Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2025

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE SENSORES AGRÍCOLAS CON PYTHON Y GRAPHVIZ

202400111 – Richard Steven Arizandieta Yol

Resumen

El presente ensayo documenta el desarrollo del Sistema de Agricultura, una aplicación en Python que gestiona información proveniente de estaciones y sensores agrícolas. El tema resulta vigente a nivel internacional debido a la creciente necesidad de optimizar recursos en la agricultura mediante tecnologías de monitoreo y análisis de datos. La novedad del proyecto radica en el uso de estructuras de datos personalizadas, como listas enlazadas y matrices, para organizar y procesar información de manera modular y eficiente.

El sistema adopta una postura técnica orientada a la simplicidad y escalabilidad, permitiendo cargar archivos XML de entrada, generar matrices de frecuencias, reducir datos mediante patrones binarios y exportar resultados en formato XML acompañado de gráficas. Esta aproximación tiene impactos positivos en el plano técnico, al reforzar competencias en programación orientada a objetos; en el ámbito económico, al optimizar la interpretación de datos agrícolas.

Palabras clave
Agricultura, Sensores, Python, XML, Datos.

Abstract

This essay documents the development of the Agriculture System, a Python application designed to manage information from agricultural stations and sensors. The topic is highly relevant at the international level due to the growing need to optimize resources in agriculture through monitoring technologies and data analysis. The novelty of the project lies in the use of customized data structures, such as linked lists and matrices, to organize and process information in a modular and efficient way.

The system adopts a technical approach focused on simplicity and scalability, allowing users to load XML input files, generate frequency matrices, reduce data through binary patterns, and export results in XML format along with graphical representations. This approach has positive impacts in the technical field, by strengthening object-oriented programming skills; in the economic domain, by improving the interpretation of agricultural data.

Keywords
Agriculture, Sensors, Python, XML, Data.

Introducción

Este ensayo documenta el desarrollo del Sistema de Agricultura, un software en Python para gestionar datos de campos agrícolas, estaciones y sensores de suelo y cultivo. Su objetivo es organizar, procesar, reducir y visualizar información agrícola mediante estructuras de datos propias.

El sistema está estructurado en clases orientadas a objetos. SistemaAgricultura funciona como controlador principal: carga datos desde XML, construye estructuras, procesa información y genera salidas en XML y gráficas. Las clases Campo, Estacion, Sensor, Frecuencia, Matriz y ListaEnlazada representan los elementos del dominio agrícola y las estructuras de soporte.

El desarrollo incluyó el diseño de listas enlazadas y matrices personalizadas, ofreciendo mayor control en la manipulación de datos y un enfoque práctico en programación orientada a objetos. El ensayo describe la estructura del código, los algoritmos principales y se apoya en diagramas de clases y actividades para ilustrar la lógica aplicada.

Objetivos del Sistema

- Proporcionar una herramienta que cargue datos desde un archivo XML y los convierta en estructuras manipulables en Python.
- Implementar estructuras de datos personalizadas para manejar información agrícola.
- Facilitar la reducción de matrices, unificando estaciones con patrones de frecuencia semejantes.
- Exportar resultados procesados en un archivo XML de salida.
- Permitir la visualización gráfica de las relaciones entre estaciones y sensores.

Diagrama de clases

Los respectivos diagramas se encuentran adjuntos en la sección de Anexos del documento.

- SistemaAgricultura

Clase principal del sistema. Carga datos desde XML, procesa campos, genera matrices reducidas, exporta resultados y crea gráficas con Graphviz. Centraliza la ejecución y conecta datos con el usuario.

- Campo

Representa un campo agrícola. Contiene estaciones, sensores (suelo y cultivo) y matrices de frecuencias.

- Estacion

Base de monitoreo con id y nombre. Sirve como referencia para relacionar frecuencias de sensores.

- Sensor

Dispositivo de suelo o cultivo. Asociado a frecuencias que registran valores en distintas estaciones.

