test driven development

**CU-01. Autenticación de usuario**

**Actor principal:** Usuario (Industria, Ciudadano, Autoridad, Administrador)  
**Descripción:** Permite al usuario crear una cuenta e iniciar sesión en la plataforma, validando credenciales y rol asignado.  
**Flujo principal:**

1. El usuario accede a la pantalla de login o registro.
2. Completa los campos requeridos (correo y contraseña).
3. El sistema valida la información y autentica al usuario.
4. Redirige a la interfaz correspondiente según su rol.  
   **Postcondición:** El usuario accede al sistema con permisos acordes a su perfil.  
   **Flujos alternos:**  
   – Si las credenciales son inválidas, se muestra mensaje de error.  
   – Si el usuario está bloqueado o no verificado, se notifica el estado.

**CU-02. Gestión de perfiles y permisos**

**Actor principal:** Administrador  
**Descripción:** Administra roles, permisos y accesos diferenciados según tipo de usuario.  
**Flujo principal:**

1. El administrador selecciona un usuario del listado.
2. Modifica o asigna rol (ciudadano, industria, autoridad).
3. El sistema actualiza los privilegios y registra el cambio.  
   **Postcondición:** Los permisos quedan actualizados y reflejados en la sesión siguiente.

**CU-03. Consulta de mapa urbano con pronóstico ambiental**

**Actor principal:** Inspectora Municipal / Autoridad Ambiental.  
**Descripción:** Visualiza el mapa con niveles actuales y predicciones de contaminación por zona.  
**Flujo principal:**

1. El usuario ingresa al módulo Mapa Ambiental.
2. El sistema carga las zonas registradas con **RiesgoActual** y **RiesgoPronosticado 24 h**.
3. Se muestran categorías de riesgo (verde, amarillo/naranja, rojo) según umbrales oficiales.
4. El usuario puede aplicar filtros (zona, contaminante, rango horario).
5. Al seleccionar una zona, se despliega un panel lateral con detalle de valores, hora y fuente.  
   **Postcondición:** Se muestra información actualizada del nivel de riesgo por zona.  
   **Criterios de aceptación:**  
   – Accesibilidad WCAG nivel AA.  
   – Tooltips navegables con teclado.  
   – Actualización dinámica del mapa al cambiar filtros.

**CU-04. Configuración de reglas de alerta preventiva**

**Actor principal:** Encargado Ambiental (Industria)  
**Descripción:** Define umbrales y condiciones para recibir alertas cuando las predicciones superan límites críticos.  
**Flujo principal:**

1. El encargado accede al módulo “Alertas preventivas”.
2. Crea una nueva regla seleccionando contaminante, zona, umbral y canal (email/in-app/SMS).
3. El sistema valida los datos (umbral numérico > 0, canal confirmado).
4. Registra la regla con estado **Activa**.  
   **Flujos alternos:**  
   – Si el correo no está verificado, se requiere confirmación.  
   – Si ya existe una regla duplicada, el sistema rechaza la creación.  
   **Postcondición:** La regla queda disponible para el monitoreo automático.

**CU-05. Emisión automática de alertas**

**Actor principal:** Sistema / Servicio de Monitoreo de Alertas  
**Descripción:** Evalúa continuamente las predicciones y emite alertas preventivas cuando se superan umbrales definidos.  
**Flujo principal:**

1. El servicio evalúa predicciones ambientales por zona.
2. Si un valor excede el umbral, genera el evento **AlertaEmitida**.
3. El sistema registra el evento en **LogDeAlertas** con fecha, valor y canal.
4. Envía la notificación al destinatario configurado.  
   **Flujo alterno:**  
   – Si existe una alerta similar en las últimas 2 h (ventana cooldown), no se emite nuevamente.  
   **Postcondición:** El evento queda registrado y visible en el historial.

**CU-06. Pausar o reactivar reglas de alerta**

**Actor principal:** Encargado Ambiental  
**Descripción:** Permite pausar temporalmente una regla sin eliminarla.  
**Flujo principal:**

1. El encargado accede a sus reglas activas.
2. Selecciona la regla y elige “Pausar”.
3. El sistema actualiza el estado a **Pausada**.
4. En cualquier momento puede reactivarla.  
   **Postcondición:** La regla no emite alertas mientras esté pausada.

