Informe App2: Gestión Agrícola en Java (PPO)

Autor: Renato Galleguillos, José Pablo Bernal, Vicente Zapata T

Asignatura: Lenguaje de Programación y Paradigmas

Docente: Paulina González

Fecha: 3-5-2025

https://github.com/RenatoGalleguillosC/App2

Índice

1. Arquitectura del Sistema

El trabajo presentado se conforma de 9 archivos. Es posible encontrar el código principal del programa dentro del archivo "App2.java", en el cual se importan el resto de los archivos a utilizar, los cuales corresponden a objetos creados, y los métodos correspondientes a cada uno. De esta forma, es posible desarmar el código para distintas tareas, como la lectura del archivo CSV, la gestión de parcelas, y gestión de cultivos. Por lo tanto, la App2 solamente sirve para estructurar el interfaz que se muestra al usuario, y el resto del código correspondiente a las funciones a realizar, se encuentran dentro del resto de los archivos. A forma de ejemplo, para la gestión de cultivos se solicita que permita funciones como listar cultivos existentes, crear y eliminar cultivos, y editar la información básica de un cultivo, las cuales, dentro del código, se encuentran en el archivo "GestorCultivos.java", en el cual cada función corresponde a un método del objeto GestorCultivos.

El cómo se estructuran los archivos se presentan en el siguiente esquema:

App2/

- App2.java
- cultivos.csv
- utils/
    - LeerCSV.java
- models/
    - Actividad.java
    - Cultivo.java
    - ElementoAgricola.java
    - Parcela.java
- services/
    - GestorActividades.java
    - GestorCultivos.java
    - GestorParcelas.java

2. Diseño de Clases

Para el diseño de clases se optó por la creación de 3 paquetes, cada uno conformado por distintos tipos de clases. En primer lugar, está el paquete *utils*, que contiene solamente la clase "LeerCSV", la cual nos permite leer el archivo *csv*. En segundo lugar, esta el paquete *models*, que representa la estructura de los datos que se

utilizan en la aplicación, es decir, las actividades, cultivos, elementos agrícolas, y parcelas. Por último, está el paquete *services* que contiene las clases utilizadas para las interacciones entre componentes del sistema, y la gestión de las actividades, cultivos, y parcelas.

3. Modificadores de Acceso y Encapsulamiento

Se utilizaron modificadores de acceso para las clases, interfaces, variables, métodos, y constructores. De esta forma, se protegen los datos a accesos no solicitados. Mayormente se utilizaron los modificadores public y private. Además, utilizando Encapsulamiento, se estructura el código de manera lógica y organizada. Como se mencionó anteriormente, se agruparon atributos y comportamientos de las distintas entidades, para crear las clases y sus respectivos métodos. Así, clases como "GestorCultivos.java" tendría los métodos de gestión de cultivos, mientras que la clase "GestorParcelas.java", tendría los métodos de gestión de parcelas. Por otro lado, también se utilizaron clases para las entidades de "Actividad", "Cultivo", "ElementoAgricola", y "Parcela"; de los cuales es posible crear múltiples objetos que conforman la estructura esperada en cada una.

4. Reflexión Final / Autoevaluación

Durante el desarrollo de esta tarea, dos integrantes del grupo contaban previamente con conocimientos en el lenguaje Java. El principal desafío fue lograr que el tercer miembro se pusiera al mismo nivel, lo cual resultó menos complejo que en trabajos anteriores, dado que Java es un lenguaje relativamente más accesible y estructurado.

En cuanto a la implementación del código, si bien logramos un funcionamiento correcto del programa, somos conscientes de que su eficiencia podría mejorarse. La solución final incluye 17 estructuras *case*, lo cual, aunque funcional, no resulta visualmente atractivo ni óptimo desde el punto de vista del diseño del menú principal. No obstante, consideramos que esta fue la estrategia más directa y manejable para cumplir con los requerimientos de la tarea sin mayores complicaciones.

Reflexiones personales:

Vicente Zapata:

Personalmente, me gusta trabajar más con lenguajes de programación orientados a objetos, debido que las reglas a las que se adhieren, como el encapsulamiento, y la creación de objetos y clases, se lleva a cabo de forma

estructurada y lógica. Esto permite un código fácil de pensar y modificar. Siguiendo esta idea, esta tarea, a mi parecer fue más simple que la anterior, debido que el cambio de C a Java es muy radical, y se demuestra una mucha mayor simplicidad del lenguaje en el último mencionado.

Por otro lado, una observación que me gustaría realizar acerca de los lenguajes POO, es que me sorprendió lo fácil que era dividir el código a realizar, debido a la forma en la que se programa en estos lenguajes. Este aspecto fue de agrado, debido que el trabajo en equipo en este tipo de tareas se ve complejizado por la naturaleza la creación de códigos que se realizan mayoritariamente de forma individual.

Renato Galleguillos:

Desde mi perspectiva, el principal desafío fue aprender el lenguaje Java, ya que partí sin conocimientos previos más allá de saber de su existencia. Hasta ahora, mi experiencia había sido principalmente con Python y, recientemente, con C, aunque aún no tengo dominio completo de este último. Igual que en la tarea anterior, el principal obstáculo fue el proceso de aprendizaje en sí.

Esta vez el proceso fue más complejo, no por la dificultad del lenguaje, sino por la falta de tiempo disponible. A pesar de contar con un plazo extenso, mis otras evaluaciones y compromisos académicos y personales me impidieron dedicar tiempo prolongado al trabajo, teniendo rara vez más de dos horas seguidas para enfocarme en la tarea. Personalmente, procuré no ser una carga para el equipo, y me esforcé por ponerme al día, considerando que mis compañeros ya contaban con formación previa en Java.

José Pablo Bernal:

Por mi parte, en general, no se presentaron grandes complejidades durante la implementación de la tarea, ya que contaba con conocimientos previos en programación orientada a objetos con Java. Esto fue una gran ventaja, ya que me permitió abordar con seguridad los conceptos fundamentales como la herencia, la encapsulación y la relación entre objetos. Gracias a esta base sólida, el diseño e implementación de las se desarrolló de manera fluida sin mayores obstáculos técnicos.

En cuanto al manejo de errores, pruebas y depuración, se optó por una metodología de uso intensivo de la impresión por consola (System.out.println), la cual fue útil al

momento de comprobar asociaciones, ya que permitió identificar posibles inconsistencias o comportamientos inesperados.

5. Uso de Inteligencia Artificial (si aplica)

La inteligencia artificial utilizada fue, principalmente, ChatGPT, que actuó como un tutor virtual para la corrección de errores en las líneas de código y para el soporte en la estructura general del programa. Su aporte fue relevante en las partes relacionadas con la lectura adecuada del archivo CSV, en particular del archivo *LeerCSV.java*, el cual debía permitir no solo visualizar el contenido del archivo, sino también editarlo o eliminar características específicas. Cabe destacar que, en muchas ocasiones, las soluciones propuestas por la IA resultaron ser más eficientes o claras que las que hicimos como grupo, por lo que consideramos lógico y provechoso incorporarlas al desarrollo del proyecto.

6. Bibliografía / Fuentes

Programación ATS. (2017). *Curso de Java desde cero* [Lista de reproducción]. YouTube. https://www.youtube.com/playlist?list=PLF_XbndBUd9egN5xV5wslww2u-3QHbPv1