DOCUMENTACIÓN DEL TRABAJO PRÁCTICO INTEGRADOR

LIBRERÍA PARA GESTIÓN Y PROCESAMIENTO DE DATOS TABULARES

INTEGRANTES – GRUPO 5

CÁCERES LUDMILA
FLORIDIA MICAELA
SILVETTI SANTINO
VILLANUEVA LEANDRO

FECHA DE ENTREGA

14 DE JUNIO

ASIGNATURA

ALGORITMOS I

ÍNDICE

INTRODUCCIÓN

PRESENTACIÓN

El presente trabajo integrador nos desafía a aplicar los conocimientos adquiridos a lo largo de la materia para construir, desde cero, una librería de manipulación y análisis de datos en formato tabular, utilizando Java como lenguaje de programación.

Si bien el enfoque principal recae en el diseño orientado a objetos y en la implementación del sistema, valoramos profundamente la fase de análisis como una oportunidad crucial para la reflexión, la organización de ideas y el consenso en equipo.

Nuestro grupo se compromete a realizar un trabajo de calidad, no solo en términos técnicos, sino también en el proceso de colaboración. Contamos con experiencias, habilidades y perspectivas diversas, y confiamos en que esta pluralidad enriquecerá el proyecto mediante la construcción conjunta.

Aspiramos a que esta experiencia nos brinde un aprendizaje significativo que será de gran utilidad para nuestro futuro académico y profesional. Con esta convicción, nos embarcamos en este desarrollo.

DOCUMENTACIÓN DE ANÁLISIS

OBJETIVO

El objetivo principal de este trabajo es desarrollar una librería en Java que permita gestionar y analizar datos estructurados en formato tabular (bidimensional), facilitando la ejecución de tareas básicas y avanzadas de procesamiento de datos.

La librería se diseñará bajo los principios de la programación orientada a objetos, priorizando la extensibilidad, la mantenibilidad y la claridad. Esto con el propósito de asegurar su evolución futura y su utilidad en contextos tanto educativos como profesionales.

Mediante este desarrollo, buscamos consolidar los conocimientos adquiridos en esta materia, profundizar en las buenas prácticas de programación y fortalecer el trabajo colaborativo en equipo.

ALCANCE

Funcionalidades incluidas

- Generación de estructuras tabulares a partir de archivos CSV, estructuras nativas de Java o copia de otras instancias existentes.
- Modificación de datos: añadir o eliminar columnas y filas, y actualizar celdas.
- Visualización de la información tabular en formato textual, incluyendo opciones de configuración.
- Consulta de propiedades de la estructura: filas, columnas, etiquetas y tipos de datos.
- Acceso indexado a datos por fila, columna o celda.
- Selección de subconjuntos de datos mediante etiquetas (slicing).
- Filtrado de datos por condiciones lógicas aplicadas a columnas.
- Agregación de datos por grupo y aplicación de funciones de resumen (suma, media, conteo, etc.).
- Copia profunda de estructuras de datos.
- Concatenación de múltiples estructuras tabulares.
- Ordenamiento de filas por valores en columnas específicas.
- Imputación de valores nulos o faltantes (NA).
- Muestreo aleatorio de filas por porcentaje.

Funcionalidades excluidas

- Visualización gráfica de los datos.
- Integración con otros formatos de archivo (Excel, JSON, etc).
- Uso de librerias externas sin previa aprobación docente.
- Implementación en lenguajes distintos a Java.

DESCRIPCIÓN DE ALTO NIVEL DEL SISTEMA

El sistema propuesto es una librería de manipulación y análisis de datos tabulares desarrollada en Java. Está diseñada para operar sobre estructuras bidimensionales, análogas a una hoja de cálculo o una tabla de base de datos, que serán gestionadas mediante estructuras de datos nativas de Java y almacenadas temporalmente en la memoria del sistema durante su ejecución.

La librería se organizará en módulos funcionales, cada uno con responsabilidades claramente definidas, lo que favorecerá su comprensión, mantenimiento y futura evolución.

A continuación, se describe cada uno de los módulos clave que conformarán la arquitectura general del sistema:

Módulo de Gestión de Datos

Este módulo constituye el núcleo fundamental de la librería. Su propósito principal es la creación, modificación y eliminación de datos tabulares, garantizando la estructura, integridad y coherencia de los mismos.

Está compuesto por clases esenciales que representan las estructuras de datos fundamentales, como Tabla, Columna y Celda. Estas clases son las encargadas de almacenar y organizar los datos, asociando a cada Celda y Columna su respectivo tipo de dato (definido por la clase TipoDato). Además, este módulo incluye un subsistema robusto de validación de tipos, con clases como ValidadorTipoDato y validadores específicos (ValidadorBooleano, ValidadorNA, ValidadorNumerico, ValidadorString), que aseguran que los valores ingresados y procesados se ajusten a las expectativas de sus tipos, manteniendo así la consistencia de la información en toda la tabla.

Módulo de Entrada/Salida de Datos

El Módulo de Entrada/Salida de Datos es responsable de la interacción de la librería con archivos externos en formato CSV. Permite la carga (cargarCSV) y descarga (guardarCSV) de datos tabulares.

Ofrece configuración flexible para el proceso de lectura y escritura, ya que el usuario puede definir el delimitador de campos (por defecto, la coma “,”) y se puede especificar si la tabla incluye una fila de encabezado o si las columnas se etiquetarán automáticamente con números enteros al cargar.

