

Arquitectura del Sistema

CheckPoint

Laboratorio de Desarrollo de Software

GVR



*La Arquitectura del software, comprende el conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del software en tiempo de ejecución.*

*Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto. Este documento describe en términos generales, las ideas principales detrás de la arquitectura escogida para el mismo.*

Tabla de contenido

[Introducción 3](#_Toc257618836)

[Propósito 3](#_Toc257618837)

[Alcance 3](#_Toc257618838)

[Definiciones, Acrónimos, y Abreviaturas 3](#_Toc257618839)

[Referencias 3](#_Toc257618840)

[Panorama General 3](#_Toc257618841)

[Representación Arquitectónica 3](#_Toc257618842)

[Objetivos Arquitectónicos y Restricciones 3](#_Toc257618843)

[Objetivos Generales 3](#_Toc257618844)

[Objetivos Específicos 3](#_Toc257618845)

[Descripción de Procesos 3](#_Toc257618846)

[Vista de Caso de Uso 3](#_Toc257618847)

[Descripción de los Actores 3](#_Toc257618848)

[Contexto del sistema 3](#_Toc257618849)

[Vista Lógica 3](#_Toc257618850)

[Perspectiva General 3](#_Toc257618851)

[Paquetes de Diseño importantes arquitectónicamente 3](#_Toc257618852)

[Vista de Procesos 3](#_Toc257618853)

[Vista de Liberación 3](#_Toc257618854)

[Vista de Implementación 3](#_Toc257618855)

[Perspectiva General 3](#_Toc257618856)

[Capas 3](#_Toc257618857)

[Vista de Datos (opcional) 3](#_Toc257618858)

[Tamaño y Rendimiento 3](#_Toc257618859)

[Calidad 3](#_Toc257618860)

[Diagramas 3](#_Toc257618861)

[Diagramas de Despliegue 3](#_Toc257618862)

[Diagrama de Objetos 3](#_Toc257618863)

[Diagramas de Paquetes 3](#_Toc257618864)

Arquitectura del Sistema

Introducción

Este documento pretende dar una visión general (de alto nivel) sobre la arquitectura del sistema en desarrollo, los objetivos y restricciones, los casos de uso significativos, los patrones de arquitectura aplicados, las principales decisiones de diseño.

Nos da, además una vista general del resto de los artefactos generados en el proceso de desarrollo.

Propósito

Este documento de arquitectura de software (por sus siglas en inglés, SAD) tiene como propósito especificar claramente la arquitectura de software a ser usada para el desarrollo del sistema checkpoint.

Se pretenden plasmar en términos arquitectónicos y de diseño todos los requerimientos definidos en el Documento de Especificación de Requerimientos de Software (SRS).

La especificación de la arquitectura se da por medio de diferentes vistas de la misma para ilustrar diferentes aspectos del sistema.

Este documento va dirigido a distintos tipos de actores involucrados en el desarrollo del proyecto, tales como Desarrolladores, Tutores y clientes. Los desarrolladores pueden utilizar este documento como base para la documentación del desarrollo de productos de software en proyectos de diferente porte. El tutor puede tomar este documento como base para mostrar la importancia de la arquitectura en el desarrollo de software, así como el rol del arquitecto de software. Es de nuestro interés intercambiar con ellos ideas y resultados. Desde el punto de vista de un desarrollador, este documento le brindará, con certeza, una buena razón para darle a la arquitectura de software la importancia que tiene en todo proyecto de desarrollo.

Alcance

Este documento profundiza principalmente en las vistas de casos de uso y lógica, incluyendo algunos elementos significativos de las otras vistas.

Se les dará principal importancia a los casos de uso de la funcionalidad principal del sistema. Finalmente, este documento es la descripción de arquitectura del caso de estudio, no es un instructivo de cómo elaborar un documento de arquitectura del sistema; en otras palabras, el lector no encontrará aquí comentarios sobre qué tipo de información debe ser incluida en el documento ni cuáles son los criterios a utilizar en casos generales.

También son detallados y direccionados otros aspectos importantes con respecto a los requerimientos no funcionales como lo son el tamaño, desempeño y la calidad de la arquitectura del sistema.

En este documento se definen los módulos a desarrollar y la manera en que éstos se relacionan con los componentes adquiridos. El sistema contará con las funcionalidades definidas en el SRS y mapeadas al software en este documento.

