UNSAdA

UNIVERSIDAD NACIONAL ◆ SAN ANTONIO de ARECO

Programación declarativa

Docente: Emanuel Lazzari

Alumnos: Agustín Moscato, Katherine Molina

Fecha de presentación: 27/11/2025

Título del proyecto: EcoSense

Índice

1. Introducción

Creemos que a día de hoy existe una notable falta de herramientas tecnológicas versátiles y asequibles para simplificar la recolección y el análisis de datos, tanto para los investigadores como para los especialistas en campos variados, por ejemplo la biología marina, las ciencias biológicas, la agroindustria, la geología o la gestión medioambiental.
En estas situaciones, contar con datos precisos y actualizados en tiempo real sobre aspectos como la humedad del suelo, la humedad ambiental o la temperatura es esencial para realizar estudios, crear modelos predictivos y tomar decisiones basadas en evidencias.
Frecuentemente, las tareas de monitoreo o registro de estas variables se llevan a cabo manualmente, rellenando planillas en papel o formularios físicos que posteriormente deben ser trasladados a sistemas digitales. Este proceso, además de ser vulnerable a equivocaciones humanas y a la pérdida de datos, implica un esfuerzo administrativo adicional debido a que es necesario volver a introducir los datos en instrumentos como Excel u otras aplicaciones de análisis.
Así, incluso un procedimiento tan sencillo como hacer anotaciones de campo o tomar mediciones normalizadas puede sufrir las consecuencias del empleo de registros en papel, que tienen la posibilidad de romperse, ensuciarse o mojarse, lo cual obstaculiza la confiabilidad y trazabilidad de los datos recolectados.
En el marco de la asignatura “Programación Declarativa”, se propone el diseño e implementación de ***EcoSense***, una plataforma IoT de monitoreo ambiental, escalable y adaptable a diferentes tipos de recolección de datos, tanto de fuentes automáticas, como sensores físicos conectados a dispositivos IoT, como de fuentes manuales, mediante formularios web o interfaces digitales.

2. Definición y descripción del problema

El problema que aborda ***EcoSense*** es la falta de una herramienta unificada que permita recolectar y gestionar datos ambientales provenientes tanto de sensores IoT como de registros manuales. Actualmente, estos datos suelen quedar dispersos en planillas, aplicaciones aisladas o plataformas que solo admiten una de las dos fuentes, dificultando la trazabilidad, la integridad y el análisis continuo.

En la actualidad, la trazabilidad, la integridad y el análisis continuo se ven obstaculizados porque estos datos tienden a estar dispersos en hojas de cálculo, aplicaciones independientes o plataformas que solo permiten una de las dos fuentes como se mencionó en el párrafo anterior.

Por ello, el sistema tiene que satisfacer condiciones como: aceptar lecturas por medio de HTTP, posibilitar la carga manual a través de formularios, realizar transformaciones y validaciones, guardar datos en una base relacional y facilitar el acceso a los mismos

mediante una API. El proyecto se lleva a cabo con limitaciones, como el empleo de nodos simulados en vez de hardware físico y un alcance restringido a un prototipo operativo creado para evidenciar la viabilidad del método.

La importancia del problema se debe a que los analistas, investigadores y maestros requieren datos confiables, coherentes y centralizados para investigaciones de tipo ambiental o productivo. La propuesta es novedosa ya que fusiona dos fuentes de información disímiles en un único flujo declarativo y emplea Elixir para procesar datos de forma concurrente, escalable y con reglas, lo cual no es frecuente en plataformas IoT académicas o comerciales.

3. Justificación del paradigma o paradigmas declarativos utilizados

Se optó por la programación funcional en Elixir para desarrollar ***EcoSense*** debido a que este paradigma ofrece ventajas específicas para sistemas IOT, los cuales necesitan procesar una gran cantidad de datos en tiempo real. Esta fue una de las razones principales por las que se seleccionó este paradigma de programación. Cada lectura que llega, ya sea desde un sensor físico o desde un formulario web, puede tratarse como un proceso independiente, lo que evita bloqueos y hace posible encolar, ordenar y validar la información sin que el sistema se sobrecargue. Esto vuelve la plataforma mucho más escalable, estable y capaz de soportar picos de actividad.