Al cargar un archivo CSV, el módulo realiza una inferencia automática del TipoDato para cada columna (numérico, booleano o cadena) y construye una instancia de Tabla. Maneja valores faltantes como NA y registra advertencias si se detectan inconsistencias de tipo con los datos inferidos o establecidos.

Incluye manejo robusto de excepciones para errores de lectura/escritura de archivos y de formato de datos, así como validaciones para la consistencia de los datos importados. Su objetivo principal es asegurar la interoperabilidad de la librería con diversas fuentes de datos CSV.

Módulo de Selección

El Módulo de Selección habilita la exploración y la obtención de subconjuntos específicos de una Tabla. Este módulo permite el acceso y la selección de datos utilizando etiquetas de filas y columnas. Es posible realizar selecciones parciales (slicing) que generan vistas temporales de los datos a través de la clase TablaParcial, lo que significa que se trabaja sobre los datos originales sin duplicarlos en memoria. Para una exploración rápida de grandes conjuntos de datos, el módulo también ofrece las operaciones head(n) y tail(n), que permiten visualizar las primeras o últimas n filas de una tabla, respectivamente. Es importante destacar que todas las operaciones de selección de este módulo (head, tail, seleccionar) directamente imprimen la Tabla resultante en la salida estándar, proporcionando una visualización inmediata del subconjunto de datos.

Módulo de Filtros

El Módulo de Filtros permite seleccionar filas específicas de una tabla basándose en diversas condiciones. Ofrece la capacidad de aplicar filtros simples de comparación (como igual, mayor o menor), combinar condiciones mediante operadores lógicos (AND, OR, NOT), y detectar valores nulos. Este módulo proporciona un control centralizado para la aplicación de estas condiciones, generando como resultado una nueva tabla que contiene únicamente las filas que cumplen con los criterios establecidos.

Módulo de Manipulación

El Módulo de Manipulación agrupa las funcionalidades que permiten transformar y preparar los datos contenidos en una Tabla. Diseñado bajo un patrón de estrategia mediante la clase abstracta ManipuladorDatos, este módulo

ofrece la capacidad de ordenar datos, permitiendo reordenar las filas de una tabla basándose en los valores de una columna específica. También incluye la imputación de valores faltantes (NA), facilitando el reemplazo de celdas vacías o no definidas por un valor dado, siempre y cuando este sea compatible con el tipo de dato de la columna. Además, posibilita el muestreo aleatorio, que permite la extracción de una muestra aleatoria de filas de la tabla, ajustando su tamaño según un porcentaje especificado. Es importante destacar que todas las operaciones dentro de este módulo realizan modificaciones directamente sobre la tabla original que se les proporciona. Su objetivo general es ofrecer herramientas básicas para la preparación y el pre-análisis de datos directamente sobre la estructura tabular.

Módulo de Visualización

El Módulo de Visualización tiene como propósito principal representar una Tabla (o sus subconjuntos) en formato de texto, asegurando que la información sea legible y fácil de inspeccionar para el usuario. La clase Visualizador gestiona el formato de salida, aplicando ajustes automáticos como el cálculo del ancho óptimo de las columnas, el truncamiento del contenido de las celdas (limitado por una longitud máxima de caracteres por celda), y la restricción del número de filas y columnas que se muestran simultáneamente para facilitar la visualización en entornos de consola. Es importante destacar que estos límites y criterios de visualización son definidos internamente por el módulo a través de constantes. Adicionalmente, el módulo puede proporcionar una función para mostrar información resumida de la tabla, incluyendo su cantidad de filas, columnas y los tipos de datos de cada columna.

Módulo de Copia y Concatenación

Este módulo brinda funcionalidades para duplicar y combinar estructuras tabulares. Incluye una operación de copia profunda, que genera una nueva tabla totalmente independiente de la original, y una función de concatenación, que une dos tablas compatibles en una sola estructura, preservando la coherencia de tipos y etiquetas. Facilita así la reutilización y expansión de datos dentro de la librería.

Módulo de Agregación

El Módulo de Agregación permite transformar tablas detalladas en resúmenes significativos. Su función principal es agrupar las filas de una tabla basándose en columnas específicas y luego aplicar diversas operaciones de agregación a los datos agrupados. Entre estas operaciones se incluyen la suma, el máximo, el mínimo, el conteo, la media, la

varianza y el desvío estándar. El resultado es una nueva tabla que condensa la información original, asegurando la compatibilidad de los tipos de datos en los cálculos.

Módulo de Gestión de Errores

El Módulo de Gestión de Errores es fundamental para el manejo robusto de situaciones excepcionales dentro de la librería. Define una jerarquía clara de excepciones específicas (ExcepcionCargaDatos, ExcepcionOperacionNoValida, ExcepcionValidacion), todas ellas derivadas de una excepción base común (ExcepcionTabular). Estas excepciones permiten una notificación precisa de los errores que puedan surgir durante las diversas operaciones tabulares. Adicionalmente, el módulo incluye una clase utilitaria, GestionErrores, que actúa como un punto centralizado para el registro (logging) de errores y advertencias. Las clases de este módulo están diseñadas para ser utilizadas por cualquier otro componente del sistema que necesite reportar o manejar condiciones anómalas, estableciendo así dependencias dinámicas que se manifiestan en tiempo de ejecución y no se representan explícitamente en el diagrama de clases.

Figura: Vista Modular

La estructura modular permite aislar funcionalidades, minimizar el acoplamiento entre clases, y facilitar la aplicación de principios de diseño como SRP (Single Responsibility Principle),

lo cual es clave para asegurar la calidad de la solución.