**CU-07. Panel de control industrial**

**Actor principal:** Encargado Ambiental  
**Descripción:** Visualiza métricas de cumplimiento y riesgos ambientales de su industria.  
**Flujo principal:**

1. El usuario accede al panel.
2. El sistema muestra KPIs de cumplimiento, tendencias y eventos recientes.
3. El usuario puede filtrar por fecha, contaminante o zona.  
   **Postcondición:** Se obtiene una vista consolidada del rendimiento ambiental de la empresa.

**CU-08. Consulta ciudadana de calidad del aire**

**Actor principal:** Ciudadano  
**Descripción:** Permite consultar el nivel de calidad del aire por barrio y recibir recomendaciones.  
**Flujo principal:**

1. El ciudadano busca su zona en el mapa.
2. El sistema muestra el índice ICA simplificado y recomendaciones.  
   **Postcondición:** El usuario conoce el nivel de exposición y medidas preventivas.

**CU-09. Registro de quejas ciudadanas**

**Actor principal:** Ciudadano  
**Descripción:** Permite reportar incidentes ambientales (humo, ruido, olores).  
**Flujo principal:**

1. El ciudadano completa el formulario con tipo de queja, ubicación y evidencia.
2. El sistema clasifica automáticamente la queja (reglas/IA ligera).
3. Asigna el caso a la autoridad o industria responsable.  
   **Postcondición:** La queja se registra con número de caso y estado **Abierta**.

**CU-10. Seguimiento de quejas**

**Actor principal:** Autoridad Ambiental  
**Descripción:** Gestiona las quejas registradas y actualiza su estado.  
**Flujo principal:**

1. La autoridad revisa los casos abiertos.
2. Evalúa evidencia y ejecuta acciones correctivas.
3. Actualiza el estado a **En gestión** o **Cerrada**.  
   **Postcondición:** El denunciante recibe notificación del cambio de estado.

**CU-11. Generación automática de reportes**

**Actor principal:** Sistema / Autoridad  
**Descripción:** Genera y envía reportes automáticos semanales o mensuales con indicadores clave.  
**Flujo principal:**

1. El job programado ejecuta la generación del reporte.
2. Se compilan métricas (alertas, ICA promedio, zonas críticas).
3. Se genera archivo PDF/Excel y se envía por email a los destinatarios.  
   **Postcondición:** El reporte queda registrado en el historial y archivado para auditoría.

**CU-12. Auditoría y trazabilidad**

**Actor principal:** Auditor Interno  
**Descripción:** Permite consultar registros de alertas, cambios de reglas y actividad de usuarios.  
**Flujo principal:**

1. El auditor filtra por fecha, usuario o tipo de evento.
2. El sistema muestra los registros con detalles antes/después y responsable.  
   **Postcondición:** Se garantiza la trazabilidad completa de las acciones dentro del sistema.

## Behaviour driven development

**Funcionalidad #1 – Autenticación de usuarios**

Característica: Acceso seguro y validado por rol

Actor: Usuario (ciudadano, industria, autoridad, administrador)

Escenario 1 – Inicio de sesión exitoso

*Dado que el usuario se encuentra en la pantalla de inicio de sesión*

*Y ha ingresado un correo y contraseña válidos*

*Cuando presiona el botón “Iniciar sesión”*

*Entonces el sistema autentica sus credenciales*

*Y redirige al panel correspondiente según su rol asignado.*

Escenario 2 – Credenciales inválidas

*Dado que el usuario ingresa un correo o contraseña incorrectos*

*Cuando intenta iniciar sesión*

*Entonces el sistema muestra el mensaje “Credenciales inválidas”*

*Y permanece en la pantalla de autenticación.*

**Funcionalidad #2 – Gestión de perfiles y permisos**

Característica: Administración de roles de usuario

Actor: Administrador del sistema

Escenario – Actualizar rol de usuario

*Dado que el administrador accede al módulo de gestión de usuarios*

*Y selecciona un usuario existente*

*Cuando asigna un nuevo rol y guarda los cambios*

*Entonces el sistema actualiza los permisos asociados*

*Y registra el cambio en la bitácora de auditoría.*

**Funcionalidad #3 – Mapa urbano con pronóstico ambiental**

Característica: Visualización de niveles actuales y pronosticados de contaminación

