Universidad Nacional de Costa Rica

Paradigmas de Programación

Tema: Examen en Parejas – Integración Secuencial y POO

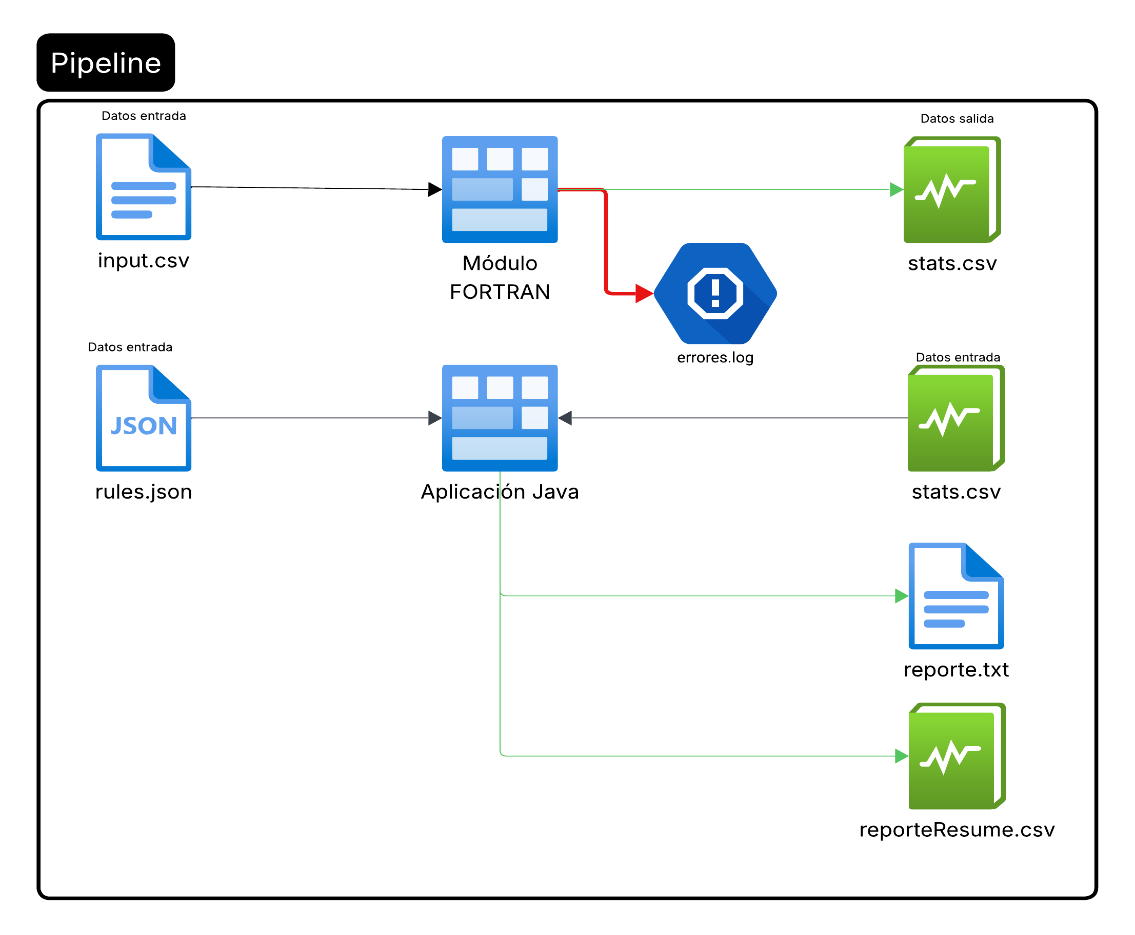
Estudiantes: Joshua Barrantes Mena, José Francisco Rodríguez Arias

Docente: M.Sc. José Pablo Noguera Espinoza

Fecha: 22 de setiembre, 2025

# Arquitectura de Integración

Diagrama del Pipeline



Esta arquitectura de integración es un pipeline secuencial. Dividido en dos 2 etapas con un archivo intermediario donde se detalla la forma de los datos. La primera etapa consiste en una implementación en Fortran, cumpliendo objetivos de cálculos numéricos pesados y exportación de resultados. La ultima etapa en Java, dirige todo el proyecto con la lógica de negocio. Se encuentran aceptaciones, errores y generación de reportes.

Flujo de datos

1. Entrada: Se inicia con un archivo input.csv que está lleno de números en filas y columnas.
2. Fortran paso 1: El programa en Fortran lee ese archivo y calcula estadísticas clave para cada columna como variable, mean, median, stddev, outliers.
3. La comunicación: Fortran guarda estos resultados en un archivo de resumen llamado stats.csv. Este archivo actúa como el puente de comunicación entre Fortran y Java.
4. Javas paso 2: El programa en Java lee el resumen de stats.csv y un archivo de configuración llamado rules.json, que contiene las reglas para validar la calidad de los datos.
5. Secuencia final: Usando las reglas, Java determina si los datos de cada columna son aceptables y genera dos archivos de salida: reporte.txt y reporte\_resumen.csv..

Estructura de los archivos clave

* stats.csv: Es un archivo de texto simple que usa punto y coma para separar los valores. Cada fila contiene los cálculos para una de las columnas originales con nombre; variable, mean, median, stddev, outliers.
* rules.json: Este archivo define las reglas de calidad por ejemplo; no más de 5 outliers o verificar la cantidad de valores atipicos. Su ventaja es que podemos cambiar las reglas aquí sin necesidad de reprogramar la aplicación Java
* errors.log: Cada que encuentra un valor no numérico o erróneo, guarda en errors.log la fila y columna donde encontró el error incluido en valor que dio el error.

