

Análisis y Diseño de Sistemas II

Curso	Análisis y Diseño de Sistemas II (4690)
Formato	Manual de curso
Autor Institucional	Cibertec
Páginas	121 p.
Elaborador	Guzmán Mariluz, Gisella
Revisor de Contenidos	

Índice	
Presentación	5
Red de contenidos	6
UNIDAD DE APRENDIZAJE 1: MODELADO ÁGIL PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS	
1.1 Tema 1 : Modelado Ágil para el análisis y diseño de sistemas	8
1.1.1 : Elementos claves de la metodología ágil para el análisis y diseño de sistemas	9
1.1.2 : Ventajas y desventajas	10
1.2 Tema 2 : Disciplina Modelado según AUP – Modelado de la Solución	13
1.2.1 : Modelado de la solución según AUP	14
1.2.2 : Actividades, Roles y Artefactos	15
1.2.3 : La importancia del prototipado	16
1.3 Tema 3 : Creación de perfiles y estereotipos para el análisis y diseño de sistemas	22
1.3.1 : Perfiles y estereotipos en un entorno CASE	22
1.3.2 : Definiendo perfiles para el análisis de sistemas	24
1.3.3 : Definiendo perfiles para el diseño de sistemas	25
1.3.4 : Agregando perfiles en un proyecto	26
UNIDAD DE APRENDIZAJE 2: ANÁLISIS DE SISTEMAS	
2.1 Tema 4 : Análisis de Sistemas	31
2.1.1 : Análisis de sistemas según AUP	31
2.1.2 : Artefactos	32
2.1.3 : Reglas básicas de nomenclaturas	32
2.1.4 : Actividades para identificar los artefactos del modelo de análisis de sistema	33
2.2 Tema 5 : Desarrollo del Modelo de Análisis Funcional de sistemas para un caso	35
2.2.1 : Especificaciones de Casos de Uso del Caso Fashion Importaciones	35
2.2.2 : Configuración de perfiles para el Modelo de Análisis de Sistemas - Caso: Venta de Productos de Fashion Importaciones	40
2.3 Tema 6 : Análisis de la Arquitectura	51
2.3.1 : Análisis de la Arquitectura de sistemas	42
2.3.2 : Análisis de la Arquitectura para el caso Fashion Importaciones	42
2.3.3 : Análisis de Casos de Uso para un dato maestro y un dato transaccional	44
2.3.4 : Análisis de los Casos de Uso para el caso Fashion Importaciones	44

2.4 Tema 7	:	Análisis de Casos de Uso	49
2.4.1	:	Análisis de Casos de Uso para un dato maestro y dato transaccional	49
2.4.2	:	Análisis de Casos de Uso para el caso Fashion Importaciones	55
UNIDAD DE APRENDIZAJE 3: MODELADO DE DATOS			
3.1 Tema 8	:	Modelo de datos	58
3.1.1	:	Modelo Conceptual	59
3.1.2	:	Modelo Lógico	66
3.1.3	:	Modelo Físico	69
3.2 Tema 9	:	Auditoría en base de datos	75
3.2.1	:	Definición, Importancia y Objetivos.	75
3.2.2	:	Modelo de Datos con tablas de auditoría	76
UNIDAD DE APRENDIZAJE 4: DISEÑO DE SISTEMAS			
4.1 Tema 10	:	Diseño de Sistemas	79
4.1.1	:	Diseño de sistemas según AUP	80
4.1.2	:	Artefactos	81
4.1.3	:	Reglas básicas de nomenclaturas	81
4.1.4	:	Actividades para identificar los artefactos del modelo de diseño de sistema	81
4.2 Tema 11	:	Desarrollo del Modelo de Diseño de sistemas para un caso	83
4.2.1	:	Configuración de perfiles para el Modelo de Diseño de Sistemas - Caso: Venta de Productos de Fashion Importaciones	83
4.3 Tema 12	:	Patrones de Diseño	85
4.3.1	:	Importancia de aplicar patrones de diseño	85
4.3.2	:	Catálogo de Patrones de Diseño más utilizados en proyectos de sistemas	87
4.4 Tema 13	:	Diseño de la Arquitectura	92
4.4.1	:	Diseño de la Arquitectura de sistemas	92
4.4.2	:	Diseño de la Arquitectura para el caso Fashion Importaciones	95
4.5 Tema 14	:	Diseño de Casos de Uso	96
4.5.1	:	Diseño de Casos de Uso para un dato maestro	100
4.5.2	:	Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte I	105
4.5.3	:	Diseño de Casos de Uso para un dato transaccional	105
4.5.4	:	Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte II	106
Bibliografía			120

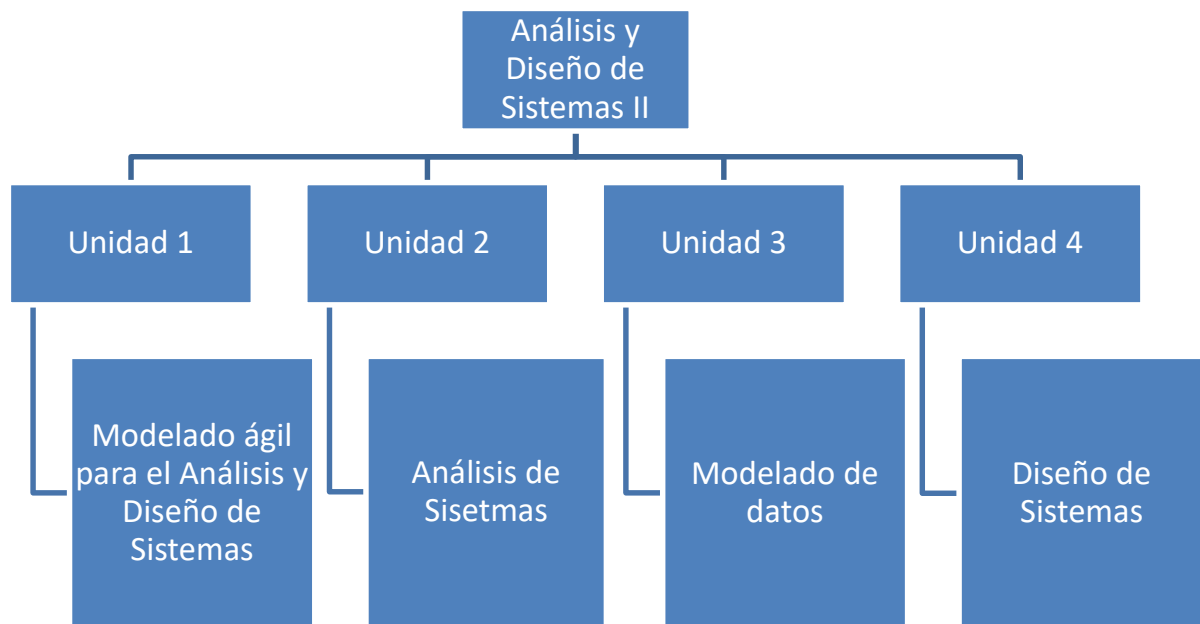
Presentación

Actualmente, el análisis y diseño de sistemas ha evolucionado en el ciclo de vida de desarrollo de software, específicamente si el equipo de desarrollo ha considerado trabajar con metodologías ágiles. El enfoque ágil permite que el analista de sistemas se concentre en artefactos relevantes para plasmar una arquitectura de sistemas robusta que pueda ser lo suficientemente flexible.

El curso de Análisis y Diseño de Sistemas II, pertenece a la línea formativa y se dicta en la carrera de Computación e Informática. El curso imparte conocimientos relacionados al desarrollo de software de calidad basado en la metodología AUP, el cual se enfoca en el análisis y diseño de sistemas de la disciplina Modelado ágil de AUP (Agile Unified Process).

El curso es teórico-práctico: consiste en un taller de desarrollo de proyectos de software. En primer lugar, se inicia con una descripción general del enfoque ágil para el Análisis y Diseño de Sistemas. Continúa con el análisis funcional de sistema como parte de la disciplina de modelado de AUP. Luego, se presenta el flujo de trabajo para realizar el modelado de datos. Por último, se desarrolla el diseño de sistemas según la primera disciplina de AUP.

Red de contenidos





MODELADO ÁGIL PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno es capaz de identificar las ventajas y desventajas del modelado ágil AUP (Agile Unified Process) para el análisis y diseño de sistemas. Asimismo, el alumno es capaz de crear perfiles y estereotipos personalizados para diagramas UML en una Herramienta CASE.

TEMARIO

- 1.1 Tema 1 : Modelado Ágil para el análisis y diseño de sistemas**
 - 1.1.1 : Elementos claves de la metodología ágil para el análisis y diseño de sistemas
 - 1.1.2 : Ventajas y desventajas
- 1.2 Tema 2 : Disciplina Modelado según AUP – Modelado de la Solución**
 - 1.2.1 : Modelado de la solución según AUP
 - 1.2.2 : Actividades, Roles y Artefactos
 - 1.2.3 : La importancia del prototipado
- 1.3 Tema 3 : Creación de perfiles y estereotipos para el análisis y diseño de sistemas**
 - 1.3.1 : Perfiles y estereotipos en un entorno CASE
 - 1.3.2 : Definiendo perfiles para el análisis de sistemas
 - 1.3.3 : Definiendo perfiles para el diseño de sistemas
 - 1.3.4 : Agregando perfiles en un proyecto

1.1. MODELADO ÁGIL PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS

El análisis y diseño de sistemas que los analistas de sistemas llevan a cabo busca comprender qué necesitan los usuarios para analizar la entrada o el flujo de datos de manera sistemática, procesar o transformar los datos, almacenarlos y producir información en el contexto de una organización específica. Mediante un análisis detallado, los analistas buscan identificar y resolver los problemas correctos. Además, el análisis y diseño de sistemas se utiliza para analizar, diseñar e implementar las mejoras en el apoyo para los usuarios y las funciones de negocios que se puedan llevar a cabo mediante el uso de sistemas de información. (Kendall, 2011).

Como especialistas en el campo de la Ingeniería de software debemos tener en cuenta que si un sistema se instala sin una planificación apropiada, a menudo los usuarios quedan muy insatisfechos y muy pronto dejan de usar el sistema. En cualquier proceso de desarrollo de software, el análisis y diseño añade estructura a los sistemas, y constituye una actividad costosa que de otra manera se realizaría al azar. Se puede considerar como una serie de procesos que se llevan a cabo en forma sistemática para mejorar los procesos de una organización mediante el uso de sistemas de información. Por consiguiente, implica trabajar con los usuarios actuales y eventuales de los sistemas de información para ofrecerles soporte tecnológico en sus procesos organizacionales.

En definitiva, la participación del usuario en el proyecto es imprescindible para el desarrollo exitoso de los sistemas de información que requieren, y ello es enfocado en cualquier metodología de desarrollo de software. Sin embargo, existen metodologías que toman mucho tiempo en estas actividades de análisis y diseño, es decir, antes de iniciar la implementación del software el equipo está concentrado en realizar modelos y documentación para establecer una arquitectura estable de sistemas.

Bajo un enfoque ágil, según lo expresado por Scott Ambler (Ambler, 2006), el objetivo del esfuerzo inicial de modelado **es simplemente es modelar lo suficiente**, es decir, no se necesita modelar muchos detalles. Si nos enfocamos en la arquitectura de sistema a implementar, entonces, su modelo inicial es tratar de identificar una arquitectura que tenga una buena oportunidad de funcionar. Esto permitirá establecer una dirección viable para el proyecto de desarrollo de software y proporcionará información suficiente para organizar el equipo en torno a la arquitectura del sistema, la cual es particularmente importante si se tendrá un equipo grande o distribuidos.

El modelador ágil, creará diagramas de formato libre que exploren la infraestructura técnica, los modelos de dominio iniciales para explorar las principales entidades empresariales y sus relaciones y, opcionalmente, los casos de cambios para explorar los posibles requisitos de nivel de arquitectura que el sistema puede necesitar admitir algún día. En iteraciones posteriores, tanto sus requisitos iniciales como sus modelos de arquitectos iniciales tendrán que evolucionar a medida que aprenda más, pero por ahora el objetivo es obtener algo que apenas sea lo suficientemente bueno para que el equipo de desarrollo pueda ponerse en marcha. En versiones posteriores, pueden decidir acortar la fase de Inicio a varios días, varias horas o incluso eliminarlo por completo según lo defina la situación en estudio. El secreto es mantener las cosas simples. Recuerde que **“no se necesita modelar muchos detalles, simplemente se necesita modelar lo suficiente”**. En este sentido, para una arquitectura, un boceto de pizarra que describe cómo se construirá el sistema de extremo a extremo es lo suficientemente bueno. Sin embargo, luego de tener una versión un poco más estable, será ideal llevarlo a un entorno CASE que permita elaborarlo de una forma rápida a fin de mapearlos en cada incremento.

Muchos desarrolladores tradicionales tendrán dificultades con un enfoque ágil para el modelado inicial porque durante años se les ha dicho que necesitan definir modelos integrales al principio

de un proyecto. El desarrollo ágil de software no es en serie, es iterativo e incremental (evolutivo). Con un enfoque evolutivo, el modelado detallado se realiza justo a tiempo (JIT) durante las iteraciones de desarrollo en las sesiones de *storming* o de lluvia de ideas del modelo.

1.1.1. Elementos claves de la metodología ágil para el análisis y diseño de sistemas

AUP es una metodología de desarrollo de software que se basa en **valores y principios** que forman parte del manifiesto ágil. Los cuatro valores **son comunicación, simpleza, retroalimentación y valor**, tal como se muestra en la siguiente figura. Este enfoque recomienda que los analistas de sistemas adopten estos valores en todos los proyectos que emprendan y no sólo cuando adopten la metodología ágil. De este modo, los sistemas que se diseñan se pueden desarrollar con rapidez.



Figura 1: Los valores son imprescindibles para la metodología ágil
Fuente. - Tomado de (Kendall, 2011)

Por otro lado, los principios del manifiesto ágil que se muestran en la figura 2, permiten a los responsables del análisis y diseño de sistemas acercarse a un mundo perfecto o ideal donde los clientes y el equipo de desarrollo de software logan tener una excelente comunicación: se verían directamente a los ojos y no sería necesaria ninguna otra forma de comunicación, en donde ambas partes estarían de acuerdo en todo momento.



Figura 2: Principios del enfoque ágil
Fuente. - Tomado de <http://www.scrumhuntersca.com/los-12-principios-del-manifiesto-agil/>

1.1.2. Ventajas y desventajas

En este punto, se tiene claro cuáles son los aspectos claves de un enfoque ágil para lograr tener éxito en un proyecto. Sin embargo, como todo paradigma o metodología es importante conocer las ventajas y desventajas de aplicarlo.

Las ventajas hacen referencia a los beneficios que consigue el equipo de desarrollo al adoptar el enfoque ágil que les permite dividir el proyecto en etapas y así centrarse en cada una de forma individual:

- **Gestión de las expectativas del usuario.** Los usuarios pueden participar en cada una de las etapas del proceso y proponer soluciones. De hecho, el proceso en su conjunto está pensado para un tipo de evaluación conjunta.
- **Resultados anticipados.** Cada etapa del proceso arroja una serie de resultados. No es necesario, por tanto, que el cliente espere hasta el final para ver el resultado.
- **Flexibilidad y adaptación a los contextos.** Se adapta a cualquier contexto, área o sector de la gestión. Es decir, no es una técnica exclusiva de ninguna disciplina.
- **Gestión sistemática de riesgos.** Del mismo modo, los problemas que aparecen durante los procesos de gestión que pueden afectar a un proyecto son gestionados en el mismo momento de su aparición. Esto es posible debido a que la intervención de los equipos de trabajo puede ser inmediata.

Las desventajas o limitaciones son las siguientes:

- **Funciona mucho mejor con equipos reducidos.** Si las empresas grandes optan por aplicarlo, es preciso que sean sectorizadas o divididas en grupos que tengan objetivos concretos. De lo contrario, en la práctica, el efecto de la técnica se perderá.
- **Requiere una exhaustiva definición de las tareas y sus plazos.** La división del trabajo en cada etapa, y de éstas en tareas específicas, son la esencia de esta metodología. De lo contrario, traerá dificultades en la gestión de los riesgos.
- **Exige que quienes la utilicen cuenten con una alta cualificación o formación.** El equipo necesita tener una base sólida y habilidades, las cuales son productos de la experiencia.
- **Falta de atención a la documentación.** Ello puede dificultar que los nuevos miembros del equipo accedan a la misma. Por tal motivo los expertos de la metodología sugieren documentar con un entorno CASE (gráficamente) o procesador de texto (textual) los modelos estables y los que son simplemente necesarios.
- **Alto nivel de interacción entre el cliente y los desarrolladores.** Se requiere una comunicación constante entre ambos para asegurar el éxito. Si la comunicación no es posible o presenta dificultades, Scott nos sugiere contar por lo menos con un usuario clave que esté disponible; de lo contrario, el proyecto estaría en riesgo.

ACTIVIDADES PROPUESTAS

- Complete los requisitos funcionales, casos de uso y actores para las actividades de negocio especificadas en la siguiente matriz de trazabilidad:

Matriz de Actividades Vs. Requisitos Funcionales					
Proceso	Actividades del Negocio	Responsable de Negocio	Requisitos	Casos de Uso	Actores
Venta de Productos	Elabora Cotización de productos	Vendedor			
	Registra Clientes	Producto			
	Busca datos de Clientes	Producto			
	Verifica Productos	Producto			
	Emite Comprobante de Pago	Producto			
	Imprime Comprobante de Pago	Producto			

Figura 3: Matriz de Actividades Vs. Requisitos Funcionales

Fuente. – Elaboración propia

- Identifique por lo menos un Objetivo de Negocio por cada problema registrado de un proceso de ventas. Recuerde describir objetivos SMART. Asimismo, marque con una “X” el aspecto que impacta cada objetivo enunciado.