Definiciones, Acrónimos, y Abreviaturas

Las palabras desconocidas o ambiguas que son utilizadas por primera vez en este documento se encuentran definidas en el documento “Glosario del Proyecto”, se debe remitir al mismos para encontrar allí términos importantes de este documento.

Referencias

[1] G. Reese, *Database Programming with JDBC and Java,* Second Edition, O’Reilly, Noviembre

de 2000, 128‐133

[2] C. Reynoso, N. Kiccillof, *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*,

Universidad de Buenos Aires, Marzo de 2004, 17‐19

[3] S. Burbeck. *“Application programming in Smalltalk‐80: How to use Model‐View‐Controller*

*(MVC)”.*University of Illinois in Urbana‐Champaign, Smalltalk Archive, http://stwww.

cs.uiuc.edu/users/smarch/st‐docs/mvc.html.

[4] Enterprise Solution Patterns: Model‐View‐Controller. Microsoft Patterns & Practices,

http://msdn.microsoft.com/practices/type/Patterns/Enterprise/DesMVC/, 2003.

[5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern‐Oriented Software*

*Architecture. A System Of Patterns,* JOHN WILEY & SONS. Inglaterra, Octubre de 1996, 31‐

52

[6] D. Garlan, M. Shaw. *An introduction to software architecture*. CMU Software Engineering

Institute Technical Report, CMU/SEI‐94‐TR‐21, ESC‐TR‐94‐21, 1994.

[7] Object Management Group, Inc. UML® Resource Page [página de Internet]. 2009

[14/09/09]. Disponible en: http://www.uml.org/

[8] KRUCHTEN, P, Architectural Blueprints—The “4+1” View Model of Software Architecture,

IEEE Software 12 (6), November 1995, p. 42‐50.

[9] B. Bruegge, A.H. Dutoit. *Object‐Oriented Software Engineering, Conquering Complex and*

*Changing Systems,* Prentice Hall, 1999, p. 14

[10] ISO, International Organization for Standardization, "ISO 9126‐1:2001, Software

engineering ‐ Product Quality, Part 1: Quality model", 2001.

[11] Frank Buschmann – Regine Meunier – Hans Rohnert – Peter Sommerland – Michael Stal,

*Pattern Oriented Software Architecture*, Volumen 1, AG Germany, WILEY, 1996, p. 457.

[12] JClic [Homepage en Internet] Cataluña ‐ España: XTEC [01/12/2009]. Disponible en:

http://clic.xtec.cat/es/index.htm

[13] Designing a Three‐Tier Architecure Pattern Language Design Fest, EuroPLoP 2001

Kloster Irsee/Germany, July 4‐8 Proceedings:

http://posa3.org/workshops/ThreeTierPatterns/;http://www.hillside.net/patterns/EuroPL

oP2001/FocusGroups\_DesignFests/three‐tier‐design‐fest.htm

[14] Pattern‐Oriented Software Architecture On Patterns and Pattern Languages Frank

Buschmann, Siemens, Munich, Germany Kevlin Henney, Curbralan, Bristol, UK Douglas C.

Schmidt, Vanderbilt University, Tennessee, USA Volume 5

[15] Documento de Descripción Arquitectónica de Estudiantes de Arquitectura de Software de

la Pontificia Universidad Javeriana del semestre 2008‐3.

[16] IEEE Recommended Practice for Software Requirements Specifications, 830‐1998.

[17] IEEE Recommended Practice for Software Design Descriptions, 1016‐1998.

[18] Bustacara Medina, Cesar Julio – [homepage de Internet] Plantilla Documento SAD

[01/05/2009]. Disponible en: http://sophia.javeriana.edu.co/~cbustaca/

[19] Camargo, E; Cardeso, F; Nuñez M. Guia de Estudio[Articulo en internet]Venezuela;

[4/12/2009]. Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS‐

6116/Guia%20Arquitectura%20v.2.pdf

Implementación de un sistema de información basado en componentes con J2EE. S. Bengochea, F. Fajardo, J. Ferré. Reporte técnico 03-17, InCo-PEDECIBA, 2003. UML Components. J. Cheesman, J. Daniels. Addison Wesley, 2001. Design Patterns. E. Gamma, et al. Addison Wesley, 1995. Architectural Blueprints—The “4+1” View Model of Software Architecture, P. Kruchten, Rational Software Corp. IEEE Software 12 (6), November 1995. Applying UML and Patterns. C. Larman. Prentice-Hall, 2002. IBM Rational Unified Process. http://www.rational.com/rup, 2003. Unified Modeling Language. OMG. http://www.omg.org/uml, 2003. Yoopeedoo (UPEDU). http://www.yoopeedoo.org, 2003.

Panorama General

Este documento está organizado de forma tal que el lector conforme va leyendo, puede encontrar la descripción de la arquitectura del sistema con ayuda de diferentes aspectos importantes, estos están distribuidos de la siguiente manera. La sección 2. Representación Arquitectónica, describe que arquitectura de software será utilizada para el desarrollo del sistema y como será representada. La sección 3. Objetivos y Restricciones Arquitectónicas, muestra como los requerimientos no funcionales impactan en la arquitectura de software seleccionada. La sección 4. Vista de Casos de Uso, contiene los casos de uso del sistema. La sección 5. Vista Lógica, tiene una descripción de las partes más importantes de la arquitectura.

La sección 6. Vista de Procesos, muestra la descripción del sistema en los procesos que ejecuta.

La sección 7. Vista de Despliegue, describe la manera como se va a ejecutar el sistema. La sección 8. Vista de Implementación, describe el modelo de implementación del sistema. La sección 9. Vista de Datos, muestra la manera cómo van a ser almacenados los datos del sistema. La sección 10. Tamaño y Desempeño, describe las principales características del tamaño y del desempeño del sistema. La sección 11. Calidad, muestra los atributos de calidad que son tenidos en cuenta.

Representación Arquitectónica

La características inherentes al sistema hace que la vista de casos de uso y la vista lógica sean las más relevantes y por ello serán las más extensas. La arquitectura está representada por diferentes vistas utilizando notación UML de forma que permitan visualizar, entender y razonar sobre los elementos significativos de la arquitectura e identificar las áreas de riesgo que requieren mayor detalle de elaboración. Este documento es una forma de comunicar el modelo del subsistema, presentando la información y discusiones estructuradamente. La arquitectura del subsistema se descompone en las siguientes dimensiones:

Requerimientos: Requerimientos funcionales y no-funcionales del sistema.

Elaboración: Representación lógica del sistema y representación de tiempo de ejecución.

Implementación: Vista de módulos implementados, potenciales escenarios de infraestructura y el deployment de los módulos.

La siguiente sección detalla las vistas de la arquitectura que serán utilizadas para cubrir las dimensiones mencionadas.

Objetivos Arquitectónicos y Restricciones

[Esta sección describe los requerimientos y objetivos que tienen algún impacto significante en la arquitectura; por ejemplo, seguridad, garantía, privacidad, portabilidad, distribución, reuso. Captura restricciones especiales que puedan aplicar estrategias de diseño e implementación, herramientas de desarrollo, estructura del equipo, programación, código legado, etc.]

Objetivos Generales

El sistema requiere de una fuerte interacción datos cargados por los usuarios de la interfaz móvil, y datos almacenados en la base de datos por medio de las interfaces web que permitan plasmar las valoraciones sobre los servicios y generar información útil para los usuarios que corresponda. En este contexto, el objetivo general del proyecto será: Desarrollar un sistema de valoración de servicios para los distintos encargados de servicios de la unidad académica que lo soliciten, que permita a los mismos un manejo simple y se presente como una herramienta útil para el monitoreo y evaluación de la calidad de los servicios de los que se encargan, además de permitirle a los administradores la generación de los distintos servicios con sus respectivos encargados y a los usuarios valoradores brindarles la posibilidad de expresar su percepción de acuerdo a la lista de valoraciones prefijadas sobre un determinado servicio del cual son consumidores.

Objetivos Especificos

Para lograr el objetivo mencionado, se describen metas específicas que se proponen alcanzar.

1. Investigar framework Ionic para desarrollo de aplicaciones móviles multiplataforma.

* Forma en que se utiliza la cámara como lector de código QR
* Forma en la que se conecta por medio de PHP a la base de datos externa
* Forma en la que se genera y utiliza una base de datos interna

1. Investigar UARGflow y su arquitectura para reutilizarla en el desarrollo de interfaz web.
2. Diseñar un esquema de base de datos eficiente y acorde a las necesidades de los usuarios, de manera de plasmar correctamente la información obtenida en las etapas de relevamiento.

