Grupo Trivial 3a

UNIVERSIDAD DE OVIEDO  - Arquitectura del Software

**DOCUMENTACION TRIVIAL 3a**

Versión 2.0

# **DOCUMENTACIÓN – Parte 1**

## **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.

## **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.* |
|  |  |  |

## **Lista de atributos de calidad**

* AT001 Facilidad de cambio de los mecanismos de parseo y/o almacenamiento de datos por si es necesario modificarlos en el futuro. Modificabilidad.
* AT002 Escalabilidad del sistema para poder añadirlo a otro tipo de aplicaciones. Modificabilidad.
* AT003 Facilidad para probar el correcto funcionamiento del sistema, garantía de que los datos se han procesado correctamente. Testabilidad.
* AT004 Tiempo de desarrollo corto. Time to markey.
* AT005 Facilidad de uso de la aplicación. Usabilidad.
* AT006 Coste de desarrollo bajo. Coste-Beneficio.
* AT007 Protección del equipo que ejecute la aplicación. Confiabilidad.
* AT008 Garantizar que la aplicación procese los datos correctamente y en un tiempo óptimo. Rendimiento

## **Atributos de calidad e interesados**

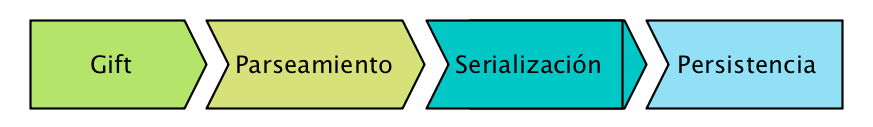
|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos vs Interesados** | **ST-01** | **ST-02** | **ST-03** |
| AT001 |  |  | X |
| AT002 | X |  | X |
| AT003 |  |  | X |
| AT004 | X |  | X |
| AT005 |  | X |  |
| AT006 | X |  |  |
| AT007 | X | X |  |
| AT008 | X |  |  |

## 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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Escenario Nº** | **Fuente de estímulo** | **Estímulo** | **Entorno** | **Artefacto** | **Respuesta** | **Medición de la respuesta** | **Atributo de calidad afectado** |
| 1 | Nuevo mecanismo de parseo | Parseo de datos | Explotación | Parser de datos | Actualización de parser | 100% de preguntas procesadas correctamente | AT001 |
| 2 | Nuevo sistema a integrar | Cambio de sistema que utiliza el parser | Explotación | Parser de datos y nuevo módulo donde integrarlo | Sincronización de ambos módulos | 0% de fallos en la sincronización | AT002 |
| 3 | Central control | Recepción de datos procesados | Explotación | Parser de datos y funcionalidad de testeo | Fichero procesado a Json | El 100% de las preguntas del fichero están correctamente procesadas | AT003 |
| 4 | Sistema en desarrollo | Fin del desarrollo | Desarrollo | Parser de datos | Tiempo de desarrollo | Antes del día 4 de Marzo | AT004 |
| 5 | Uso del sistema | Uso por usuario final | Explotación | Parser de datos | Opinión de usuarios beta-tester | 90% de votos a favor de la aplicación | AT005 |
| 6 | Sistema en desarrollo | Fin del desarrollo | Desarrollo | Parser de datos | Coste de desarrollo | Sin coste monetario | AT006 |
| 7 | Uso del sistema | Cualquier momento de uso del sistema | Explotación | Parser de datos e interfaz del mismo | Medidas estándar de protección | No hay amenaza potencial | AT007 |
| 8 | Uso del sistema | Uso ordinario | Explotacion | Parser de datos | Relación rendimiento óptima | Medidas de Benchmark de menos de 20s | AT008 |

# **DOCUMENTACIÓN – Parte 2**

Ámbito y alcance del problema

Para esta segunda practica se nos plantea un problema ligeramente diferente y mas completo al planteado en la primera entrega. Los requisitos han sido ampliados de manera sustancial con el objetivo de lograr una aplicación completamente funcional para su uso en el escritorio.

Las limitaciones de esta segunda entrega vienen a ser que la aplicación solamente funcionara en el escritorio, siendo además necesario que los jugadores estén registrados en el juego para poder comenzar una partida.

En cuanto al alcance del problema nos encontramos con que tenemos que llevar a cabo el desarrollo de una aplicación completamente funcional que permita jugar partidas de trivial a diferentes usuarios y con diferentes modos de juego. Además se plantea que han de registrarse una serie de datos de interés acerca de las partidas jugadas por cada usuario con el objetivo de conocer datos estadísticos sobre su rendimiento durante las partidas.

Planteamiento del problema

En esta ocasión la empresa NoGame nos plantea la continuación de la primera parte del juego ya desarrollada. Esta vez el desarrollo consistirá en la realización de una aplicación completa de escritorio.

