Trabajo Práctico N°2: Monitoreo de Eventos

Iteración 1 - Primera Entrega (26/04/2018)

GRUPO N°8:

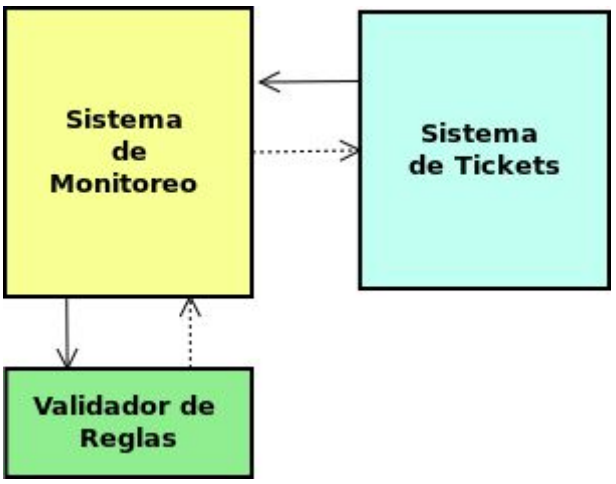
FATUR, Iván Andrés (84491)
MASRIAN, Viken (96438)

Índice:

Diseño propuesto

Luego de leer el enunciado se determina que se contará con 3 sistemas interactuando entre sí. Los sistemas son:

- Sistema Validador de Reglas
- Sistema de Monitoreo de Eventos
- Sistema de Manejo de Tickets



Para llevar a cabo la funcionalidad que se propone en el enunciado, se determinan formas para que los sistemas puedan interactuar y cumplir el objetivo. Se propone una solución del tipo cliente-servidor, es decir, el cliente ante cada acción que busca llevar a cabo debe acudir a un servicio del servidor.

Las interacciones se realizan a través de la invocación directa de funciones o por medio de servicios REST.

El sistema de “Monitoreo de eventos” será notificado a través de un servicio REST por el sistema de “Manejo de tickets” ante cualquier evento. Esta forma de interactuar nos permite la capacidad que más de un sistema de manejo de tickets pueda hacer uso del Sistema de Monitoreo.

El sistema “Validador de reglas” (desarrollado en Clojure) correrá dentro de la aplicación JAVA de “Monitoreo de eventos” haciendo uso de la interoperabilidad entre ambos lenguajes. En consecuencia cuando el sistema de monitoreo tenga la necesidad de conocer el nuevo estado de uno o varios dashboards asociados al proyecto, este simplemente invocará a las funciones que brinda el Validador de reglas.

Los usuarios serán capaces de interactuar con cualquiera de los dos sistemas por medio de aplicaciones web, de forma tal que sea un medio rápido, intuitivo y separado del modelo, es decir, estas aplicaciones van a reflejar las respuestas de los servicios.

Un aspecto importante que se propone es la posibilidad de observar cambios en tiempo real en el sistema de monitoreo, para esto determinamos que los dashboards cada cierto intervalo de tiempo definido por el usuario, consultará a los servidores.

Interfaces

Se prevé la necesidad de implementar los siguientes métodos en el “Monitor de eventos”:

POST
- de un nuevo ticket.
- de un nuevo dashboard.
- de una nueva consulta en un dashboard.

GET
- de un ticket ya cargado.
- de consultas pertenecientes a un dashboard.
- de valor de “consulta perteneciente a dashboard”.
- de conjunto de valores históricos de “consulta perteneciente a dashboard”.

PUT
- de un dashboard.
- de consultas pertenecientes a un dashboard.

DELETE
- de un dashboard.

Se prevé la necesidad de implementar los siguientes métodos en el “Sistema de Tickets”:

POST
- de un nuevo ticket.
- de un nuevo comentario en ticket.
- de un nuevo proyecto de tickets.
- de usuarios.
- de roles.

GET
- de un ticket ya cargado.
- de proyecto.