Descripción de Procesos

Actualmente no existe un proceso definido, tratándose de un sistema novedoso el proceso se describe en función de las necesidades del cliente.

La persona que está a cargo de gestionar un determinado servicio que la unidad académica brinda a la comunidad universitaria puede sentir la necesidad de evaluar la calidad que se percibe sobre el mismo, en ese sentido, es posible consultar a la comunidad universitaria sobre la base de un conjunto de valoraciones definidas por el (de acuerdo a los conceptos que desee evaluar).

La comunidad se expide seleccionando alguna de estas valoraciones prefijadas de acuerdo a la realidad del servicio en operación, así mismo es deseable que el que realizo la valoración reciba una realimentación al respecto para sentir que está siendo escuchado y no desalentar la participación.

Al cabo de algún tiempo con una cierta cantidad de valoraciones hechas e información recabada el encargado será capaz de generar informes que le permitan evaluar y mejorar el servicio y su gestión.

Vista de Caso de Uso

Esta vista presenta la percepción que tiene el usuario de las funcionalidades del sistema. Se presenta el proceso de negocio más importante y los casos de uso críticos que se derivan de éste. Este capítulo provee el contexto y determina el alcance del resto del documento. Primeramente se describe el Negocio. Luego se presenta el modelo del dominio para el Sistema. Se identifican actores y se detallan los casos de uso significativos.

Descripción de los Actores

|  |  |
| --- | --- |
| *Nombre del Actor:* | Administrador del Sistema |
| *Definición:* | Es el encargado de crear servicios en el sistema a pedido de los encargados de servicios. Además crean las ubicaciones.  Puede generar informes estadísticos sobre todos los servicios del sistema. |

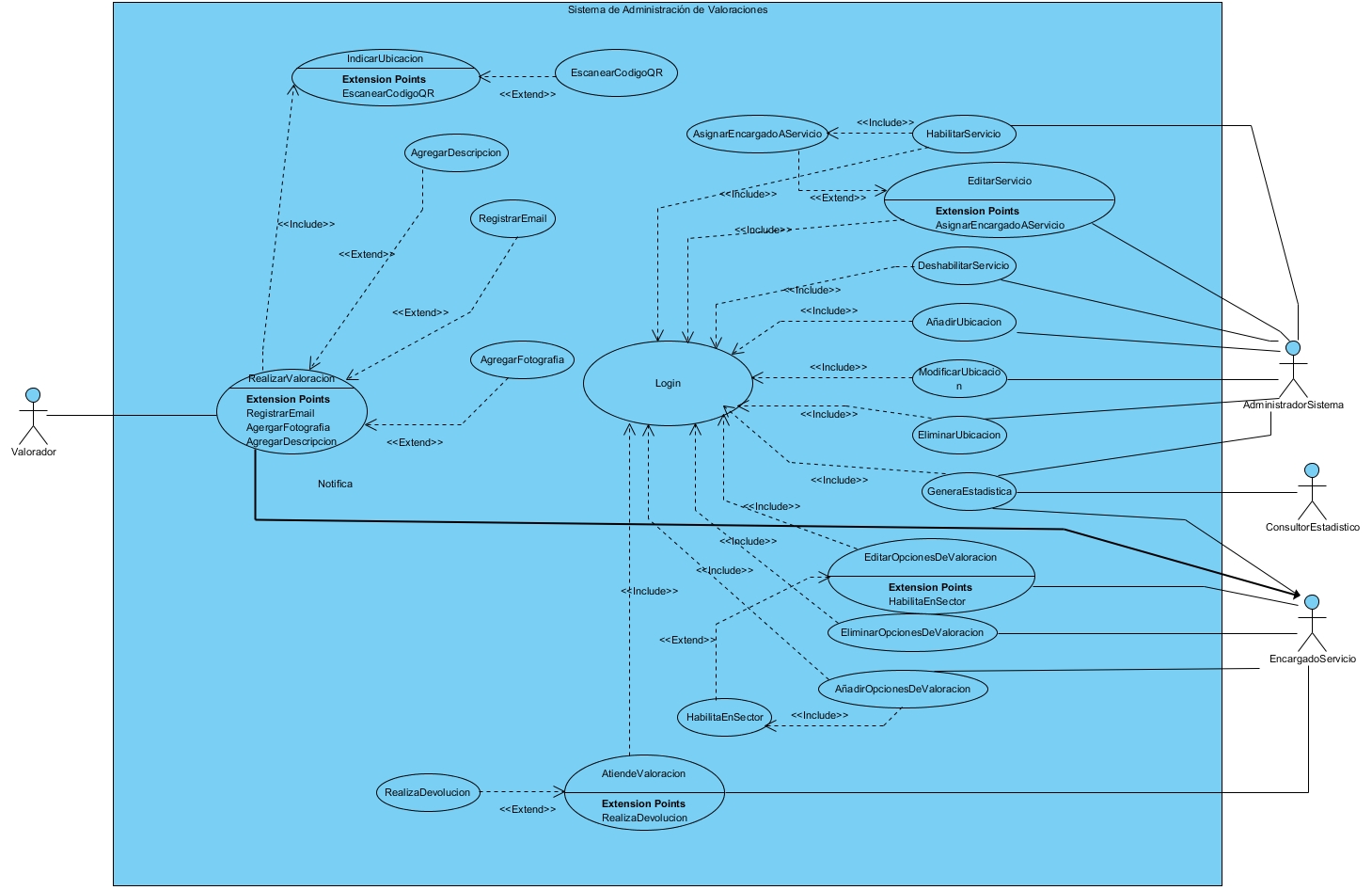
|  |  |
| --- | --- |
| *Nombre del Actor:* | Encargado de Servicio |
| *Definición:* | Es la persona encargada de gestionar el servicio. Este se encarga de crear las valoraciones que crea convenientes para su servicio y de asociar esas valoraciones a las ubicaciones donde dicho servicio efectivamente es brindado.  Puede generar informes estadísticos sobre su servicio. |