Actor: Inspectora municipal / Autoridad ambiental

Escenario – Consulta de niveles pronosticados

*Dado que existen zonas registradas con valores de riesgo actual y pronosticado*

*Y la fuente de datos es válida y actualizada*

*Cuando la inspectora consulta el mapa ambiental*

*Entonces cada zona muestra su nivel de riesgo actual y pronosticado a 24 h*

*Y el color de la zona corresponde a su categoría según los umbrales oficiales.*

Escenario – Filtro temporal y actualización dinámica

*Dado que la inspectora se encuentra en el mapa ambiental*

*Cuando selecciona una hora específica dentro del rango de 0 a 24 horas*

*Entonces el sistema actualiza los valores visualizados*

*Y consulta la memoria local de pronósticos para refrescar los datos.*

**Funcionalidad #4 – Configuración de alertas preventivas**

Característica: Creación de reglas de alerta ambiental

Actor: Encargado ambiental de una industria

Escenario – Crear una regla de alerta válida

*Dado que el encargado ambiental accede al módulo de alertas preventivas*

*Y completa todos los campos requeridos con valores válidos*

*Cuando guarda la regla de alerta*

*Entonces el sistema valida los datos*

*Y registra la regla con estado “Activa”.*

Escenario – Intento de duplicar regla existente

*Dado que ya existe una regla de alerta para el mismo contaminante y zona*

*Cuando el encargado intenta registrar una nueva regla idéntica*

*Entonces el sistema rechaza la operación*

*Y muestra el mensaje “Regla duplicada no permitida”.*

**Funcionalidad #5 – Emisión automática de alertas**

Característica: Generación y notificación de alertas preventivas

Actor: Sistema de monitoreo

Escenario – Generar alerta cuando se supera el umbral

*Dado que existe una regla activa con umbral configurado*

*Y el sistema recibe predicciones ambientales superiores al límite*

*Cuando el servicio de monitoreo ejecuta la evaluación*

*Entonces se genera el evento de dominio “AlertaEmitida”*

*Y se registra en el historial de alertas con hora, valor y canal utilizado.*

Escenario – Evitar alerta duplicada en ventana de enfriamiento

*Dado que ya se emitió una alerta del mismo tipo en las últimas 2 horas*

*Cuando el sistema detecta nuevamente el mismo evento*

*Entonces no genera una nueva alerta*

*Y registra la acción en el log de auditoría como “Bloqueada por cooldown”.*

**Funcionalidad #6 – Pausar o reactivar reglas**

Característica: Gestión del estado de las reglas de alerta

Actor: Encargado ambiental

Escenario – Pausar regla activa

*Dado que el encargado visualiza su lista de reglas activas*

*Cuando selecciona una regla y elige la opción “Pausar”*

*Entonces el sistema cambia el estado de la regla a “Pausada”*

*Y esta deja de generar alertas mientras dure la pausa.*

Escenario – Reactivar regla pausada

*Dado que existe una regla en estado “Pausada”*

*Cuando el encargado selecciona la opción “Reactivar”*

*Entonces el sistema la cambia a estado “Activa”*

*Y reanuda su evaluación normal en el monitoreo.*

**Funcionalidad #7 – Panel de control industrial**

Característica: Visualización de métricas y cumplimiento ambiental

Actor: Encargado ambiental

Escenario – Consultar indicadores de cumplimiento

*Dado que el usuario ha iniciado sesión como encargado ambiental*

*Cuando accede a su panel de control*

*Entonces el sistema muestra indicadores de cumplimiento, tendencias y riesgos*

*Y permite filtrar los datos por fecha, contaminante y zona.*

**Funcionalidad #8 – Consulta ciudadana**

Característica: Información pública de calidad del aire

Actor: Ciudadano

Escenario – Consultar calidad del aire por barrio

*Dado que el ciudadano accede a la aplicación web*

*Y selecciona su zona o barrio*

*Cuando consulta la calidad del aire*

*Entonces el sistema muestra el índice ICA simplificado*

*Y presenta recomendaciones de salud según el nivel obtenido.*

**Funcionalidad #9 – Registro de quejas ciudadanas**

Característica: Recepción y clasificación automática de denuncias

Actor: Ciudadano

Escenario – Registrar una nueva queja

*Dado que el ciudadano completa el formulario de queja con datos válidos*

*Y adjunta una evidencia (imagen o enlace)*

*Cuando presiona el botón “Enviar”*

*Entonces el sistema registra la queja*

*Y asigna automáticamente una categoría y responsable según reglas predefinidas.*