- Frecuencia

Valor de medición de un sensor en una estación. Incluye id y valor numérico.

Clases de estructuras de datos

- Matriz

Modela la tabla de frecuencias entre estaciones y sensores. Incluye métodos para establecer y obtener valores, mostrar matrices en consola, binarizar los datos, sumar filas, copiar y generar versiones reducidas.

- ListaEnlazada y Nodo

Implementan la estructura de datos básica utilizada en todo el sistema. Cada ListaEnlazada almacena nodos que pueden contener estaciones, sensores, frecuencias o incluso filas de la matriz. Nodo es la unidad mínima que contiene un dato y un puntero al siguiente.

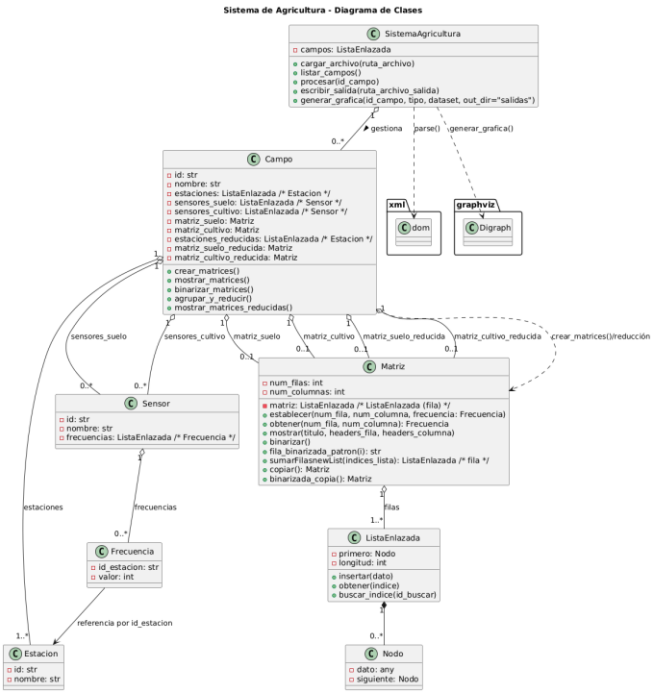

Figura 1. Diagrama de clases para el Sistema de Agricultura

Fuente: elaboración propia.

Carga de archivos

1. Inicio desde el menú

El usuario selecciona "Cargar archivo" en Principal.py, ingresa ruta y nombre, y el programa llama a sistema.cargar_archivo(archivo).

2. Parseo del XML

SistemaAgricultura.cargar_archivo usa xml.dom.minidom.parse(ruta) para leer el documento y obtener nodos <campo>. Si ocurre una excepción, se muestra Error al cargar archivo: {e}.

3. Construcción de objetos

Para cada <campo> se crea un objeto Campo(id, nombre). Dentro: se recorren <estacion> para crear Estacion(id, nombre), y <sensorS> (suelo) y <sensorT> (cultivo); en cada <frecuencia> se crea Frecuencia(idEstacion, valor) que se inserta en el Sensor correspondiente.

4. Creación de matrices de frecuencias

Con estaciones y sensores cargados, se ejecuta campo.crear_matrices() para formar matrices de suelo y cultivo, colocando las frecuencias según estación y sensor.

5. Inserción o reemplazo del campo

Si existe un campo con el mismo ID, se reemplaza; si no, se inserta en self.campos. En ambos casos el sistema informa la acción en consola.

6. Listado y visualización

Al finalizar, el sistema lista los campos disponibles (listar_campos()) y muestra en consola las matrices de cada campo (campo.mostrar_matrices()) con encabezados de estaciones y sensores.

