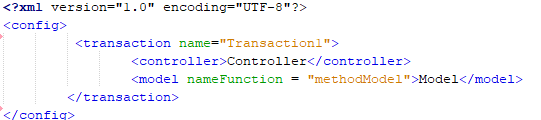
**Documentación framework MVC**

**Descripción uso del framework**

**Objetivo principal:**

Representar el patrón arquitectónico MVC, sin notarse explícitamente en el código, el framework consiste en ejecutar un método de una clase que usted considere parte del modelo o el modelo mismo conociendo el nombre completo de la clase y el método en específico a ejecutar, cabe destacar que en dicho método del modelo podrá acceder al controlador que se tenga definido en el archivo.

Toda esta información estará incluida en un archivo de configuración de tipo XML como el siguiente:



**Funcionalidades:**

1. Ejecutar una transacción definida en el archivo de configuración por su nombre definido en el atributo name
2. Ejecutar método de una clase modelo la cual dentro de las etiquetas <model>nombre de la clase</modelr> se encontrará el nombre completo de la clase la cual hace referencia al modelo y el tributo nameFunction sirve para especificar que método se ejecutará en base a su nombre.
3. Utilizar la clase controlador-definida en <controller>nombre de la clase</controller> dentro del método del modelo.
4. Definir el comportamiento de un archivo bitácora, por el archivo *configLog.properties*
5. Guardar información de la ejecución de las transacciones.

**Ejemplo de uso:**

**Descripción de componentes**

*Nombre:* ManagerReflection

*Descripción:* Componente encargado de ejecutar los métodos de las clases controlador y modelo mediante la API de Java “Reflection”

Dependencias con otros componentes: Ninguno

*Interfaces de Salida:*

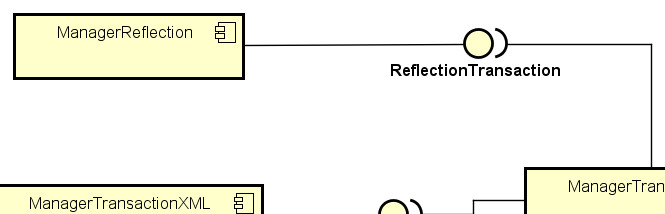
Interface ReflectionProcessor

* getInstanceClass
* runMethodModel

Interfaces de Entrada: Ninguna

Artefactos: API Reflection java

*Diagrama:*



*Nombre:* ManagerXMLTransaction

*Descripción:* Componente encargado de leer la información del archivo de configuracion config.xml, transforma todas las transacciones definidas a una lista de objetos de tipo Transaction

Dependencias con otros componentes: Ninguno

*Interfaces de Salida:*

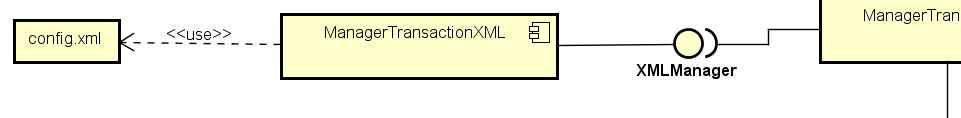
Interface XMLManager

* readXMLFile

Interfaces de Entrada: Ninguna

Artefactos: archivo config.xml

*Diagrama:*



*Nombre:* LogManagerTransaction

*Descripción:* Componente encargado de administrar el archivo log#.txt el cual escribe en un formato sencillo la fecha de ejecución de una transacción, la transacción ejecutada y todas las transacciones disponibles.

Por defecto el nombre del primer archivo es log0.txt

De igual manera revisa de un archivo de configuración el tamaño máximo del archivo y si se encuentra activado el log, si un archivo sobrepasa el límite de tamaño se divide en otro archivo el cual cambia a ser llamada log1.txt cada vez que se sobrepasa avanza en una unidad su nombre log2.txt…logn.txt.