Dicho aplicación/juego tendrá interfaz grafica y permitirá que los usuarios se registren y loguen en la aplicación. Dado que cada usuario quedara guardado en el sistema también se guardaran datos relativos a las partidas que haya jugado. De esta forma podremos llevar a cabo un análisis estadístico de su rendimiento durante el juego. En la interfaz grafica se mostrara un tablero similar al tablero de trivial clásico con el objetivo de que los usuarios puedan conocer en que posiciones se encuentran y sus puntuaciones. Aparte de esto existirá un usuario administrador que podrá consultar las estadísticas y las puntuaciones de todos los jugadores registrados en el sistema.

La interfaz grafica llevara a cabo la representación visual de todos los datos contenidos en la base de datos utilizada por la aplicación, tales como usuarios registrados, preguntas, etc

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. Sus tareas son hacer que el sistema sea tan fácil de modificar y escalar como sea posible. Dicho objetivo viene motivado por el intento de conseguir que el coste de desarrollo sea el mínimo posible.
* Usuario administrador:
  + El administrador o administradores son los encargados de tratar con la aplicación de una forma mas amplia que los usuarios corrientes. Tienen la función de controlar que la aplicación este funcionando bien así como que no se producen efectos inesperados en las estadísticas durante el funcionamiento del juego.
* Equipo de desarrollo:
  + Los programadores tratan de lograr un diseño modular que permita construir la aplicación del modo mas independiente posible para que así los módulos existentes no se vean influidos por el desarrollo de otros módulos.
* Jugadores / Usuarios de la aplicación:
  + Los usuarios de la aplicación son aquellas personas interesadas en descargar y dar uso al juego que hemos desarrollado. Son la clave para que una aplicación triunfe o no por lo que el diseño ha de ser tan sencillo como sea posible. Además el usuario tiene que encontrar entretenida la aplicación porque si no lo hace dejara de usarla.

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 de los distintos componentes del juego para poder ampliar funcionalidad en un futuro sin que esto suponga un coste elevado. |
| ST-02 | Usuario administrador | Simplificación de aprendizaje y utilización. Fácil interacción con la interfaz.  Rápida detección de posibles problemas y rápida y sencilla solución de ellos |
| ST-03 | Equipo de desarrollo | Proyecto escalable y modularizado. Reparto de tareas y desarrollo eficiente. |
| ST-04 | Jugadores / Usuarios de la aplicación | Facilidad de uso ante todo.  Entretenimiento además de buena experiencia de uso. |

Lista de atributos de calidad

|  |  |  |
| --- | --- | --- |
| Código | Tipo de atributo | Descripción |
| AT001 | Posibilidad de cambio del look&feel del juego. | Modificabilidad |
| AT002 | Posibilidad de modificar el tipo de tablero así como el numero de jugadores que participan en una partida. | Modificabilidad |
| AT003 | Garantizar que la aplicación procese los datos correctamente y en un tiempo  óptimo | Rendimiento |
| AT004 | Las estadísticas de cada usuario solo pueden ser conocidas por el mismo y por los administradores del juego | Confidencialidad |
| AT005 | Coste de desarrollo bajo. Coste-Beneficio | Coste-Beneficio |
| AT006 | Facilidad de uso de la aplicación | Usabilidad |
| AT007 | Facilidad de visualización de las estadísticas | Usabilidad |
| AT008 | Evitar la inconsistencia de los datos por accesos simultáneos | Seguridad |
| AT009 | Los usuarios no registrados y logeados no podrán acceder a la aplicación | Seguridad |
| AT010 | Las BBDD han de funcionar a tiempo completo sin producirse fallos de lectura/escritura | Disponibilidad |
| AT011 | Tiempo de desarrollo corto | Time to market |
| AT012 | El juego ha de estar dividido en componentes lo suficientemente pequeños como para ser reutilizados en otras aplicaciones de la empresa | Adaptabilidad |
| AT013 | La aplicación debe de ser fácilmente probada con el objetivo de encontrar posibles bugs lo mas rápida y fácilmente posible. | Testeabilidad |

Atributos de calidad e interesados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributos vs Interesados** | **ST-01** | **ST-02** | **ST-03** | **ST-04** |
| **AT001** | X |  |  | X |
| **AT002** | X |  |  | X |
| **AT003** | X |  |  | X |
| **AT004** | X | X |  | X |
| **AT005** | X |  |  |  |
| **AT006** |  |  |  | X |
| **AT007** |  | X |  |  |
| **AT008** | X | X | X | X |
| **AT009** | X | X |  |  |
| **AT010** | X |  |  | X |
| **AT011** | X |  |  |  |
| **AT012** | X |  | X |  |
| **AT013** |  |  | X |  |