REQUERIMIENTOS FUNCIONALES MÁS RELEVANTES

A continuación, se detallan los requerimientos funcionales más relevantes del sistema, organizados en una estructura jerárquica que agrupa funcionalidades relacionadas bajo macro-requerimientos. Esta clasificación busca facilitar el análisis, diseño e implementación modular del sistema.

Macro-requerimiento 1: Gestión de Datos

Este módulo constituye el núcleo del sistema y se encarga de la creación, edición y organización interna de las estructuras tabulares.

Requerimiento Funcional 1.1:

La librería debe permitir generar una estructura tabular desde:

- Un archivo CSV.
- Una estructura de dos dimensiones nativa de Java.
- Una secuencia lineal de datos nativa.

Requerimiento Funcional 1.2:

La librería debe permitir modificar estructuras existentes mediante:

- Inserción de filas o columnas.
- Eliminación de filas o columnas.
- Actualización de valores de celdas.

Macro-requerimiento 2: Entrada/Salida de Datos

Este módulo asegura la interoperabilidad con archivos externos en formato CSV.

Requerimiento Funcional 2.1:

La librería debe permitir cargar datos desde un archivo CSV, especificando:

- El delimitador de columnas.
- Si el archivo incluye una fila de encabezado con etiquetas de columnas.

Requerimiento Funcional 2.2:

La librería debe permitir exportar la estructura actual a un archivo CSV, permitiendo definir delimitadores y el formato de salida.

Macro-requerimiento 3: Selección

Este módulo proporciona interfaces para acceder a los datos y explorar subconjuntos de forma eficiente.

Requerimiento Funcional 3.1:

La librería debe permitir el acceso indexado por etiquetas o índices numéricos a:

- Una fila completa.
- Una columna completa.
- Una celda específica.

Requerimiento Funcional 3.2:

La librería debe permitir seleccionar subconjuntos de filas y/o columnas (slicing) utilizando listas de etiquetas.

Macro-requerimiento 4: Filtrado

Este módulo permite seleccionar filas específicas de una tabla basándose en diversas condiciones.

Requerimiento Funcional 4.1:

La librería debe permitir aplicar filtros condicionales (query) sobre columnas, utilizando operadores relacionales y lógicos (>, <, =, AND, OR, NOT).

Macro-requerimiento 5: Manipulación

Este módulo agrupa funcionalidades asociadas al análisis y transformación de datos.

Requerimiento Funcional 5.1:

La librería debe permitir ordenar filas en función de los valores de una o más columnas, en orden ascendente o descendente.

Requerimiento Funcional 5.2:

La librería debe permitir la imputación de valores faltantes (NA) en celdas, mediante un valor literal proporcionado por el usuario.

Requerimiento Funcional 5.3:

La librería debe permitir realizar muestreo aleatorio de filas, devolviendo un subconjunto proporcional al porcentaje indicado.

Macro-requerimiento 6: Visualización

Este módulo facilita la presentación legible de los datos y la exploración inicial del conjunto.

Requerimiento Funcional 6.1:

La librería debe permitir consultar propiedades básicas de la estructura: cantidad de filas y columnas, etiquetas, y tipos de datos por columna.

Requerimiento Funcional 6.2:

La librería debe permitir la visualización textual de los datos, con opciones configurables de límite de filas/columnas mostradas y tamaño máximo de celda.

Macro-requerimiento 7: Copia y Concatenación

Este módulo provee funcionalidades para duplicar y combinar estructuras tabulares existentes.

Requerimiento Funcional 7.1:

La librería debe permitir la creación de una copia profunda (Deep copy) de una estructura tabular, garantizando la duplicación completa de los datos en memoria.

Requerimiento Funcional 7.2:

La librería debe permitir la concatenación de dos estructuras tabulares, siempre que sus columnas sean compatibles (por nombre, tipo y orden).

Macro-requerimiento 8: Agregación

Este módulo permite transformar tablas detalladas en resúmenes significativos mediante operaciones de agrupación y cálculo.

Requerimiento Funcional 8.1:

La librería debe permitir agrupar filas de una tabla por una o más columnas y aplicar funciones de resumen (suma, media, conteo, máximo, mínimo, varianza, desvío estándar) sobre los datos agrupados.