ID_Prob	Problemática	Objetivos de negocio (SMART)	Aspecto que impacta el Objetivo			
			Tiempo	Costo	Calidad	Servicio
P_001	Errores al momento de generar el CDP					
P_002	Errores al momento de verificar el stock					
P_003	Demora en la generación del CDP					
P_004	Altos costos en impresión de CDP					

Figura 4: Matriz de Problemas Vs. Objetivos de negocio

Fuente. – Caso y Plantilla elaborados por profesor Jorge Aguilar Hernando

Resumen

1. El análisis y diseño de sistemas requiere de una planificación con el objetivo de establecer una arquitectura robusta y estable de un sistema de información y de satisfacción para el usuario que lo requiere.
2. El enfoque ágil recomienda que durante el proceso de análisis y diseño de sistemas “no se necesita modelar muchos detalles, simplemente se necesita modelar lo suficiente”. En este sentido, para definir una arquitectura de sistemas se puede realizar un boceto de pizarra e incluso utilizar una herramienta CASE para describir gráficamente el comportamiento del sistema de un modo rápido, sin considerar muchos detalles.
3. AUP es una metodología de desarrollo de software que se basa en valores y principios que forman parte del manifiesto ágil. Los cuatro valores son comunicación, simpleza, retroalimentación y valor.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <http://www.ambysoft.com/unifiedprocess/agileUP.html>
- <https://ebf.com.es/blog/ventajas-y-desventajas-de-las-metodologias-agiles-y-su-aplicacion-en-el-trabajo/>
- <https://www.omg.org/spec/UML/2.5.1/PDF>

1.2. DISCIPLINA MODELADO SEGÚN AUP – MODELADO DE LA SOLUCIÓN

El proceso unificado ágil (PUA) o Metodología AUP, por sus siglas en inglés, es una versión simplificada de RUP. Adopta una filosofía **“en serie para lo grande”** e **“iterativa para lo pequeño”** a fin de construir sistemas basados en computadora, liberando entregables incrementales (Ambler, 2006). Al adoptar las actividades en fases clásicas del RUP (concepción, elaboración, construcción y transición), el AUP brinda un revestimiento en serie (por ejemplo, una secuencia lineal de actividades de ingeniería de software) que permite que el equipo visualice el flujo general del proceso de un proyecto de software. Sin embargo, dentro de cada actividad, el equipo repite con objeto de alcanzar la agilidad y entregar tan rápido como sea posible incrementos de software significativos a los usuarios finales.

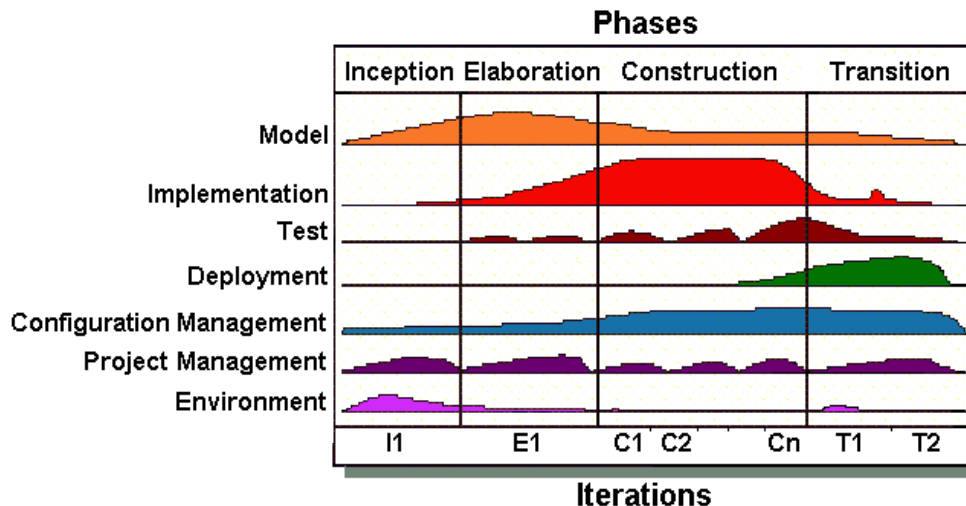


Figura 5: Estructura de AUP: Fases y disciplinas

Fuente. - Tomado de <http://www.ambysoft.com/unifiedprocess/agileUP.html#Serial>

Como bien sabemos y tal como se muestra en la estructura de AUP, la primera disciplina es “Modelo o Modelado”. Esta disciplina se centra en crear representaciones de UML de los dominios del negocio y el problema. En este sentido, los procesos que se llevan a cabo son los siguientes:

- **Entender el negocio** de la organización,
- Explorar el **dominio del problema**
- **Modelar una solución viable** para tratar el dominio del problema

Para conservar la agilidad, los modelos realizados en cada proceso deben ser “solo lo suficientemente buenos” para permitir que el equipo avance. En este apartado se explicará sobre el último proceso de la disciplina de modelado: “Modelar la solución”, en el cual se realiza el análisis y diseño de la arquitectura de sistemas.

1.2.1. Modelado de la solución según AUP

Como bien se precisó anteriormente, la primera disciplina de AUP es “Modelo o Modelado” cuyos roles, actividades y artefactos se muestran en la siguiente figura. Aquí se resalta las actividades del “Modelado de la solución”, donde se concentra el esfuerzo del análisis y diseño de sistemas para definir una arquitectura.

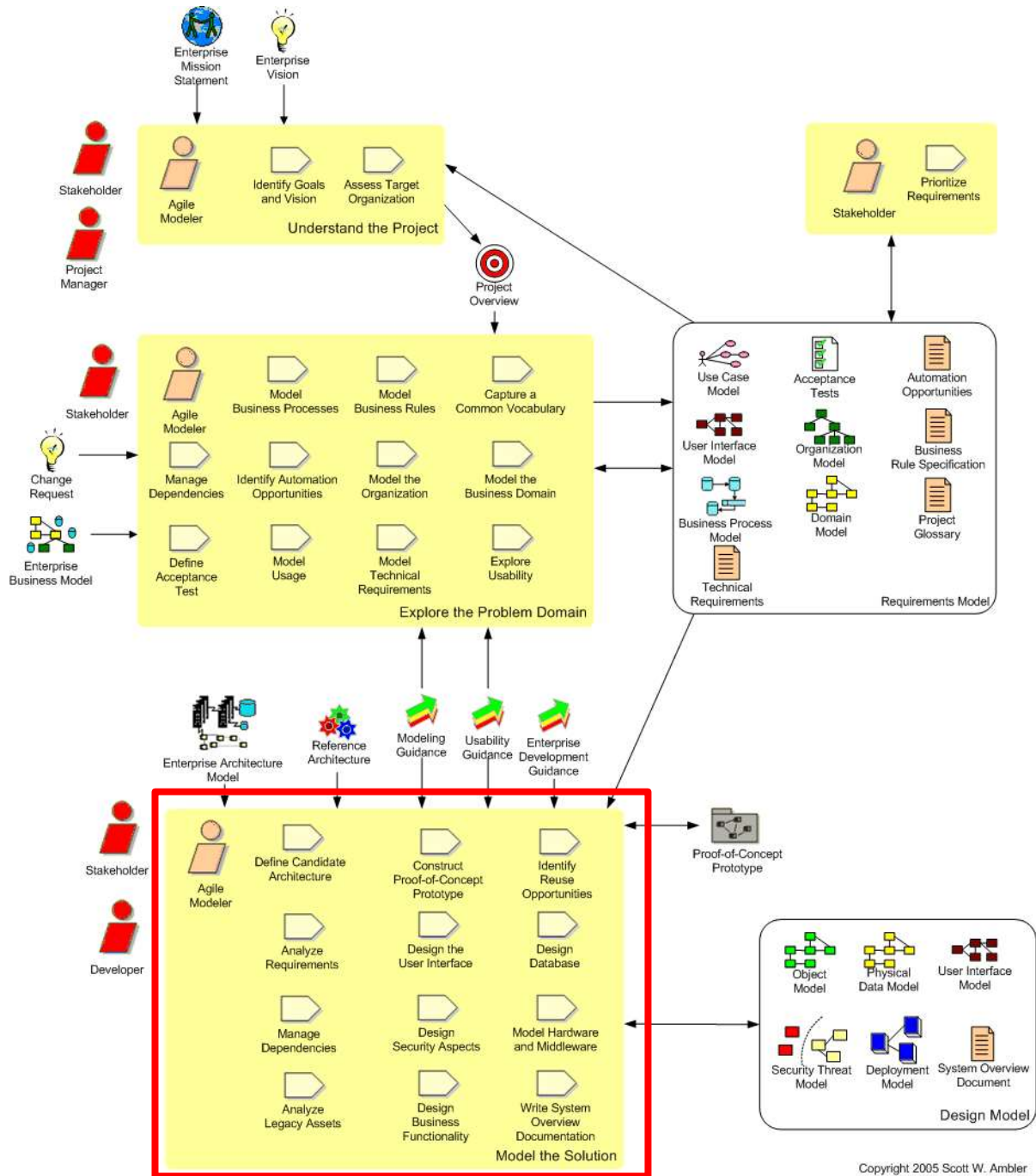
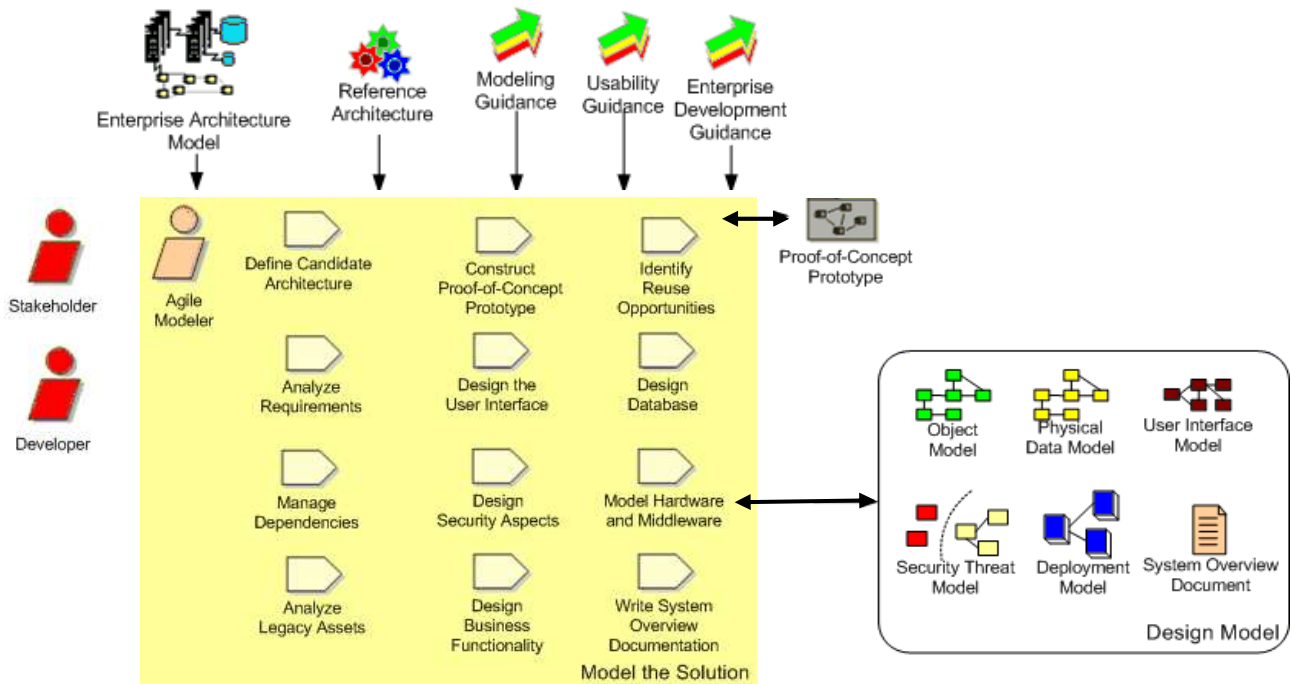


Figura 6: Disciplina de Modelado del AUP

Fuente. - Tomado de <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>

1.2.2. Actividades, Roles y Artefactos

La siguiente figura muestra las actividades, roles y artefactos generados en el Modelado de la solución según AUP:



Copyright 2005 Scott W. Ambler

Figura 7: Modelado de la solución

Fuente. - Tomado de <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>

El objetivo principal de las actividades en esta etapa “Modelado de la solución” es identificar una solución viable para abordar el dominio del problema. Esta solución viable se enfoca en la definición de una **arquitectura** a nivel hardware y software del sistema a desarrollar.

La arquitectura es un aspecto importante de los esfuerzos de desarrollo de software ágil, al igual que los esfuerzos tradicionales, y es una parte crítica del escalado de enfoques ágiles para satisfacer las necesidades del mundo real de las organizaciones modernas. Sin embargo, los agilistas se acercan a la arquitectura de una forma diferente a como lo hacen los enfoques tradicionales y fundamentalmente, lo realizan en conjunto con los stakeholders.

La arquitectura proporciona la base a partir de la cual se construyen los sistemas y un modelo arquitectónico define la visión en la que se basa su arquitectura. El ámbito de la arquitectura puede ser el de una sola aplicación, de una familia de aplicaciones, de una organización o de una infraestructura como Internet compartida por muchas organizaciones. Independientemente del ámbito, la experiencia de Scott Ambler sugiere que se puede adoptar un enfoque ágil para el modelado, desarrollo y evolución de una arquitectura.

Para resumir la práctica del enfoque ágil a la hora de determinar la arquitectura del sistema, éste **se centra en la participación activa de los stakeholders**, la cual es fundamental para el éxito cuando se trata de identificar los requisitos arquitectónicos puesto que los requisitos provienen de ellos, no de los desarrolladores.

Para entender la labor de cada rol en el Modelado de la solución, a continuación, se muestra una tabla con la descripción de cada uno de ellos.

Rol	Descripción
Modelador ágil	Rol que crea y desarrolla modelos, ya sean bocetos, tarjetas de índice o modelos complejos en herramientas CASE, de una manera evolutiva y colaborativa.
Stakeholders	Es cualquier persona que es un usuario directo, usuario indirecto, gerente de usuarios, gerente sénior, miembro del personal de operaciones, miembro del personal de soporte (help desk), desarrolladores que trabajan en otros sistemas que integran o interactúan con el que está en desarrollo, o profesionales de mantenimiento potencialmente afectados por el desarrollo y/o implementación de un proyecto de software.
Desarrollador	Un desarrollador escribe código fuente, prueba y construye software.

Figura 8: Tabla con Roles AUP del Modelado de la Solución
Elaboración propia

1.2.3. La importancia del prototipado

La creación de prototipos de interfaz de usuario, como bien lo precisa (Ambler, 2006), es una técnica de análisis iterativa, que debe ser considerado desde la fase de elaboración de AUP, en la que los usuarios deben participar activamente en la simulación de la interfaz de usuario para un sistema. Los prototipos de interfaz de usuario tienen varios propósitos:

- Como un artefacto de requisitos para visualizar inicialmente el sistema.
- Como un artefacto de análisis que le permite explorar el ámbito del problema con los stakeholders.
- Como un artefacto de diseño que le permite explorar el ámbito de solución del sistema.
- Un vehículo para que pueda comunicar los posibles diseños de interfaz de usuario del sistema.

Como se puede ver en el diagrama de actividades que se muestra en la figura 9, hay cuatro pasos de alto nivel en el proceso de creación de prototipos de la interfaz de usuario.

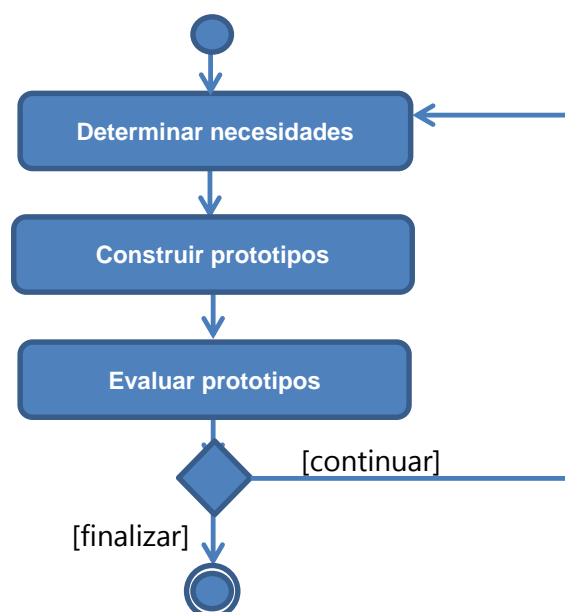


Figura 9: Proceso de creación de prototipos de interfaz de usuario
Elaboración propia

Paso1. Determinar necesidades

El primer paso es analizar las necesidades de interfaz de usuario de los diferentes usuarios. El modelado de la interfaz de usuario pasa de la definición de requisitos al análisis en el momento en que decide evolucionar todo o parte del prototipo esencial de la interfaz de usuario en un prototipo de interfaz de usuario tradicional. Esto implica que los dibujos a mano lo convierten en algo un poco más sustancial. Este proceso se inicia mediante la toma de decisiones de plataforma, que en efecto es una decisión arquitectónica. Por ejemplo, ¿tiene la intención de implementar el sistema para que se ejecute en un explorador de Internet, como una aplicación con una interfaz gráfica de usuario basada en Windows, como una aplicación Java multiplataforma o como un conjunto de pantallas basado en mainframe? Diferentes plataformas conducen a diferentes herramientas de creación de prototipos.

Mientras se determina las necesidades de los stakeholders, se puede decidir transformar los prototipos de interfaz de usuario esenciales, en el caso que estos se hayan empezado a crear con bocetos. La figura 10 representa una interfaz de usuario esencial y la figura 11 un boceto de dos pantallas potenciales o páginas HTML basadas en ese prototipo.