Escenarios de calidad.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Escenario Nº*** | ***Fuente de estímulo*** | ***Estímulo*** | ***Entorno*** | ***Artefacto*** | ***Respuesta*** | ***Medición de la respuesta*** | ***Atributo de calidad afectado*** |
| **1** | Usuario | Comienzo de una partida | Explotación | Sistema | El sistema muestra el tablero elegido y le presenta al usuario preguntas | Tiempo no superior a 1 segundo desde que se lleva a cabo la acción | AT009  AT006  AT001 |
| **2** | Usuario | Seleccionar una apariencia de tablero distinta | Explotación | Interfaz | Cambio de la interfaz dependiendo de la opción seleccionada | Tiempo no superior a 3 segundos desde que se lleva a cabo la acción | AT006  AT002  AT001 |
| **3** | Usuario | Acceso a sus estadísticas de juego | Explotación | Sistema | El sistema muestra al usuario las estadísticas de sus juegos | Datos bien calculados a partir de los datos almacenados sobre el usuario | AT010  AT007  AT004 |
| **4** | Administrador de la aplicación | Acceso al panel de estadísticas globales | Explotación | Sistema | Se presentan en pantalla los datos relativos a las estadísticas globales del juego | Datos bien calculados a partir de los datos almacenados sobre las partidas jugadas por los usuarios | AT010  AT007  AT006  AT004 |
| **5** | Directivos | Cambio de las reglas del juego | Desarrollo | Sistema | Cambio de la funcionalidad de la aplicación hasta que cumpla con lo pedido | Coste desarrollo < 6000€  Tiempo desarrollo < 15 días  Reutilización de módulos ya implementados | AT011  AT012  AT005 |
| **6** | Directivos | Ampliación de la funcionalidad del juego | Desarrollo | Sistema | Implementación de las nuevas características requeridas | Coste desarrollo < 9000€  Tiempo desarrollo < 45 días | AT011  AT012  AT005 |
| **7** | Usuario | Juego simultaneo de varios usuarios | Explotación | MongoDB | No se pueden producir fallos de lectura ni dejar la BBDD en un estado inconsistente | Sin errores.  Tiempo de lectura < 500ms  Tiempo de escritura < 800ms | AT008  AT003 |
| **8** | Herramientas de desarrollo | Ejecución de test | Explotación | Modelo de datos.  Lógica de Negocio | El sistema nos produce unos resultados de todo lo ocurrido durante la ejecución de los test | Test ejecutados con cada build.  Cobertura de los test > 50%  Test todos en verde | AT013  AT011  AT012 |

Descripción de negocio de la solución

Para el desarrollo de esta segunda ampliación de nuestra aplicación Trivial hemos optado por implementar una solución basada en la arquitectura MVC (Modelo – Vista – Controlador). Esta decisión ha sido tomada en base a que se deja intuir que el próximas versiones del juego la representación visual del mismo variará. Con MVC solucionamos de un plumazo el problema de tener que hacer una aplicación diferente para cada tipo de vista, ya que lo que vamos a tener de esta manera será un modelo que no variará (en la medida de lo posible) junto a un controlador que es el encargado de comunicar el modelo con las diferentes vistas que va a tener el juego.



Al haber seguido este modelo de manera mas o menos fiel la estructura del proyecto resultante esta compuesta de las siguientes partes (entre otras):

* Modelo
  + Aquí esta representada la lógica de negocio y el estado. Es independiente del controlador y de las distintas vistas
* Business / Controlador
  + Esta parte es la encargada de procesar los eventos de los jugadores y realizar las acciones acordes a esos eventos.
* Presentación / Vista
  + Muestra los contenidos del modelo

Además de las tres partes principales que ya han sido nombradas se usan otra seria de patrones del software y metodologías con la intención de que nuestra aplicación posea un alto grado de cohesión y poco acoplamiento.

Tecnologías utilizadas.

* Java: Lenguaje de programación principal elegido por su robustez y la base de conocimiento de la que los miembros del equipo tenemos
* Eclipse: Entorno de desarrollo que hemos utilizado para desarrollar la aplicación en combinación con algunos plugins como el plugin de Git y el de Gradle
* Github**:**  es un repositorio de Git basado en la web para alojar proyectos utilizando dicho sistema de [control de versiones](http://es.wikipedia.org/wiki/Control_de_versiones).
* MongoDB: Base de datos documental Open-source que provee un gran rendimiento y una alta disponibilidad además de ser fácilmente escalable.
* LibGDX: Framework hecho en java para el desarrollo de videojuegos que provee una API unificada que funciona en todas las plataformas que soporta. Permite un entorno para el prototipado rápido ya que se puede ejecutar y debuguear el juego sobre escritorio de forma nativa.
* Travis CI: es un servidor de integración continua que esta integrado con github y ofrece soporte para un gran numero de lenguajes
* Cucumber: Framework de pruebas que permite a los equipos de desarrollo describir como debe comportarse el programa en texto plano. Este texto es escrito en un lenguaje de dominio especifico en el que resulta fácil leer los términos de negocio y sirve de documentación.
* Gradle: es una herramienta software para automatizar la construcción de nuestros proyectos, como las tareas de compilación, testeo, empaquetado y despliegue.
* GSON: Es una librería java que puede ser usadas para convertir objetos java a su representación JSON. También puede ser usada para convertir desde un Sting JSON a su objeto java equivalente.

Vistas

En esta parte de la documentación pasamos a comentar las diferentes vistas que hemos realizado con el objetivo de representar de forma visual las diferentes partes de la aplicación y que además nos sirven para entender de una forma mas intuitiva el funcionamiento del juego en su conjunto.