Escenario – Validar campos obligatorios

*Dado que el ciudadano deja campos vacíos en el formulario*

*Cuando intenta enviar la queja*

*Entonces el sistema muestra un mensaje de error*

*Indicando los campos que deben completarse.*

**Funcionalidad #10 – Seguimiento de quejas**

Característica: Actualización y cierre de casos ambientales

Actor: Autoridad ambiental

Escenario – Cierre de queja atendida

*Dado que una queja está en estado “En gestión”*

*Cuando la autoridad finaliza las acciones correspondientes*

*Entonces actualiza el estado a “Cerrada”*

*Y el ciudadano recibe una notificación del resultado.*

**Funcionalidad #11 – Reporte automático**

Característica: Generación y envío de reportes programados

Actor: Sistema / Autoridad

Escenario – Generar reporte semanal

*Dado que el job de reportes está programado para los lunes a las 08:00*

*Y los datos de la semana anterior están disponibles*

*Cuando se ejecuta el proceso automático*

*Entonces el sistema genera un reporte en formato PDF*

*Y lo envía por correo electrónico a las autoridades correspondientes.*

**Funcionalidad #12 – Auditoría y trazabilidad**

Característica: Consulta de logs y cambios del sistema

Actor: Auditor interno

Escenario – Revisar historial de modificaciones

*Dado que el auditor accede al módulo de auditoría*

*Y selecciona un rango de fechas y tipo de evento*

*Cuando ejecuta la búsqueda*

*Entonces el sistema muestra los registros con detalles de usuario, acción y hora*

*Y permite exportar los resultados en formato CSV.*

## Domain driven development

* **Entidad Principal: ReglaDeAlerta**

**Descripción:**  
Representa la configuración establecida por el encargado ambiental para vigilar un contaminante dentro de una zona específica y generar alertas cuando las predicciones superen los umbrales definidos. Está asociada al usuario del sistema (Encargado Ambiental) y coordina la emisión de alertas a través del servicio de dominio.

**Atributos Clave:**

* **zona:** área geográfica monitoreada.
* **contaminante:** indicador ambiental supervisado.
* **umbralPermitido:** valor límite que define riesgo alto.
* **canalNotificacion:** medio de envío de alerta (Email, In-App, SMS).
* **estado:** activa, pausada o silenciada.
* **ventanaEvaluacion:** intervalo temporal de monitoreo.
* **fechaUltimaAlerta:** registro de la última alerta emitida.

**Reglas de Negocio Asociadas:**

1. Una **ReglaDeAlerta** no puede duplicarse para el mismo usuario, zona y contaminante.
2. Se genera una alerta sólo cuando el valor pronosticado supera el **umbralPermitido** dentro de la **ventanaEvaluacion**.
3. No se emite una nueva alerta del mismo tipo antes de 2 horas (anti-spam).
4. Cada regla requiere al menos un canal de notificación válido; si es correo, debe estar confirmado.
5. Las reglas **pausadas** conservan configuración pero no pueden emitir alertas.

* **Entidad: AlertaPreventiva**

**Descripción:**  
Representa un evento de dominio generado cuando una **ReglaDeAlerta** se cumple, indicando que una condición ambiental superó su umbral y se envió una notificación al usuario correspondiente.

**Atributos Clave:**

* **id:** identificador único del evento.
* **reglaAsociada:** referencia a la regla que originó la alerta.
* **valorDetectado:** nivel de contaminante registrado.
* **fechaEmision:** hora y fecha de la generación.
* **canal:** medio utilizado (email, SMS, notificación interna).
* **estado:** enviada, fallida o bloqueada (por cooldown).

**Reglas de Negocio Asociadas:**

1. Cada alerta debe vincularse a una única **ReglaDeAlerta**.
2. Si el mismo tipo de alerta fue emitido en menos de 2 horas, el sistema registra el evento pero no lo reenvía.
3. Las alertas deben registrarse con sello de tiempo y datos de fuente.
4. Toda alerta genera un registro en el **LogDeAlertas** para trazabilidad.

* **Entidad: PrediccionAmbiental**

**Descripción:**  
Almacena los valores pronosticados de contaminantes por zona y hora, obtenidos de fuentes externas o modelos de machine learning.

**Atributos Clave:**

* **zona:** referencia geográfica donde aplica el pronóstico.
* **contaminante:** tipo de sustancia monitoreada.
* **valorPronosticado:** cantidad o índice estimado.
* **fechaHora:** momento de la predicción.
* **fuenteDatos:** API o modelo que la originó.