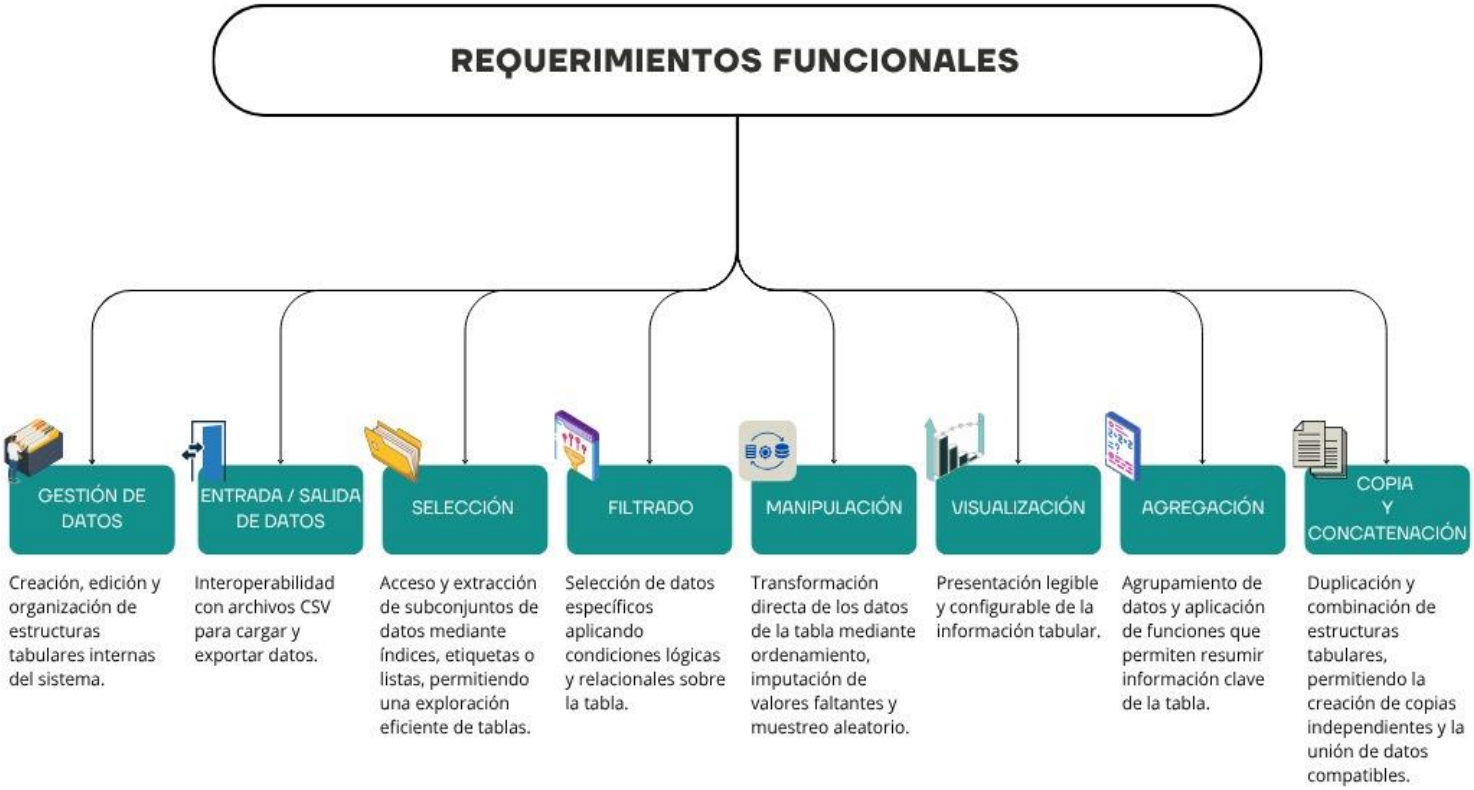


Figura:

Representación jerárquica de los requerimientos funcionales de la librería. Cada grupo reúne funcionalidades relacionadas bajo un macro-requerimiento, facilitando su análisis, diseño e implementación modular.

REQUERIMIENTOS NO FUNCIONALES

Los siguientes atributos de calidad orientan el desarrollo de la librería, con el objetivo de asegurar su correcta funcionalidad, mantenimiento a largo plazo y una óptima experiencia de uso. Han sido redactados siguiendo las recomendaciones y sugerencias brindadas en las consignas del Campus.

Extensibilidad:

- La librería será diseñada siguiendo los principios SOLID, lo que facilitará la adición de nuevas funcionalidades sin requerir modificaciones extensivas en el código existente.
- Se priorizará el uso de interfaces y/o clases abstractas para facilitar la evolución de su arquitectura.

Mantenibilidad:

- Las clases exhibirán alta cohesión y bajo acoplamiento.
- Se evitará la duplicación de código, promoviendo la reutilización.
- Se buscará que cada clase tenga una única responsabilidad clara, siguiendo el principio SRP.

Gestión de errores:

- El sistema capturará y clasificará errores comunes mediante una jerarquía de excepciones personalizadas.
- Toda excepción será manejada internamente o propagada con información útil para el usuario/desarrollador.

Rendimiento:

- Se espera que las operaciones de carga, filtrado y selección se completen en menos de 5 segundos sobre estructuras de gran volumen.
- Se minimizarán operaciones de copia innecesarias en consultas parciales (ej. Slicing, filtrado) para optimizar el uso de recursos.

Seguridad (encapsulamiento):

- Se procurará que todos los atributos sean privados, accediendo a ellos mediante métodos públicos (getters/setters) con la lógica de validación necesaria.
- No se permitirá acceso directo a la estructura interna de datos fuera de los métodos provistos por la librería.

Portabilidad:

- La librería será compatible con Java 8 o superior, y su ejecución debe funcionar correctamente en cualquier sistema operativo que soporte dicha versión de Java.

Usabilidad y documentación:

- La API contará con una estructura consistente y nombres intuitivos para facilitar su uso.
- La librería será documentada, incluyendo ejemplos de uso para guiar a los usuarios.