A eso se suma que la programación funcional permite expresar la lógica del sistema mediante funciones puras, es decir, funciones que no dependen de estados internos ni producen efectos secundarios. Esto hace que las reglas de validación, las transformaciones y las alertas sean más fáciles de entender, probar y mantener, reduciendo muchos de los errores habituales de los enfoques imperativos.

Si lo comparamos con otros paradigmas, la programación imperativa se basa en modificar estados compartidos y ejecutar pasos en un orden específico, lo que complica bastante el trabajo cuando hay muchos sensores enviando datos al mismo tiempo. Por su parte, la orientación a objetos suele mezclar comportamiento y estado, lo que puede generar acoplamientos innecesarios y dificultades para mantener la coherencia en contextos tan paralelos.

Característica	Enfoque imperativo	Enfoque funcional
Enfoque del programador	Cómo realizar tareas (algoritmos) y cómo realizar el seguimiento de cambios de estado.	Información deseada y transformaciones necesarias
Cambios de estado	Importante	Inexistente
Orden de ejecución	Importante	Baja importancia
Control del flujo primario	Bucles, elementos condicionales y llamadas a funciones (métodos).	Llamadas a funciones, incluyendo la recursividad.
Unidad de manipulación primaria	Instancias de estructuras o clases	Funciones como recopilaciones de datos y objetos de primera clase

Por todas estas razones, el paradigma funcional y Elixir se adaptan mejor a las necesidades de ***EcoSense***. Brindan modelos más nítidos, transformaciones declarativas y una concurrencia mucho mayor, lo que permite abordar cada entrada de datos como una unidad independiente del sistema. La plataforma, gracias a eso, tiene la capacidad de funcionar de manera confiable, escalable y robusta, incluso al procesar volúmenes grandes de datos en tiempo real.

4. Estado del arte

Hoy en día existen muchas herramientas para el monitoreo ambiental y la recolección de datos, pero la mayoría presenta limitaciones importantes cuando se necesitan flujos más flexibles o pensados para investigación.

Las plataformas IoT comerciales, entre ellas ThingSpeaK, Blynk IoT y Ubidots, posibilitan la recepción de datos y su visualización en paneles simples; sin embargo, no son compatibles con procesos científicos adaptables y suelen ser costosos.

Por eso, muchos investigadores todavía dependen de cargar sus datos en Google Sheets o en Excel, que aunque son accesibles generan problemas de trazabilidad, fidelidad y alto riesgo de errores al cargar y combinar datos manuales con datos tomados por sensores.

5. Objetivos del trabajo

5.1 Objetivo general

Diseñar e implementar ***EcoSense***, una plataforma declarativa capaz de unificar, procesar y visualizar datos ambientales provenientes tanto de sensores IoT como de registros

manuales, garantizando escalabilidad, trazabilidad y automatización del flujo de información.

5.2 Objetivos específicos

- Implementar un servidor en Elixir/Phoenix que reciba lecturas mediante endpoints dedicados.
- Procesar los datos aplicando validaciones, transformaciones y asignación de metadatos desde un enfoque funcional.
- Diseñar un modelo de base de datos relacional en MySQL para almacenar lecturas históricas, configuraciones y observaciones.
- Simular nodos adicionales mediante Postman o datos hardcodeados para evaluar el comportamiento y la escalabilidad del sistema ante múltiples fuentes de datos.
- Desarrollar un dashboard web básico que permita consultar y visualizar las métricas recolectadas mediante API REST.
- Evaluar el comportamiento del sistema bajo múltiples fuentes de datos concurrentes y analizar su escalabilidad.
- Documentar el diseño declarativo, las decisiones de arquitectura y los resultados experimentales.

Asimismo, se espera que el desarrollo de ***EcoSense*** contribuya a que distintos investigadores, docentes y equipos de trabajo puedan recolectar, almacenar y visualizar información de manera digital y automatizada, reduciendo errores humanos y mejorando la disponibilidad, integridad y trazabilidad de los datos para su posterior análisis.

6. Diseño y documentación del proceso de desarrollo

6.1 Análisis y modelado del problema

Los formularios manuales y los sensores de IoT son las dos fuentes de datos que el sistema tiene que integrar. Los dos producen lecturas ambientales que necesitan ser validadas, normalizadas y almacenadas de manera segura. La evaluación estableció que el flujo general abarca: la recepción de lecturas, la implementación de reglas declarativas, la conversión funcional y la persistencia en una base de datos relacional. Se estructuró el dominio de

manera clara y coherente al modelar entidades clave como nodos, sensores, lecturas, usuarios y observaciones manuales.