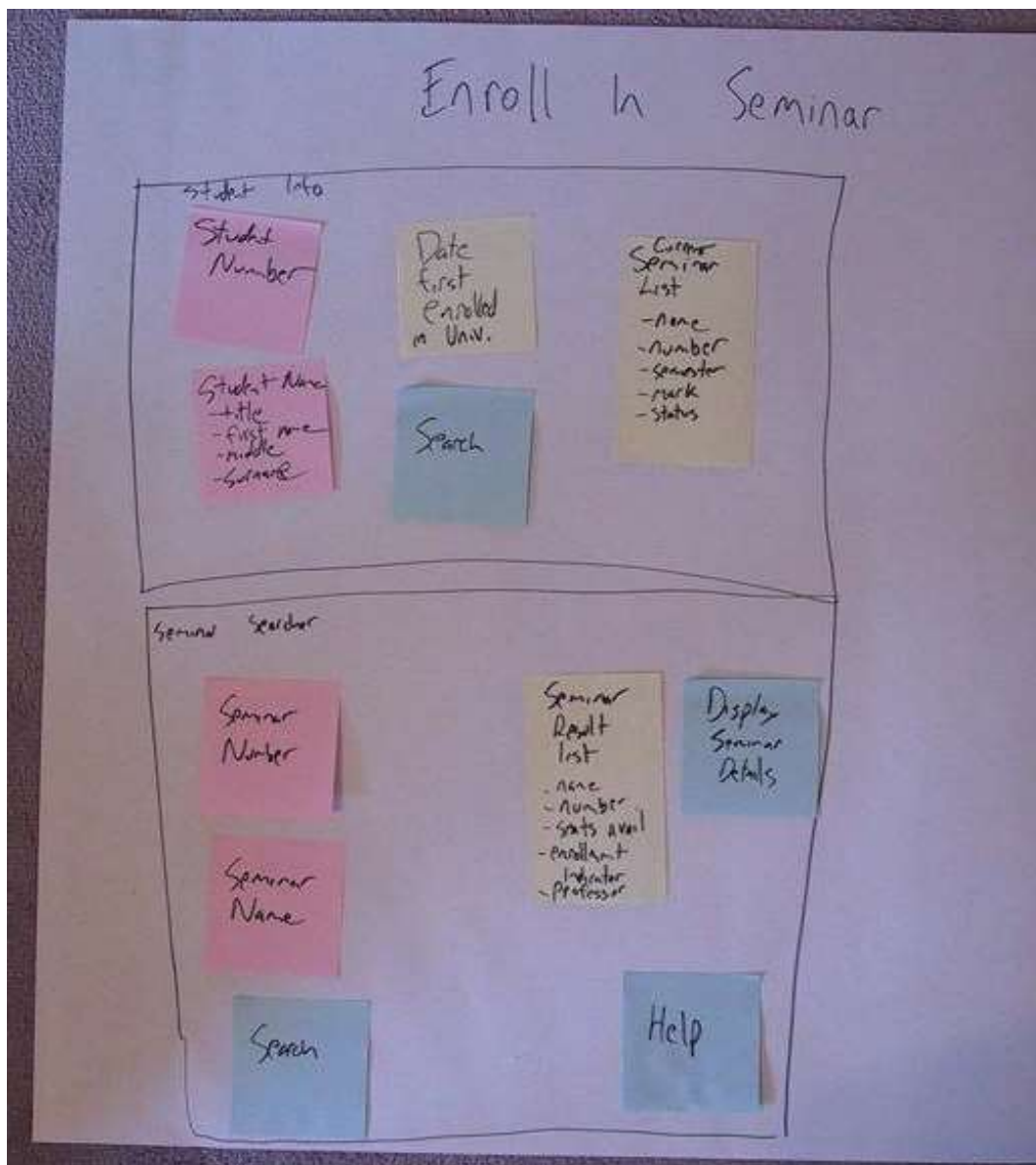


Figura 10: Interfaz de usuario esencial

Fuente. - Tomado de <http://www.agilemodeling.com/artifacts/uiPrototype.htm>

En el ejemplo de la figura 11, el modelador ágil eligió dividir el prototipo en dos para cuestiones de cohesión: se prefirió pantallas que cumplan un solo propósito, en este caso capturando información básica del estudiante e inscribiendo a un estudiante en seminarios, respectivamente. Podría decirse que este es un problema de diseño (hay una delgada línea entre el análisis y el diseño que cruzará todo el tiempo). Los bocetos proporcionan un nivel final de detalle que los prototipos de papel, es mucho más fácil tener una idea de cómo se implementará la pantalla que de la Figura 10; sin embargo, el boceto es importante cuando ya se tiene una interfaz más madura.

Student Information

Student Number: 789-567-234

First Name: Scott

Middle: William

Surname: Ambler

Salutation: Mr.

Date first Enrolled: June 14 2003

Seminars:

Seminar	Term	Grade	Status
CSC 100 Intro to CS	Fall 2003	A+	Passed
CSC 200 Intro to AI	Fall 2003	A	Passed
CSC 203 Advanced AI	Spring 2004	-	Enrolled

Buttons: Add, Drop, Transcript, Close

Add a seminar

Seminar Number: CSC #

Name: Agile X

Search

Results

Seminar	Term	Sects/Avail	Professor
CSC 250 Agile Techniques	Fall 2004	4	Smith, J.
CSC 300 Agile ERP	Spring 2005	17	Jones, S.
CSC 310 Agile Database techniques	Spring 2004	0	Johnson, M.

Course description:

CSC 310 Agile Database Techniques

This course describes evolutionary development strategies for data oriented development. See www.agiledb.org for details.

This course currently has 39 people waitlisted for it.

Buttons: Add, Close

Figura 11: Bocetos de pantalla

Fuente. - Tomado de <http://www.agilemodeling.com/artifacts/uiPrototype.htm>

A medida que se itera a través de la creación de prototipos de interfaz de usuario, a menudo se descubrirá información mejor capturada en otros artefactos. El desarrollo ágil de software es un proceso evolutivo, por lo que esto es normal crear varios modelos en paralelo.

Paso2. Construir prototipos

Una vez que se comprende las necesidades de interfaz de usuario de los stakeholders o partes interesadas en español, el siguiente paso es crear realmente un prototipo. Con una herramienta de creación de prototipos o un lenguaje de alto nivel, se desarrolla las pantallas, las páginas y los informes que necesitan los usuarios. Es posible que se desee crear bocetos como los que se ven en la figura 11 o ir directamente a una implementación concreta, como la página HTML que se muestra en la figura 12. Los bocetos son más inclusivos, los stakeholders pueden participar

activamente en su creación, aunque la página HTML real está mucho más cerca del código de trabajo, el cual es el objetivo principal del equipo de desarrollo.

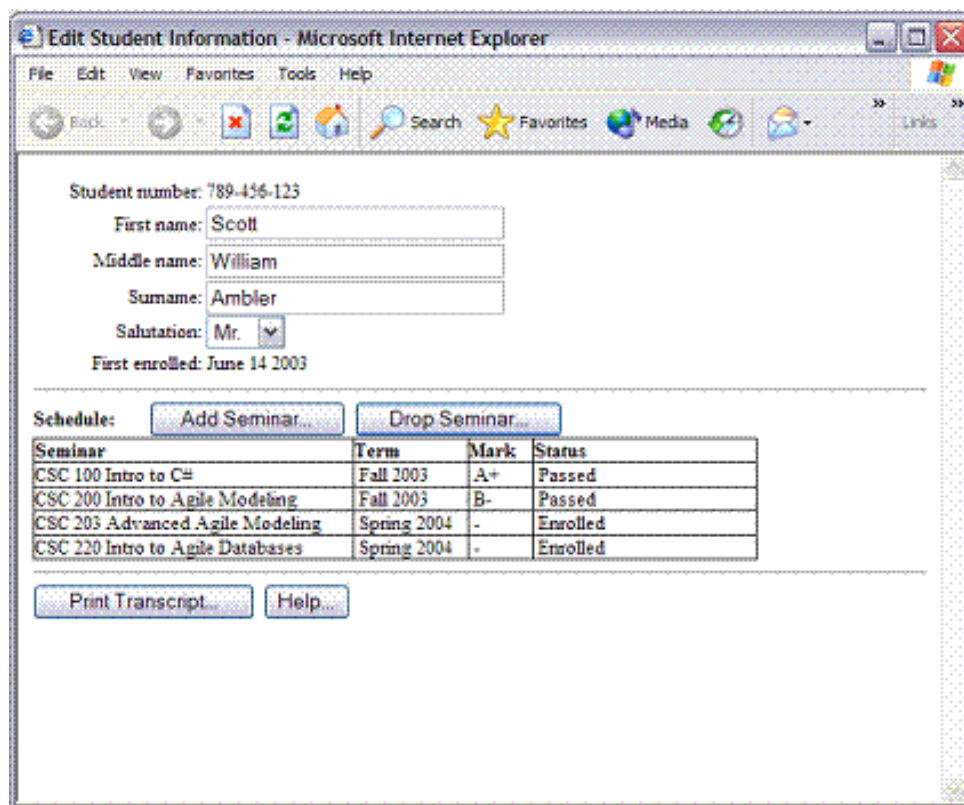


Figura 12: Prototipo concreto de interfaz de usuario

Fuente. - Tomado de <http://www.agilemodeling.com/artifacts/uiPrototype.htm>

Es fundamental comprender que no es necesario crear un prototipo para todo el sistema. Es muy común crear prototipos de una pequeña parte de la interfaz de usuario, quizás una sola pantalla o página HTML, antes de pasar a implementarla. Recordemos que los desarrolladores ágiles trabajan de una manera evolutiva, es decir, no necesitan definir todo por adelantado antes de continuar. A veces se necesitará crear un prototipo de una gran parte del sistema, tal vez como parte de un ejercicio de previsión o tal vez incluso para ayudar a definir el alcance del sistema para que pueda obtener financiamiento del proyecto.

Paso 3. Evaluar prototipos

Una vez creada una versión del prototipo de interfaz de usuario, debe ser evaluada por los stakeholders para comprobar que satisface sus necesidades. A veces esto es tan fácil como pedirle a alguien que dedique unos minutos a ver lo que ha construido y otras veces es tan complicado como programar una reunión para que pueda demostrar el software a un grupo de personas. Es preferible el primer enfoque.

Al evaluar un prototipo de interfaz de usuario, el enfoque ágil sugiere realizar las siguientes preguntas puesto que proporcionan comentarios significativos de los stakeholders:

- ¿Qué tiene de bueno el prototipo de interfaz de usuario?
- ¿Qué tiene de malo el prototipo de interfaz de usuario?
- ¿Qué falta en el prototipo de interfaz de usuario?

Paso 4. ¿Continuar o finalizar?

Después de evaluar el prototipo, es posible que se necesite desechar partes de este, modificar partes e incluso agregar piezas nuevas. Se sugiere detener el proceso de creación de prototipos de interfaz de usuario cuando se encuentre que el proceso de evaluación ya no genera nuevas ideas o está generando un pequeño número de ideas no tan importantes. De lo contrario, vuelva a explorar las necesidades de la interfaz de usuario de los stakeholders.

ACTIVIDADES PROPUESTAS

1. Marque con un aspa (X) cuál es el objetivo principal del modelado de la solución según AUP:
 - a. Crear prototipos de cada funcionalidad o caso de uso
 - b. Modelar la arquitectura del sistema
 - c. Construir código fuente del sistema
2. Mencione una descripción breve de los roles del modelado de la solución.
3. Describa brevemente los 4 pasos del proceso de creación de prototipos de interfaz de usuario según Scott Ambler:

Pasos	Descripción

Figura 13: Descripción del Proceso de Creación de prototipos
Elaboración propia

Resumen

1. El modelado de la solución consiste en identificar una solución viable para abordar el dominio del problema. Esta solución viable se enfoca en la definición de una arquitectura a nivel hardware y software del sistema a desarrollar.
2. El proceso de creación de prototipos de interfaz de usuario que sugiere el enfoque ágil presenta 4 pasos: determinar las necesidades de los stakeholders, construir prototipos, evaluar prototipos y decidir si se continúa o finaliza.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>
- <http://www.agilemodeling.com/essays/agileArchitecture.htm>
- <http://www.agilemodeling.com/artifacts/uiPrototype.htm>

1.3. CREACIÓN DE PERFILES Y ESTEREOTIPOS PARA EL ANÁLISIS Y DISEÑO DE SISTEMAS

Un perfil en el ámbito de modelado de sistemas, específicamente cuando se requiere comunicar una vista del sistema a los stakeholders del proyecto, es un mecanismo de extensión para atender particularidades de modelado según la necesidad del equipo de desarrollo. Un perfil contiene un conjunto de estereotipos que va acompañado de una notación gráfica y/o texto que describe su uso de aplicación. Un estereotipo debe entenderse entonces, como una etiqueta que aplicada a un elemento o relación de un diagrama tiene un significado adicional.

La especificación de un estereotipo puede definirse en una tabla que contenga los siguientes datos:

Nombre	
Aplica a	
Significado	

Figura 14: Especificación del estereotipo
Elaboración propia

Por ejemplo:

Nombre	Include
Aplica a	Dependencias entre casos de uso
Significado	El caso de uso base refiere al caso de uso incluido como parte de su flujo de eventos.

Figura 15: Especificación del estereotipo *include*
Fuente. - Tomado de <https://synergix.wordpress.com/2008/07/20/estereotipos-en-uml/>

Algunos perfiles están definidos en el estándar UML como parte de un enfoque de desarrollo. Por ejemplo, el Modelado de Negocio (o business modeling en inglés) de RUP (Proceso Unificado de Rational) es un perfil utilizado para representar la vista externa e interna del negocio u organización en estudio. Este perfil presenta varios estereotipos para representar cada aspecto relevante del negocio, tal como se muestra en la siguiente tabla:

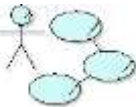

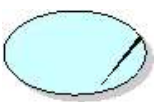
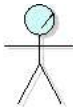
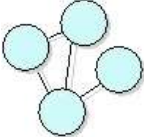

Perfil	Estereotipos	
	Modelos	Elementos
Modelado de Negocio (Business Modeling)	 Modelo de Casos de Uso de Negocio	   Objetivo del Negocio Caso de uso de Negocio Actor de Negocio
	 Modelo de Análisis de Negocio	 Trabajador de Negocio

Figura 16: Tabla de estereotipos del perfil Modelado de Negocio
Elaboración propia

1.3.1. Perfiles y estereotipos en un entorno CASE

La creación de perfiles y estereotipos para un propósito dependerá de lo que requiera el equipo de desarrollo y el procedimiento de configuraciones en un entorno CASE dependerá de la misma herramienta.

Desde la aparición del UML como un lenguaje estándar de modelado de sistemas, los entornos CASE la incorporan para la creación de modelos básicos. Además, algunas herramientas CASE incluyen un conjunto de perfiles y otras permiten incorporar perfiles que el equipo de desarrollo crea con diversos propósitos.

Ejemplos:

El Rational Rose incluye perfiles de modelado de negocio y de diseño para aplicaciones Java Web. Por otro lado, el Rational Software Architect tiene incluido el perfil de Análisis RUP, pero no el de modelado de negocio ni el de diseño. Sin embargo, permite incorporar estos perfiles. Finalmente, otro ejemplo de CASE es el Rhapsody que no incluye ninguno de los perfiles mencionados, pero permite importar los perfiles que el equipo de desarrollo requiera; para ello, se configura un archivo con extensión “.sbsx” que contiene los estereotipos de un perfil específico.

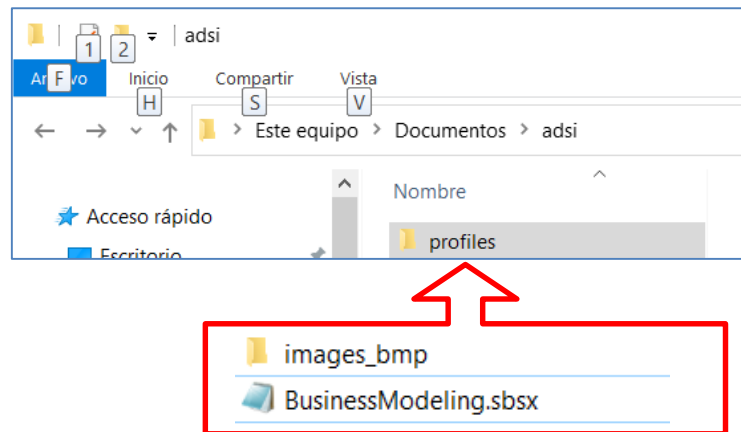


Figura 17: Carpeta que contiene perfil del Business Modeling con extensión *.sbsx
Elaboración propia

```
<RHAPSODY-MODEL>
  <IProfile type="e">
    ...
    <AggregatesList type="e">
      <value> </value>
    </AggregatesList>
  </IProfile>
  <IStereotype type="e">
    ....
  </IStereotype>
</RHAPSODY-MODEL>
```

Figura 18: Estructura básica de configuración de perfil en archivo *.sbsx
Elaboración propia

1.3.2. Definiendo perfiles para el análisis de sistemas

En este apartado se muestra a modo de resumen el perfil de Análisis RUP, el cual presenta tres estereotipos tomando como base el elemento “clase” de un Diagrama de clases: boundary, control y entity. Estas clases de análisis del enfoque RUP se pueden aplicar en un proyecto ágil para describir los elementos que deben implementarse para cada funcionalidad del sistema, sin atender detalles de programación. Específicamente, estos elementos siguen un esquema de 3 capas o layers de un patrón arquitectónico conocido como MVC (Modelo Vista Controlador): una o más clases del tipo entidad (*entity*) que modela el dato a ser manipulado, una o más clases de interfaz que modelan la vista o interfaces gráficas de usuario, y una clase control que representa la lógica interna de la funcionalidad.