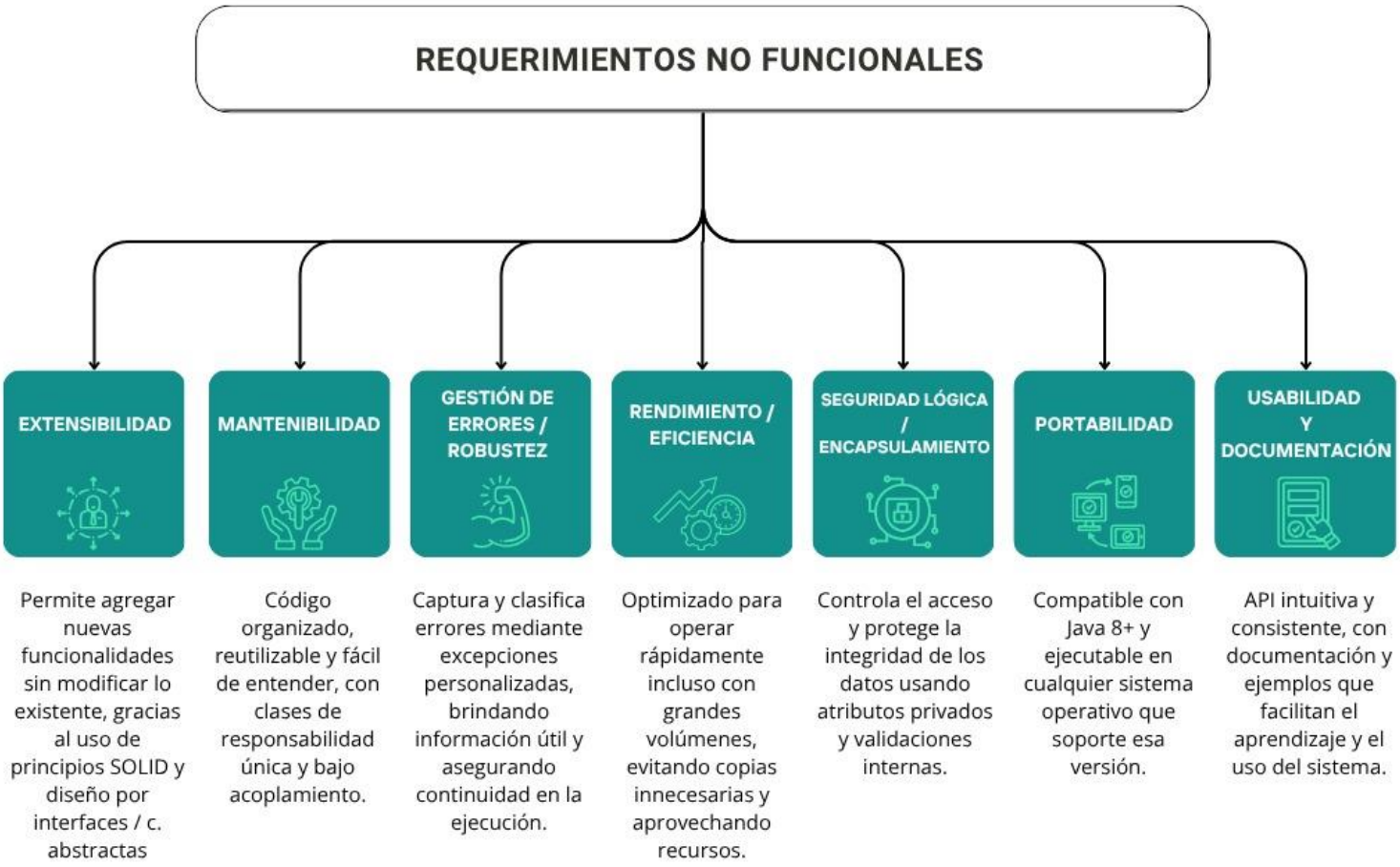
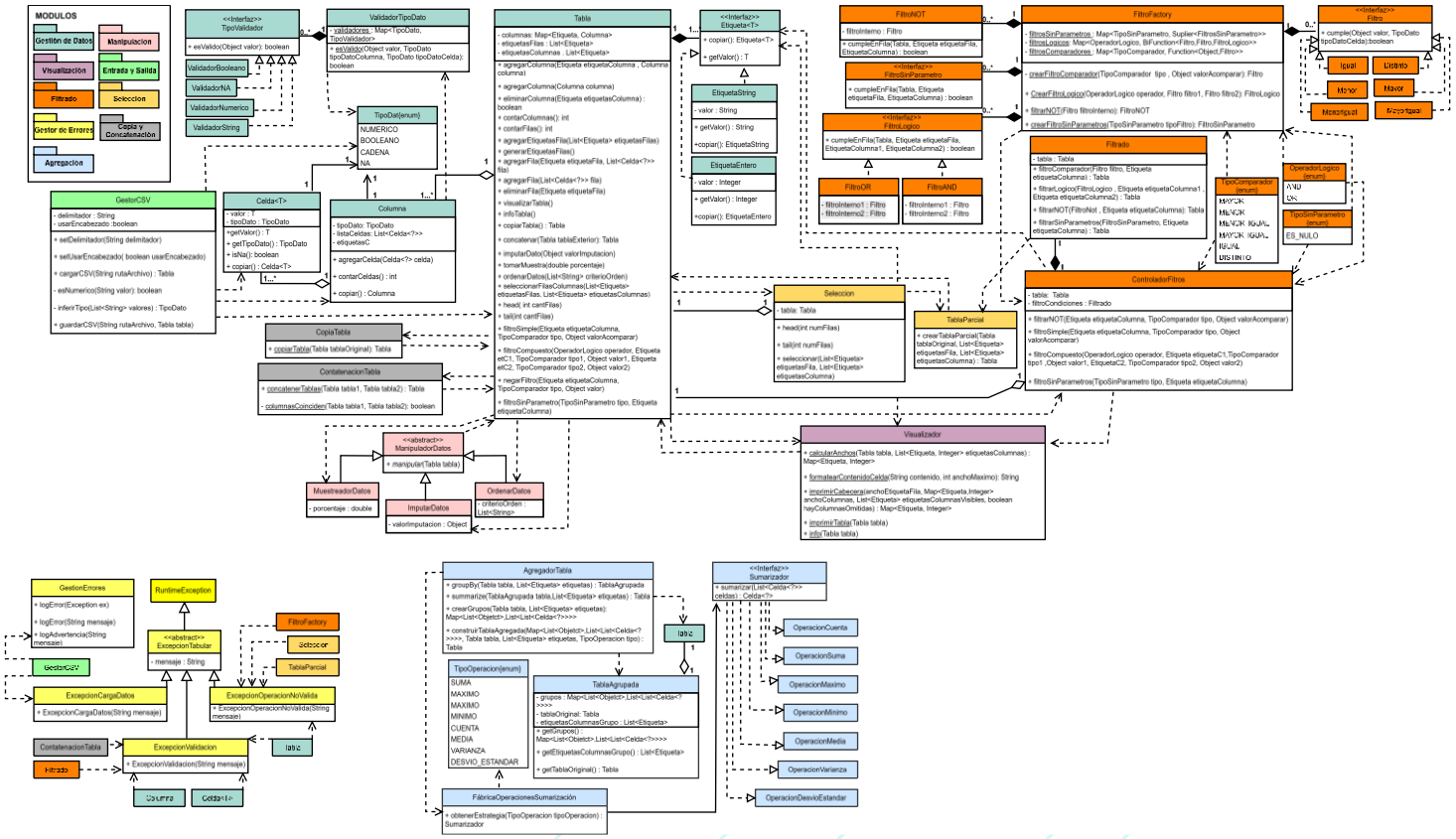