Forma de manejar los errores

* El programa en fortran está diseñado para no detenerse por datos incorrectos. Si encuentra texto en lugar de un número. Guarda en stats.csv un valor por defecto (-9999) en lugar de omitirlo, esto para referencia que hubo un error en esa columna y que necesita ser verificada
* La aplicación en Java está preparada para fallos comunes. Si no encuentra los archivos que necesita o si el archivo de reglas está mal escrito, no falla de manera abrupta. En su lugar, muestra un mensaje de error claro y se detiene de forma segura.

# Decisiones de diseño

¿Por qué el tramo Fortran se resuelve secuencialmente y no con clases?

Primeramente, debido a su rendimiento, porque Fortran esta dedicado a el tema numérico y científico. Según Maldonado (2024) redactó que “es su capacidad para ejecutar cálculos numéricos a gran escala. Este lenguaje es utilizado en áreas que requieren procesamiento intensivo, como la simulación climática, la modelización de sistemas físicos complejos y la investigación científica” (parr. 6). Esto demuestra que irse a los cálculos secuenciales es más optimizado que hacer sobrecarga de objetos, herencia, implementaciones los cuales no darían el acceso al hardware directo porque el compilador lo vuelve más rápido en lenguaje máquina. Por otro lado, el tema de simplicidad y un código más enfocado en los cálculos tiene una única función, evitando crear clases para separar funciones.

¿Por qué Java usa clases, herencia y polimorfismo?

Seguidamente, con Java lo más importante es el tema de la organización, por ejemplo organizar ColumnMetric y QualityRule para representar las ideas más centrales del problema. Dando como resultado las clases un factor ordenado y fácil de entender. Segundo, las interfaces ayuda a realizar nuevas reglas. Esto nos permite hacer modificaciones solamente en esta parte y no interferir en código que si funciona.

¿Por qué CSV para la comunicación entre módulos?

Con este apartado en csv se presenta un texto plano. Esto permite abrir el stats.csv en cualquier momento para ver si los cálculos de Fortran eran correctos. Esto nos ahorró mucho tiempo de depuración. Por último, la factibilidad de la portabilidad de un csv brinda texto funciona igual en Windows, Linux o Mac, sin problemas de compatibilidad.

# Complejidad y rendimiento

# Pseudocódigo

# 

# Complejidad temporal y espacial

La complejidad temporal consiste en el tiempo que tarda el código en realizar ciertas operaciones, según que se le pida, el tiempo que tarda aumenta de forma considerable o puede no aumentar, en el caso del pseudocódigo fortran, consta de 4 operaciones, lectura del csv, calculo estadístico, calculo de outliers y escritura de csv.

La lectura del csv va en función de la cantidad de filas(M) y columnas(K) que existan, seguido del cálculo de estadísticas que también va en función de las filas y columnas, el calculo de outliers también requiere de lectura de todas las filas de cada columna, finalizando con los resultados derivados de cada columna completa, las cuales se imprimen en un csv final en función de la cantidad de columnas que tenia el csv inicial. Por ende, la complejidad total va en función de (M x K) durante toda la ejecución.

Por otro lado, la complejidad espacial es la cantidad de espacio que requiere realizar los cálculos durante la ejecución, siendo en el pseudocódigo fortran utilizado por la matriz de datos en memoria que se mantiene en constante lectura y escritura durante todo el pseudocódigo, mas las columnas auxiliares utilizadas necesarias para los cálculos finales, dando un resultado de (M x K) + (M x K).

Medidas empíricas

Un ejemplo del calculo de las medidas empíricas es la siguiente:

Filas = 100000  
columnas = 50

Bytes por valor entero = 4

Espacio usado por la matriz = (100000 x 50) x 4 = 20000000 ≈ 20Mb

Espacio usado por variables auxiliares: mean, median, stdevs, outliers < 1Kb

Factible en una computadora común.

Cuellos de botella

* Según ciertos casos pueden ser los relacionados a la lectura y escritura de datos, pueden tomar tiempo si se toman CSVs descomunalmente grandes, acompañado de que la entrada/salida de disco suele ser mas tardado que los cálculos en memoria RAM.
* Guardar toda la matriz en memoria puede ser contraproducente, millones de filas y columnas pueden hacerla crecer demasiado y colapsar la memoria.
* En el pseudocódigo actual, la media y la desviación se calculan por separado, existiendo un segundo cálculo de la media para la desviación.

Optimizaciones

* Se puede calcular la media y la varianza en un solo bucle, ahorrando tiempo de cálculo.
* Aprovechar más las funciones incluidas de fortran como el SUM().
* Paralelizar los cálculos de las diferentes columnas.

# Referencias

Maldonado, R. (4 de septiembre de 2024). Historia de Fortran: el primer lenguaje de programación. *KeepCoding.* [*https://keepcoding.io/blog/fortran-el-primer-lenguaje-de-programacion/#:~:text=Fortran%20en%20la%20computación%20de%20alto%20rendimiento&text=Uno%20de%20los%20aspectos%20más,en%20evaluación%20de%20supercomputadores%20cuánticos*](https://keepcoding.io/blog/fortran-el-primer-lenguaje-de-programacion/#:~:text=Fortran%20en%20la%20computación%20de%20alto%20rendimiento&text=Uno%20de%20los%20aspectos%20más,en%20evaluación%20de%20supercomputadores%20cuánticos)*.*