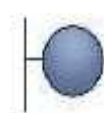

Estereotipos del perfil “Análisis RUP”		Capas del MVC
Notación gráfica	Nombre	
	Entity Clase Entidad	Model Modelo
	Boundary Clase de Interfaz	View Vista
	Control Clase Control	Controller Controlador

Figura 19: Tabla de correspondencia de estereotipos del perfil Análisis RUP con capas del patrón arquitectónico MVC
Elaboración propia

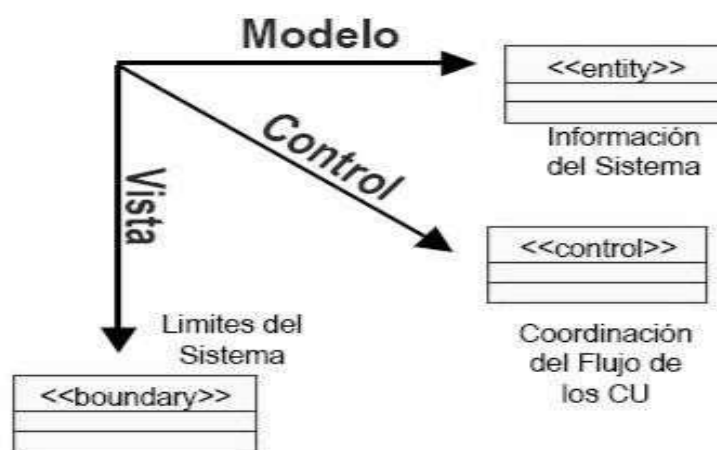


Figura 20: Clases de análisis enfocadas al patrón arquitectónico MVC
Elaboración propia

Nota: Para el curso, el perfil de RUP de análisis se incluirán en el perfil **AnalysisProfile**. Ver carpeta **profiles** compartido en la sesión.

1.3.3. Definiendo perfiles para el diseño de sistemas

En esta sección se muestra a modo de resumen el perfil de diseño J2EE y WebModeler para representar componentes de una aplicación web Java Enterprise Edition (JEE), el cual presenta varios estereotipos tomando como base el elemento “clase” de un Diagrama de clases: HttpServlet, ServerPage, ClientPage y Form. Estas clases estereotipadas permiten representar los componentes que se utilizan para implementar una aplicación Java Web. Sin embargo, si el equipo de desarrollo decide implementar el sistema con otro lenguaje de programación podrá optar por utilizar otros estereotipos.

Además es importante resaltar que los estereotipos de diseño utilizados deben seguir un patrón de diseño, tema que será detallado más adelante. Sin embargo, a modo de tener una idea general sobre cada estereotipo que se aplicará en nuestros proyectos, podemos categorizarlos a partir del mismo esquema de 3 capas o layers de un patrón arquitectónico conocido como MVC (Modelo Vista Controlador) visto en la sección anterior. A continuación, se muestran las tablas con la correspondencia entre estos elementos (los de los perfiles y capas del patrón arquitectónico).


Estereotipo del perfil “JavaEE”		Capas del MVC
Notación gráfica	Nombre	
	HttpServlet	Controller Controlador

Figura 21: Tabla de correspondencia de estereotipos del perfil JavaEE con capas del patrón arquitectónico MVC

[Elaboración propia](#)




Estereotipos del perfil “WebModeler”		Capas del MVC
Notación gráfica	Nombre	
	ServerPage	View Vista
	ClientPage	
	HTMLForm	

Figura 22: Tabla de correspondencia de estereotipos del perfil WebModeler con capas del patrón arquitectónico MVC

[Elaboración propia](#)

Nota: Para el curso, con fines didácticos, ambos perfiles (JavaEE y WebModeler) se incluirán en el perfil **DesignProfile**. Ver carpeta **profiles** compartido en la sesión.

1.3.4. Agregando perfiles en un proyecto

La creación de perfiles en una herramienta CASE dependerá de la configuración que se requiere realizar en el entorno. Por ejemplo para IBM Rhapsody se realiza el siguiente procedimiento para crear perfiles:

1. Se crea un paquete desde un proyecto activo.
2. Se define como **profile** al paquete creado.
3. Desde el perfil, se agregan los estereotipos (**stereotypes** en inglés).
4. Se definen las características de cada estereotipo: imagen asociada y artefacto aplicable, como mínimo.
5. Finalmente, desde el explorador de Windows podrá observar que en el proyecto se ha creado un archivo con extensión *.sbsx, el cual contiene la configuración del perfil creado.

Luego de crear los perfiles y sus estereotipos, estos se aplican en los proyectos creados.

A continuación, realizaremos una práctica sobre cómo aplicar perfiles personalizados:

Paso 1: A modo de ejemplo, aplicaremos el perfil del Business Modeling, utilizado en el curso de ADSI, para trabajarlo en el nuevo proyecto a crear. Para ello, desde el explorador de windows, agregue la carpeta **profiles** al directorio del workspace del Rhapsody. La carpeta **profiles** contiene el **perfil BusinessModelingProfile**.

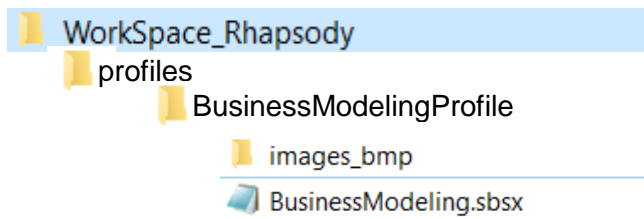


Figura 23: Perfil del Business Modeling con extensión *.sbsx
Elaboración propia

Paso 2: A continuación, se crea el proyecto “ProyectoADSII”.

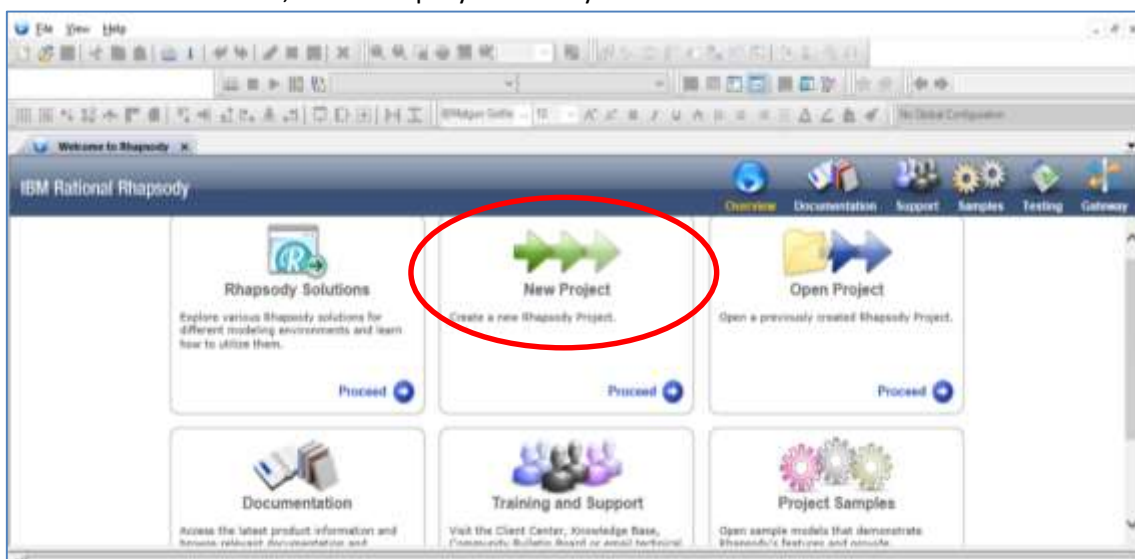


Figura 24: Creación de proyecto desde página de Bienvenida
Elaboración propia

Paso 3. Ahora, seleccione el proyecto del browser de proyectos; luego, seleccione menú “File” y opción “Add Profile to Model...”. Finalmente, se ubica el archivo del perfil con extensión *.sbsx.

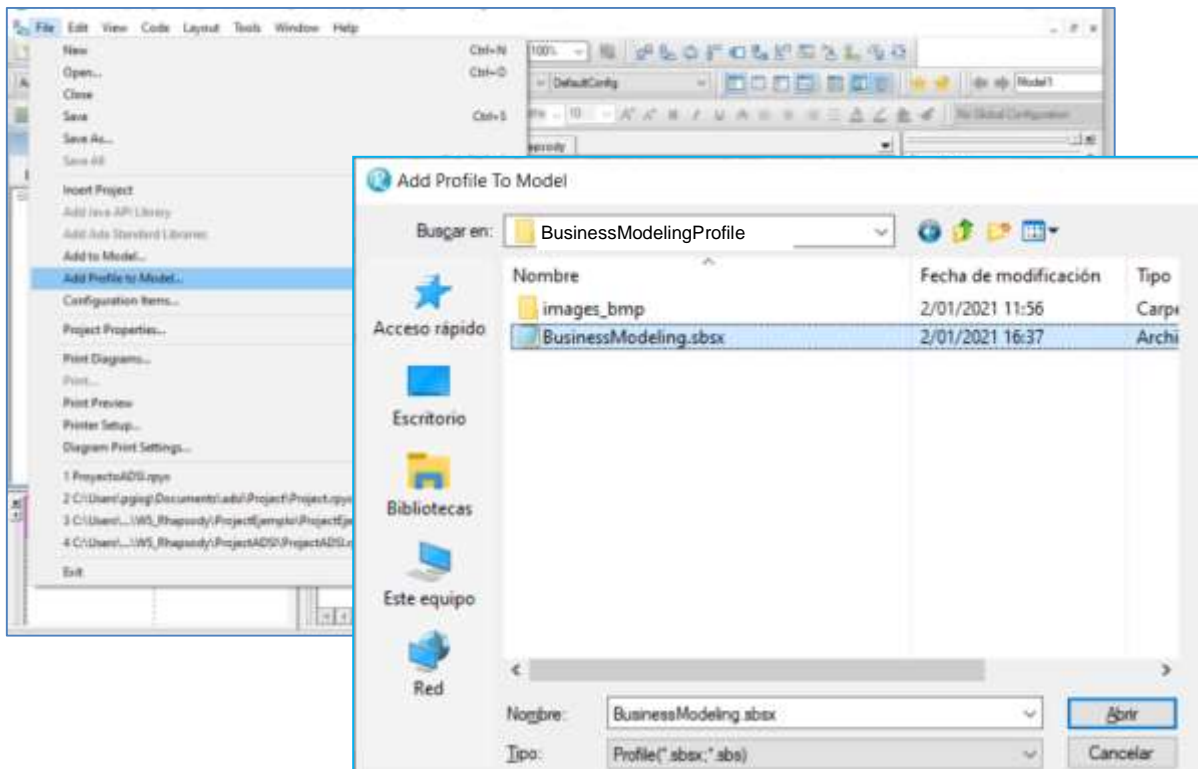


Figura 25: Agregar perfil con extensión *.sbsx a proyecto
Elaboración propia

Paso 3: A continuación, se visualizará el perfil cargado en la carpeta **Profiles** del proyecto.

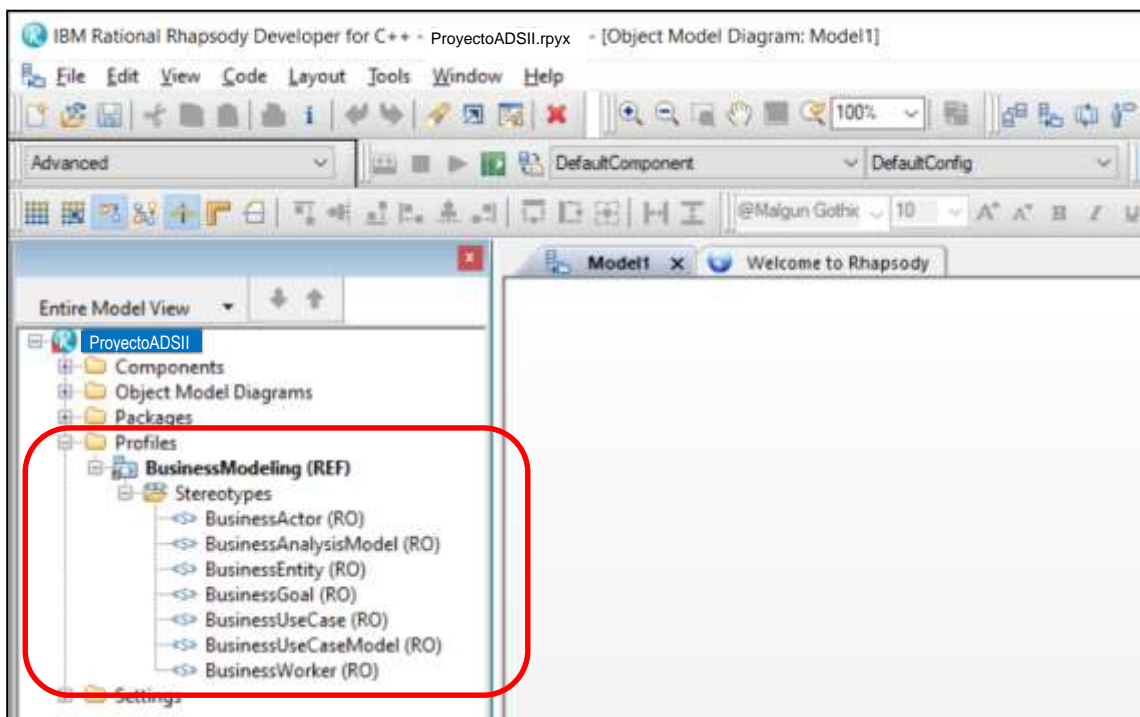


Figura 26: Proyecto con carpeta Profiles
Elaboración propia

ACTIVIDADES PROPUESTAS

1. Aplique los perfiles de análisis y diseño al proyecto “ProyectoADSII”. Previamente, desde el explorador de Windows, verifique que las carpetas **AnalysisProfile** y **DesignProfile** se incluyen en la carpeta **profiles** ubicado en el ProyectoADSII.
2. Marque con un aspa (X) cuáles son estereotipos de clases de análisis:
 - a. Modelo, vista y controlador
 - b. HttpServlet, ServerPage y ClientPage
 - c. entity, control y boundary
 - d. HttpServlet, ServerPage, ClientPage y HTMLForm
3. Mencione una descripción breve de las capas del patrón arquitectónico MVC.
4. Indique el nombre de cada estereotipo:







Imagen de estereotipo	Nombre
	
	
	
	
	
	

Figura 27: Descripción del Proceso de Creación de prototipos
Elaboración propia

Resumen

1. El perfil de RUP Análisis presenta tres estereotipos de clases: entity, control y boundary que modela el dato que se manipula en la funcionalidad, lógica de la aplicación e interfaz, respectivamente.
2. El perfil de Java EE presenta un estereotipo de clase: HttpServlet que representa un servlet donde se implementa la lógica de la aplicación.
3. El perfil de WebModeler presenta tres estereotipos de clases: JavaServerPage, ClientPage y Form que modelan las clases de diseño. El JavaServerPage representa el componente que se crea en el lado del servidor y los estereotipos ClientPage y Form representan los componentes que se ejecutan o despliegan por el lado del cliente.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>
- <http://www.agilemodeling.com/essays/agileArchitecture.htm>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=profiles-creating-stereotypes>



ANÁLISIS DE SISTEMAS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno elabora la documentación del Análisis funcional de un caso de estudio utilizando una Herramienta CASE.

TEMARIO

2.1 Tema 4 : Análisis de Sistemas

- 2.1.1 : Análisis de sistemas según AUP
- 2.1.2 : Artefactos
- 2.1.3 : Reglas básicas de nomenclaturas
- 2.1.4 : Actividades para identificar los artefactos del modelo de análisis de sistema

2.2 Tema 5 : Desarrollo del Modelo de Análisis Funcional de sistemas para un caso

- 2.2.1 : Especificaciones de Casos de Uso del Caso Fashion Importaciones
- 2.2.2 : Configuración de perfiles para el Modelo de Análisis de Sistemas - Caso: Venta de Productos de Fashion Importaciones

2.3 Tema 6 : Análisis de la Arquitectura

- 2.3.1 : Análisis de la Arquitectura de sistemas
- 2.3.2 : Análisis de la Arquitectura para el caso Fashion Importaciones

2.4 Tema 7 : Análisis de Casos de Uso

- 2.4.1 : Análisis de Casos de Uso para un dato maestro y dato transaccional
- 2.4.2 : Análisis de Casos de Uso para el caso Fashion Importaciones

2.1. ANÁLISIS DE SISTEMAS

En este apartado se describirá cómo se desarrolla el análisis de sistemas según AUP. Además, se mencionarán los artefactos relevantes que se representan así como las reglas básicas de nomenclaturas. Finalmente, se describirá las actividades para identificar los artefactos del modelo de análisis de sistema.

2.1.1. Análisis de sistemas según AUP

Debido a que AUP toma como base el enfoque de RUP, Scott Ambler sugiere que no debe permitirse que el equipo realice todos los artefactos que se definen en la disciplina de análisis de RUP, sino solo los que realmente brinden apoyo durante la tarea de análisis para entender el dominio del problema. En este sentido, el objetivo en esta etapa es identificar una solución viable a nivel arquitectónico para atender el dominio del problema, sin necesidad de representar detalles de implementación de alguna tecnología puesto que éste será desarrollado en la etapa de diseño.

2.1.2. Artefactos

En la siguiente tabla se muestra la lista de artefactos del Modelo de Análisis con una breve descripción. Estos artefactos serán los que se desarrollarán en el curso.

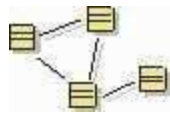




Artefacto	Descripción
 Modelo de Análisis	Representa la vista interna del sistema. Define un modelo de objetos que describe la vista interna de casos de uso, y que sirve como una abstracción del modelo de diseño.
 Paquete de Análisis	Los paquetes del análisis proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables. Un paquete de análisis contiene clases y diagramas de interacción de las funcionalidades del sistema a nivel de análisis.
 Clase de Interfaz	Es una clase utilizada para modelar la interacción entre el entorno del sistema y su funcionamiento interno. Modela las partes del sistema que dependen de su entorno.
 Clase Control	Representa la lógica de negocio de la aplicación, es decir, el control, la coordinación y la secuencia entre objetos. Encapsula el comportamiento de uno o más casos de uso.
 Clase Entidad	La entidad es una clase utilizada para modelar la información y comportamiento asociado que deben ser almacenados. Modela las partes del sistema que son independientes de su entorno.