|  |  |
| --- | --- |
| *Nombre del Actor:* | Consultor de Estadísticas |
| *Definición:* | Se trata de la persona que solo puede generar informes estadísticos sobre cualquiera de los servicios en el sistema |

|  |  |
| --- | --- |
| *Nombre del Actor:* | Usuario Valorador |
| *Definición:* | Una persona de la comunidad universitaria que hace uso de los servicios que esta brinda, puede seleccionar alguna de las valoraciones predefinidas sobre algún servicio y enviarla.  Puede recibir una respuesta de tratamiento de la misma si lo desea. |

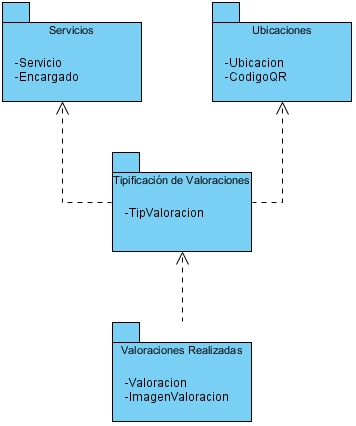
Contexto del sistema

Los casos de uso críticos para el proceso se describen en esta sección. Primero se indica las relaciones entre los casos de usos detectados y luego se presenta la versión expandida de los mismos. En el siguiente diagrama se puede observar como los distintos casos de uso interactúan unos con otros y como está compuesto el contexto general del sistema.



Vista Lógica

Según lo diseñado el sistema está compuesto por un conjunto de paquetes, cada uno con una función específica. A continuación vemos los distintos paquetes interactuando entre si.



Perspectiva General

[Esta sección describe la composición general de modelo de diseño en términos de jerarquía y capas.]

Paquetes de Diseño importantes arquitectónicamente

[Para cada paquete significante, incluye una sección con su nombre, una descripción breve y un diagrama con estructuras significantes y paquetes contenidos dentro del paquete.

Para cada estructura significante en el paquete, incluye su nombre, una breve descripción, y, opcionalmente, una descripción de algunas de sus principales responsabilidades, operaciones y atributos.]

Vista de Procesos

[Esta sección describe la descomposición del sistema dentro del proceso ligeros (simple hilos de control) y procesos pesados (grupos de procesos ligeros). Organice la sección por grupos de procesos que se comunican e interactúan. Describe el principal modo de comunicación entre procesos, tal como mensajes, interrupciones.]