Figura: Este conjunto de requerimientos no funcionales establece criterios de calidad que orientan el diseño, la mantenibilidad y la experiencia de uso de la librería. Algunos fueron tomados de la consigna, y otros propuestos a partir de buenas prácticas y recomendaciones que están en el campus, priorizando siempre decisiones que acompañen nuestro proceso de aprendizaje.

DOCUMENTACIÓN DE DISEÑO

DESCRIPCIÓN GENERAL DEL DIAGRAMA DE CLASES



Para ver el diagrama con mayor detalle, consulte el archivo adjunto.

El diagrama de clases presentado ofrece una vista general estática del sistema, cuyo objetivo es implementar una librería en Java para la manipulación y el análisis de datos tabulares. Esta representación permite identificar la estructura del sistema y las relaciones entre los distintos módulos funcionales que lo componen. El sistema se encuentra modularizado en nueve bloques funcionales principales, cada uno con responsabilidades bien definidas: gestiondedatos, entradaysalida, seleccion, filtros, manipulacion, visualizacion, copiayconcatenacion, agregacion y gestiondeerrores. Esta modularización promueve una alta cohesión interna y un bajo acoplamiento entre los módulos.

En el diseño del sistema, se observa una aplicación consistente del polimorfismo y la generalización a través de relaciones de herencia y realización, junto con diversas dependencias y asociaciones entre módulos. Las clases OrdenadorDatos, ImputadorDatos y MuestreadorDatos extienden la clase abstracta ManipuladorDatos, estableciendo una jerarquía que permite la aplicación polimórfica de diversas operaciones sobre los datos tabulares.

Las clases CopiaTabla y ConcatenadorTabla, pertenecientes al módulo de copiaryconcatenacion, presentan relaciones de uso (dependencia) con la clase Tabla del módulo de gestiondedatos. Ambas clases operan sobre instancias de Tabla para duplicar o combinar estructuras tabulares, pero no las mantienen como atributos permanentes, lo que subraya su rol como funciones que manipulan datos externos.

La clase Visualizador, ubicada en el módulo de visualizacion, se relaciona con Tabla del módulo de gestiondedatos mediante una dependencia. Su propósito principal es representar visualmente la información contenida en las tablas, utilizando instancias de la clase Tabla como parámetro en sus métodos, pero sin almacenarlas como un atributo permanente. Este enfoque reduce el acoplamiento, ya que Visualizador no depende permanentemente del estado interno de una instancia específica de Tabla.