Figura 28: Tabla de artefactos de Análisis – Parte I

Fuente. – Elaboración propia

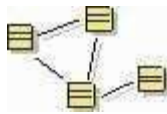
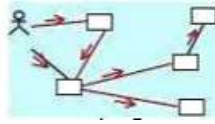
Artefacto	Descripción
 <p>Diagrama de Clases</p>	El diagrama de clases describe la estructura de un caso de uso o funcionalidad.
 <p>Diagrama de Comunicación</p>	Diagrama de interacción que describe el comportamiento del caso de uso centrado en la responsabilidades y colaboraciones entre los objetos.

Figura 29: Tabla de artefactos de Análisis – Parte II

Fuente. – Elaboración propia

2.1.3. Reglas básicas de nomenclaturas

La siguiente tabla muestra los artefactos del modelo de análisis que requiere atender ciertas reglas en su nomenclatura:





Artefacto	Descripción
 <p>Paquete de Análisis</p>	Los paquetes del análisis podrían seguir la misma nomenclatura de un paquete de diseño, es decir, nombrarlos en minúscula y a partir de la segunda palabra, en caso de tener un nombre compuesto, empieza con mayúscula. El nombre debe ser un sustantivo que representa los datos a manipular por los casos de uso que contiene el paquete o la transacción que se manipula en sus casos de uso.
 <p>Clase de interfaz</p>	El nombre de este artefacto empezará con el prefijo CI, a continuación se agregará el nombre de la interfaz gráfica de usuario o del sistema externo que representa.
 <p>Clase Control</p>	El nombre de este artefacto empezará con el prefijo CC, a continuación se agregará el nombre del paquete de análisis donde se ubique. Se recomienda crear una clase control por cada paquete de análisis.
 <p>Clase Entidad</p>	El nombre de este artefacto será un sustantivo que denote el dato a manipular: maestro o transaccional.

Figura 30: Tabla de nomenclaturas de artefactos de Análisis

Fuente. – Elaboración propia

2.1.4. Actividades para identificar los artefactos del modelo de análisis

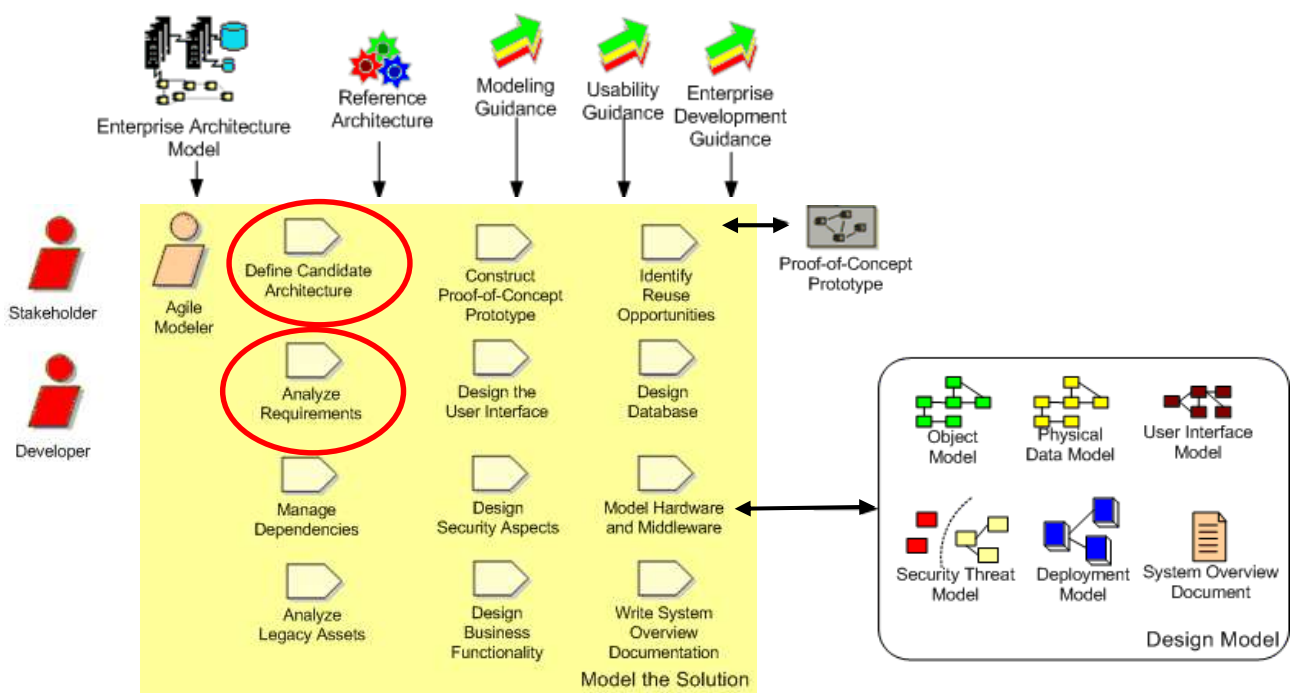
El modelo de análisis es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del modelo de diseño.

Durante el análisis, se identifica, de manera continua, nuevos paquetes del análisis, clases y requisitos comunes a medida que el modelo de análisis evoluciona, y los paquetes de análisis concretos continuamente se refinan y mantienen.

Este modelo de análisis no es un diagrama final que describe todos los posibles conceptos y sus relaciones, es un primer intento por definir los conceptos claves que describen el sistema. Este artefacto es opcional, pero también tiene a su vez la propiedad de ser temporal en el caso en que se planea su desarrollo. Su utilidad radica en que permite una apreciación global conceptual del sistema.

La siguiente figura resalta las actividades que se desarrollarán en el curso para elaborar el modelo de análisis:

- **Definir una arquitectura candidata** del sistema donde se hace un Análisis de la Arquitectura.
- **Analizar los requisitos**, específicamente los funcionales, donde se realizará un Análisis de Casos de Uso



Copyright 2005 Scott W. Ambler

Figura 31: Modelado de la solución – Análisis según AUP

Fuente. - Tomado de <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>

Resumen

1. El análisis se traduce en el modelo de análisis, el cual es usado para representar la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del modelo de diseño.
2. El modelo de análisis no es un diagrama final que describe todos los posibles conceptos y sus relaciones, es un primer intento por definir los conceptos claves que describen el sistema. Este artefacto es opcional, pero también tiene a su vez la propiedad de ser temporal en el caso en que se planea su desarrollo. Su utilidad radica en que permite una apreciación global conceptual del sistema.
3. Los artefactos claves para describir la funcionalidad o caso de uso del sistema son los siguientes: clase de interfaz o **boundary**, clase **control** y clase de entidad o **entity**.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>

2.2. DESARROLLO DEL MODELO DE ANÁLISIS FUNCIONAL DE SISTEMAS PARA UN CASO

En esta sección se presentará 2 especificaciones de casos de uso para el caso Fashion importaciones. Luego, se importará el proyecto que contiene el modelado de negocio y el modelo de casos de uso del caso de estudio. Finalmente, se configurará el perfil de análisis y se creará el paquete que incluirá el modelo de análisis.

2.2.1. Especificaciones de Casos de Uso del Caso Fashion Importaciones

En primer lugar, se describe el caso de estudio.

CASO: FASHION IMPORTACIONES

Usted ha sido contratado para trabajar en el proyecto: Sistema de Venta de productos importados para mujeres emprendedoras. Esta compañía tiene como principal propósito captar y entrenar a emprendedoras para la venta de carteras, joyas y accesorios, para lo cual diseñan catálogos de ventas de forma digital para facilitar la venta a través de ellas. Su misión es ayudar a la mujer peruana a poder generar ingresos extras y alcanzar la independencia económica, apoyándolas y entrenándolas en las herramientas digitales, haciendo de su pasión por las carteras, joyas y accesorios su mejor negocio. Actualmente en el Perú, Fashion Importaciones tiene 3 sucursales con almacenes: Lima, Cajamarca y Chiclayo.

Con el fin de agilizar sus ventas y atención de pedidos, así como incrementar el número de emprendedoras, el Gerente de la empresa requiere un sistema que permita a sus emprendedoras registrar, pagar y hacer seguimiento de sus pedidos de los productos correspondientes a un catálogo vigente. Asimismo, requieren definir una ruta óptima de entregas de estos pedidos a fin de minimizar los costos de combustible de sus móviles. A continuación, se describen los procesos.

El proceso de afiliación de una emprendedora inicia cuando una postulante se contacta con la empresa para acceder a la capacitación. Una Recepcionista atiende a la postulante y le solicita su nombre completo, DNI, dirección, números de contacto y correo electrónico. A continuación, la recepcionista le brinda las fechas de capacitación presencial o virtual. Si la postulante está de acuerdo con alguna fecha, el Asesor registra su participación en la fecha indicada. Al término de la capacitación, la postulante es evaluada sobre los temas que ha sido entrenada. En caso de aprobar la evaluación, la postulante recibe una constancia de la capacitación y puede acceder al catálogo digital. Asimismo, está lista para realizar sus pedidos.

El proceso de venta de productos inicia cuando una emprendedora se contacta con la empresa para realizar su pedido el cual tiene asignado los datos de la emprendedora y de los productos solicitados. Un Asesor de Ventas toma el pedido y lo registra como "GENERADO" y confirma el monto total para que la emprendedora realice el pago a una cuenta bancaria. Una vez, realizado el pago del pedido, la emprendedora envía la constancia de pago al Asesor de ventas, quien cambia de estado del pedido a "PAGADO". A continuación, genera el Comprobante de Pago (CDP) y lo envía a la emprendedora vía correo electrónico.

La atención de pedidos inicia cuando la Supervisora de ventas entrega la lista de pedidos de las emprendedoras al área de despacho. El asistente de despacho elabora un cronograma de salidas de la semana junto con la ruta a seguir para entregar los pedidos. El asistente de despacho entrega al transportista el cronograma de salidas, la hoja de ruta y una guía de remisión de los pedidos; en este momento los pedidos están "EN RUTA". Una vez que el transportista entrega

los productos a la emprendedora, le solicita que firme la copia de la guía de remisión. Por otro lado, existe un proceso de evaluación de pedidos al crédito que permite a la Supervisora de Ventas, evaluar los pedidos de las emprendedoras que tiene a cargo una Asesora de ventas.

Finalmente, el encargado de despacho recibe las copias de las guías de remisión firmadas confirmando que el pedido ha sido “ENTREGADO” y genera un informe de despacho, el cual es enviado a la Gerente General.

Ahora, se describen dos Especificaciones de Casos de Uso:

- Mantener Productos
- Evaluar Pedidos al Crédito

ESPECIFICACIÓN DE CASO DE USO: Mantener Productos

1. Descripción

El caso de uso permite mantener actualizado el registro de los productos importados de la empresa. De acuerdo con su necesidad, el Asistente de almacén puede agregar, actualizar y desactivar un producto.

2. Actor(es)

Asistente de almacén.

3. Flujo de Eventos

3.1. Flujo Básico

1. El caso de uso se inicia cuando el Asistente de almacén selecciona la opción “Productos” en la interfaz del menú principal.
2. El sistema muestra la interfaz “MANTENER PRODUCTO” con la lista de Productos con los campos: código, nombre, categoría, descripción, fecha de registro, fecha de actualización y estado. Además, muestra las opciones: Agregar Producto, Actualizar Producto y Desactivar Producto.
3. Si el Asistente de almacén elige un Producto
 - a. Si elige “Actualizar” ver el Subflujo Actualizar Producto.
 - b. Si elige “Desactivar” ver el Subflujo Desactivar Producto.
4. Si el Asistente de almacén NO elige un Producto
 - a. si elige “Agregar” ver el Subflujo Agregar Producto.
5. El Asistente de almacén selecciona “Salir” y el caso de uso finaliza.

3.2. Subflujos

3.2.1. Agregar Producto

1. El sistema muestra la interfaz PRODUCTO con los siguientes campos: código (solo lectura), nombre, categoría, descripción breve, fecha de registro (sólo lectura) y fecha de actualización (solo lectura). Además, muestra las opciones: Aceptar y Cancelar.
2. El Asistente de almacén ingresa los datos del Producto.
3. El Asistente de almacén selecciona la opción Aceptar.
4. El sistema valida los datos ingresados.
5. El sistema genera un nuevo código de Producto y obtiene la fecha del sistema para la fecha de registro y la fecha de actualización.
6. El sistema graba un nuevo registro de Producto y muestra el MSG “Producto creado con código Nro. 999999”.
7. El Asistente de almacén cierra la interfaz PRODUCTO y regresa a la interfaz MANTENER PRODUCTO con la lista de Productos actualizada y el subflujo finaliza.

3.2.2. Actualizar Producto

1. El sistema muestra los datos del Producto seleccionada en la interfaz PRODUCTO: código (sólo lectura), nombre, categoría, descripción breve, fecha de registro (sólo lectura) y fecha de actualización (solo lectura). Además muestra las opciones: Aceptar y Cancelar.
2. El Asistente de almacén actualiza los datos del Producto.
3. El Asistente de almacén selecciona la opción Aceptar.
4. El sistema valida los datos ingresados del Producto.
5. El sistema obtiene la fecha del sistema para la fecha de actualización, actualiza el registro de Producto y muestra el MSG “Producto actualizado satisfactoriamente”.
6. El Asistente de almacén cierra la interfaz PRODUCTO y regresa a la interfaz MANTENER PRODUCTO con la lista de Productos actualizada y el subflujo finaliza.

3.2.3. Desactivar Producto

1. El sistema muestra el MSG: “¿Está seguro de que desea desactivar el(los) Producto(s) seleccionado(s)?”.
2. El Asistente de almacén selecciona la opción YES para confirmar la desactivación.
3. El sistema actualiza el registro del(los) Producto(s) en estado “Desactivado”.
4. El sistema muestra la interfaz MANTENER PRODUCTO con la lista de Productos actualizada y termina el subflujo.

3.3. Flujos Alternativos

1. Datos del Producto Inválidos

Si los datos ingresados son nulos o inválidos, tanto en los subfujos Agregar como en Actualizar Producto, el sistema muestra el MSG: “Se han encontrado datos inválidos” y los subflujos continúan en el paso 2.

2. Producto ya existe

Si el sistema detecta que el Producto ya existe en el paso 4 del subflujo Agregar Producto, muestra el MSG: “Producto ya existe” y el subflujo finaliza.

3. No confirma Desactivación

Si el Asistente de almacén selecciona NO en el paso 2 del subflujo Desactivar Producto, finaliza el subflujo.

4. Precondiciones

1. El Asistente de almacén está identificado en el sistema.
2. Lista disponible de Productos.

5. Poscondiciones

1. En el sistema quedará registrado el nuevo Producto.
2. En el sistema quedará actualizado el registro del Producto.
3. En el sistema quedará desactivado el Producto.

6. Puntos de Extensión

Ninguno.

7. Requisitos Especiales

Ninguno.

Especificación de Caso de uso: Evaluar pedidos al crédito**1. Breve descripción**

El caso de uso permite a la supervisora de venta evaluar los pedidos, pudiendo aprobar o denegar el pedido de una emprendedora.

2. Actor(es)

Supervisora de venta

3. Flujo de Eventos**3.1. Flujo Básico**

1. El caso de uso comienza cuando el Supervisora de Venta solicita “Evaluar Pedidos al Crédito” en el menú principal.
2. El sistema muestra la interfaz “Evaluar Pedidos” con la lista de todos los pedidos que aún no fueron evaluados. La lista de pedidos contiene los siguientes datos: Nro del pedido, Código y Nombre del Cliente, Vendedor, Fecha y Monto total. Además, posee las opciones Aceptar y Salir.
3. La Supervisora de Ventas selecciona uno de los pedidos a evaluar.
4. La Supervisora de Ventas selecciona Aceptar.
5. El sistema muestra la Interfaz “Pedido al Crédito” donde se muestra detalladamente los datos del Pedido (Número del pedido, fecha del pedido, monto total del pedido y nombre del Asesor de Ventas). Asimismo:
Datos de la emprendedora: código, DNI, nombres, apellidos y línea de crédito;
Lista con el detalle del pedido (Código y nombre del producto, cantidad, precio Total Ítem y Sub total) IGV y Monto Total;
Además, incluye un cuadro de texto para colocar las observaciones y 2 opciones de selección (Aprobar y Denegar) y los botones Grabar y Salir.
6. La Supervisora de Ventas ingresa las observaciones.
7. La Supervisora de Ventas Selecciona Aprobar.
8. La Supervisora de Ventas Selecciona grabar.
9. El sistema actualiza el Pedido como aprobado.
10. El sistema muestra el MSG “Pedido Aprobado” y cierra la interfaz Automáticamente.

3.2. Flujos Alternativos**<Solicitud Denegada>**

Si en el punto 7, el Supervisora de Ventas selecciona Denegar.