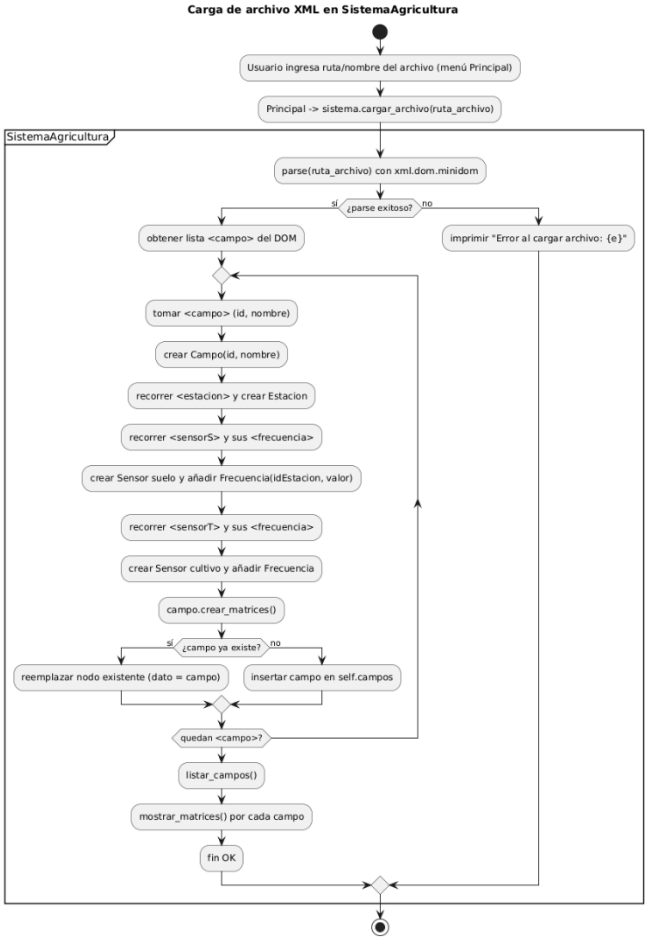

Figura 2. Carga de Archivo XML en Sistema Agricultura

Fuente: elaboración propia.

Agrupar y reducir estaciones

Se agrupan estaciones con el mismo patrón binario de activación de sensores y se construyen matrices reducidas sumando las frecuencias de cada grupo. El

patrón se obtiene concatenando la fila binaria de suelo y la de cultivo con “#”.

1. Cálculo de patrones por estación

Para cada estación se calcula:

- patron_suelo = matriz_suelo.fila_binarizada_patron(idx)
- patron_cultivo = matriz_cultivo.fila_binarizada_patron(idx)

y luego patron = patron_suelo + "#" + patron_cultivo. Luego patron = patron_suelo + “#” + patron_cultivo. La binarización (1 si frecuencia ≠ 0, 0 en otro caso) la hace fila_binarizada_patron, y la concatenación la realiza _patron_por_estacion en Campo.

3. Construcción de grupos

Se recorre la lista de estaciones:

- Si ya existe un grupo con el patrón, se añade el índice.
- Si no, se crea un nuevo grupo con ese patrón.

4. Inicialización de estructuras reducidas

Con cant_grupos se crean: estaciones_reducidas, matriz_suelo_reducida y/o matriz_cultivo_reducida con tantas filas como grupos y el mismo número de columnas que las originales.

5. Población de estructuras reducidas

Para cada grupo:

- Se toma el id de la primera estación y se concatena el nombre de todas para formar nombre_reducido.
- Se inserta Estacion(id_reducido, nombre_reducido) en estaciones_reducidas.
- En matrices reducidas, fila_sum = matriz_original.sumarFilasnewList(indices) y se asigna al grupo. La suma por columna la hace Matriz.sumarFilasnewList.

6. Resultado

Campo contiene estaciones_reducidas con los grupos y matrices reducidas de suelo y/o cultivo con frecuencias consolidadas, listas para mostrar o exportar desde SistemaAgricultura.escribir_salida.

