Documento de Arquitectura de Software

Integración de PGE con tecnologías GIS

Luciana Canales

Alejandro Remiro

Maximiliano Felix

Contenido:

[1 Introducción. 3](#_Toc303693280)

[2 Vista de Casos de Uso 4](#_Toc303693281)

[2.1 Diagrama de Casos de Uso Críticos 4](#_Toc303693282)

[2.2 Actores 5](#_Toc303693283)

[2.3 Especificación de Casos de Uso Críticos 6](#_Toc303693284)

[3 Vista Lógica 7](#_Toc303693285)

[3.1 Estilo Arquitectónico. 7](#_Toc303693286)

[3.2 Subsistemas. 8](#_Toc303693287)

[3.3 Diagramas de Interacción 9](#_Toc303693288)

[4 Vista de Distribución (Deployment). 11](#_Toc303693289)

[4.1 [Escenario 1] 12](#_Toc303693290)

[4.2 [Escenario 2] 12](#_Toc303693291)

[5 Vista de Procesos. 13](#_Toc303693292)

[6 Vista de Implementación. 14](#_Toc303693293)

[7 Referencias 15](#_Toc303693294)

# Introducción.

[

En esta sección se realiza una introducción al sistema cuya arquitectura se describe.

Para la elaboración de este plantilla, se tomó como base el modelo 4+1 [10].

]

# Vista de Casos de Uso

[

Los elementos de las cuatro vistas desde las que se analiza la arquitectura son “ejercitados” por un pequeño subconjunto de casos de uso (o escenarios de esos casos de uso) que llamamos casos de uso críticos. Este subconjunto se debe elegir cuidadosamente, utilizando principalmente dos criterios:

* Que intervengan el mayor número de componentes arquitectónicos posibles
* Que intervengan los componentes que se vislumbran como críticos o más complejos.

]

## Diagrama de Casos de Uso Críticos

[

En esta sección se utilizan los Diagramas de Casos de Uso de UML para mostrar los casos de uso críticos para la arquitectura. Los elementos a utilizar en estos diagramas son:

|  |  |
| --- | --- |
| Caso de Uso | Secuencia de acciones con un fin específico (Ej. Enroll in University) |
| Actor | Persona, organización o sistema externo que interactúa con el sistema (Ej. Student) |
| Asociación | Relación entre un Caso de Uso y un Actor que participa en él. |
| Frontera del Sistema | La Frontera del Sistema se indica con un rectángulo que encierra los Casos de Uso. |
|  |  |

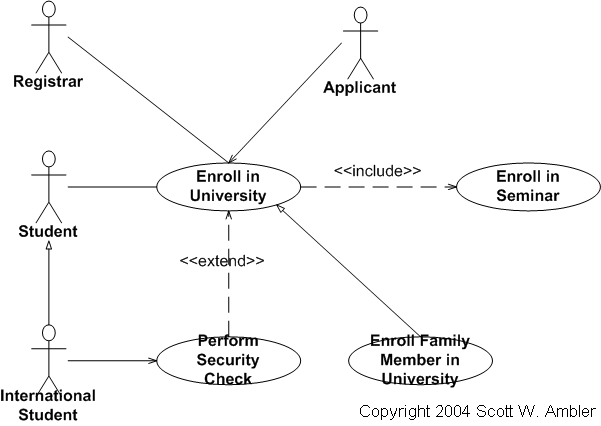


Diagrama de Ejemplo 1 (Ver [2])

]

## Actores

[

En esta sección se realiza una descripción de cada actor perteneciente al diagrama anterior.

]

### [Actor 1]

### [Actor 2]

## Especificación de Casos de Uso Críticos

[

En esta sección se incluye un ítem por cada caso de uso mostrado en el diagrama anterior. De cada caso de uso debe brindarse su descripción, así como uno o más flujos de eventos que detallen la interacción actor-sistema para cada escenario relevante del caso de uso. Opcionalmente, pueden especificarse pre y postcondiciones para cada caso. Las precondiciones son las condiciones que deben cumplirse para que el flujo del caso de uso pueda realizarse. Las postcondiciones son las condiciones que se cumplen al terminar el flujo del caso de uso (modificaciones en el estado del sistema).

]

### [Caso de Uso Crítico 1]

##### Descripción

##### Pre-condiciones

##### Flujo de Eventos

|  |  |
| --- | --- |
| Acción de Actor | Respuesta del Sistema |
| 1 - |  |
|  | 2 - |
| 3 - |  |

##### Post-condiciones

### [Caso de Uso Crítico 2]

# Vista Lógica

[

La vista lógica permite describir el sistema en base a abstracciones fundamentales del diseño orientado a objetos para dar soporte a los requerimientos funcionales. En un enfoque top-down, se comienza por descomponer el sistema en un conjunto de subsistemas “grandes”, como ser las “capas” (*layers*) si se utiliza una arquitectura en capas, y a partir de ellos se realizan sucesivos refinamientos hasta llegar a las unidades lógicas más pequeñas.

]

Tomando en cuenta la solución propuesta por Raquel Sosa en su tesis de maestría [ref tesis] se tiene un conjunto de 5 subsistemas independientes que interactúan para resolver los casos de uso. Con respecto a la figura 6, de izquierda a derecha se tiene: El cliente GIS, el CTP de entrada llamado RestConnector, la PGE propiamente dicha, el CTP de salida llamado SoapConnector y el servidor de mapas que se quiere enmascarar.

Figura 6

Cliente GIS: Cualquier programa utilizado para consumir datos geográficos utilizando protocolos WMS y WFS. Puede ser desde una aplicación web usando OpenLayers hasta Gvsig.

CTP RestConnector: es un sub sistema encargado de recibir pedidos REST especificados según los protocolos WMS y WFS. Transforma estos pedidos a formato SOAP para que atraviesen la PGE. Y los encamina hacia ella. También según la configuración agregará información a los pedidos que requieran seguridad y otros datos exigidos por la PGE.

PGE: Es la plataforma de gobierno electrónico del estado uruguayo [ref pge].

CTP SoapConnector: Este sistema estará registrado en la PGE como un proveedor de servicios, al cual se encaminarán los pedidos WMS y WFS previamente transformados por el CTP RestConnector. Su función es volver a componer el pedido REST para enviarlo al servidor de mapas configurado, y transformar la respuesta del mismo en un mensaje SOAP para que recorra el camino inverso. La idea es que exista un SoapConnector por cada servicio GIS expuesto a través de la PGE.

Servidor de mapas: Es un servidor que soporte los protocolos WMS y WFS. En principio habrá soporte para GeoServer [ref geo] y MapServer [ref ms].

## Estilo Arquitectónico.

En esta sección se comentará el estilo arquitectónico que respecta a los componentes CTP, para los demás sistemas referirse a la documentación específica de cada uno.

Ambos CTP están implementados como una orquestación de servicios dentro de un ESB. Por tanto respetan la arquitectura provista por los ESB que es en definitiva la arquitectura de los sistemas de mensajería.

Los sistemas de mensajería utilizan una arquitectura de pipeline de filtros, donde cada filtro afecta el mensaje que se transmite al siguiente filtro o bien hace trabajo de ruteo cambiando el orden o los filtros que se ejecutan sobre un mensaje particular.

Los CTP están implementados utilizando la tecnología JBOSS ESB, la cual provee una interfaz de filtro que se implementa, y luego se define un servicio donde se declaran los filtros que utiliza y en qué orden se ejecutan.

[

En esta sección se describe el estilo arquitectónico elegido para el sistema (capas estricto, capas no estricto, etc.). En el lenguaje de modelado UML 2, se utiliza el Diagrama de Componentes para representar componentes lógicos, creados en tiempo de diseño, y no componentes físicos (ejecutables, bibliotectas, etc.) creados en tiempo de implementación, como sucede con el Diagrama de Componentes de UML 1. Se aconseja utilizar diagramas de componentes para representar subsistemas (considerar el clasificador <<subsystem>> como alternativa a <<component>>.

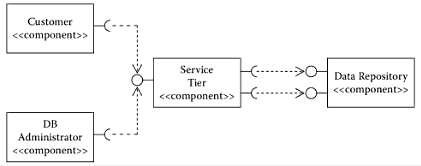


Diagrama de Ejemplo 2 (Ver )

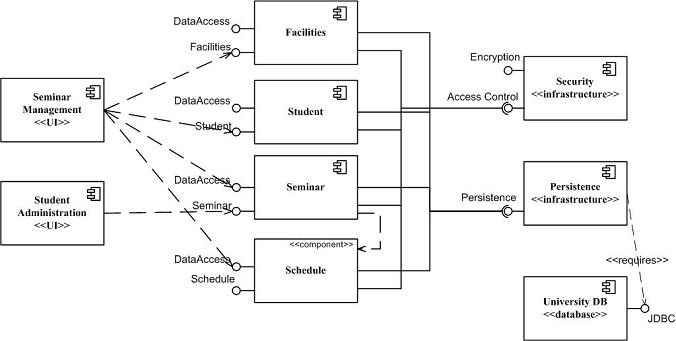


Diagrama de Ejemplo 3 (Ver )

]

## Subsistemas.

En esta sección se describe cada uno de los subsistemas, debido a que son el objeto de ese proyecto se describen los CTPs y como se utiliza una versión reducida y minimalista de la PGE implementada específicamente para simular la PGE real en el contexto de este proyecto también se incluirán sus detalles en el siguiente apartado.

[

En esta sección se describe cada uno de los subsistemas. El diagrama de componentes muestra en detalle los componentes que corresponden a ese subsistema. Para cada componente, deben detallarse las interfaces que implementa.

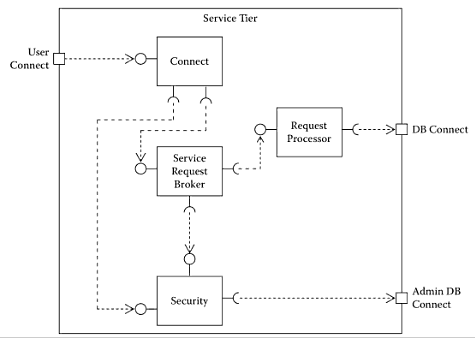


Diagrama de Ejemplo 4 (El subsistema es la capa de servicios)

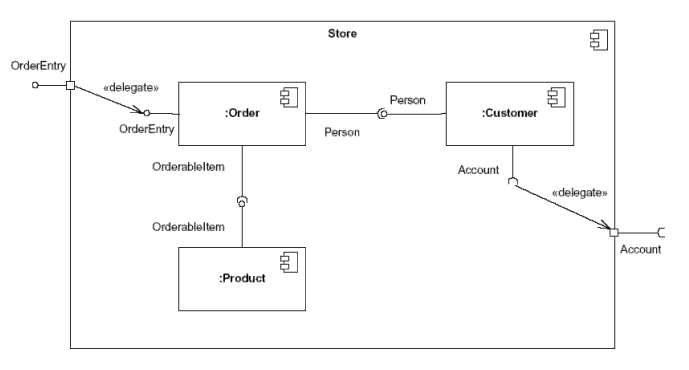
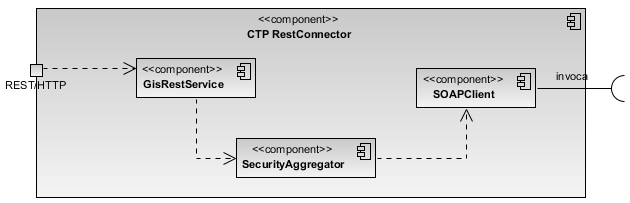


Diagrama de Ejemplo 5 (El subsistema es el componente Store)

]

#### CTP RestConnector.

##### Diagrama de Componentes.



##### Descripción.

El RestConnector está compuesto de tres filtros con responsabilidades bien separadas que van agregando información al mensaje ESB para finalmente transformar el pedido de REST en un mensaje SOAP y enviarlo hacia la PGE.

El punto de entrada es un Getway HTTP cuya implementación es provista por la plataforma ESB. Este recibe las conexiones HTTP en una URI particular del servidor y lo transforma en un mensaje ESB que se transmite al primer filtro y se va pasando entre estos procesándolo como sea debido.

El primer filtro es el GisRestService, este se encarga de parsear el pedido WMS o WFS para averiguar a que servicio geográfico expuesto a través de la PGE se quiere invocar. Para esto toma datos del pedido y consulta una base de conocimiento. Carga toda la información necesaria y delega al siguiente filtro.

El SecurityAggregator es el encargado de resolver las necesidades de autenticación y autorización que el servicio de la PGE requiera. Con la información proporcionada por el filtro anterior conformará el token de seguridad SAML que se requiere para atravesar la PGE.

Finalmente el SOAPClient utiliza un cliente de webservice generado a partir del webservice SOAP que provee el CTP SOAPConnector, para invocar el servicio geográfico con los datos necesarios incluyendo los parámetros del pedido original y los agregados por la PGE. Con la diferencia de que el endpoint invocado no es el propio SoapConnnector sino el proxy del servicio expuesto por la PGE.

Al recibir la respuesta se coloca en el mensaje ESB lo enviado por el proveedor y el servidor ESB se encarga automáticamente de convertir eso en una respuesta HTTP adecuada, ya que se ha terminado de invocar los filtros configurados para ese servicio.

#### CTP SoapConnector.

##### Diagrama de Componentes.



##### Descripción.

El SoapConnector también está implementado en la tecnología ESB pero es bastante diferente al anterior. En este caso se configura un servicio cuyo punto de acceso en un procesador SOAP provisto por el ESB, el mismo recibe pedidos HTTP que contengan un mensaje SOAP, lo procesa e invoca a la clase que lo implementa, en este caso el componente GisSoapService, el cual tiene declarados como métodos todos los definidos por los estándares WMS y WFS. Este componente recibe toda la información y utiliza la configuración para armar la URL del servidor de mapas configurado. Dado que se utilizará una instancia de este sub sistema por cada servicio geográfico, es decir por cada servidor de mapas que se quiera exponer, solo se configura un servidor y su URL, IP, puerto y la URI. Esta URI puede variar según el protocolo y según la implementación del servidor, por ejemplo existen variantes entre GeoServer y MapServer.

El componente RestInvoker es el encargado de armar el pedido HTTP correspondiente, basado en los parámetros del pedido orginal y la URL provista por GisSoapService.

Para la respuesta, simplemente se retorna lo enviado por el servidor de mapas, los componentes provistos por ESB que implementan el webservice se encargan de convertirlo en un mensaje SOAP Response. Solo se debe tener en cuenta que para el método GetMap la respuesta es una imagen en formato binario y no un XML como en todos los demás métodos.

#### Simulador de Plataforma de Gobierno Electronico.

[TODO]

##### Diagrama de Componentes.

##### Descripción.

El

## Diagramas de Interacción

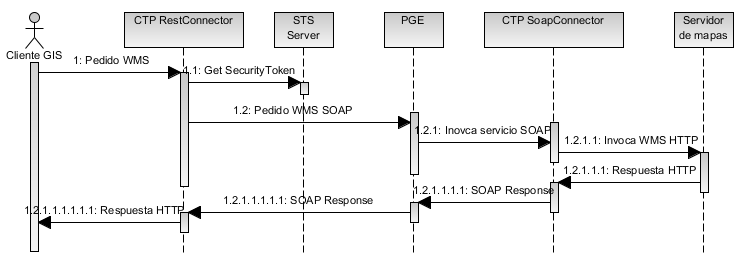


Diagrama 8 (Diagrama de Secuencia para caso de uso Público General, ver [3])

Para el resto de los casos de uso, la interacción entre los sistemas y componentes es igual, lo que diferencia a cada caso de uso es el despliegue de los subsistemas que cambia según el entorno en que se encuentra el cliente.

# Vista de Distribución (Deployment).

Uno de los puntos interesantes en el diseño fue encontrar la manera de desplegar los diferentes componentes teniendo en cuenta los requerimientos planteados en los escenarios de uso de la tesis de Raquel Sosa, así como también la interacción con los sistemas preexistentes, es decir tanto la PGE, como la REDuy y los servicios e infraestructura con los que hoy cuentan los diferentes organismos estatales que utilizan a diario información geográfica.

## Escenario Organismo cliente

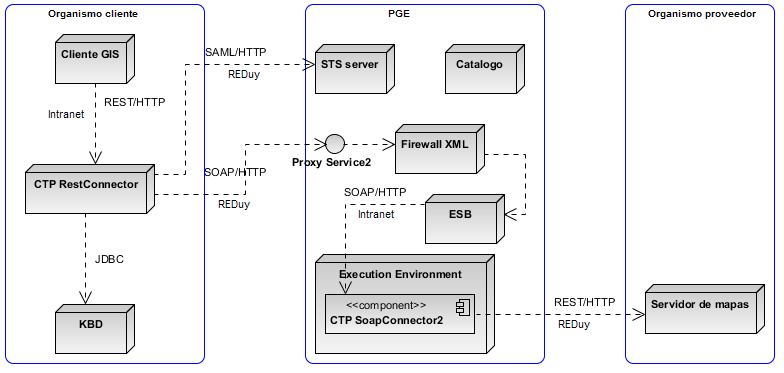


Diagrama 9, despliegue del escenario Organismo cliente.

#### Descripción.

#### Nodos.

#### Conexiones.

## Escenario Público general

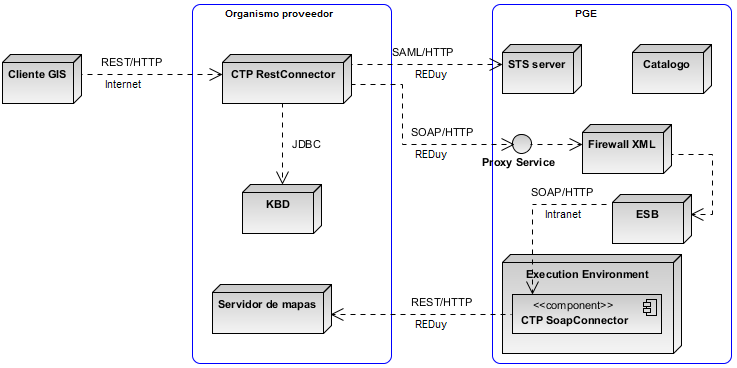


Diagrama 10, despliegue del escenario Público general.

#### Descripción.

#### Nodos.

#### Conexiones.

# Vista de Procesos.

[

La Vista de Procesos permite analizar el sistema a través de su descomposición en unidades de ejecución, es decir, procesos del sistema operativo e hilos (threads). Esta vista **no es requerida** dentro de este documento.

La vista puede dividirse en una parte que muestre la estructura estática de hilos y procesos, así como en una parte dinámica enfocada a la interacción de los mismos (sincronización, comunicación, etc.).

Los tipos de diagramas que pueden utilizarse para esta vista son básicamente los mismos que se utilizan para la Vista Lógica, pero centrados en las clases activas que representan a los hilos y procesos.

Para mostrar la estructura estática se sugiere utilizar el Diagrama de Clases de UML, con los estereotipos <<thread>> y <<process>>.