1. El sistema actualiza el Pedido como Denegado 2.

4. Requerimientos Especiales

Ninguno

5. Pre Condiciones

1. Supervisora de ventas logeado en el sistema.
2. Lista de Pedidos pendientes.

6. Post Condiciones

1. En el sistema queda actualizada el pedido

7. Puntos de extensión

Ninguno

2.2.2. Configuración de perfiles para el Modelo de Análisis de Sistemas - Caso: Venta de Productos de Fashion Importaciones

Paso 1: Importe el proyecto “ProyFashionImportaciones”:

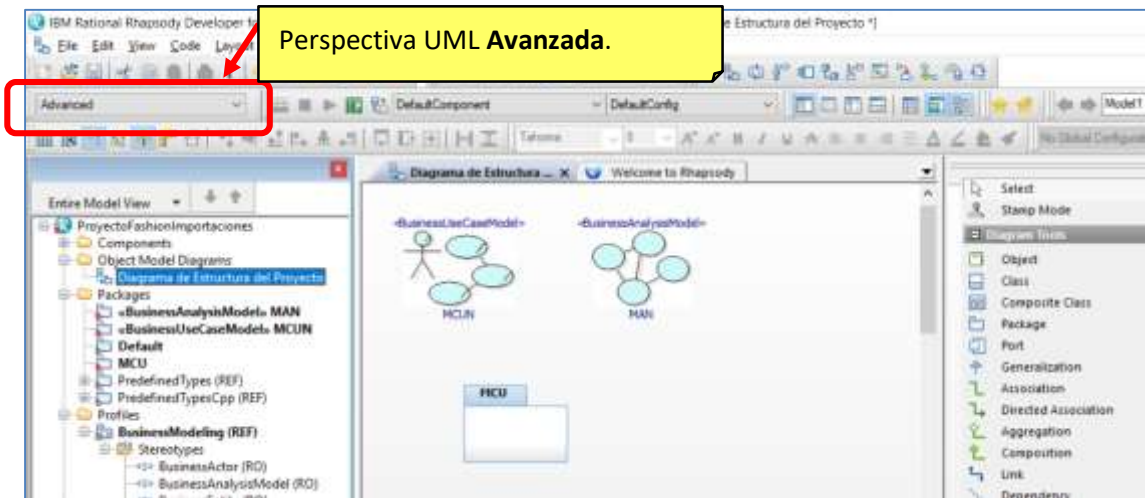


Figura 33: Estructura de Proyecto del caso “Fashion Importaciones”.

Elaboración propia

Paso 2: Siga los pasos de la Unidad de Aprendizaje 1 para agregar el perfil de análisis en el proyecto Fashion Importaciones. A continuación, cree el paquete del modelo de análisis (coloque “MA” como *Name* y “Modelo de Análisis” como *Label* para mostrarlo). Luego, asigne el estereotipo correspondiente “AnalysisModel”.

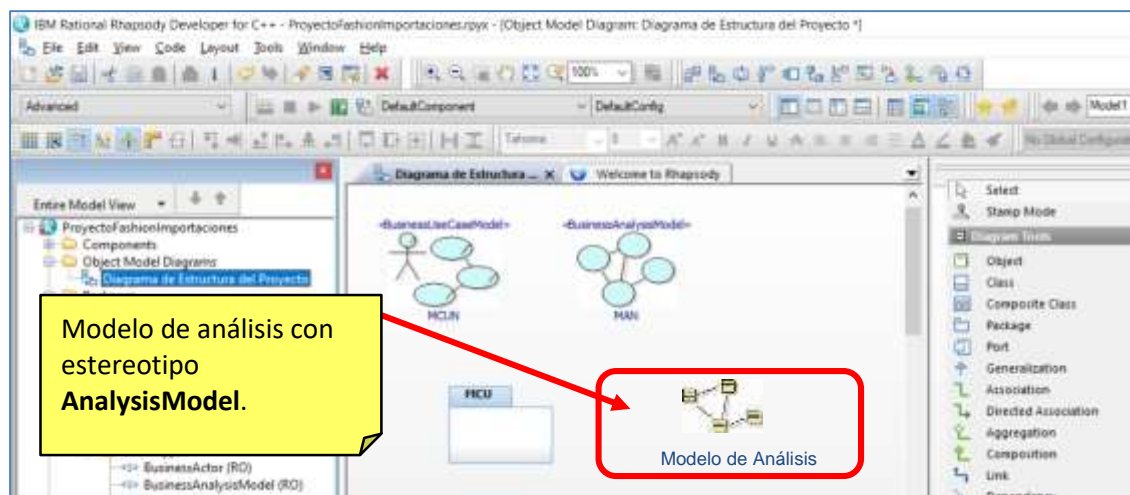


Figura 34: Estructura de Proyecto del caso “Fashion Importaciones” con el modelo de análisis

Elaboración propia

Resumen

1. La herramienta CASE de IBM conocida como IBM Engineering Systems Design Rhapsody presenta características que permite personalizar perfiles para propósitos diferentes de un proyecto.
2. La estructura de un proyecto de sistemas que incluye artefactos hasta el proceso de análisis incluye los siguientes modelos:
 - a. Modelo de Casos de uso de Negocio
 - b. Modelo de Análisis de Negocio
 - c. Modelo de Casos de Uso
 - d. Modelo de Análisis

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- https://www.ibm.com/support/knowledgecenter/SSB2MU_8.4.0/com.ibm.rhp.uml.diagrams.doc/topics/rhp_t_dm_creating_rhap_projects.html
- https://www.ibm.com/support/knowledgecenter/SSB2MU_8.4.0/com.ibm.rhp.harmonyse.doc/topics/rhp_c_modeling_with_harmony_se_profile.html

2.3. ANÁLISIS DE LA ARQUITECTURA

En esta sección se presentará la solución del análisis de la arquitectura de un caso de estudio y a continuación, se solicita realizar el análisis de la arquitectura del caso “Fashion Importaciones”.

2.3.1. Análisis de la Arquitectura de sistemas

El rol responsable de esta tarea es el Arquitecto de software. Esta tarea permite definir una arquitectura candidata basada en la experiencia obtenida de sistemas similares o en dominios del problema similares, y restringir las técnicas arquitectónicas a ser usadas en el sistema. Cabe destacar que analizar la arquitectura resulta beneficioso en el caso donde se desarrollen sistemas que no se hayan hecho antes.

El propósito del análisis de la arquitectura es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes del análisis, clases de análisis de entidad evidentes y requisitos especiales comunes.

Se realizan los pasos siguientes:

- Organización de casos de uso según análisis
- Identificación de los paquetes de análisis
- Definición de las dependencias entre los paquetes de análisis
- Identificación de clases de entidad obvias por cada paquete de análisis

Organización de casos de uso según análisis

Este paso implica la preparación del Diagrama General de Casos de Uso que se encuentra en el Modelo de Casos de Uso, para adaptarla a la Arquitectura de Análisis en donde se identifica los paquetes de análisis distribuidos en dos capas: Específica y General.

Identificación de paquetes de análisis

Los paquetes de análisis constituyen una división del sistema de software que tiene sentido desde el punto de vista de los expertos en el dominio. La descomposición del software en paquetes se establece cuando uno tiene una idea que considera suficientemente fiable de la cantidad de trabajo y del número, y la complejidad de los diagramas, situación a la cual se puede haber llegado tanto al principio de la etapa de análisis como un tiempo después. Una identificación inicial de los paquetes del análisis se hace de manera natural basándonos en los requisitos funcionales y en el dominio del problema, es decir, en la aplicación o negocio que estamos considerando.

Entre las asignaciones adecuadas de casos de uso a un paquete en concreto, se tiene los siguientes criterios:

1. Los casos de uso requeridos para dar soporte a un determinado proceso de negocio.
2. Los casos de uso requeridos para dar soporte a un determinado actor del sistema.

Para identificar los paquetes, se considera lo siguiente:

1. Tener un diagrama de casos de uso con los roles bien definidos.
2. Los casos de uso que estén bajo la responsabilidad de un actor deben tener contenidos estrechamente relacionados.
3. Los casos de uso que están relacionados mediante relaciones de generalización deben pertenecer al mismo paquete.
4. Los casos de uso relacionados mediante relaciones de extensión y que solo se extienden a partir de un caso de uso base deben pertenecer al mismo paquete del caso de uso base.

5. Los casos de uso incluidos tienden a generar su propio paquete la mayor parte de veces. Si los casos de uso base que incluyen al caso de uso son funcionalidades con distintos contenidos, entonces, se debe crear un paquete para el caso de uso incluido.

Definición de dependencias entre paquetes de análisis

El objetivo es conseguir paquetes que sean relativamente independiente y débilmente acoplados, pero que posean una cohesión interna alta. Es inteligente intentar reducir el número de relaciones entre clases en paquetes diferentes, debido a que ello reduce las dependencias entre paquetes.

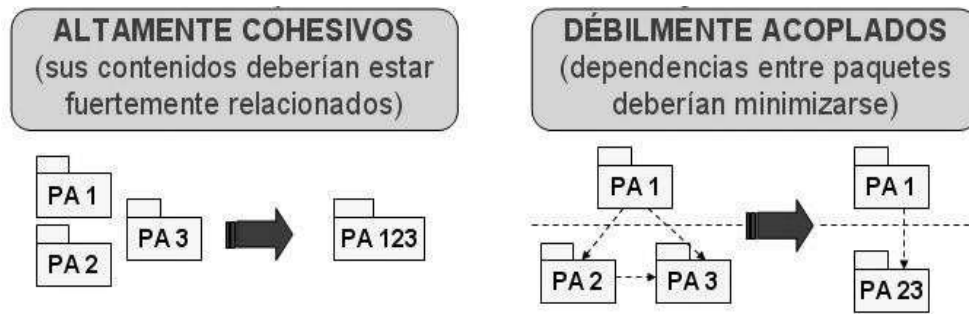


Figura 35: Características de los paquetes de análisis.

[Elaboración propia](#)

Los paquetes identificados se organizarán en la Capa de Aplicación, la cual se subdivide en dos capas internas:

- a) **Capa Específica:** Aquí se ubican los paquetes correspondientes al proceso del negocio (core) de la empresa identificados (Nivel Superior).
- b) **Capa General:** Aquí se ubican los paquetes de maestros de información, paquetes de servicio, seguridad y casos de apoyo del sistema (Nivel Inferior).

Para identificar las dependencias entre paquetes, es conveniente revisar el diagrama de casos de uso estructurado según análisis, esto con el fin de ubicar las relaciones que existen entre los casos de uso. Las dependencias se crean a partir de los paquetes de análisis que contienen los casos de uso base. A continuación, se muestra un ejemplo de distribución de paquetes en las capas de la aplicación y sus dependencias para definir la arquitectura de análisis.



Figura 36: Arquitectura de análisis.

[Elaboración propia](#)

1. Organización de los Casos de Uso según Análisis

En el Modelo de Análisis, se coloca el Diagrama General de Casos de Uso en un nuevo Diagrama de modelo de objetos de nombre **Organización de los Casos de Uso según Análisis**; ello implica asignar colores diferentes a los casos de uso. Para ello, podemos tomar en cuenta estas consideraciones:

- Los casos de uso que están relacionados mediante relaciones de generalización deben pertenecer al mismo paquete, por lo tanto presentan el mismo color.
- Los casos de uso relacionados mediante relaciones de extensión y que solo se extienden a partir de un caso de uso base deben pertenecer al mismo paquete del caso de uso base. Por lo tanto, presentan el mismo color
- Los casos de uso incluidos tienden a generar su propio paquete la mayor parte de veces. Si los casos de uso base que incluyen al caso de uso son funcionalidades con distintos contenidos, entonces, se debe crear un paquete para el caso de uso incluido. En este sentido, los casos incluidos y los casos de uso base presentan colores diferentes.

Finalmente, se obtiene el siguiente resultado:

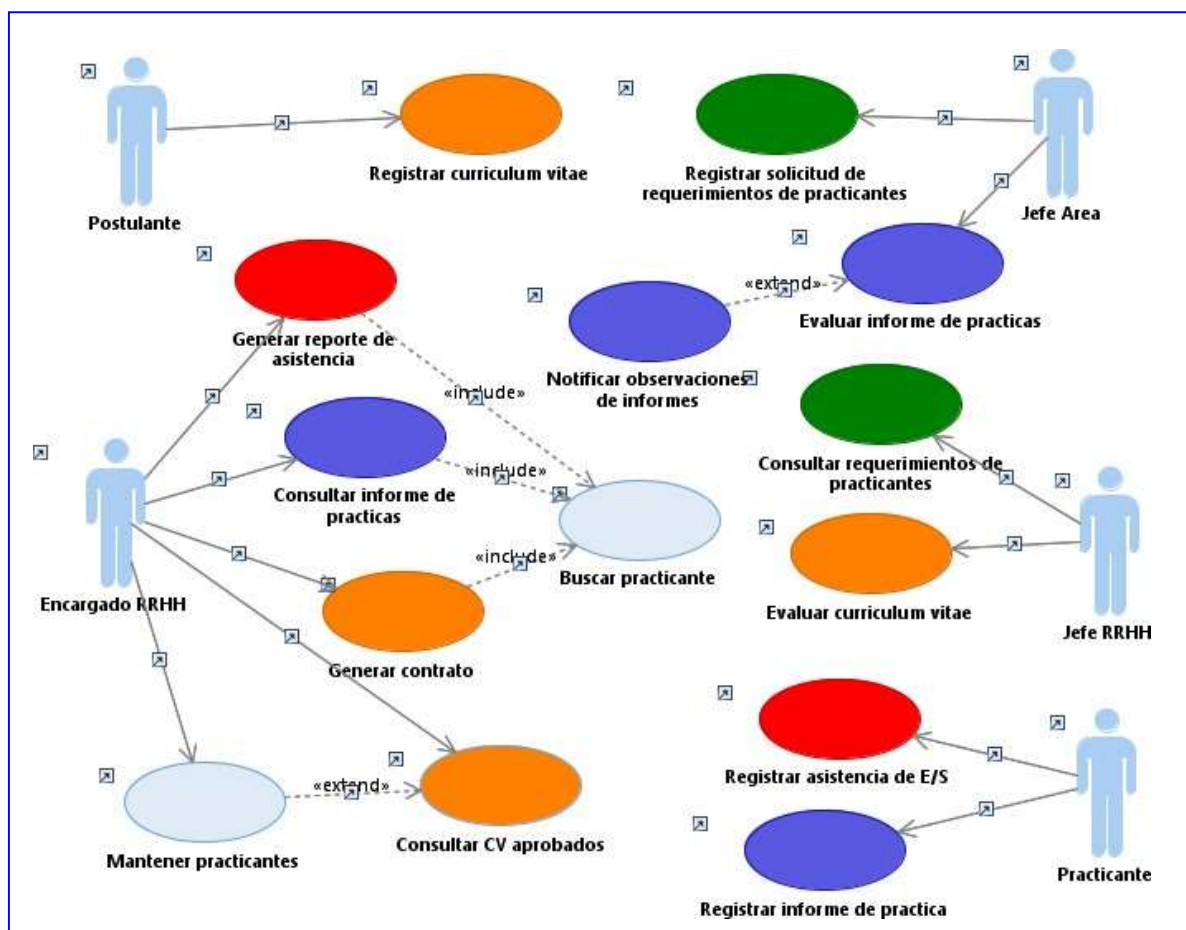


Figura 39: Organización de casos de uso según análisis.
Elaboración propia

2. Arquitectura de Análisis

En el Modelo de Análisis, luego de identificada la nueva relación de casos de uso en la organización de casos de uso según análisis, se confeccionan los paquetes de análisis distribuidos en la capa específica y capa general, creando con ello la arquitectura de análisis, al que se muestra a continuación:

PASO 1: Cree el diagrama de modelo de objetos “Arquitectura de Análisis”, desde el Browser de Proyecto. Para ello, clic derecho sobre el modelo de Análisis (MA) y seleccione **Add/Diagram.../Object Model Diagrams**.

PASO 2: Cree los paquetes de análisis desde el Browser de Proyecto: Asistencia, Selecccion, Requerimientos, Practicas y Practicantes. Para ello, clic derecho sobre el modelo de Análisis (MA) y seleccione **Add Package**. Luego, arrastre los paquetes al Diagrama de “Arquitectura de Análisis”.

PASO 3: Desde el diagrama “Arquitectura de Análisis” agregue texto para cada capa y división. Además, asigne los colores de cada paquete.

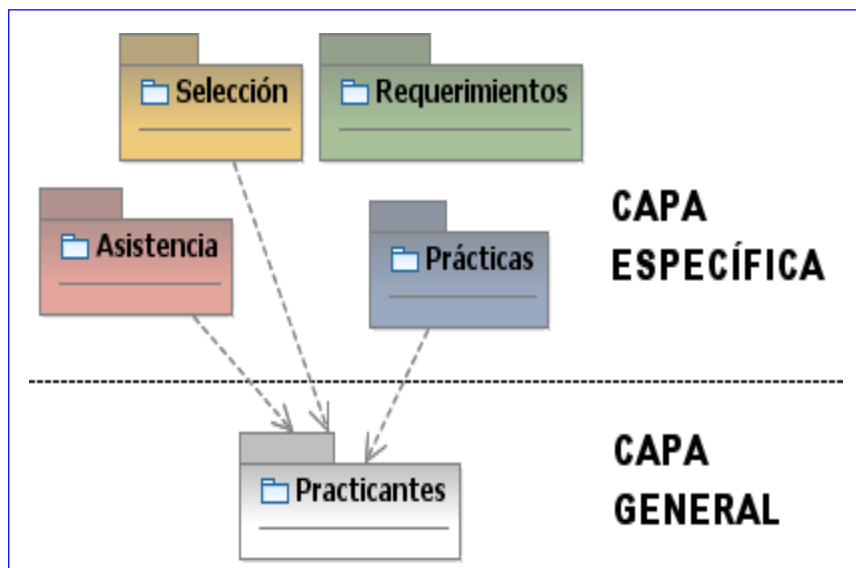


Figura 40: Arquitectura de análisis.
Elaboración propia

ACTIVIDAD PROPUESTA

Ahora, es tu turno de realizar los siguientes artefactos en el modelo de análisis para el caso de estudio Fashion Importaciones:

- Organización de CU según Análisis
- Arquitectura de Análisis

Resumen

1. El Modelo de Casos de Uso es un input para realizar el modelo de análisis. De modo que, debemos completarlo antes de iniciar la confección del modelo de análisis.
2. Los pasos para realizar el análisis de la arquitectura son los siguientes:
 - Organización de casos de uso según análisis
 - Identificación de los paquetes de análisis
 - Definición de las dependencias entre los paquetes de análisis
 - Identificación de clases de entidad obvias por cada paquete de análisis
3. Los paquetes se usan para organizar el modelo de análisis en piezas más manejables, que representan abstracciones o subsistemas y una primera vista del diseño.
4. Un paquete de análisis debe ser débilmente acoplado y altamente cohesivo.
5. Los paquetes de análisis identificados se organizan en la Capa de Aplicación, la cual se subdivide en dos capas internas:
 - Capa Específica: Aquí se ubican los paquetes correspondientes al proceso del negocio (core) de la empresa identificados (Nivel Superior).
 - Capa General: Aquí se ubican los paquetes de maestros de información, paquetes de servicio, seguridad y casos de apoyo del sistema (Nivel Inferior).