6.2 Esquemas de funciones, estructuras de datos y reglas

Se definieron funciones puras para validar y transformar las lecturas entrantes, evitando efectos colaterales y facilitando el razonamiento sobre el flujo. Entre ellas se incluyen:

- Validación de rangos y tipos
- Normalización de unidades
- Asignación de timestamps y metadatos
- Detección de alertas mediante reglas declarativas.

Las estructuras de datos se basan en esquemas de Elixir (Ecto) y corresponden al modelo del DER: nodes, sensors, readings, forms, thresholds y users.

6.3 Decisiones de diseño y justificación

Se eligió Elixir/Phoenix por su capacidad de gestionar varios procesos simultáneamente, lo cual posibilita que cada lectura que ingresa se maneje como un proceso autónomo. Esta propiedad posibilita que la escalabilidad se produzca y previene bloqueos. La sencillez para modelar relaciones y la estabilidad de MySQL fueron las razones por las cuales se eligió esta base de datos. La comunicación con sensores simulados se llevó a cabo utilizando HTTP debido a su facilidad y capacidad de interoperar. Se eligió un enfoque declarativo para los módulos de validación y reglas, porque posibilita expresar la lógica del dominio con claridad, lo cual propicia el testeo y disminuye las equivocaciones.

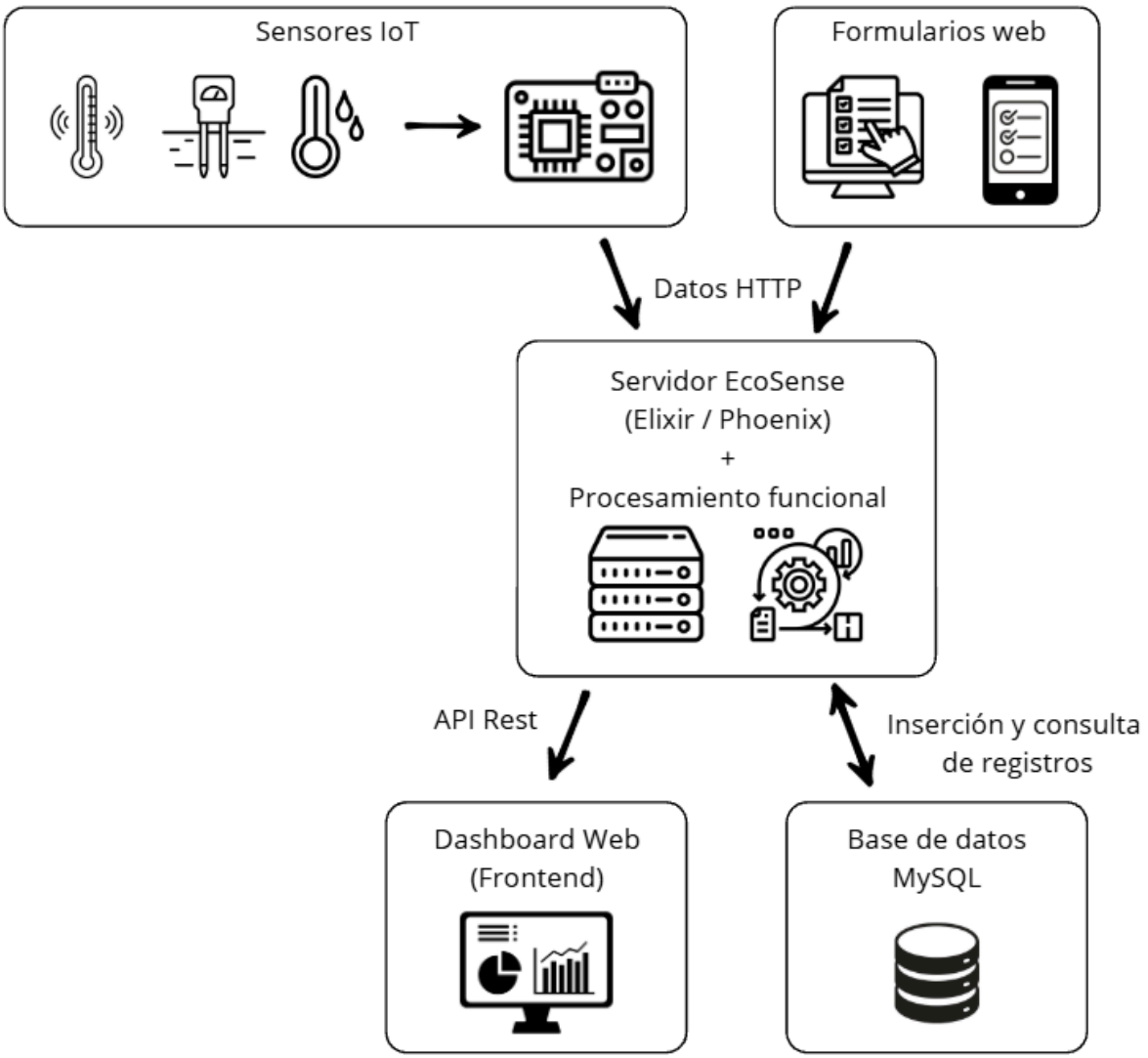
6.4 Prototipos y versiones intermedias

Durante el desarrollo se creó un prototipo incremental:

1. **Primera versión:** servidor Phoenix recibiendo lecturas simuladas con Postman.
2. **Segunda versión:** integración completa con MySQL y pruebas de validación funcional.
3. **Tercera versión:** diseño de un dashboard básico para visualizar los datos mediante consultas a la API.

Estas iteraciones permitieron validar la arquitectura, ajustar la estructura de datos y comprobar la escalabilidad del sistema bajo múltiples eventos simultáneos.

7. Diagramas


Sensores IoT
Formularios web
Datos HTTP
Servidor EcoSense
(Elixir / Phoenix)
+
Procesamiento funcional
API Rest
Inserción y consulta
de registros
Dashboard Web
(Frontend)
Base de datos
MySQL

8. Implementación

9. Conclusiones

10. Anexos

11. Bibliografía

Dar formato:
https://learn.microsoft.com/es-es/dotnet/standard/linq/functional-vs-imperative-programming