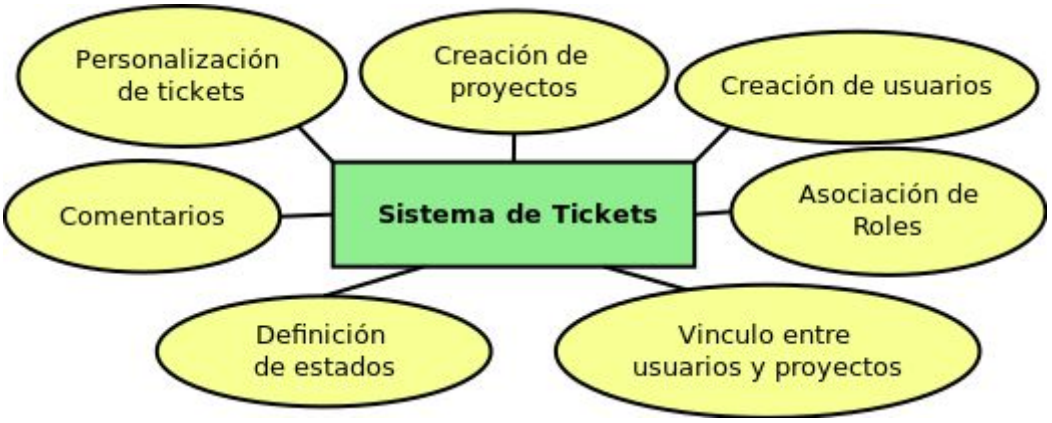
PUT
- de un ticket ya cargado.

Servicios a Usuarios

En esta sección se describen los servicios que se brindan por cada uno de los sistemas, estos pueden ser utilizados por el usuario. Para su mayor claridad se decidió su representación a través de gráficos.

En la siguiente imágen se busca presentar los servicios disponibles por el usuario en el Sistema de Monitoreo, estos son indicados en forma de óvalos.

En la siguiente imágen se busca presentar los servicios disponibles por el usuario en el Sistema de Tickets, estos también son indicados en forma de óvalos.

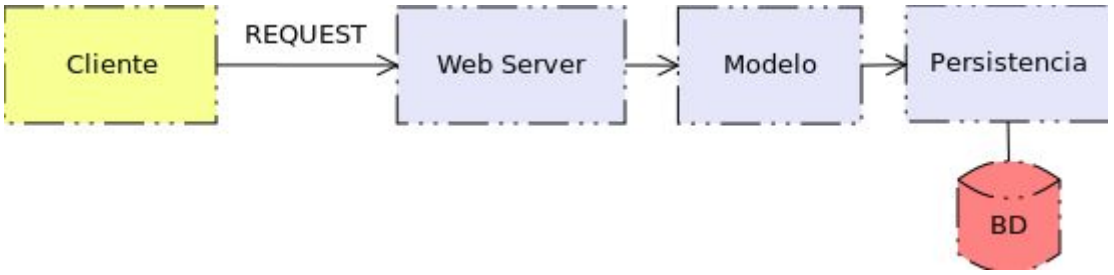


Patrón de Arquitectura

Tanto para el Sistema de Monitoreo como para el Sistema de Tickets, determinamos tomar como referencia el patrón de arquitectura *Enterprise* Application *Architecture*. Esta decisión se toma ya que nuestros sistemas cuentan con las siguientes características:

- Usuarios Concurrentes
- Conexión con otros sistemas
- Interfaces Estandar
- Reusabilidad
- Validación de reglas del modelo
- Mapeo entre tablas y objetos

En relación a lo dicho se intenta representar al Sistema de Monitoreo y al Sistema de Tickets a través del siguiente esquema general.



En este esquema se puede ver como el cliente realiza una petición al servidor web, luego este le transfiere la solicitud al modelo. Este verifica las reglas correspondientes, si es válido pasa a realizar la solicitud, esta puede desencadenar un cambio en las entidades del modelo. Esto puede provocar la persistencia de datos en la base de datos. Si la realización de la petición fue exitosa, el Webserver le responderá al cliente. De cierta manera se puede decir que se encarga de responder con las vistas adecuadas.

Tanto para el Sistema de Monitoreo como el de Tickets cumpliran el anterior esquema, la diferencia radica en el modelo de cada uno de ellos, donde para el primero se consideran las entidades, relaciones y reglas necesarias para llevar a cabo un sistema de monitoreo mientras que para el segundo se busca modelar un sistema de tickets.

Procesos del Sistema

Se describe el flujo principal de nuestro sistema considerando que ya fueron definidos los tickets y las consultas en el Sistema de Manejo de Tickets y Sistema de Monitoreo de Eventos respectivamente.

El siguiente flujo nos permite llevar a cabo el monitoreo en tiempo real:

1. **Sistema de Manejo de Tickets:** Se produce un evento sobre algún ticket. Un ejemplo podría ser una transición de estado.

2. **Sistema de Manejo de Tickets:** notifica al Sistema de Monitoreo de tal evento, brinda la información correspondiente.

3. **Sistema de Monitoreo de Eventos:** toma el ingreso de nuevos datos. También toma las reglas definidas en el dashboard del proyecto correspondiente.