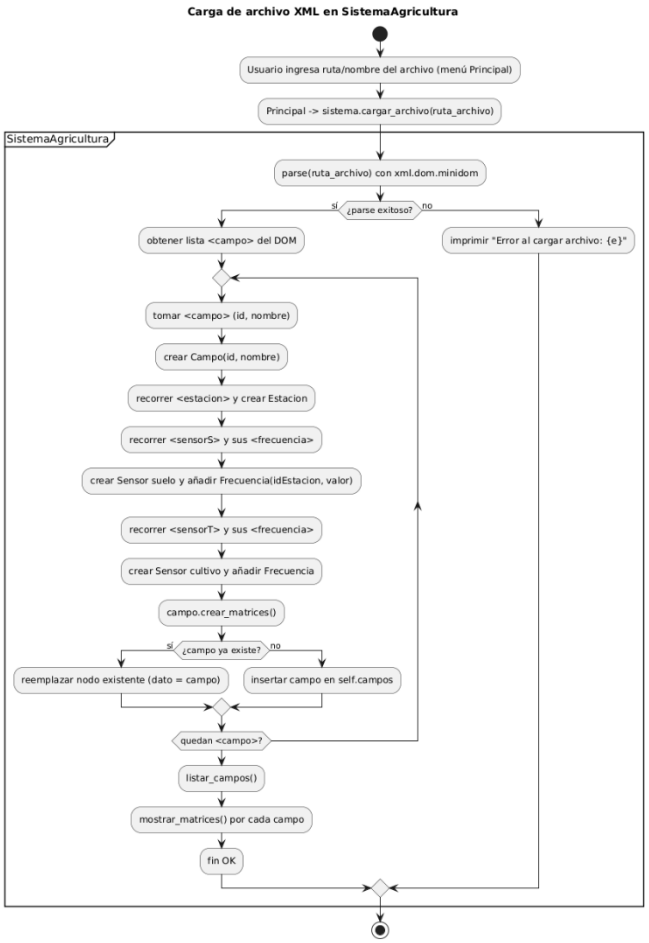

Figura 1. Agrupamiento y Reducción de Estaciones

Fuente: elaboración propia.

Generación de ArchivoSalida.xml

1. Inicialización del documento XML

Se crea un Document() con nodo raíz <camposAgricolas>. Si no hay campos en self.campos, se muestra “No hay campos cargados” y termina el proceso.

2. Asegurar procesamiento previo

Para cada campo: si no tiene matrices reducidas, se ejecuta campo.agrupar_y_reducir() con aviso para generar estaciones_reducidas y matrices de suelo/cultivo.

3. Nodo <campo>

Se crea un nodo <campo> con atributos id y nombre, añadido al root como contenedor de la información reducida.

4. Estaciones reducidas

Si existen, se crea <estacionesBaseReducidas> y dentro un <estacion> por cada agrupación con id y nombre.

5. Sensores de suelo

Si el campo tiene sensores de suelo:

- Se crea <sensoresSuelo>.
- Para cada sensor (columna j), se agrega <sensorS id, nombre>.
- Si hay matriz_suelo_reducida, se recorren filas i y se consulta freq.
- Si freq ≠ 0, se añade <frecuencia idEstacion=...> con el id de la estación reducida y el valor (entero o flotante según corresponda).

6. Sensores de cultivo

El flujo es análogo para <sensoresCultivo> y matriz_cultivo_reducida, generando <sensorT> y sus <frecuencia> solo cuando el valor es diferente de cero.

6. Escritura del archivo

Al finalizar el armado del DOM, se llama a doc.toprettyxml(indent=" ") y se intenta escribir en la ruta solicitada. Si la escritura es exitosa, se informa la ruta; si ocurre una excepción, se imprime un mensaje de error correspondiente.