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <http://www.ambyssoft.com/unifiedprocess/agileUP.html#Philosophies>
- <http://www.agilemodeling.com/essays/agileArchitecture.htm>

2.4. ANÁLISIS DE CASOS DE USO

2.4.1. Análisis de Casos de Uso para un dato maestro y dato transaccional

El Análisis de Caso de Uso es el proceso de examina los casos de uso para descubrir los objetos y clases de análisis del sistema a desarrollar. Las clases identificadas deben agruparse en los paquetes según criterios de Arquitectura de Software.

Para llevar a cabo el análisis de casos de uso, se realiza lo siguiente:

- Crear un paquete por cada caso de uso a analizar.
- Encontrar clases de análisis según el comportamiento del caso de uso.
- Crear el diagrama de clases (estructura del caso de uso).
- Crear el diagrama de comunicación (comportamiento del caso de uso).

Crear un paquete por cada caso de uso a analizar

Este paquete incluirá los diagramas que describen cómo un caso de uso en particular es modelado, primero su estructura con el diagrama de clases y luego por cada uno de sus flujos descritos en su ECU, se construye un diagrama de comunicación. De este modo, se podrá entender su compartamiento.

Encontrar clases de análisis

Este paso se realiza por cada caso de uso. Para ello, se analizan los escenarios de Caso de Uso para identificar las clases que participan en ellos.

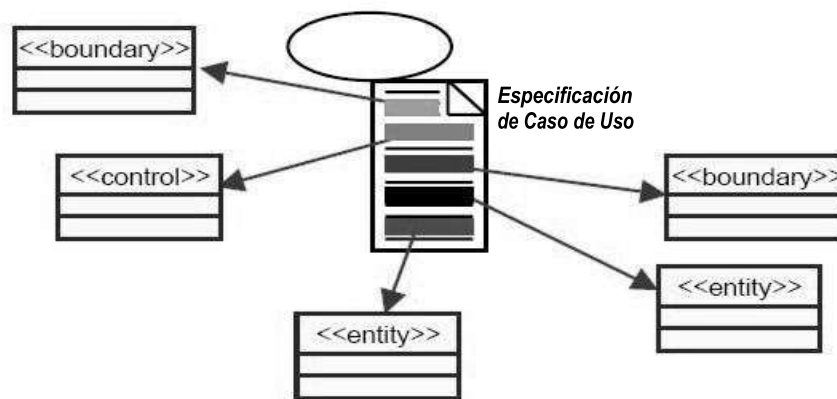


Figura 41: Un flujo completo de un caso de uso debe distribuirse en clases de análisis.

Elaboración propia

Tal como se muestra en la figura anterior, a partir de una especificación de caso de uso (ECU) se pueden obtener las clases de análisis. Existen tres tipos de clases de análisis: boundary, control y entity. A continuación, se describe a cada una de ellas con mayor detalle:

- **Boundary.** Describe una interacción entre el sistema con los usuarios y con otros sistemas. Pueden representar abstracciones de formularios, de protocolos de comunicaciones con otros sistemas o interfaces de dispositivos. Por regla general, al menos una clase boundary sirve como medio de comunicación entre un actor y el correspondiente caso de uso.

Ejemplos:

Si se tiene el caso de uso “Procesar Facturación” donde existe información que debe ser enviada a un Sistema de Facturación externo. Se puede crear una clase de interfaz llamada `CI_SistemaFacturacion` para representar la interfaz al sistema externo.

Si se tiene el caso de uso “Generar Expediente” que será utilizado por un Asesor financiero, se requiere crear la clase interfaz llamada `CI_GeneraExpediente` para representar la interfaz donde el actor ingresará los datos.

- **Entity.** Se emplean para modelar aquella información o comportamiento que posee una vida larga en el sistema. Normalmente, están asociadas a algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso del mundo real.

El número de clases entidad variará dependiendo de los conceptos que requieren almacenamiento persistente dentro del caso de uso. Estas clases sufren un proceso de refinamiento a medida que se ubica a la misma clase entidad dentro de distintas realizaciones de caso de uso.

Ejemplos:

Sea el caso de uso “Mantener empleados” en el cual se puede registrar, modificar o desactivar empleados, es evidente que la información que debe ser manipulada es del empleado. Para ello, se crea una clase entidad `Empleado`.

Sea el caso de uso “Consultar horario de docentes”, es evidente que la información a manipularse en dicha funcionalidad es la del docente y del horario. Para ello se crean las clases entidad `Docente` y `Horario`.

- **Control.** Se utilizan para modelar la coordinación, secuencia, transacciones y control de otros objetos. También para representar derivaciones y cálculos complejos, cómo la lógica de negocio, que no pueden asociarse a ninguna información concreta de larga duración almacenada por el sistema.

Por regla general, se trata de encapsular la lógica de control de un caso de uso dentro de una clase `Control`. Suele ser un buen hábito de diseño utilizar únicamente una clase control por cada caso de uso, y así encapsular en un único elemento la lógica del caso de uso correspondiente. Por otro lado, todos los casos de uso ubicados en un mismo paquete de análisis comparten la misma clase control.

Ejemplo:

En un paquete de análisis denominado Evaluación se ubican los casos de uso “Evaluar empleado”, “Procesar evaluación de desempeño” y “Consultar estadísticas de Evaluaciones”. Para los tres casos de uso se crea una clase control `CC_Evaluacion` que coordina el trabajo de los tres casos de uso.

Crear diagramas de clases de análisis

Este paso permite representar las clases participantes y sus relaciones para un determinado caso de uso.

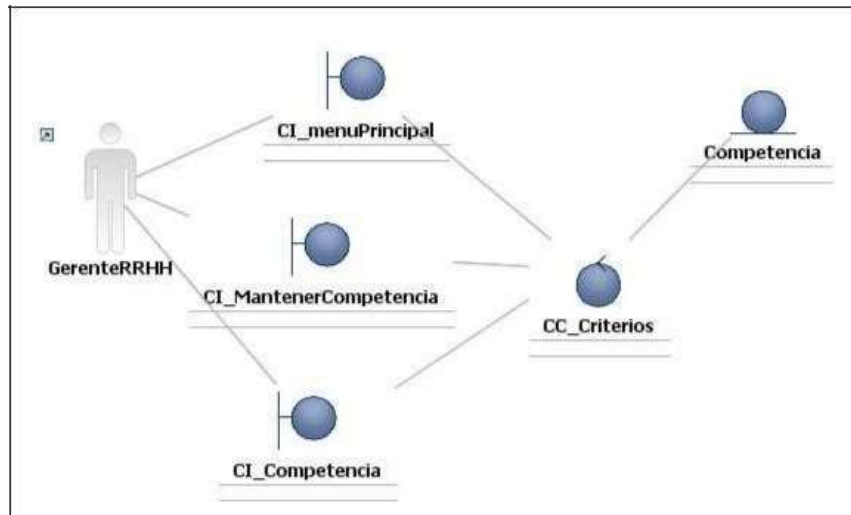


Figura 42: Diagrama de clases de análisis.
Elaboración propia

Crear diagramas de comunicación

El diagrama de comunicación es un tipo de diagrama de interacción. En esta etapa, no se usa diagramas de secuencia, porque no es importante la cronología de las interacciones. Un diagrama de comunicación muestra la colaboración dinámica entre los objetos, es decir, describe el comportamiento de un caso de uso mostrando explícitamente las relaciones de los objetos participantes.

Un caso de uso puede tener uno o más diagramas de comunicación, esto es debido a que se representa el flujo básico, sub- flujos y flujos alternativos. Por ejemplo, a continuación, se muestra el diagrama de comunicación para describir el comportamiento del flujo básico de un caso de uso llamado Mantener competencias.

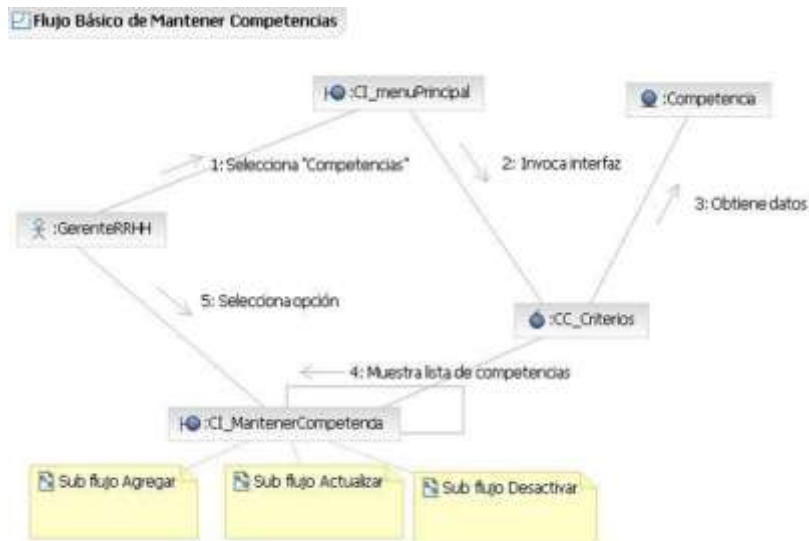


Figura 43: Diagrama de comunicación del flujo básico de un caso de uso
Elaboración propia

2.4.2. Análisis de Casos de Uso para el caso Fashion Importaciones

Para desarrollar la solución del análisis de casos de uso del caso Fashion Importaciones, previamente, revisemos la solución de otro caso de estudio.

CASO RESUELTO

A continuación, se muestra la ECU del CU Mantener Cajero de un sistema de Ventas:

ESPECIFICACIÓN DE CASO DE USO: Mantener Cajero**1. Descripción**

El caso de uso permite mantener actualizado el registro de los cajeros de la clínica. De acuerdo a su necesidad, el Administrador de la Clínica puede agregar, actualizar y desactivar un cajero.

2. Actor(es)

Administrador.

3. Flujo de Eventos**3.1. Flujo Básico**

1. El caso de uso se inicia cuando el Administrador selecciona la opción "Cajeros" en la interfaz del menú principal.
2. El sistema muestra la interfaz "MANTENER CAJERO" con la lista de cajeros con los campos: código, nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro, fecha de actualización y estado. Además, muestra las opciones: **Agregar Cajero, Actualizar Cajero y Desactivar Cajero.**
3. Si el Administrador elige un cajero
 - a. Si elige "Actualizar" ver el Subflujo Actualizar Cajero.
 - b. Si elige "Desactivar" ver el Subflujo Desactivar Cajero.
4. Si el Administrador NO elige un cajero
 - a. si elige "Agregar" ver el Subflujo Agregar Cajero.
5. El Administrador selecciona "Salir" y el caso de uso finaliza.

3.2. Subflujos**3.2.1. Agregar Cajero**

1. El sistema muestra la interfaz CAJERO con los siguientes campos: código (solo lectura), nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro (sólo lectura) y fecha de actualización (solo lectura). Además, muestra las opciones: **Aceptar y Cancelar.**
2. El Administrador ingresa los datos del Cajero.
3. El Administrador selecciona la opción Aceptar.
4. El sistema valida los datos ingresados.
5. El sistema genera un nuevo código de cajero y obtiene la fecha del sistema para la fecha de registro y la fecha de actualización
6. El sistema graba un nuevo registro de cajero y muestra el MSG "Cajero creado con código Nro. 999999".
7. El Administrador cierra la interfaz CAJERO y regresa a la interfaz MANTENER CAJERO con la lista de cajeros actualizada y el subflujo finaliza.

3.2.2. Actualizar Cajero

1. El sistema muestra los datos del cajero seleccionada en la interfaz CAJERO: código (sólo lectura), nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro (sólo lectura) y

fecha de actualización (solo lectura). Además muestra las opciones:

Aceptar y Cancelar.

2. El Administrador actualiza los datos del cajero.
3. El Administrador selecciona la opción Aceptar.
4. El sistema valida los datos ingresados del cajero.
5. El sistema obtiene la fecha del sistema para la fecha de actualización, actualiza el registro de cajero y muestra el MSG "Cajero actualizado satisfactoriamente".
6. El Administrador cierra la interfaz CAJERO y regresa a la interfaz MANTENER CAJERO con la lista de cajeros actualizada y el subflujo finaliza.

3.2.3. Desactivar Cajero

1. El sistema muestra el MSG: "¿Está seguro que desea desactivar el(los) cajero(s) seleccionado(s)?".
2. El Administrador selecciona la opción YES para confirmar la desactivación.
3. El sistema actualiza el registro del(los) cajero(s) en estado "Desactivado".
4. El sistema muestra la interfaz MANTENER CAJERO con la lista de cajeros actualizada y termina el subflujo.

3.3. Flujos Alternativos

1. Datos del Cajero Inválidos

Si los datos ingresados son nulos o inválidos, tanto en los subflujos Agregar como en Actualizar Cajero, el sistema muestra el MSG: "Se han encontrado datos inválidos" y los subflujos continúan en el paso 2.

2. Cajero ya existe

Si el sistema detecta que el cajero ya existe en el paso 4 del subflujo Agregar Cajero, muestra el MSG: "Cajero ya existe" y el subflujo finaliza.

3. No confirma Desactivación

Si el Administrador selecciona NO en el paso 2 del subflujo Desactivar Cajero, finaliza el subflujo.

4. Precondiciones

1. El Administrador está identificado en el sistema.
2. Lista disponible de Cajeros.

5. Poscondiciones

1. En el sistema quedará registrado el nuevo Cajero.
2. En el sistema quedará actualizado el registro del Cajero.
3. En el sistema quedará desactivado el Cajero.

6. Puntos de Extensión

Ninguno.

7. Requisitos Especiales

Ninguno.

La estructura del Caso de Uso se presenta en el siguiente diagrama de clases:

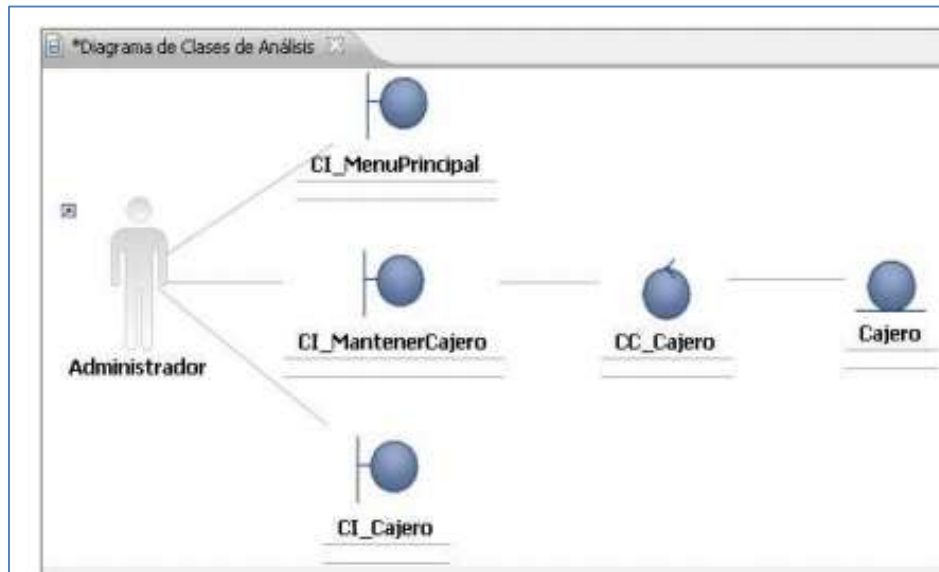


Figura 44: Diagrama de clases del CU Mantener Cajero

Elaboración propia

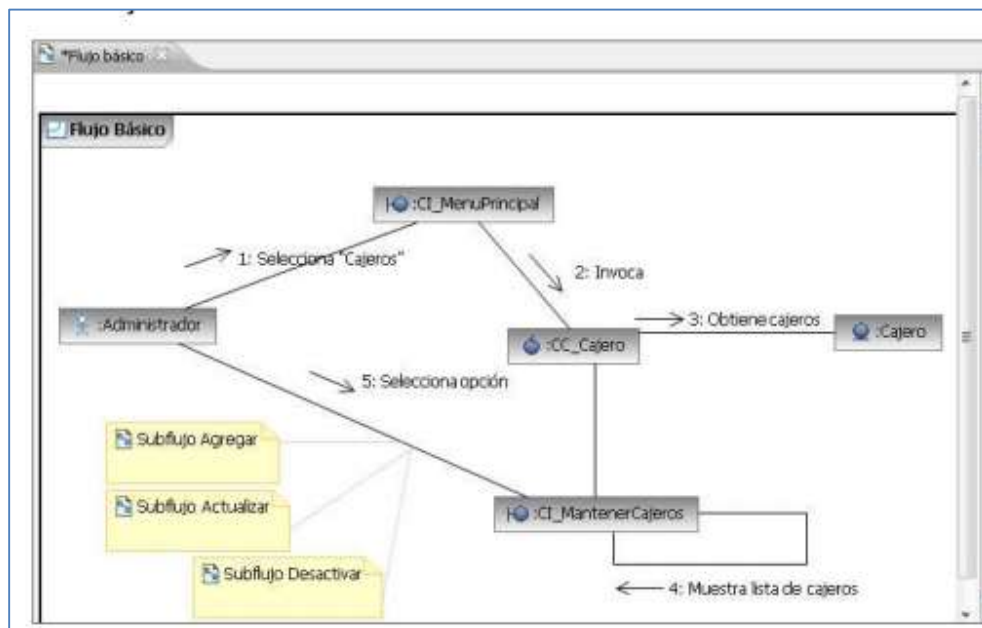


Figura 45: Diagrama de comunicación del flujo básico del CU Mantener Cajero

Elaboración propia

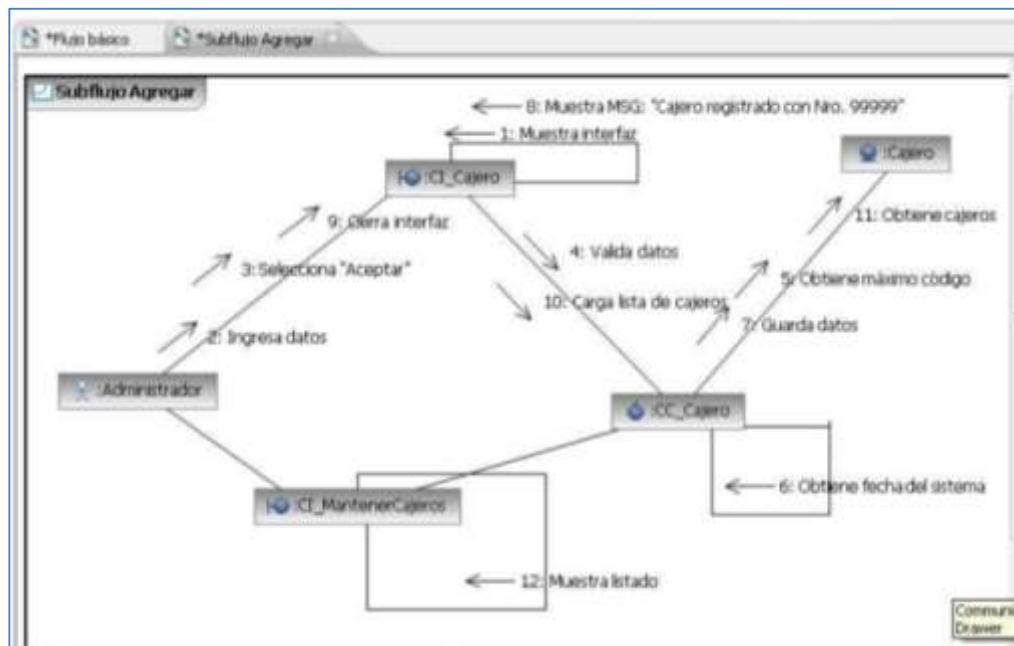


Figura 46: Diagrama de comunicación del subflujo "Agregar" del CU Mantener Cajero
Elaboración propia

ACTIVIDAD PROPUESTA

Ahora, es tu turno de realizar los siguientes artefactos en el modelo de análisis para el caso de estudio Fashion Importaciones para lo cual debe considerar las ECUs descritas en las páginas del 36 al 40:

- Mantener Productos: Caso de uso que manipula un dato maestro.
- Evaluar Pedidos al Crédito: Caso de uso que manipula un dato transaccional.

