# DOCUMENTACIÓN

## Planteamiento del problema

Una empresa dedicada a la creación de videojuegos necesita crear un nuevo juego de preguntas/respuestas de naturaleza similar al trivial clásico.

* Es primer lugar es necesario procesar los datos relativos a las preguntas y respuestas de las que constará el juego. Durante este primer procesamiento la aplicación debe informarnos de los posibles errores en los ficheros de entrada. Estos ficheros de entrada deberán de usar el formato GIFT, pero posteriormente la aplicación podría requerir del uso de ficheros de entrada con otros formatos como QTI o XML. Una vez procesados los datos han de tener una representación interna JSON en nuestra aplicación.
* Posteriormente estos datos se serializaran en una base de datos no relacional como MongoDB aunque esto es una decisión aun por tomar y podría haber variaciones.

Los pasos anteriores deberán estar claramente diferenciados para facilitar que una automatización o un operario tengan la posibilidad de ejecutar estas etapas en momentos diferentes.

Por el momento no se requiere que ninguna de estas dos etapas ofrezcan un gran rendimiento, ya que solamente van a ser utilizadas en las fase de construcción de una base de preguntas/respuestas.

## Metodología usada

El estudio y diseño de esta arquitectura se llevara a cabo mediante la utilización del método Atribute-Driven-Design (ADD) y el estándar del SEI (ANSI/IEEE 1471, 2000).

El método ADD esta basado en la identificación de los atributos de calidad y la creación de una serie de escenarios que esos atributos deben cumplir.

## Identificación de los interesados (Stakeholders)

## Identificación inicial de los atributos de calidad

## Lista de interesados (Stakeholders)

## Lista de atributos de calidad

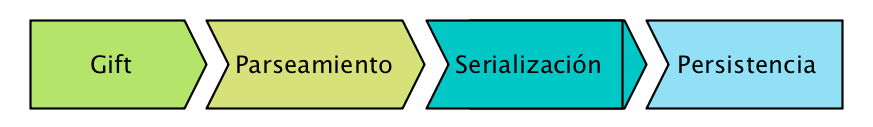
## Atributos de calidad e interesados

## Descripción de negocio de la solución

La solución que planteamos esta compuesta de varios módulos integrados todos ellos en una aplicación desarrollada en el lenguaje Java.

* Parser: el modulo encargado de la lectura y transformación a un formato de representación interna en nuestra aplicación de los datos obtenidos a partir de los ficheros de entrada en formato GIFT, QTI o XML.
* Serializador: este modulo es el encargado de pasar a formato JSON los datos que obtuvimos en la fase anterior.
* Modulo de persistencia: el modulo encargado de hacer los datos obtenidos en la fase anterior persistentes. Dicha persistencia se lleva a cabo en la base de datos no relacional MongoDB.

Cada uno de los módulos anteriormente descritos representan una etapa de las que se compone nuestra aplicación BATCH de procesamiento por lotes.



* Funcionamiento del modulo parser.

Recibe los ficheros que contienen las preguntas en uno de los formatos aceptados para procesarlos y convertirlos a datos que podamos usar en nuestra aplicación. También se encarga de avisarnos si se encuentra algún error en los datos de entrada durante su procesado.

Cuando los datos de los ficheros de entrada estén completamente procesados estos datos ya estarán listos para ser enviados al modulo de serialización.

* Funcionamiento del modulo de serialización.

Este modulo depende de los datos procesados por el parser para su funcionamiento. Una vez recibe esos datos se encarga de serializar estos datos al formato JSON. Una vez transformados a JSON los datos ya estarán preparados para ser procesados en la siguiente etapa, la persistencia.

* Funcionamiento del modulo de persistencia.

Recibe una serie de datos JSON que mediante una implementación de los métodos de la interfaz JsonSerial serán guardados en la base de datos elegida, MongoDB.

Como se puede observar esta solución elegida se pretende asemejar al estilo arquitectónico de procesamiento por lotes tradicional, el cual consiste en una serie de etapas bien diferenciadas que se ejecutan de manera secuencial, al depender las etapas del resultado obtenido de etapas anteriores.

## Escenarios de calidad