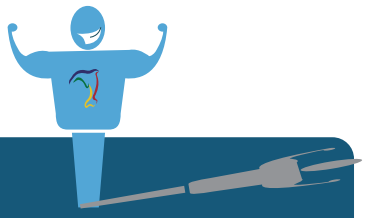
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

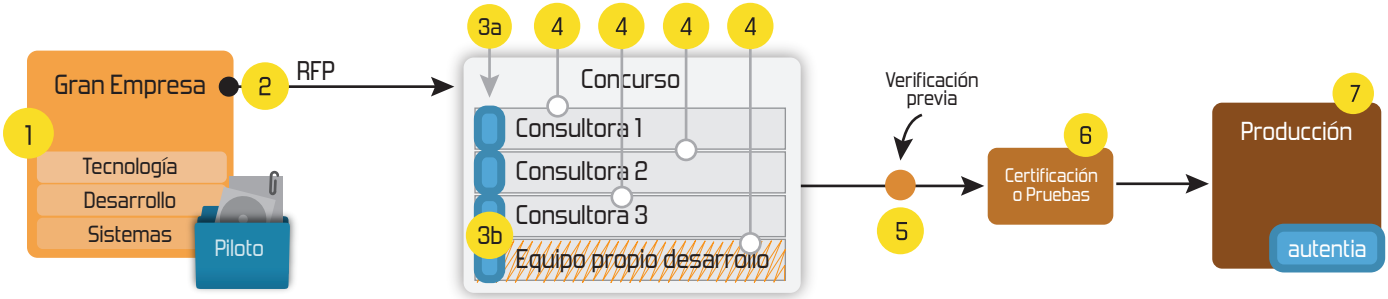
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por: Isaac Gutierre Gómez.

Isaac está actualmente trabajando como responsable de arquitectura de una importante empresa perteneciente a una gran entidad financiera. Ha liderado varios proyectos de aplicaciones para Internet, así como proyectos de telecomunicaciones y de IT para distintas industrias.

Contacta en: isaac@adictosaltrabajo.com

Descargar este documento en formato PDF umlintro.pdf

TOUR POR UML

En este artículo daremos un recorrido por las partes más importantes de UML e intentaremos haceros ver la importancia de modelar y porqué es tan importante un buen diseño de software.

Un proyecto de software con éxito es aquél que produce un software de calidad, consistente y sobre todo que satisface las necesidades de los usuarios que van a utilizar el producto resultante.

Una empresa que produce software de calidad, con un uso eficiente y efectivo de los recursos y terminar los proyectos en plazo tiene un negocio sostenible. Y tenemos que tener en cuenta que lo que importa es el producto final, que funcione bien y que cumpla los requisitos establecidos por los usuarios y no que sea muy bonito, que se hagan reuniones muy importantes o que se hayan codificado muchas líneas de código.

Hace ya tiempo leí una frase que creo que merece la pena recordar: “Para desarrollar software de calidad duradera, hay que idear una sólida base arquitectónica que sea flexible al cambio”. El modelado es una parte fundamental en esta aspecto, construimos modelos para poder visualizar el comportamiento del sistema y poder controlar su arquitectura.

Incluso para producir software de sistemas pequeños sería bueno hacer un análisis y un modelado ya que se producirían sistemas de mejor calidad, pero lo que si es cierto, es que cuanto más grande y complejos son los sistemas más importante es hacer un buen modelado ya que nos ayudará a entender el comportamiento del sistema en su totalidad y que si no tenemos modelado sería bastante difícil. Y cuando se trata de sistemas complejos el modelado nos dará una idea de los recursos necesarios (tanto humanos como materiales) para abordar el proyecto. También nos dará una visión más amplia de cómo abordar el problema para darle la mejor solución.

El Lenguaje Unificado de Modelado (Unified Modeling Languaje, UML) es el lenguaje **estándar** para realizar el modelado de los sistemas de software y es independiente del lenguaje de programación utilizado.

En este artículo no vamos a entrar en más detalles de lo que es UML y de su historia, nos vamos a centrar en las partes más significativas del lenguaje.

UML tiene tres elementos fundamentales:

- Bloques básicos de construcción
  - Elementos
  - Relaciones
  - Diagramas
- Reglas que dictan como se pueden combinar estos bloques básicos. UML tiene reglas para:
  - Nombres
  - Alcance
  - Visibilidad
  - Integridad
  - Ejecución
- Mecanismos comunes. Que se basen en algún patrón, al igual que en arquitectura se puede hablar del barroco, románico, etc..
  - Especificaciones
  - Adornos

- Divisiones comunes
- Mecanismos de extensibilidad

Como dijo Grady Booch: El 80 por ciento de la mayoría de los problemas pueden modelarse usando alrededor del 20 por ciento de UML.