Característica	Enfoque imperativo	Enfoque funcional
Enfoque del programador	Cómo realizar tareas (algoritmos) y cómo realizar el seguimiento de cambios de estado.	Información deseada y transformaciones necesarias
Cambios de estado	Importante	Inexistente
Orden de ejecución	Importante	Baja importancia
Control del flujo primario	Bucles, elementos condicionales y llamadas a funciones (métodos).	Llamadas a funciones, incluyendo la recursividad.
Unidad de manipulación primaria	Instancias de estructuras o clases	Funciones como recopilaciones de datos y objetos de primera clase

https://www.cs.us.es/cursos/pd-2022/temas/T1_Introduccion_PD_PF.pdf
tiene muchos links de interés ese pdf que podemos usar para citar en algunas partes del documento

https://www.cs.us.es/~fsancho/Blog/

IEEE Xplore

- Programación funcional para el Internet de las cosas: una pasarela LoRa-MQTT escrita en Elixir

https://ieeexplore.ieee.org/document/10368535

- Diseño de un servidor perimetral basado en Elixir para aplicaciones IoT adaptables

https://ieeexplore.ieee.org/document/10062586

- Evaluación del lenguaje de programación Elixir para IoT

https://ieeexplore.ieee.org/document/7797392

Cuevas, J. L. S., et al. (2023).
Monitoreo de temperatura y humedad ambiental con LoRaWAN.
Revista IPSUMTEC, TecNM.
Disponible en línea: revistas.milpaalta.tecnm.mx

Chanchí-Golondrino, G. E. (2022).
Sistema IoT para la monitorización de variables climatológicas en cultivos.
(Artículo en Redalyc / revistas latinoamericanas sobre agricultura de precisión con IoT).
ResearchGate

Los dos siguientes son dos papers (PDF)

Carrera, E. V. (2016).
Uso de programación declarativa (Ginga-NCL) en aplicaciones interactivas.
Artículo sobre programación declarativa en sistemas interactivos, útil para justificar el paradigma. Dialnet

González García, C. (2017).
MIDGAR: Plataforma para aplicaciones IoT con enfoque en programación declarativa.
(Tesis doctoral).
Universidad de Oviedo.
Trabaja IoT + programación declarativa, muy útil para tu marco teórico. DigiBuo

https://thingspeak.mathworks.com/prices/thingspeak_student precio de thingspeak

https://www.blynk.io/pricing precio de blynk

https://es.ubidots.com/pricing ubidots