Informe de Decisiones de Diseño

Introducción

Este sistema de gestión académica fue diseñado para demostrar conceptos fundamentales de Programación Orientada a Objetos (POO) en Java, como herencia, polimorfismo, interfaces funcionales, y manejo de excepciones. Su desarrollo forma parte del Trabajo Práctico Integrador (TPI) con el objetivo principal de implementar un sistema versátil y robusto que aplique estos conceptos de manera práctica.

Estructura del Sistema

Organización en Paquetes

El código se organizó en paquetes según las responsabilidades funcionales, facilitando la modularidad y la escalabilidad:

- excepciones: Contiene las excepciones personalizadas para gestionar errores específicos.
- gestores: Proporciona clases para manejar recursos, como GestorRecursos.
- interfaces: Define contratos funcionales con interfaces como Clasificable y Evaluable.
- recursos: Contiene las clases relacionadas con los diferentes tipos de recursos académicos (Libro, Articulo, TrabajoInvestigacion).

Jerarquía de Clases

Se diseñó una clase abstracta RecursoAcademico como base para todos los recursos. Define atributos y métodos comunes a los recursos académicos. Las clases hijas (Libro, Articulo, TrabajoInvestigacion) heredan de esta clase y añaden comportamientos específicos, cumpliendo el principio de polimorfismo.

Decisiones Clave

Informe de Decisiones de Diseño

Uso de Herencia y Polimorfismo

- La jerarquía de clases aprovecha la herencia para compartir atributos y métodos comunes.
- Las clases derivadas implementan métodos abstractos como calcularRelevancia y mostrarDetalles.
- El polimorfismo permite tratar diferentes tipos de recursos de manera uniforme, simplificando la lógica del sistema.

Implementación de Interfaces Funcionales

- Se usaron interfaces funcionales como FiltroRecurso para habilitar expresiones lambda y aumentar la flexibilidad del código.
- Los métodos de GestorRecursos emplean estas interfaces para realizar operaciones dinámicas como ordenamiento, filtrado y generación de informes.

Gestión de Recursos

- La clase GestorRecursos centraliza la administración de los recursos. Utiliza expresiones lambda para filtrar y ordenar, mejorando la modularidad y permitiendo futuras ampliaciones.

Manejo de Excepciones Personalizadas

- RecursoNoEncontradoException: Maneja recursos inexistentes.
- CategoriaInvalidaException: Gestiona errores al asignar categorías.
- LimiteRecursosException: Restringe el número máximo de recursos permitidos.

Estas excepciones refuerzan el control de errores y aseguran una ejecución predecible.

Funcionalidades Principales

Informe de Decisiones de Diseño

El sistema ofrece las siguientes funcionalidades clave:

- Registro de recursos académicos, permitiendo diferentes tipos de datos según el recurso.
- Clasificación de recursos mediante la interfaz Clasificable.
- Evaluación de relevancia y generación de informes detallados.
- Filtrado avanzado usando expresiones lambda y streams.

Conclusión

El diseño del sistema refleja una implementación sólida de conceptos de programación orientada a objetos. La correcta organización en paquetes, uso de herencia, manejo de excepciones y expresiones funcionales asegura un código mantenible y extensible. Este enfoque respalda la gestión eficiente de recursos académicos y destaca buenas prácticas de desarrollo en Java.