En todo proceso de software donde se utilice una metodología orientada a objetos y la notación UML no pueden faltar los diagramas, para representar las diferentes vistas del producto final.

Los diagramas de UML se pueden dividir en estáticos (aportan una visión estática del sistema) y dinámicos (aportan una visión dinánica del sistema).

Los diagramas estáticos:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de objetos
- Diagrama de componentes
- Diagrama de despliegue

Los diagramas dinámicos:

- Diagrama de estados
- Diagrama de actividad
- Diagramas de interacción:
  - Diagrama de secuencia
  - Diagrama de colaboración

Como se puede ver hay demasiados diagramas y en muchos proyectos no son necesarios todos los diagramas. Será la práctica y experiencia y el tipo de sistema a desarrollar lo que nos ayudará a escoger los diagramas a utilizar, por ejemplo podemos decir que para aplicaciones cliente se suelen utilizar los diagramas de casos de uso, de clase y de colaboración o de secuencia, para aplicaciones donde sean importantes los eventos se puede utilizar el diagrama de estados, para aplicaciones cliente-servidor aparte de los diagramas de casos de uso, clases y objetos también puede ser conveniente utilizar los diagramas de despliegue y componentes, y para aplicaciones complejas cliente-servidor pues si será recomendable utilizar todos los diagramas ya que dará una visión más amplia de cómo será el sistema final.

Diagramas de casos de uso.

Los diagramas de casos de uso muestran la funcionalidad del sistema desde la perspectiva que tienen los usuarios y lo que el sistema debe de hacer para satisfacer los requisitos propuestos. Pueden mostrar el comportamiento de un sistema completo o de una parte.

Los elementos básicos que se utilizan son:

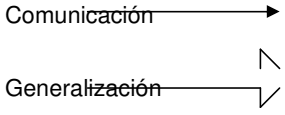
1. Actores: Son los diferentes usuarios y el papel que representan dentro del sistema.

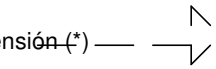
Actor

2. Caso de uso: Representan todo lo que el usuario puede realizar dentro del sistema

Caso de Uso

3. Relaciones: Para asociar los elementos anteriores.

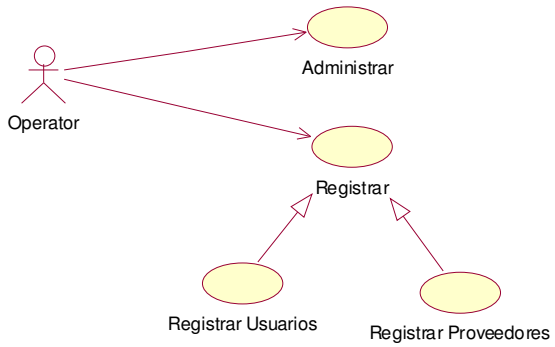


Extensión (*) 

Inclusión (*) 

(*) En estas dos se deben de poner también las letras (include, o exclude).

Ejemplo:



Diagramas de clases

Los diagramas de clases son estáticos porque no describen un comportamiento en función del tiempo. Tienen que ver con la implementación de la aplicación.

Los elementos son:

1. Clases: Se pueden definir como la descripción de un conjunto de objetos con las mismas propiedades. Puede ser un concepto del mundo real, se puede decir que es una plantilla para crear objetos.
2. Relaciones: Las relaciones pueden ser de distintos tipos asociación, agregación, herencia (generalización, especialización).

Ejemplo:

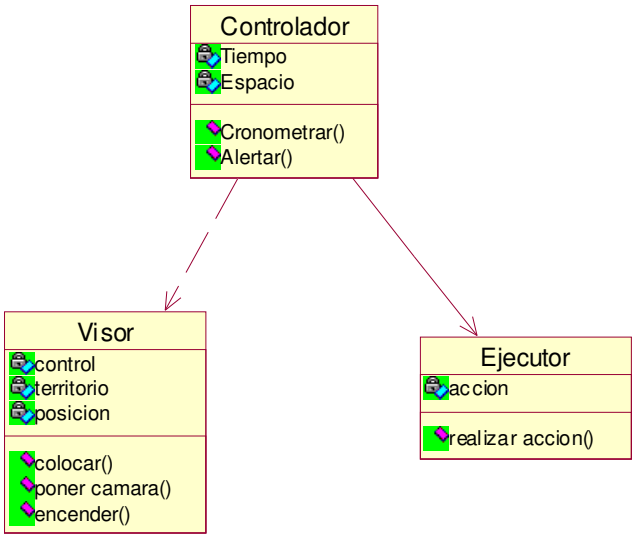


Diagrama de Estados

El diagrama de estados es un gráfico compuesto de los estados del sistema y sus transiciones.

Si se asocia a una clase describirá como una instancia de esta clase reacciona ante los eventos.

