1. Planteamiento del problema

La empresa NoGame dedicada a la creación de videojuegos quiere crear una variante del juego Trivial, aunque la idea es que puedan crear más juegos similares en el futuro.

Inicialmente se creará una base de datos MongoDB con preguntas suficientes para poder abastecer a los diferentes juegos. Para ello, construirán una aplicación que lea ficheros con las preguntas y respuestas, procese dichas preguntas indicando si hay errores o no y las almacene en la base de datos.

Los ficheros se leerán de bancos de preguntas, con formato GIFT, ya existentes. La aplicación utilizará una representación interna de las preguntas en formato JSON lo cual facilita su posterior almacenamiento en la base de datos.

La aplicación se ejecutará en dos etapas. Una primera etapa analizará los ficheros con las preguntas y generará el formato JSON; y otra etapa tomará las preguntas en JSON y las almacenará en la base de datos MongoDB. La ejecución de estas 2 etapas estará controlada por un operario de la compañía que podrá decidir cuándo ejecuta cada etapa e incluso podrá automatizar dicha ejecución para que se realice cada cierto tiempo.

La compañía no requiere que las aplicaciones sean muy eficientes ni que el proceso de conversión se realice de forma interactiva. Aunque en esta primera fase, la compañía solamente requiere la conversión de preguntas en formato GIFT, se está pensando que la solución debe admitir otros formatos en el futuro.

1. Metodología usada
2. Identificación de los interesados (Stakeholders)

Responsables de NoGame: Se trata de los equipos directivos de la empresa, son responsables de los presupuestos y toman las decisiones que comprometen fondos de dicho presupuesto.

Equipo de desarrollo del proyecto Trivial: Este equipo será el responsable de desarrollar el sistema resultante de la arquitectura.

Usuario Cliente: Elemento final, será el que utilice el juego.

Responsables de la información: Este equipo se encargará de introducir la información con las preguntas que utilizarán los distintos juegos, además de comprobar su veracidad y actualizarlas en caso de cambios.

|  |  |
| --- | --- |
| Stakeholders | Intereses |
| Responsables de NoGame | 1. Bajo coste de desarrollo, esto es, el desarrollo del proyecto debe ser corto y con un coste reducido. 2. Posibilidad de que el juego sea multiplataforma 3. Posibilidad de crear más juegos con formato preguntas/respuestas, sin necesidad de empezar el proceso completo. |
| Equipo de desarrollo del proyecto Trivial | 1. Baja complejidad en los algoritmos de aleatorización de las preguntas. 2. Proyecto rentable, esto es, que permita ser desarrollado por el precio establecido con un grado de rentabilidad que haga atractivo el desarrollo. 3. Permitir la ampliación o añadidura de nuevas partes de manera sencilla al proyecto. |
| Usuario Cliente | 1. Rapidez en la visualización de las preguntas 2. Aleatoriedad de las preguntas, que no salga la misma pregunta repetida en un periodo de tiempo o en una serie de usos. 3. Veracidad actual de las preguntas. |
| Responsables de la información | 1. Máxima facilidad y rapidez a la hora de introducir preguntas, preferiblemente con tratamientos por lotes y no individualmente. 2. Máxima facilidad y rapidez para buscar una determinada pregunta. 3. Máxima facilidad y rapidez para actualizar los datos de una pregunta. 4. Poder mantener la integridad y coherencia en la información. |

1. Requisitos funcionales
   1. Se le debe poder comunicar a la aplicación por consola de donde tiene que leer la información.
   2. Se le debe poder comunicar a la aplicación por consola donde ha de devolver el resultado.
   3. La aplicación debe poder leer principalmente ficheros en un formato determinado (GIFT) con la información de las preguntas , y generar a partir de él otro fichero en un formato intermedio (JSON)
   4. La aplicación debe permitir observar los resultados intermedios de la conversión para detectar errores.
   5. La aplicación debe poder leer los ficheros en el formato intermedio (JSON) y almacenarlos en una base de datos.
   6. Estas dos operaciones podrán ser realizadas individualmente por un operario
   7. Estas dos operaciones se deben poder automatizar, programando su ejecución cada cierto tiempo.
2. Requisitos no funcionales
   1. Debe poder correr en paralelo varias instancias de la aplicación a la vez.
   2. Se le debe poder comunicar a la aplicación por consola que formato va a leer.
3. Atributos de calidad

|  |  |  |
| --- | --- | --- |
| Código | Descripción | Tipo de Atributo |
| AT001 | El sistema debe estar disponible siempre que el operario lo quiera usar, por tanto debe poder funcionar 24x7. | Disponibilidad |
| AT002 | Facilidad de cambio de los algoritmos de conversión, pues es puede ser necesario modificarlos en el futuro para admitir más formatos de entrada, o ampliar la variedad de formatos de salida. | Modificabilidad |
| AT003 | No es necesario que el proceso de conversión de la aplicación sea eficiente. | Rendimiento |
| AT004 | Es importante la seguridad en la información procesada. Los datos que se guarden son muy importantes para el correcto funcionamiento de la aplicación final. | Seguridad |
| AT005 | Es importante poder saber si se ha realizado la conversión de un formato a otro de forma correcta, y si se ha almacenado correctamente. | Estabilidad |
| AT006 | No exige una interfaz gráfica, pero sí que el operador que la use pueda, mediante comandos, realizar una serie de operaciones básicas. Interacción con el usuario. | Usabilidad |
| AT007 | Facilidad para probar la fiabilidad de sistema, garantía de que los cálculos se han realizado correctamente. | Testabilidad |

1. Escenarios de calidad

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Escenario nº | Fuente de estímulo | Entorno | Artefacto | Respuesta | Medición de la respuesta |  | Atributos de calidad afectado |
| 1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |