Grupo Trivial 3a

UNIVERSIDAD DE OVIEDO  - Arquitectura del Software

**DOCUMENTACION TRIVIAL 3a**

Versión 2.0

# **DOCUMENTACIÓN**

## **Ámbito y alcance del problema**

En esta primera toma de contacto con el problema nos encuadramos ante un entorno vacío, con limitaciones básicas como son las de crear.

Como limitaciones del parser están los formatos de salida propuestos, básicamente json y opcionalmente xml (ambos desarrollados), y un solo formato de entrada GIFT.

El alcance del problema será obtener el equivalente en el formato de salida, dado el fichero gift con los datos.

Opcionalmente se plantea un entorno de persistencia como MongoDB.

## **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)**

Este proyecto esta nombrado como “Trivial3a” por lo que los interesados están contextualizados dentro del dicho proyecto.

En este caso los interesados son:

* **Responsables de la empresa:** Los dirigentes que toman las decisiones sobre el proyecto, presupuesto y evolución.
* **Usuario administrador:** estará en contacto con el módulo de parseo para utilizarlo a la hora de cargar datos o introducir nuevos datos.
* **Equipo de desarrollo:** el diseño modular permitirá edificar la aplicación de modo que este módulo no se vea influido por el desarrollo de otros módulos.

### *Responsables de la empresa*

Se trata de los equipos directivos de la corporación, son responsables de los presupuestos y toman las decisiones que comprometen fondos de dicho presupuesto. En nuestro caso los profesores. Entre sus objetivos están:

* Bajo coste de desarrollo, esto es, el desarrollo del proyecto debe ser corto y con un coste reducido.
* Guiar el proceso de evolución de la aplicación.

### *Usuario administrador*

El/los usuarios administrador/es son un grupo de usuarios encargados del mantenimiento de la aplicación, en este módulo se encargaran de la carga del fichero mediante la interfaz del parser de modo que su trabajo será simple e intuitivo, y realizado de forma secuencia por la estructura BATCH. Entre sus objetivos:

* Trabajo lo más simple posible, con el menor número de fallos posibles.
* Aprendizaje simplificado e facilidad de la interfaz.

### *Equipo de desarrollo*

Todos los miembros del equipo, afectados de manera directa pues este módulo será una de las partes básicas de la aplicación a construir. Lo realizaremos de forma independiente para que los módulos puedan cambiarse y modificarse sin necesidad de alterar el resto de módulos. Por tanto sus objetivos simplificados son:

* Simplificar al máximo y modularizar el parser para mejorar la integración en el sistema y el desarrollo de otros módulos.
* Proyecto rentable, esto es, que permita ser desarrollado por el precio establecido con un grado de rentabilidad que haga atractivo el desarrollo. En nuestro caso obtener la mayor nota con el mínimo esfuerzo posible.

## Identificación inicial de los atributos de calidad

## **Lista de interesados (Stakeholders)**

|  |  |  |
| --- | --- | --- |
| **Código** | **StakeHolder** | **Intereses** |
| **ST-01** | Responsables de la empresa | *Bajo coste de desarrollo, esto es, el desarrollo del proyecto debe ser corto y con un coste reducido. Eficiencia e independencia del parser.* |
| **ST-02** | Usuario administrador | *Simplificación de aprendizaje y utilización. Fácil interacción con la interfaz.* |
| **ST-03** | Equipo de desarrollo | *Proyecto escalable y modularizado. Reparto de tareas y desarrollo eficiente.* |
|  |  |  |

## **Atributos de calidad e interesados**

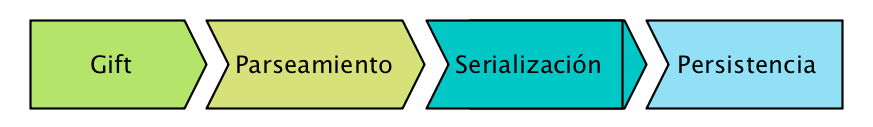
|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos vs Interesados** | **ST-01** | **ST-02** | **ST-03** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 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 módulo 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 representa una etapa de las que se compone nuestra aplicación **BATCH** de procesamiento por lotes.



* Funcionamiento del módulo 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 módulo de serialización.

* Funcionamiento del módulo 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

## Vistas