Si se asocia a un caso de uso describirá como funciona ese caso de uso con el sistema funcionando.

Estos diagramas son muy útiles para mostrar el ciclo de vida de las clases con complejidad media o alta, pero no tiene mucho sentido para clases de una complejidad simple ni tampoco para clases con una complejidad

bastante elevada.

Símbolo inicial

Símbolo de Fin

Ejemplo:

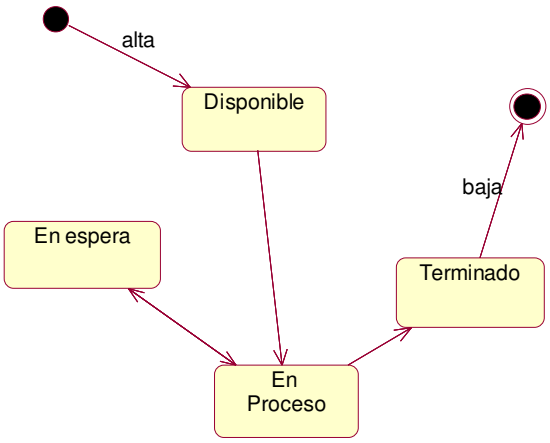


Diagrama de colaboración

Los diagramas de colaboración son útiles para mostrar los efectos que puede tener un objeto con los demás.

Los elementos básicos son clases encerradas en rectángulos, enlaces entre clases y operaciones entre clases.

La estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces.

Ejemplo:

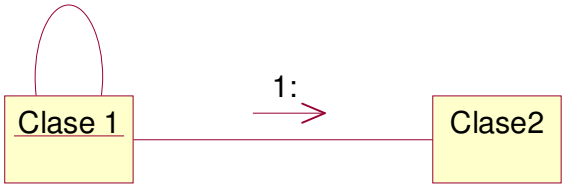
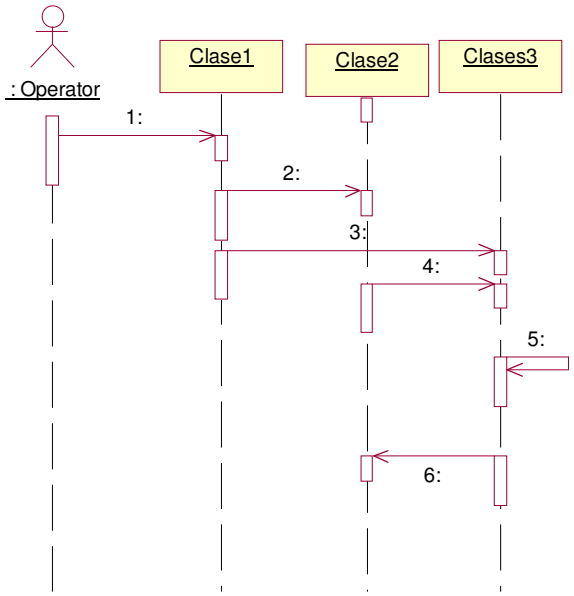


Diagrama de secuencia

Estos diagramas representan la interacción entre clases, se leen de izquierda a derecha y de arriba abajo. Muestran el comportamiento del sistema de forma cronológica. Y por esto son muy útiles cuando se está trabajando con sistemas de tiempo real.

Los elementos básicos son las clases que se representan con rectángulos, los actores, las barras de sincronización temporal que se representan con líneas discontinuas y los mensajes que se representan con flechas.

Ejemplo:



Si desea contratar formación, consultoria o desarrollo de piezas a medida puede contactar con

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Integración de Visual Paradigm en NetBeans	Os mostramos como integrar esta fantastica herramienta con Netbeans
Patrones de GRASP	Os presentamos una introducción a los patrones de asignación de responsabilides y su relación con el proceso unificado.
Modelado UML con Visual Paradigm	Os mostramos como instalar y utilizar la versión gratuita de Visual Paradigm for UML. Cabe destacar que pemite extraer elementos de diseño desde textos de análisis.
Gestión de proyectos con project	En este tutorial os enseñaremos crear un plan, realizar el seguimiento del proyecto, como cerrar el proyecto y comunicar los resultados
Problemas al planificar un proyecto	En este tutorial/artículo os presentamos una plantilla modelo (básica) para un proyecto software (orientado a aplicaciones Web/Java OOP) y os comentamos por qué es tan difícil cumplir con un plan de proyecto informático
Aplicaciones con JSPs	Os mostramos como construir una aplicación con JSP que acceda a MySQL
Repositorio CVS en Windows	Os mostramos como montar un servidor para el control de versiones CVS en Windows asi como acceder a él a través de WinCVS
Despliegue gráfico de EJBs	Os mostramos como crear y desplegar de un modo gráfico un EJB de sesión el el servidor de aplicaciones de referencia de Sun