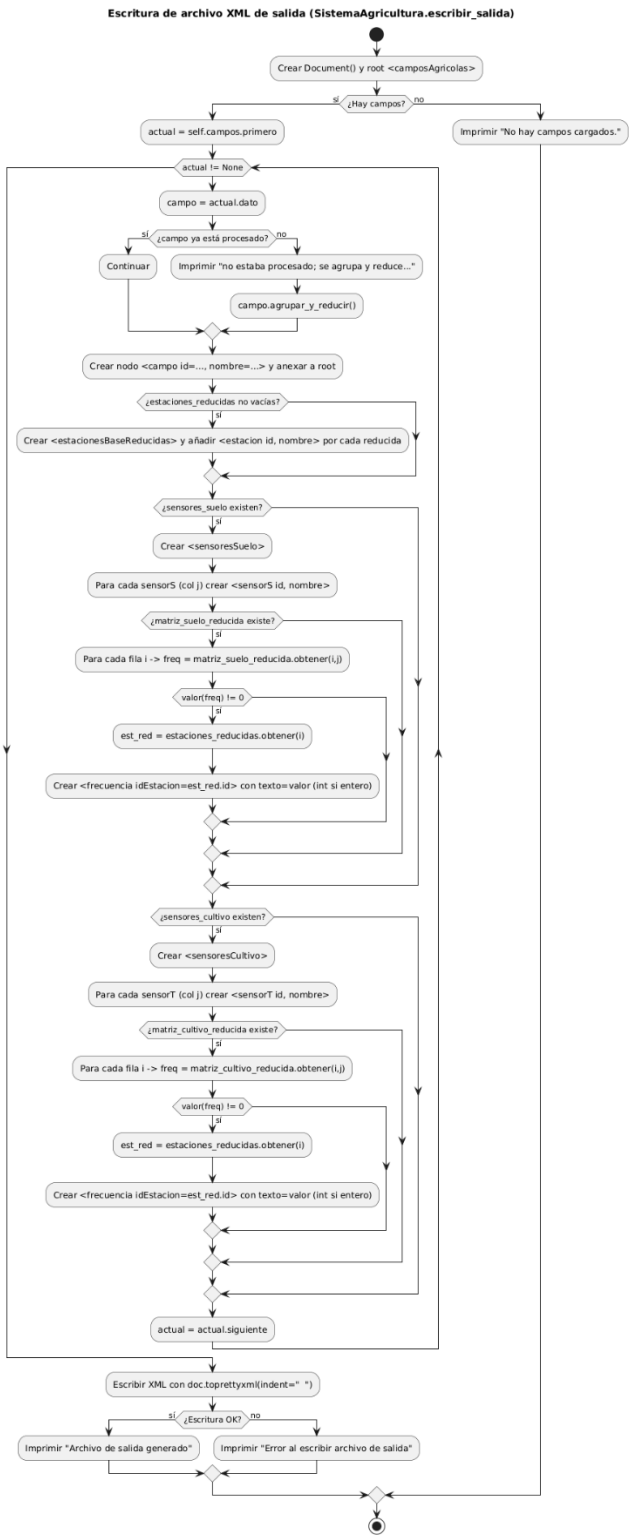

Figura 4. Escritura de archivo XML de Salida

Fuente: elaboración propia.

Conclusiones

El proyecto Sistema de Agricultura mostró la utilidad de la programación orientada a objetos y estructuras de datos personalizadas para gestionar información agrícola. Con listas enlazadas y matrices se organizaron frecuencias de sensores, se redujo la redundancia agrupando estaciones y se presentaron resultados en matrices, archivos XML y gráficas.

La modularidad del diseño, con clases para cada entidad facilitó la comprensión y escalabilidad del sistema. La reducción de datos con patrones binarios resultó eficaz para simplificar la información sin perder valores entre matrices.

Por tanto, concluyo que el sistema logró procesar un archivo XML, generar matrices de frecuencias, reducir información y exportar los resultados requeridos de forma exitosa.

Referencias bibliográficas

- Mark Lutz, (2013). Learning Python. O'Reilly Media, Inc.

- David M. Beazley, (2009). *Python Essential Reference*. Addison-Wesley Professional.

- lliotte Rusty Harold & W. Scott Means, (2004). *XML in a Nutshell*. O'Reilly Media, Inc.

- Emden R. Gansner & Stephen C. North, (2000). *An Open Graph Visualization System and Its Applications to Software Engineering*. AT&T Research.