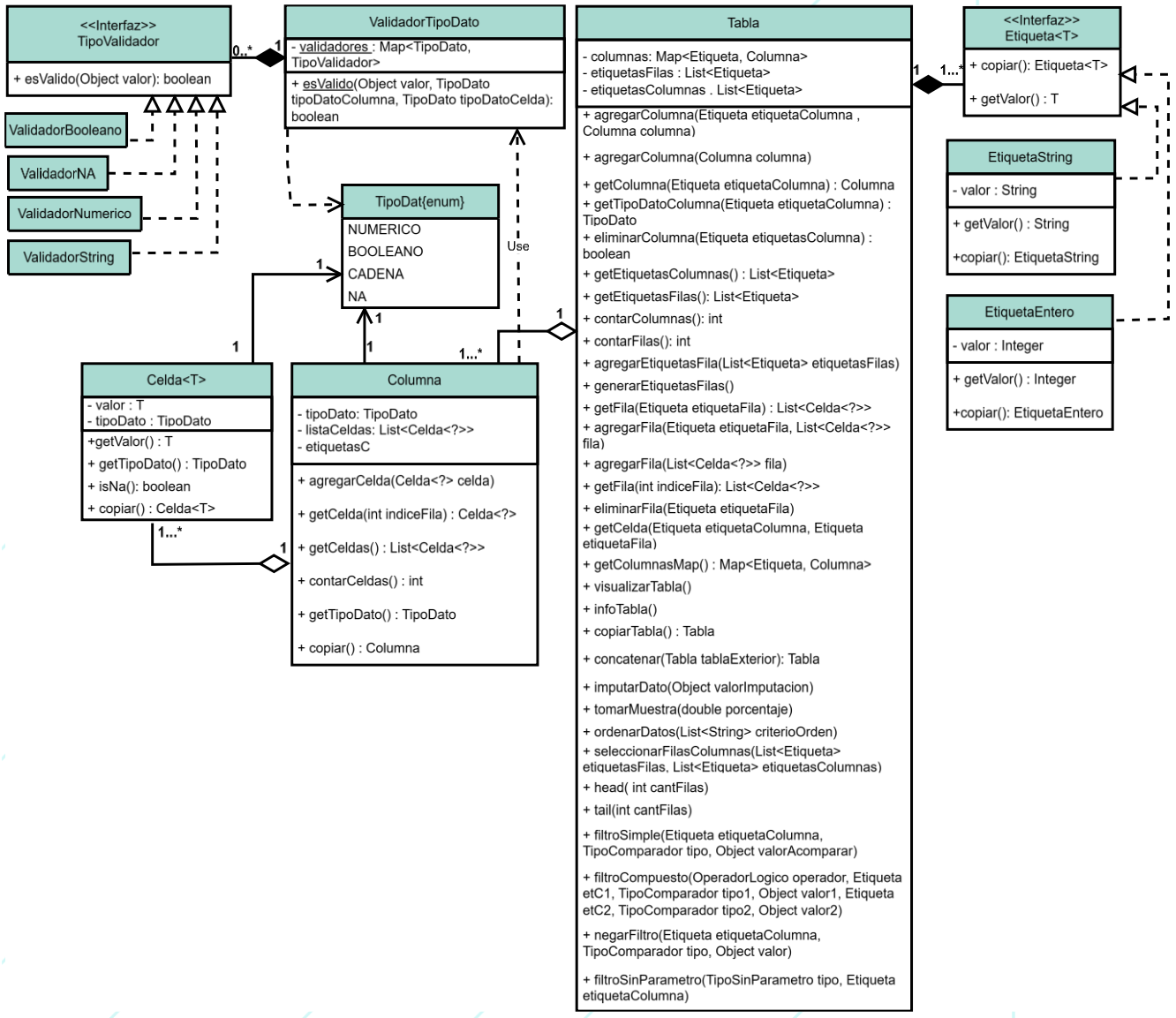
Por su parte, la clase GestorCSV, del módulo de entradaysalida, mantiene una relación de uso con la clase Tabla del módulo de gestiondedatos, ya que realiza operaciones de carga y exportación desde y hacia archivos CSV.

Además, GestorCSV mantiene una asociación con la clase GestionErrores del módulo de gestiondeerrores, delegando en ella el manejo y la gestión de posibles errores durante las operaciones de entrada y salida.
Finalmente, el módulo de gestiondeerrores define una jerarquía de excepciones. Las clases ExcepcionValidacion, ExcepcionCargaDatos y ExcepcionOperacionNoValida heredan de la clase abstracta ExcepcionTabular. Esta última encapsula comportamientos comunes asociados a errores vinculados con operaciones sobre tablas, permitiendo un manejo de errores coherente. Estas excepciones pueden ser utilizadas por cualquier módulo del sistema que requiera informar o manejar situaciones anómalas, manifestándose como dependencias dinámicas durante la ejecución del programa (por ejemplo, cuando un método lanza o captura una excepción).

El diseño general del sistema fue concebido respetando principios fundamentales de la programación orientada a objetos, lo que se traduce en una arquitectura flexible, extensible y de fácil mantenimiento. Se aplica, por ejemplo, el Principio de Sustitución de Liskov (LSP), mediante el uso de interfaces que son implementadas por diversas clases concretas. Esto garantiza que una implementación específica de una interfaz pueda reemplazar a su tipo base sin alterar el comportamiento esperado del sistema. Ejemplos claros de esta aplicación incluyen la interfaz Etiqueta, implementada por EtiquetaEntero y EtiquetaString, que asegura que cualquier tipo de identificador válido pueda ser manejado de forma transparente por las clases Tabla y Columna. Asimismo, en el sistema de validación de tipos, la interfaz TipoValidador es implementada por validadores específicos como ValidadorBoolean, ValidadorNA, ValidadorNumerico y ValidadorString, permitiendo que la lógica de validación se intercale o modifique sin impactar el código cliente que la utiliza.

Además, el diseño incorpora el Principio de Inversión de Dependencias (DIP), asegurando que los módulos de alto nivel dependan de abstracciones y no de implementaciones concretas. Esto se manifiesta en varios puntos clave. Módulos funcionales de alto nivel, como aquellos que orquestan la manipulación de datos, operan sobre la clase abstracta ManipuladorDatos en lugar de acoplarse a clases específicas como OrdenadorDatos. Dentro del módulo gestiondedatos, la clase Columna (un componente esencial para la estructura de datos) no depende directamente de validadores de tipo concretos (ValidadorBoolean, etc.). En su lugar, interactúa con la clase ValidadorTipoDato, la cual, a su vez, se basa en la interfaz TipoValidador para delegar la validación específica. Esta inversión de dependencias reduce el acoplamiento directo entre los componentes, lo que favorece significativamente la flexibilidad, extensibilidad y mantenibilidad.