Dependencias con otros componentes: Ninguno

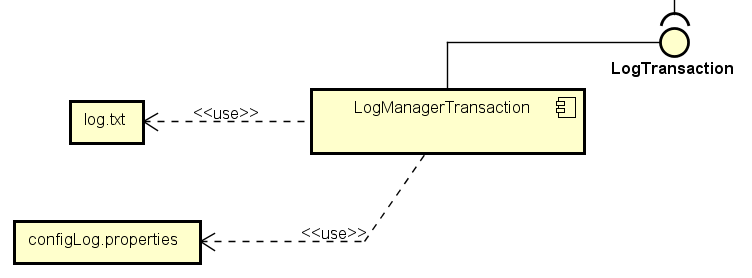
*Interfaces de Salida:*

Interface LogTransaction

* writeLogFile

Interfaces de Entrada: Ninguna

Artefactos: archivos log#.txt, configLog.properties

*Diagrama: *

*Nombre:* ManagerTransactions

*Descripción:* Componente encargado de ejecutar una transacción que se solicite por su nombre, el flujo es muy sencillo:

1. Encuentra la transacción solicitada.
2. Genera una instancia del controlador definido.
3. Luego invoca el método de una clase considerada modelo, que pasa como argumento, la vista donde se ejecuta la transacción, el controlador y algún objeto como argumento.
4. Para después guardar en el log la información de la transacción realizada.

Dependencias con otros componentes: ManagerReflection, ManagerXMLTransaction, ManagerLog

*Interfaces de Salida:*

Interface TransactionExecutor

* executeTransaction

Interfaces de Entrada:

