Programación Avanzada

Acceso, procesamiento y visualización de datos abiertos en aplicaciones Java

12 de mayo de 2025
Programación Avanzada

Carlos Gálvez Reguera

ÍNDICE

INTRODUCCIÓN

El objetivo de este trabajo es desarrollar una aplicación en Java que permita acceder, procesar y visualizar datos abiertos procedentes de una fuente oficial, utilizando una arquitectura modular basada en capas. En concreto, la aplicación se centrará en el tratamiento de datos sobre precios de carburantes en las estaciones de servicio españolas, publicados por la Administración General del Estado.

El acceso a datos abiertos se ha convertido en una herramienta fundamental para el desarrollo de soluciones tecnológicas con impacto real en la sociedad. Esta actividad permite integrar conceptos avanzados de programación orientada a objetos, diseño de interfaces gráficas y consumo de servicios web REST.

Aplicándolos a un caso práctico con datos reales. Además, fomenta el desarrollo de aplicaciones reutilizables, estructuradas y mantenibles, aspectos clave en entornos profesionales.

El desarrollo de la aplicación se ha estructurado en varias fases. En primer lugar, se ha realizado la selección del conjunto de datos y la identificación de las entidades del dominio. Posteriormente, se ha llevado a cabo el diseño del modelo de clases, dividiendo la aplicación en tres capas funcionales: acceso a datos, lógica de negocio y presentación. A nivel técnico, se ha utilizado Java como lenguaje de programación, junto con un framework de interfaz gráfica (Swing), incorporando llamadas a servicios REST para obtener datos actualizados, y representaciones gráficas para facilitar la comprensión visual de la información procesada.

2. DESARROLLO DEL CÓDIGO

2.1 Arquitectura y justificación de la estructura de clases

La aplicación se ha desarrollado siguiendo la arquitectura Modelo-Vista-Controlador (MVC), que favorece la separación de responsabilidades y mejora la mantenibilidad del software.

- **Modelo:** Incluye las clases encargadas del acceso a datos y del modelo de dominio. La clase Gasolinera representa la entidad principal, mientras que GasolineraDAO gestiona la carga de datos desde el archivo CSV.
- **Vista:** Representada por GasolineraView, es responsable de construir la interfaz gráfica y mostrar los datos en una tabla (JTable).
- **Controlador:** La clase GasolineraController actúa como intermediario entre la vista y el modelo, gestionando eventos y sincronizando los datos cargados con la interfaz.
- **Aplicación:** La clase Main inicia el sistema, ensamblando los componentes MVC y lanzando la aplicación.

Esta estructura modular permite desacoplar la lógica de negocio, la presentación y la gestión de eventos, cumpliendo con buenas prácticas de ingeniería del software.

2.2 Clase Gasolinera.java

Define la entidad Gasolinera con los atributos necesarios para representar una estación de servicio, como provincia, municipio, dirección, precios de carburantes (gasolina 95 y 98), y coordenadas. Dispone de constructor, getters y método toString(). Ha sido documentada con comentarios Javadoc para facilitar su comprensión.

Acceso código clase en el siguiente enlace:Gasolinera.java

2.3 Clase GasolineraDAO.java

Implementa el acceso a los datos a través de la lectura del archivo CSV. Utiliza la biblioteca OpenCSV para interpretar correctamente los campos, incluso cuando contienen comillas o comas internas. Gestiona excepciones y convierte los datos leídos en objetos Gasolinera. Se emplea Apache Commons Lang como dependencia auxiliar requerida por OpenCSV.

Acceso código clase en el siguiente enlace:GasolineraDAO.java

2.4 Clase GasolineraView.java

Construye la interfaz gráfica con Swing, mostrando los datos en una tabla (JTable) contenida en un JScrollPane. Incluye un botón que permite al usuario cargar los datos desde el CSV. La disposición de los componentes se ha organizado mediante BorderLayout.

Acceso código clase en el siguiente enlace:GasolineraView.java

2.5 Clase GasolineraController.java

Gestiona la acción del botón de la vista. Al activarse, invoca a GasolineraDAO para obtener la lista de gasolineras y luego actualiza el modelo de la tabla (DefaultTableModel) en la vista para mostrar los datos. Implementa manejo de errores por si el archivo CSV no está disponible.