Vista de Liberación

[Esta sección describe una o más configuraciones físicas de la red (hardware) en las cuales el software se libera y funciona. Es una vista del modelo de liberación. En un mínimo para cada configuración debe indicar los nodos físicos (computadoras, CPUs) que ejecutan el software y sus interconexiones (bus, LAN, punto al punto, etcétera.) también incluye un mapa de los procesos de la vista de procesos sobre los nodos físicos.]

Vista de Implementación

[Esta sección describe la estructura total del modelo de la puesta en marcha, la descomposición del software en capas y subsistemas en el modelo de implementación y cualquier componente arquitectónico significativo.]

Perspectiva General

[Esta subdivisión nombra y define varias capas y su contenido, las reglas que gobiernan la inclusión a una capa dada, y los límites entre las capas. Incluir un diagrama componente que demuestre las relaciones entre las capas.]

Capas

[Para cada capa, se debe incluir una subdivisión con su nombre, una enumeración de los subsistemas situados en la capa, y un diagrama de componentes.]

Vista de Datos (opcional)

[Se debe colocar una descripción del almacenamiento de datos persistentes del sistema. Esta sección es opcional si hay o no datos persistentes, o si la traducción entre el modelo del diseño y el modelo de los datos es trivial.]

Tamaño y Rendimiento

[Colocar una descripción con las características principales de la dimensión del software que afectan la arquitectura, así como las restricciones de desempeño del objetivo.]

Calidad

[Describir cómo la arquitectura del software contribuye a la capacidad general (con excepción de funcionalidad) del sistema: extensibilidad, confiabilidad, portabilidad, etcétera. Si estas características tienen significación especial, por ejemplo seguridad, garantía o implicaciones de privacidad, deben ser claramente delineadas.]

Diagramas

Diagramas de Despliegue

[Básicamente este tipo de diagrama se utiliza para modelar el Hardware utilizado en la implementación del sistema y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos, componentes y asociaciones. En el UML 2.0 los componentes ya no están dentro de nodos, en cambio puede haber artefactos (archivo, un programa, una biblioteca o Base de datos) u otros nodos dentro de nodos.

Además los Diagramas de Despliegue muestran la configuración en funcionamiento del sistema incluyendo su software y su hardware. Para cada componente de un diagrama es necesario que se deba documentar las características técnicas requeridas, el tráfico de red, el tiempo de respuesta, etc.]

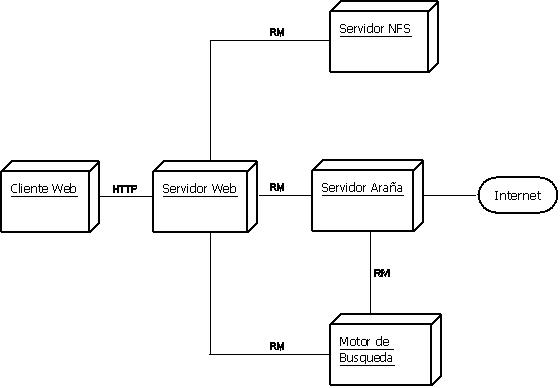


Diagrama de Objetos

[Forma parte de la vista estática del sistema. En este diagrama se modelan las instancias de la clase del Diagrama de Clases, cabe aclarar que el mismo cuenta con objetos y enlaces. En estos diagramas también es posible encontrar las clases para tomar como referencia su instanciación. En otras palabras el Diagrama de Objetos muestra un conjunto de objetos y sus relaciones en un momento concreto. Los Diagramas de Objetos son realmente útiles para modelar estructuras de datos complejas. ]



Diagramas de Paquetes

[Los diagramas de Paquetes se usan para reflejar la organización de paquetes y sus elementos. Los usos más comunes de para los diagrama de paquete son para organizar diagramas de casos de uso y diagramas de clases, estos paquetes son como grandes contenedores de clases.

Los elementos contenidos en un paquete comparten el mismo espacio de nombres, esto significa que los elementos contenidos en un mismo espacio de nombres específico deben tener nombres únicos. Como otra característica de estos diagramas, cada paquete se debe identificar con un nombre único y opcionalmente mostrar todos los elementos dentro del mismo. ]