Interface ReflectionTransaction

* getInstanceClass
* runMethodModel

Interface XMLManager

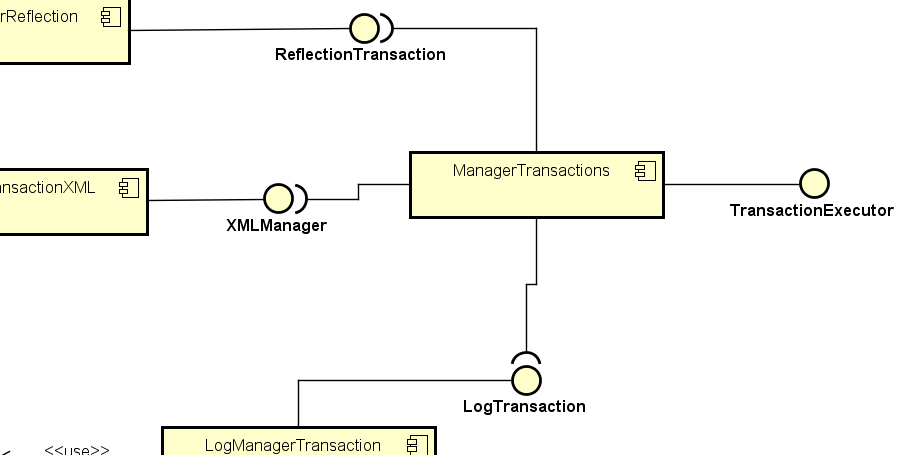
* readXMLFile

Interface LogTransaction

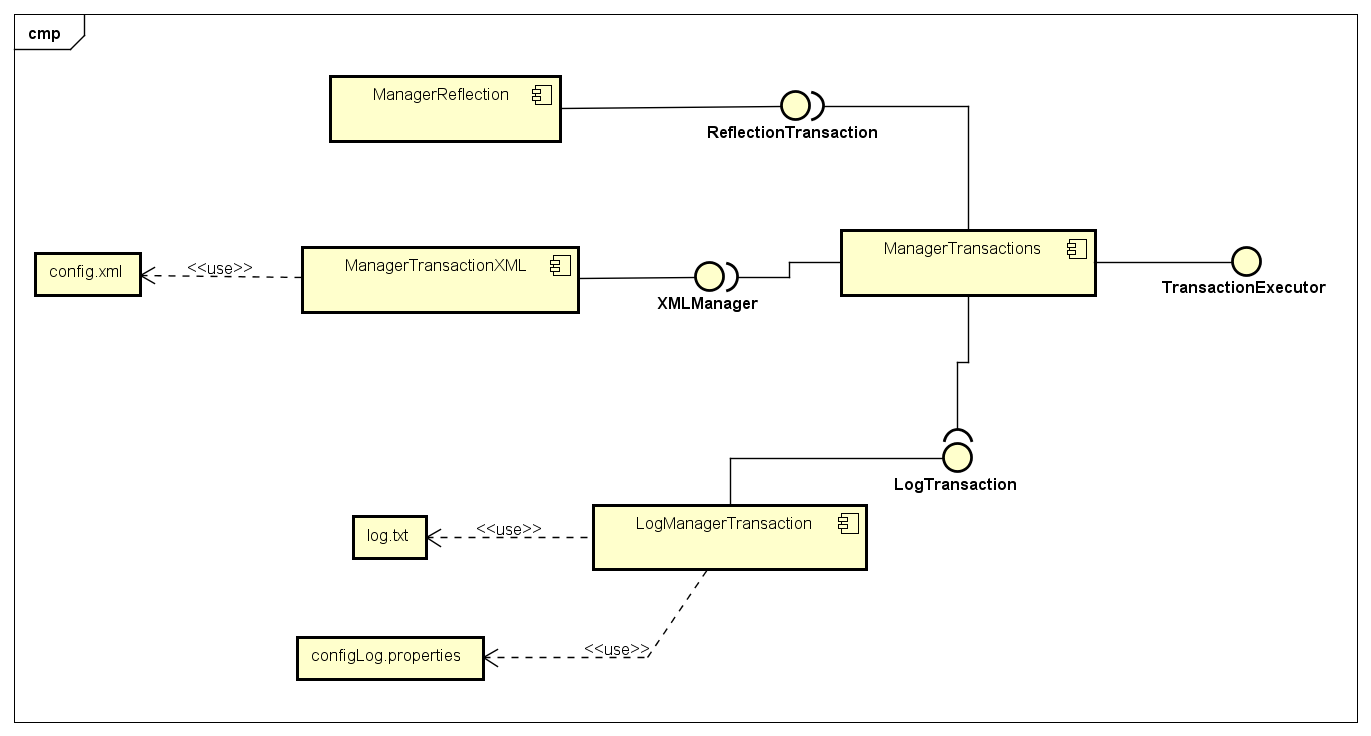
* writeLogFile

Artefactos: ninguno

*Diagrama:*

**

*Diagrama total de componentes:*

**

Descripcion de clases

Nombre de la clase:

Descripción:

Dependencias con otras clases: <<Listar las asociaciones, nombre y descripción (¿para qué se tiene dicha asociación?)>>.

Atributos: Enumerarlas y adicionar el nombre, tipo, visibilidad, valor por omisión y descripción.

Funciones: Enumerarlas y adicionar el nombre, listado de argumentos con su tipo, valor de retorno, visibilidad, sí es función pública mencionar el servicio que esta implementando (componente e interface de salida) y descripción.

Añadir diagrama de clases.

Descripcion de las secuencias

Agregar dos conjuntos de diagramas de secuencia, el primero orientado a los desarrolladores que utilizarán el framework en sus proyectos, el segundo conjunto serán diagramas de secuencia que describan la interacción de los objetos externos con los internos de la clase del framework, objetos externos son aquellos que los desarrolladores que usa el framework crean, los objetos internos son aquellos que son creados al interior del framework. Estos diagramas de secuencia serán para desarrolladores qué hará uso del Framework y aquellos que requieren saber la funcionalidad, al interior, del misma. Cada diagrama de secuencia tendrá asociado un texto donde explicará cada paso que se muestre. Los diagramas de secuencia a realizar serán definido en conjunto con el profesor.

Video de ejemplo:

Describir un ejemplo de uso del componente, es deseable que se añada un video, si es así en la documentación se tendrá un liga a youtube.