DOCUMENTO DE ARQUITECTURA

Aplicativo/Proyecto (template)

Fecha	Versión	Descripción de Cambios	Autor
29/04/2009	1.0	Primera versión	Ricardo Di Pasquale

Autor

Grupo

Indice

1 - Contexto del proyecto

En este apartado debe describirse el contexto tecnológico y funcional del proyecto, incluyendo restricciones en tiempo, costos y recursos de cualquier tipo.

2 - Requerimientos de Arquitectura

2.1 - Descripción breve de los objetivos clave

Se deben documentar los objetivos principales de la arquitectura de la aplicación del proyecto en cuestión, por ejemplo:

El objetivo primario de la arquitectura de la aplicación xxx es proveer una infraestructura que soporte una interfaz programática de herramientas cliente de terceros (proveedores) para acceder al almacenamiento de datos. Debe ofrecer:

- *Flexibildad en terminus de plataforma, deployment y configuración de cara a las herramientas cliente a desarrollar por terceros.*
- *Un framework que permita que las herramientas se acoplen (tipo plug-in) al entorno de modo de obtener feedback inmediato de las actividades del usuario de la aplicación, y proveer información útil para su posterior análisis.*
- *Proveer acceso al data store de lectura y escritura en forma simple y conveniente.*

El objetivo secundario es evolucionarla arquitectura (respecto de la primera version productiva) que pueda escalar para soportar deployments de 100 a 150 usuarios. Este objetivo debiera ser alcanzado sin alterar los bajos costos que hoy prove la plataforma.

El enfoque utilizado debe ser consistente con las necesidades del cliente y las restricciones y requirimientos no funcionales descriptos en el presente documento.

2.2 - Casos de uso de arquitectura

Describir con un enfoque funcional los casos de uso básico referentes a la arquitectura, por ejemplo:

Se registran dos casos de uso básicos referents a la API a desarrollar que se relevaron en reunions con los principales proveedores que van a desarrollar las herramientas "cliente":

- ***Acceso a datos:*** *Los queries de las herramientas clients se enfocan en las actividades de un usuario en particular. Una secuencia de consultas comienza obteniendo información sobre la asignación de trabajo de un usuario en particular, que es básicamente el proyecto al que está asignado. La navegación luego perfora (drill down) datos detallados sobre la actividad del usuario en forma cronológica.*
- ***Almacenamiento de datos:*** *Las herramientas "cliente" necesitan tener disponible la posibilidad de almacenar información la base de datos del sistema, de manera que puedan compartir datos sobre sus actividades. Se necesita un mecanismo de notificación para que las herramientas cliente comuniquen la disponibilidad de nuevos datos. Los datos son diversos en estructura y contenido, por lo que se supone que deben tener algún mecanismo "descubrible" de meta-data incluido.*

2.3 - Requerimientos de arquitectura de los actores

Se deben estudiar los requerimientos de la arquitectura desde las perspectives de los actors principales de la aplicación, aunque considerando no solamente los actores productivos (usuario, proveedores, clientes, etc), sino teniendo en cuenta también actores de otras fases del proceso de desarrollo, cómo pueden ser el equipo de desarrollo, proveedores y otros actores internos.

Los requerimientos desde la perspective de los principales actors se cubren de la siguiente manera:

- *Terceros (proveedores)*
  - *Facilidad de acceso a los datos: El esquema relacional de la base de datos posee al rededor de 50 tablas, con algunas relaciones complejas. Por lo que no es trivial la conformación de las consultas. También se prevee que los cambios al esquema relacional durante la evolución de la plataforma no podrán evitarse. Por estas razones, se requiere un mecanismo para facilitar a los terceros el manejo de los datos de manera independiente de la base de datos.*
  - *Soporte de plataformas heterogéneas*
  - *Notificaciones de enventos instantánea*
- *Desarrolladores*
  Desde la perspectiva del desarrollador, la API debiera:
  1. *Se fácil e intuitiva*
  2. *El código debiera ser fácil de modificar.*
  3. *Proveer un modelo conciso y conveniente para implementer casos de uso comunes (cross)*
- *Equipo de desarrollo interno*
  Desde la perspectiva de los desarrolladores internos, la arquitectura debe:
  - *Abstraer completamente la estructura de la base de datos relacional.*
  - *Soportar acceso concurrente*
  - *Proveer performance escalable: Debiera ser possible escalar el sistema sin cambios a la implementación actual.*
  - *No ser innecesariamente caro de testear.*

2.4 - Restricciones

Este tipo de restricciones solo refiere a las que tengan que ver con la arquitectura, cómo por ejemplo el ambiente de ejecución, tiempos, etc. Por ejemplo:

- *Debe corer en entornoes Linux*
- *Debe utilizar el esquema Legacy actual de la base de datos*

2.5 - Requerimientos no funcionales

Describe requerimientos no funcionales básicos relacionados con la arquitectura aplicativa . Por ejemplo:

- ***Performance:*** *Debiera responder en consultas de hasta 1000 registros en menos de 5 segundos.*
- ***Confiabilidad:*** *La arquitectura debiera ser defensive de cara a la introducción de errors por parte de los proveedores de herramientas cliente. Se establecerán los mecanismos de trade-off y auditoria necesarios para el caso.*
- ***Simplicidad:*** *Dado que los requerimientos funcionales todavía son vagos o ambiguous se prefiere la simplicidad en el diseño, quizás resignando complejidad.*

2.6 - Riesgos

Se require la enumeración de los riesgos arquitectónicos del proyecto, su estrategia de mitifación, su impacto en el proyecto (en caso de ocurrir) y la posibilidad (estadística o mejor probabilística si es posible) de que el mismo ocurra.

3 – Solución

3.1 - Patrones arquitectónicos relevantes

Se describe la utilización de patrones arquitectónicos (y patrones de diseño relevantes) en este proyecto.

3.2 - Breve descripción de la arquitectura

Incluye una descripción de la arquitectura, y los diagramas y modelos que el arquitecto recomiende. El objetivo principal de esta sección es proveer un enfoque de nivel cero a la arquitectura de la aplicación en cuestión:

3.3 - Vistas Estructurales

Describe todos los componentes de relevancia structural de la arquitectura. Se debiera incluir:

- Logical view (componentes)
- Deployment view

3.4 - Vistas de comportamiento

Describe los temas de comportamiento referents a la arquitectura. Debe incluir los comportamientos más relevantes o generales de la aplicación. Para esto debe incluir varios diagramas de secuencia, interacciones, actividades, etc (que el arquitecto considere necesarios)

3.5 - Temas de implementación

Incluye todos los “issues” técnicos relacionados con la elaboración o construcción del proyecto. Típicamente, debiera incluir temas de seguridad, temas transaccionales, temas concretos referents al lenguaje de programación, los frameworks a utilizar y las apis o librerías.

4 - Análisis de la arquitectura

4.1 - Análisis de escenarios

Se deben analizar los escenarios más generales de cara al futuro de la aplicación o del entorno. Por ejemplo:

- ***Modificar la organización de la base de datos:*** *Los cambios realizados a la base de datos necesitarán cambios en los componentes EJB del servidor.*
- ***Mover la arquitectura a otro JEE vendor:*** *Puede que la empresa decida salir de la línea de productos IBM WebSphere. Para esto con poco esfuerzo se tratará de no implementer comportamiento privado (del producto) alguno de forma de independizar lo más possible la arquitectura.*
- ***Escalar el deployment a 150 usuarios:*** *Este scenario require de una cuidada capacidad de planeamiento basada en la especificación del hardware disponible. Debe ser factible particionar la base de datos sin impactar a la arqutiectura.*