Acceso código clase en el siguiente enlace:GasolineraController.java

2.6 Clase Main.java

Punto de entrada del programa. Inicializa los componentes del modelo, vista y controlador, y los conecta entre sí. Lanza la interfaz en el hilo de eventos de Swing utilizando SwingUtilities.invokeLater().

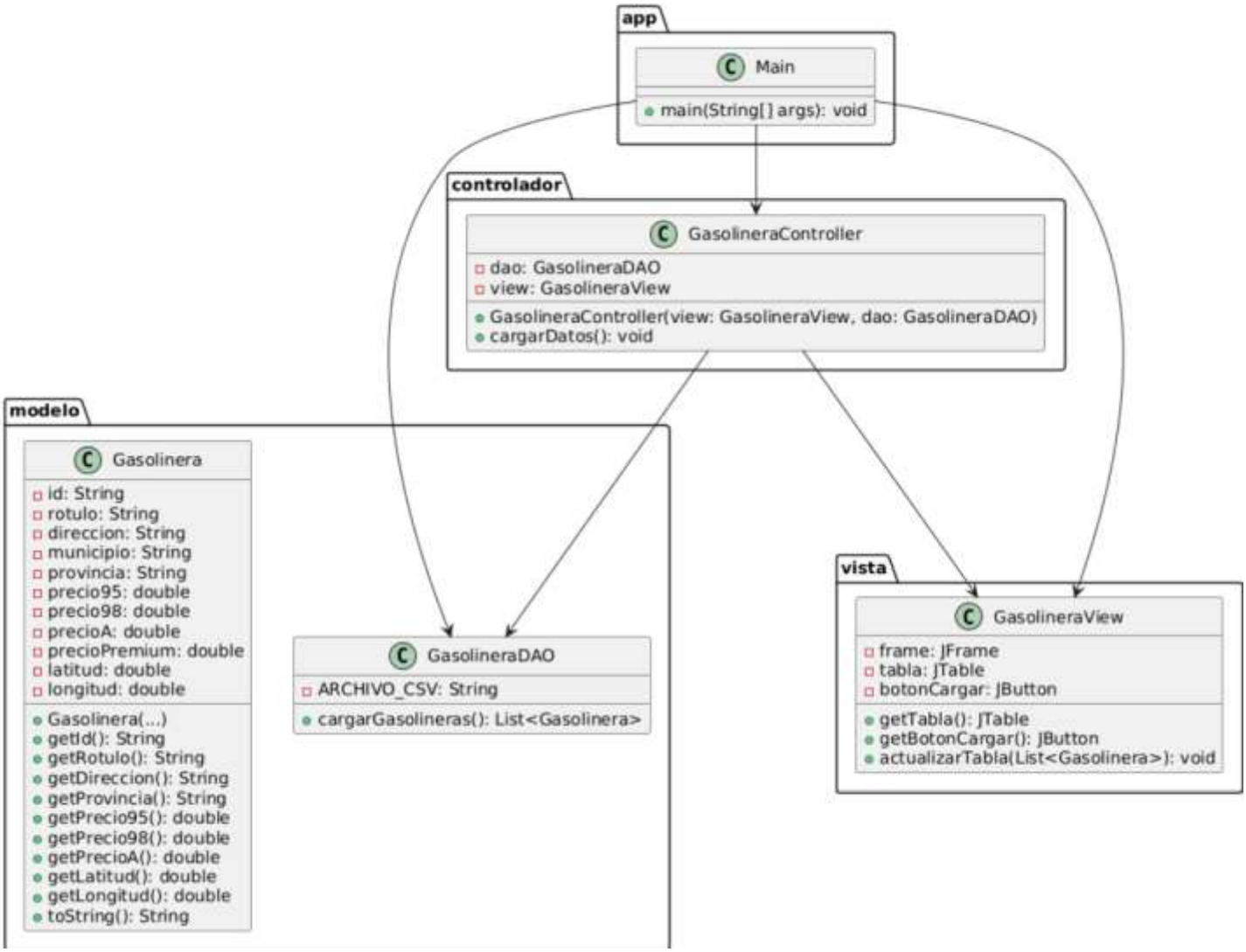
Acceso código clase en el siguiente enlace: Main.java