SUBSISTEMA "MÓDULO GESTIÓN DE DATOS"



Este diagrama de clases detalla la estructura interna y el comportamiento del módulo gestiondedatos, encargado de la representación, la gestión fundamental y la validación de tipos de datos dentro de las estructuras tabulares de la librería.
Las clases e interfaces principales que lo componen son: Tabla, Columna, Celda, Etiqueta (interfaz) y sus implementaciones (EtiquetaEntero, EtiquetaString), el enumerable TipoDato, la interfaz TipoValidador y sus implementaciones (ValidadorBoolean, ValidadorNA, ValidadorNumerico, ValidadorString), y la clase ValidadorTipoDato.

La clase Tabla actúa como la entidad central para la representación de datos tabulares. Contiene como atributo una colección de objetos Columna. Sus métodos permiten la manipulación de columnas (como inserción, eliminación, modificación), el acceso a datos por etiquetas o índices, y la consulta de las dimensiones de la tabla.

La clase Columna representa una columna específica dentro de una Tabla. Posee una Etiqueta (para su identificación), un TipoDato (que define el tipo de datos que debe contener) y una colección de objetos Celda. Sus métodos están orientados a la gestión de las celdas que la componen y a asegurar la consistencia de tipos, utilizando el sistema de validación de tipos centralizado.

La clase Celda es la unidad mínima de almacenamiento de datos dentro de la tabla. Contiene un valor de tipo Object y está asociada a una instancia de TipoDato, que define el tipo de dato esperado para ese valor.

La clase TipoDato es un enumerable que define los tipos de datos permitidos en el sistema: numérico, booleano, cadena y NA (valor especial que indica datos faltantes). Estos tipos son fundamentales para establecer la estructura de las columnas y validar el contenido de las celdas.

La interfaz TipoValidador establece el contrato para la validación de un valor específico. Sus implementaciones (las clases concretas como ValidadorBoolean, ValidadorNA, ValidadorNumerico y ValidadorString) implementan esta interfaz, proporcionando la lógica de validación específica para cada tipo de dato.

La clase ValidadorTipoDato actúa como un punto centralizado y un orquestador para la validación de tipos. Emplea un mecanismo de mapeo estático para asociar cada enumeración TipoDato con su implementación de TipoValidador correspondiente. Proporciona una funcionalidad estática que se encarga de verificar la compatibilidad de un valor con los tipos de datos esperados para una columna y para una celda, delegando la

validación específica al TipoValidador adecuado. Además, gestiona la lógica para valores nulos, considerándolos válidos únicamente si el tipo de la celda es NA.

La interfaz Etiqueta actúa como un contrato que asegura la consistencia en la gestión de identificadores. Las clases EtiquetaString y EtiquetaEntero implementan esta interfaz, proporcionando tipos de etiquetas basados en cadenas de texto y números enteros respectivamente. Las clases Tabla y Columna tienen una asociación con la interfaz Etiqueta para sus respectivas etiquetas, permitiendo el polimorfismo en el manejo de identificadores.

RELACIONES ENTRE CLASES:

Tabla y Fila / Columna:

La clase Tabla se relaciona con la clase Columna mediante una relación de composición. Esto significa que una instancia de Tabla contiene una colección de objetos Columna, y si la tabla se elimina, las columnas también se eliminan. Esta relación garantiza que las columnas no existan de forma independiente a la tabla que las contiene.

MULTIPLICIDAD:

- Una instancia de *Tabla* se relaciona con una o muchas (1…*) instancias de *Columna*.
- Cada instancia de *Columna* se relaciona con una única instancia de clase *Tabla*.

Columna y Celda:

La clase Columna se relaciona con la clase Celda mediante una relación de composición. Una columna está compuesta por una colección de celdas, y estas no pueden existir fuera de su columna, lo que refuerza la integridad de la estructura de datos.

MULTIPLICIDAD:

- Una instancia de *Columna* se relaciona con una o muchas (1…*) instancias de *Celda*.
- Cada instancia de *Celda* está asociada a una única instancia de *Columna*.

Columna y ValidadorTipoDato:

La clase Columna mantiene una relación de uso (dependencia) con la clase ValidadorTipoDato.
Al insertar un valor, Columna invoca métodos de ValidadorTipoDato para comprobar que el valor sea compatible con el tipo de dato esperado.
Esta relación indica que Columna utiliza los servicios de ValidadorTipoDato, pero no es responsable de su creación ni de su ciclo de vida.

Celda y TipoDato:

La clase Celda mantiene una asociación con el enumerable TipoDato.
Cada celda almacena un valor que tiene un tipo de dato determinado (numérico, booleano, cadena o NA). Esta asociación establece que una celda puede vincularse a uno de los tipos definidos en TipoDato.

Etiqueta y sus Implementaciones:

La interfaz Etiqueta es implementada por las clases EtiquetaString y EtiquetaEntero. Esta interfaz actúa como contrato que restringe los tipos de etiquetas utilizados en las clases Tabla, Fila y Columna, garantizando que sean exclusivamente de tipo String o entero.

IMPLEMENTACIÓN

Junto a este documento se adjunta el archivo comprimido que contiene todo el código fuente de la librería desarrollada a lo largo de este trabajo práctico.

Además, el código de implementación se encuentra disponible en un repositorio online de GitHub. Este repositorio fue la herramienta fundamental para la gestión de versiones (utilizando Git), la colaboración y la organización del equipo durante todo el proceso de desarrollo del proyecto. Es posible visitar el repositorio para acceder a la última versión del código, consultar el historial de cambios o clonar el proyecto.

Por último, se adjunta la documentación técnica de los módulos y métodos más importantes y relevantes del sistema.