4. **Sistema de Monitoreo de Eventos:** invoca al Validador de reglas, con la información considerada en el punto 3.

5. **Validador de reglas:** realiza el procesamiento considerando las reglas y los datos.

6. **Validador de reglas:** retorna un nuevo estado al Sistema de Monitoreo de Eventos.

7. **Sistema de Monitoreo de eventos:** procesa el resultado que le brinda el Validador.

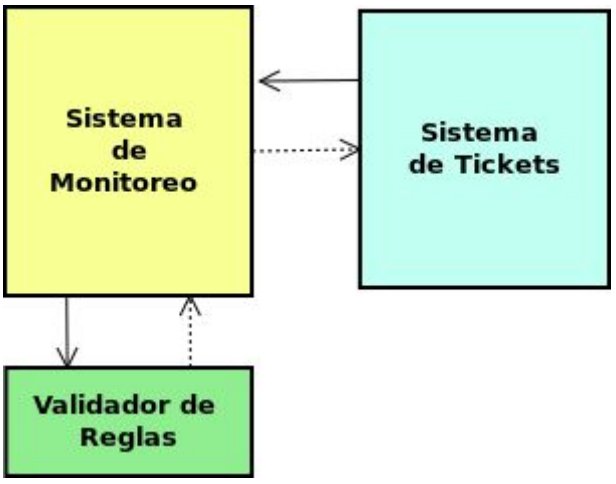
8. **Sistema de Monitoreo de eventos:** refleja el nuevo estado en el dashboard correspondiente.

Despliegue

En esta sección se busca describir las tecnologías y topologías que se utilizan para llevar a cabo el sistema.

Como se aclaró en secciones anteriores , la idea es llevar a cabo la implementación del sistema haciendo uso de la interoperabilidad (JAVA - Clojure) y de servicios REST.

Como se sabe el Sistema de Monitoreo de Eventos será el nexo entre el Sistema de Validación de Reglas y el Sistema de Tickets.



Se describe en detalle cada una de las partes.

Sistema de Monitoreo de Eventos

Se determinó usar como lenguaje para desarrollar el Sistema de Monitoreo de Eventos a JAVA. La elección de JAVA se debe principalmente a cinco factores:

1. Nos permite hacer uso de la POO, de esta forma podremos modelar los distintos requisitos que se necesitan para este sistema
2. Nos permite brindar servicios web
3. Se trata del lenguaje con el que se desarrollan aplicaciones móviles para el sistema operativo Android, ampliamente utilizado en el mercado
4. Soporte a gran cantidad de librerías
5. Extensa documentación

Para llevar a cabo la persistencia se determina hacer uso de un motor de base de datos Relacional. En nuestro caso se elige PostgreSQL.

En resumen se utilizarán las siguientes tecnologías/herramientas:

- Jersey: Implementación de JAX-RS. Se trata de un framework que nos facilita el desarrollo de servicios tipo REST.
- JDK: Nos provee de herramientas de desarrollo.
- Tomcat: Módulo que nos provee de un servidor web, de esta forma se resuelven las peticiones de los distintos servicios.

- Maven: Herramienta que nos permite gestionar las dependencias del proyecto.
- PHP+HTML+Javascript: Lenguajes utilizados para permitir la interacción del usuario con el Sistema de Monitoreo
- PostgreSQL: Base de datos utilizada para persistir datos

Validador de Reglas

El validador de Reglas se encuentra implementado en Clojure, con lo cual decidimos aprovechar esta característica y hacer uso de la interoperabilidad entre Clojure y JAVA. Esto es posible ya que Clojure realiza su ejecución sobre la máquina virtual de JAVA.

Mediante el uso de la herramienta lein, se generará un jar (JAVA Archive). Este será agregado a nuestro proyecto en JAVA (Sistema de Monitoreo), en consecuencia podremos hacer uso de las funciones que nos brinda el Validador.

En resumen se utilizarán las siguientes tecnologías/herramientas:

- Lein: Herramienta que nos permite gestionar un proyecto Clojure

Sistema de Manejo de Tickets

Se decidió hacer uso de JAVA para el Sistema de Manejo de Tickets, este provee de una interfaz web para que el usuario interactúe con el sistema a través de servicios REST. En consecuencia las herramientas a utilizar son las mismas que las definidas en el Sistema de Monitoreo.

Consideramos que como se busca una solución escalable, de forma tal que se trabajará con grandes cantidades de datos se determinó utilizar como medio de persistencia una base de datos No Relacional.