(NOTA: *En caso de* ***no poder acceder visualizar los códigos,*** *a través de los enlaces, desde el punto 7 de esta memoria, accediendo a Javadoc, seleccionas el package,pulsar el nombre de la clase, si se encuentra en la pestaña* CLASS *puede clicar al nombre de la clase y se mostrará el código*)

3. DIAGRAMA UML

En el siguiente apartado se adjunta una captura del diagrama UML que representa las clases desarrolladas y sus relaciones. Se ha seguido el enfoque de modelado orientado a objetos, indicando atributos y métodos públicos relevantes para la práctica.

El diagrama ha sido generado utilizando PlantUML, una herramienta que permite crear diagramas a partir de código, lo que facilita su mantenimiento y legibilidad. En esta versión detallada se incluye el modelo de dominio completo para la entidad Gasolinera, con sus atributos clave como id, provincia, precio95, precio98, latitud y longitud, además de los métodos de acceso y utilidad como getters y toString().



4. LIBRERÍAS UTILIZADAS

En el desarrollo de esta aplicación se han empleado las siguientes librerías:

4.1 Java Swing (API estándar)

La interfaz gráfica de usuario ha sido construida utilizando el framework Swing, incluido en el JDK de Java. Esta API proporciona componentes como JFrame, JTable, JButton y JScrollPane, que permiten representar visualmente los datos de forma estructurada y dinámica. Swing es una herramienta orientada a eventos y permite el diseño modular de interfaces de usuario sin necesidad de librerías externas.

4.2 OpenCSV

Para el procesamiento del archivo CSV, se ha utilizado la librería OpenCSV. Esta herramienta permite leer archivos CSV de forma robusta, gestionando correctamente campos con comas internas, comillas y datos vacíos. Su integración ha sido clave para garantizar la carga precisa de datos en la aplicación.
Grupo: com.opencsv
Clases utilizadas: CSVReader, CSVParserBuilder
Funcionalidad principal: Lectura línea a línea del archivo preciosEESS_es.csv y parseo estructurado a objetos Gasolinera.

4.3 Apache Commons Lang

La librería Apache Commons Lang es una dependencia indirecta de OpenCSV. Su clase ObjectUtils es utilizada internamente por el parser de OpenCSV para validar y gestionar objetos nulos. Fue necesario incorporar esta librería al Build Path para evitar errores de ejecución relacionados con clases no encontradas.
Grupo: org.apache.commons.lang3
Clases clave: ObjectUtils

6. ACCESO AL JAVADOC

El proyecto ha sido documentado mediante comentarios estructurados con anotaciones Javadoc, lo que permite la generación automática de la documentación técnica del código fuente. Se puede acceder a la documentación creada en Javadoc desde el siguiente enlace:

ACCESO_RAIZ_JAVADOC

7. VERIFICACIÓN DE EJEFUCIÓN POR CONSOLA

Se puede comprobar la ejecución y resultado del programa en los siguiente enlaces:

EJECUTAR_ACTIVIDAD_2

(NOTA: *Para ejecutar la aplicación automáticamente, puede utilizar el archivo* ***execute_ACT2_PAV.bat****, accesible desde la carpeta GALVEZ_REGUERA_CARLOS_ACTIVIDAD_2. Este script lanza la aplicación Java de forma automatizada desde el entorno de desarrollo o desde los .class compilados.)*

8. CONCLUSIÓN

La presente actividad ha permitido explorar un conjunto de tecnologías ampliamente consolidadas en el desarrollo de aplicaciones Java orientadas a la gestión y visualización de datos. El uso del patrón de arquitectura Modelo-Vista-Controlador ha facilitado una separación clara de responsabilidades entre la lógica de negocio, la presentación de datos y el control de eventos, lo que resulta especialmente útil en proyectos con crecimiento funcional o mantenimiento evolutivo.