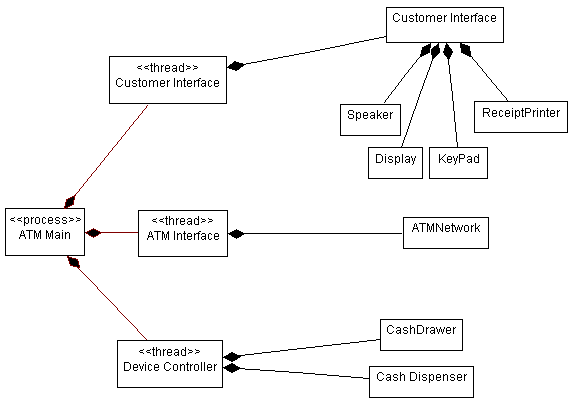


Diagrama de Ejemplo 8

Si se considera necesario mostrar la parte dinámica, se sugiere utilizar el Diagrama de Actividad de UML.

]

# Vista de Implementación.

[

La vista de implementación se focaliza en los componentes en tiempo de ejecución que forman el sistema (ejecutables, archivos de clases, bibliotecas, frameworks, etc.), que son la implementación de los componentes lógicos (provenientes de Diseño).

Dentro de esta vista, interesa mostrar las dependencias entre componentes implementados (utilizando los “artefactos”de UML).

También pueden mostrarse la distribución de los artefactos en los nodos.

En ambos casos se utiliza el Diagrama de Deployment de UML.



Diagrama de Ejemplo 9 (Dependencias entre artefactos, ver [6])

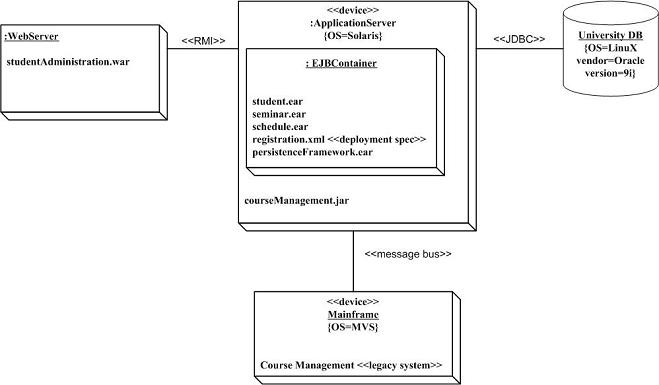


Diagrama de Ejemplo 10 (Ver [6])

]

# Referencias

1. Agile Models Distilled: Potential Artifacts for Agile Modeling  
   <http://www.agilemodeling.com/artifacts/>
2. Agile Modeling. UML 2 Use Case Diagrams  
   <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>
3. Agile Modeling. UML 2 Sequence Diagrams  
   <http://www.agilemodeling.com/artifacts/sequenceDiagram.htm>
4. Agile Modeling. UML 2 Component Diagrams  
   <http://www.agilemodeling.com/artifacts/componentDiagram.htm>
5. Agile Modeling. UML 2 Deployment Diagrams  
   <http://www.agilemodeling.com/artifacts/deploymentDiagram.htm>
6. Deployed Software: Artifacts  
   <http://codeidol.com/other/learnuml2/Modeling-Your-Deployed-System-Deployment-Diagrams/Deployed-Software-Artifacts/>
7. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. N. Rozanski, E. Woods. Addison-Wesley, 2005
8. Software Architecture in practice, Second Edition. L. Bass, P. Clemens, R. Kazman. Addison-Wesley, 2003
9. An Introduction to Software Architecture. D. Garlan, M. Shaw. 1994. <http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf>
10. Architectural Blueprints — The “4+1” View Model of Software Architecture. Kruchten, Philippe. 1995. <http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>
11. Software Architecture Links. Bredemeyer Consulting.  
    <http://www.bredemeyer.com/links.htm>