**Reglas de Negocio Asociadas:**

1. Cada predicción pertenece a una **Zona** y a un **Contaminante**.
2. Debe contener información de fuente, fecha y hora.
3. El sistema sólo evalúa predicciones dentro de la **ventanaEvaluacion** activa.
4. Las predicciones expiradas se conservan con fines estadísticos, pero no generan alertas.

* **Entidad: Zona**

**Descripción:**  
Define las áreas geográficas del municipio de Santa Cruz de la Sierra utilizadas para la representación de riesgo ambiental, clasificación y localización de alertas y quejas.

**Atributos Clave:**

* **idZona:** identificador único.
* **nombre:** denominación del distrito o barrio.
* **coordenadas:** límites geográficos o polígonos del mapa.
* **categoriaRiesgo:** bajo, moderado o alto.
* **contaminantePrincipal:** sustancia predominante en el área.

**Reglas de Negocio Asociadas:**

1. Cada zona debe tener una categoría de riesgo calculada según los umbrales oficiales.
2. Las zonas se muestran en el mapa con colores estandarizados (verde, amarillo, rojo).
3. Las actualizaciones del mapa deben reflejar los últimos valores pronosticados.

* **Entidad: Usuario**

**Descripción:**  
Representa a cualquier persona que interactúa con el sistema, ya sea como ciudadano, autoridad, encargado ambiental o administrador.

**Atributos Clave:**

* **idUsuario:** identificador único.
* **nombreCompleto:** datos personales básicos.
* **correoElectronico:** dirección validada.
* **rol:** {ciudadano, industria, autoridad, administrador}.
* **estadoCuenta:** activo, bloqueado, pendiente de verificación.

**Reglas de Negocio Asociadas:**

1. El **rol** define los permisos y vistas disponibles.
2. Las cuentas inactivas o no verificadas no pueden acceder a funciones críticas.
3. Cada usuario puede tener múltiples **ReglasDeAlerta** y registros de actividad.

* **Entidad: CanalDeNotificacion**

**Descripción:**  
Especifica los medios a través de los cuales el sistema puede enviar alertas o mensajes informativos.

**Atributos Clave:**

* **tipo:** email, SMS, in-app o webhook.
* **destinatario:** dirección o número de destino.
* **confirmado:** estado de validación.
* **preferenciaHorario:** rango horario permitido para envío.

**Reglas de Negocio Asociadas:**

1. Cada canal debe estar confirmado antes de ser utilizado.
2. Las notificaciones se respetan según preferencia y horario de usuario.
3. Si un canal falla, el sistema intenta con el siguiente disponible.

* **Entidad: QuejaCiudadana**

**Descripción:**  
Registra los reportes realizados por ciudadanos sobre incidentes ambientales (humo, ruido, olores, tráfico).

**Atributos Clave:**

* **idQueja:** identificador del reporte.
* **tipo:** categoría (ruido, humo, olores, etc.).
* **descripcion:** texto y detalles del evento.
* **ubicacion:** coordenadas o zona asociada.
* **evidencia:** enlace o imagen de soporte.
* **estado:** abierta, en gestión, cerrada.
* **fechaRegistro:** día y hora de creación.

**Reglas de Negocio Asociadas:**

1. Todos los campos obligatorios deben completarse antes del envío.
2. La clasificación automática asigna tipo y responsable según reglas o IA.
3. Las quejas se pueden reabrir solo si la autoridad lo determina.
4. Cada queja debe estar vinculada a una **Zona**.

* **Entidad: ReporteAmbiental**

**Descripción:**  
Documento generado automáticamente (PDF o Excel) que resume las métricas ambientales, alertas y cumplimiento semanal o mensual.

**Atributos Clave:**

* **idReporte:** identificador del archivo.
* **periodo:** rango temporal (semanal o mensual).
* **fechaGeneracion:** sello de tiempo.
* **destinatarios:** lista de correos o roles.
* **estadoEnvio:** entregado o pendiente.

**Reglas de Negocio Asociadas:**

1. Los reportes se generan automáticamente según cronograma (lunes 08:00).
2. Deben contener indicadores validados (ICA, alertas, zonas críticas).
3. El envío fallido debe registrarse y reintentarse en el siguiente ciclo.

* **Entidad: AuditoriaSistema**

**Descripción:**  
Almacena todas las acciones realizadas en el sistema para asegurar trazabilidad, transparencia y control de calidad.

**Atributos Clave:**

* **idEvento:** identificador de registro.
* **usuario:** responsable de la acción.
* **accion:** tipo de operación (crear, editar, eliminar, iniciar sesión, generar alerta).
* **fechaHora:** registro temporal.
* **resultado:** éxito o fallo.

**Reglas de Negocio Asociadas:**

1. Toda acción relevante del sistema debe generar un registro de auditoría.
2. Los registros deben conservarse con integridad y orden cronológico.
3. Solo los administradores y auditores pueden acceder al historial completo.

# Automatización y Evidencia

## Versionamiento (Github)

Para el control de versiones y la gestión colaborativa del código fuente del proyecto **Integrador Urbano**, se utiliza la plataforma **GitHub**, donde se mantiene el repositorio público del sistema con su estructura completa y actualizada.

El repositorio se encuentra organizado bajo la rama principal **main**, la cual centraliza las actualizaciones validadas del proyecto. La estructura base incluye los directorios y archivos fundamentales para la implementación del sistema, tales como **src/**, **public/**, y archivos de configuración como **package.json**, **vite.config.ts**, **tsconfig.json** y **index.html**, que definen la arquitectura de compilación, dependencias y despliegue.

Cada commit está asociado al autor responsable (por ejemplo, *OV20408*, *JA*, *PO*, *MM*) y describe brevemente la tarea realizada, siguiendo un formato uniforme de mensaje (por ejemplo, *“maquetación de proyecto”*, *“implementación de dashboard”*, *“configuración de backend”*). Esto permite mantener trazabilidad entre las actividades del tablero **Trello** y las entregas de código, fortaleciendo la transparencia del flujo de trabajo.

Las pruebas unitarias y flujos de integración continua se ejecutan mediante GitHub Actions, garantizando la validación automática tras cada commit realizado y evidenciado.

1. **Razonamiento de elección**

La elección de **GitHub** como herramienta de versionamiento se fundamenta en las siguientes ventajas:

1. **Control de versiones distribuido:** permite que cada miembro del equipo mantenga una copia local del proyecto, facilitando el trabajo paralelo y la integración continua.
2. **Colaboración y trazabilidad:** las ramas, commits y pull requests documentan de manera detallada las contribuciones y cambios realizados por cada integrante, alineándose con las buenas prácticas de SQA.
3. **Integración con CI/CD:** se vincula directamente con **GitHub Actions**, permitiendo automatizar pruebas unitarias, despliegues y validaciones al momento de cada *push*.
4. **Historial de auditoría:** cada modificación queda registrada con autor, hora, descripción y cambios específicos, lo cual facilita auditorías internas y revisiones del código fuente.
5. **Accesibilidad y transparencia:** al tratarse de un repositorio público, facilita la revisión externa y la demostración del avance técnico del proyecto, garantizando cumplimiento con los principios de calidad y mejora continua.

A screenshot of a computer

AI-generated content may be incorrect.

## Trello

Para la organización y seguimiento de las actividades del proyecto **Integrador Urbano**, se emplea la herramienta **Trello**, configurada bajo el enfoque ágil **Kanban**, con un tablero visual que permite gestionar las tareas de manera colaborativa, transparente y ordenada.

* **Estructura del tablero**

El tablero se compone de tres columnas principales que representan los estados del flujo de trabajo:

* **Pendiente:** contiene las tareas planificadas que aún no han iniciado. Ejemplos: *Investigar modelos de machine learning (MLs)*, *Desarrollo de Backend*, *Creación del workflow de N8N*.
* **En progreso:** agrupa las tareas actualmente en desarrollo, como *Maquetación del proyecto*, *Implementación de dashboards interactivos*, *Desarrollo de login y registro*, o *Investigación de APIs públicas*.
* **Finalizado:** almacena las tareas completadas y revisadas, garantizando trazabilidad del avance y cumplimiento de entregables.

Cada tarjeta incluye etiquetas de color que indican el tipo de tarea o módulo (por ejemplo, *backend, integración, automatización*), además de la asignación del responsable mediante sus iniciales. Esto facilita la identificación de prioridades, el control del progreso individual y la gestión de la carga de trabajo en equipo.

A screenshot of a computer

AI-generated content may be incorrect.