Resumen

1. El Modelo de Análisis representa los componentes relevantes, atendiendo las tres capas arquitectónicas MVC de las funcionalidades identificadas en el sistema.
2. El análisis de la arquitectura incluye la definición de una capa general y una capa específica que incluyen las funcionalidades o casos de uso que manipulan datos reutilizables y casos de uso que manipulan datos transaccionales del core del negocio, respectivamente.
3. El análisis de casos de uso, tanto para un dato maestro como para un dato transaccional, sugiere incluir un diagrama de clases de análisis (boundary, control y entity) y un diagrama de comunicación que muestre la interacción de los componentes relevantes identificados como clases del tipo interfaz, control y entidad.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <http://www.agilemodeling.com/essays/agileArchitecture.htm>
- <https://tpetricek.github.io/Teaching/software-engineering/requirements.html#/4/6>



MODELADO DE DATOS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno crea los modelos conceptual, lógico y físico de datos que den soporte a dos procesos de negocio, utilizando las herramientas IBM Rhapsody e IDA.

TEMARIO

3.1 Tema 8 : Modelo de datos

- 3.1.1 : Modelo Conceptual
- 3.1.2 : Modelo Lógico
- 3.1.3 : Modelo Físico

3.2 Tema 9 : Auditoría en base de datos

- 3.2.1 : Definición, Importancia y Objetivos.
- 3.2.2 : Modelo de Datos con tablas de auditoría

3.1. MODELO DE DATOS

Un modelo de datos se orienta al tratamiento de una Base de Datos, el cual permite describir:

- Las estructuras de datos de la base: El tipo de los datos que se presenta en la base y la forma en que se relacionan.
- Las restricciones de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar la realidad deseada.
- Operaciones de manipulación de los datos: típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base.

Un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

Como es sabido, una Base de Datos siempre está orientada a resolver un problema determinado, por lo que los dos enfoques propuestos son necesarios en cualquier desarrollo de software.

Un modelo de datos es un lenguaje que, típicamente, tiene dos sub-lenguajes:

- Un Lenguaje de Definición de Datos o DDL (Data Definition Language), orientado a describir de una forma abstracta las estructuras de datos y las restricciones de integridad.
- Un Lenguaje de Manipulación de Datos o DML (Data Manipulation Language), orientado a describir las operaciones de manipulación de los datos.

A la parte del DML orientada a la recuperación de datos, usualmente se le llama Lenguaje de Consulta o QL (Query Language).

Una opción bastante usada a la hora de clasificar los modelos de datos es hacerlo de acuerdo con el nivel de abstracción que presentan:

Modelos de Datos Conceptuales

Son los orientados a la descripción de estructuras de datos y restricciones de integridad. Se usan fundamentalmente durante la etapa de Análisis de un problema dado y están orientados a representar los elementos que intervienen en ese problema y sus relaciones. El ejemplo más típico es el Modelo Entidad- Relación.

Modelos de Datos Lógicos

Son orientados a las operaciones más que a la descripción de una realidad. Usualmente están implementados en algún Manejador de Base de Datos. El ejemplo más típico es el Modelo Relacional, que cuenta con la particularidad de contar también con buenas características conceptuales (Normalización de bases de datos).

Modelos de Datos Físicos

Son estructuras de datos a bajo nivel implementadas dentro del propio manejador. Ejemplos típicos de estas estructuras son los Árboles B+, las estructuras de Hash, etc.

3.1.1. Modelo Conceptual

Las clases del modelo conceptual se obtienen a partir de los objetos de información que fluyen entre las actividades. Una característica importante que resaltar es que el modelado de los casos de uso del sistema y el modelado conceptual se realizan en paralelo, esto es crucial para obtener casos de uso correctos, puesto que es necesario entender bien el dominio para poder escribir casos de uso que sean realmente útiles.

El Modelo Conceptual Orientado a Objetos beneficiará a dos equipos de trabajo:

- **Equipo de Desarrolladores:** En esta etapa del desarrollo, es conveniente detenerse en la identificación de los conceptos y no tanto en las relaciones entre ellos. Este modelo incluirá los conceptos y sus relaciones y se describirá mediante un diagrama de clases UML, en el que los conceptos se representan mediante clases (del dominio).
- **Equipo de Base de Datos:** En esta etapa, luego del modelo conceptual, se obtiene el proceso del modelo lógico al diseño físico donde se podrá identificar las tablas relacionales del proyecto de Base de Datos como componente RUP.

A continuación, se describe los pasos que se realizan para la construcción del modelo conceptual, los cuales son los siguientes:

- Identificar clases persistentes con sus atributos
- Asociar clases
- Identificar agregaciones
- Definir jerarquías de clases

Paso 1. Identificar clases persistentes con sus atributos. Este paso incluye identificar los siguientes elementos:

- La Clase es la unidad básica que encapsula toda la información de un objeto (o una instancia de una clase). A través de ella, podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).
- Los atributos representan las propiedades de la clase que se encuentran en todas las instancias de la clase. Definen la estructura de una clase y de sus objetos.
- Los atributos corresponden a sustantivos y sus valores pueden ser sustantivos o adjetivos.
- Dentro de una clase, los nombres de los atributos deben ser únicos (aunque puede aparecer el mismo nombre de atributo en diferentes clases).
- Los atributos pueden representarse solo mostrando su nombre, su tipo e, incluso, su valor por defecto. Además, pueden presentar identificadores de acceso: public, private o protected.
- Para los Identificadores, en el momento de incluir atributos en la descripción de una clase se debe distinguir entre sí atributos que reflejan las características de los objetos en el mundo real y los identificadores que son utilizados por razones de implementación

Existen algunas categorías de clases que podríamos utilizar para identificarlas correctamente. A continuación, se muestra una tabla con algunos ejemplos:

Categoría	Ejemplo
Tangibles o físicos	Edificio, Producto
Especificaciones o descripciones	EspecificacionProducto, DescripcionVuelo
Lugares	Tienda, Aula, Laboratorio
Transacciones	Venta, Pago, Reserva
Líneas o detalle de transacción	LineaVenta, DetalleReserva
Registros de finanzas, expedientes	CDP, Factura, Ticket, HistoriaClinica
Roles de personas	Cajero, Piloto
Organizaciones	Departamento, Sucursal
Historiales	PrecioProducto, PrecioDolar, AtencionCitas
Registros de cambios de estados	DisponibilidadHabitacion, DisponibilidadButaca
Conceptos abstractos	RangoHora, UnidadAprendizaje
Relaciones	Amistad, Parentesco

Figura 47: Categoría de clases
Elaboración propia

Paso 2. Asociar clases. En este paso es importante considerar los siguientes aspectos:

- La asociación es una relación entre clases que indica una conexión significativa e interesante.
- Está representada como una línea entre clases con nombre. La asociación es inherentemente bidireccional.
- Es convencional leer la asociación de izquierda a derecha o de arriba hacia abajo.
- Criterios para identificar asociaciones
 - Enfocarse en aquellas asociaciones para las cuales el conocimiento de la relación necesita ser conservado en el tiempo. (Asociaciones que se necesitan saber).
 - Demasiadas asociaciones tienden a confundir.
 - Evitar mostrar asociaciones redundantes o derivables.
 - Pueden existir múltiples asociaciones entre dos clases.

a. Tipos de Asociaciones

- Asociación Binaria: Asociación entre dos clases

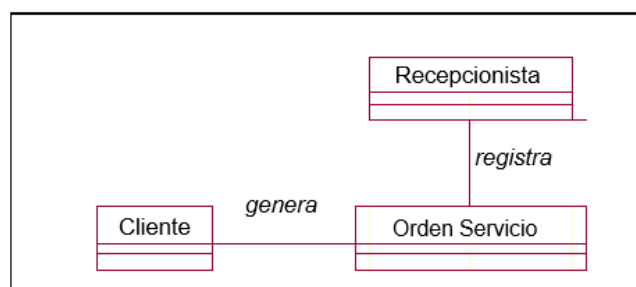


Figura 48: Asociación Binaria
Elaboración propia

- Asociación de clase: Asociación entre dos clases que contiene otra entidad. Generalmente, este tipo de asociación se utiliza para representar una relación de muchos a muchos entre dos clases.

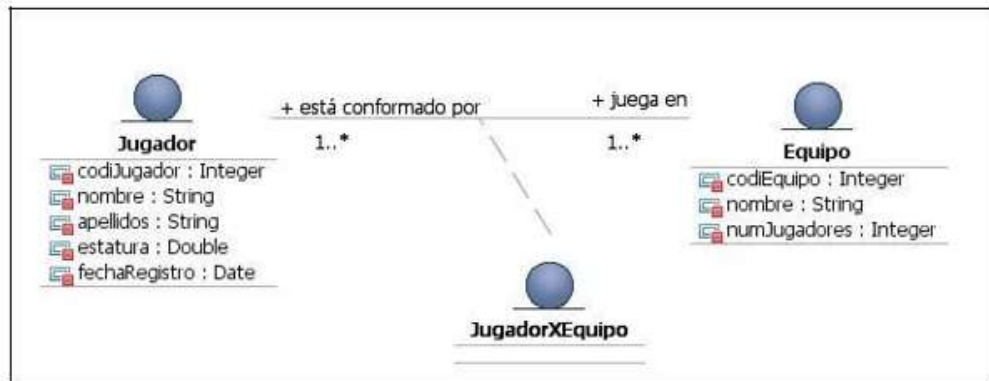


Figura 49: Asociación de clase
Elaboración propia

- Asociación Reflexiva: Se da en la misma clase.

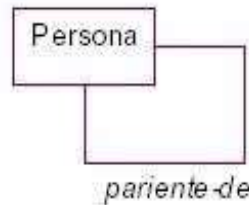


Figura 50: Asociación Reflexiva
Elaboración propia

b. Clase Asociativa

- Se da cuando uno o más atributos están relacionados con la asociación.
- Las instancias de la clase asociativa dependen del tiempo de vida de la asociación.
- Se da en una asociación de muchos a muchos entre dos clases y existe información asociada con la propia relación de asociación.

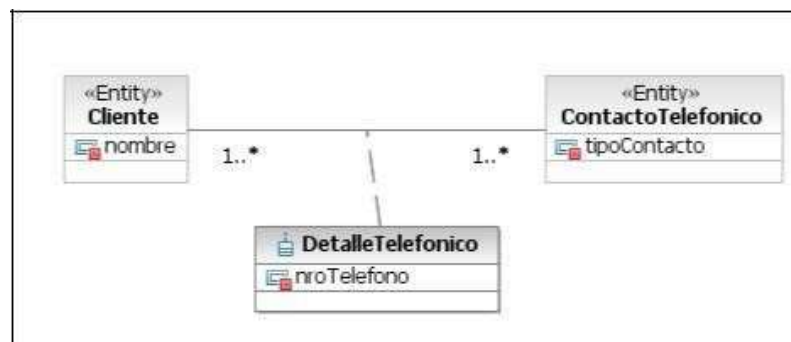


Figura 51: Clase Asociativa
Elaboración propia

c. Roles

- Dada una asociación entre dos entidades, decimos que cada entidad representa un rol en dicha asociación.
- Muchas veces, según el punto de vista de cada entidad, es posible nombrar a la asociación de manera diferente.

d. Multiplicidad

- Restringe el número de objetos de una clase que se pueden implicar en una relación determinada en cualquier momento en el tiempo. La frase “en cualquier momento en el tiempo” es vital para entender las multiplicidades.
- Define cuántas instancias de la clase A pueden estar asociadas con una instancia de la clase B.
- La multiplicidad presenta las relaciones con valores de datos de acuerdo con el detalle siguiente:

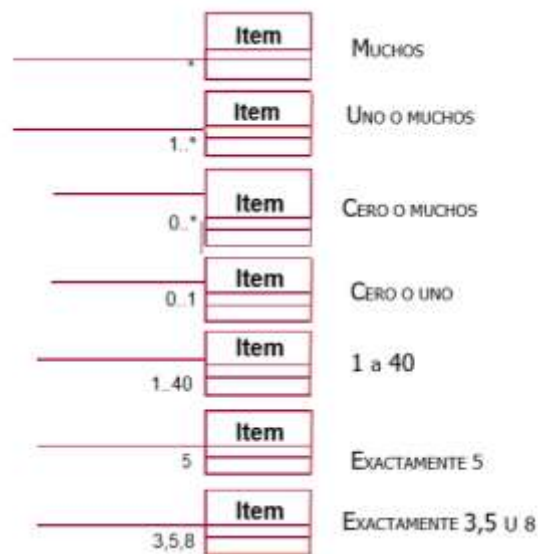


Figura 52: Multiplicidad
Elaboración propia

En el siguiente ejemplo, se representa las siguientes relaciones:

- Un jugador “juega en” muchos equipos
- Un equipo “está conformado por” varios jugadores.
- Cada jugador, dependiendo del equipo en que se encuentre tendrá un rol diferente.

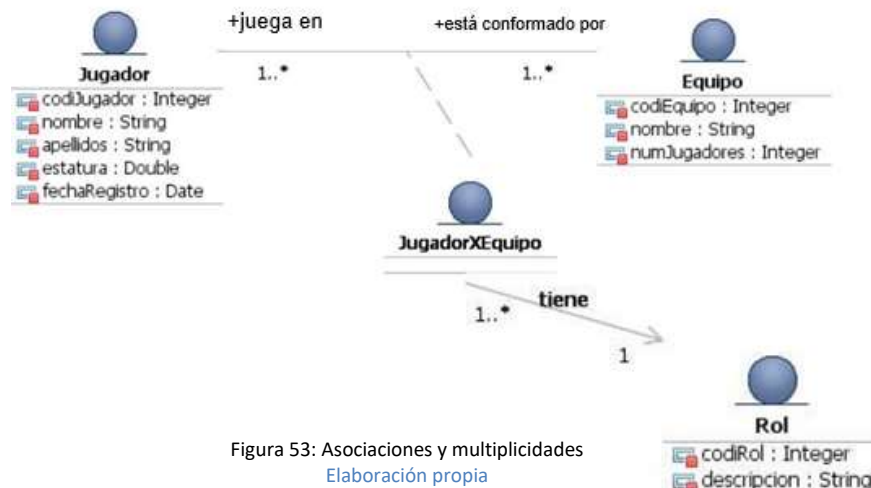


Figura 53: Asociaciones y multiplicidades
Elaboración propia

Paso 3. Identificar agregaciones.

- La agregación indica una relación de “un todo conformado por partes”
- Existen 2 tipos de agregaciones: Agregación Débil o Compartida y Agregación Fuerte o Compuesta
- La agregación representa una relación parte de entre objetos.
- En UML se proporciona una escasa caracterización de la agregación
- Puede ser caracterizada con precisión determinando las relaciones de comportamiento y estructura que existen entre el objeto agregado y cada uno de sus objetos componentes.

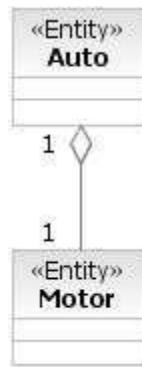


Figura 54: Agregación entre clases
Elaboración propia

A continuación, se describen y se muestra un ejemplo de cada tipo de agregación: Agregación Débil o Compartida y Agregación Fuerte o Compuesta

a. Agregación Compartida

Es un tipo de relación utilizada para modelar la relación todo-parte entre objetos. La parte puede estar simultáneamente en varias instancias del todo. La agregación existe de preferencia entre conceptos no físicos.

Ejemplo:

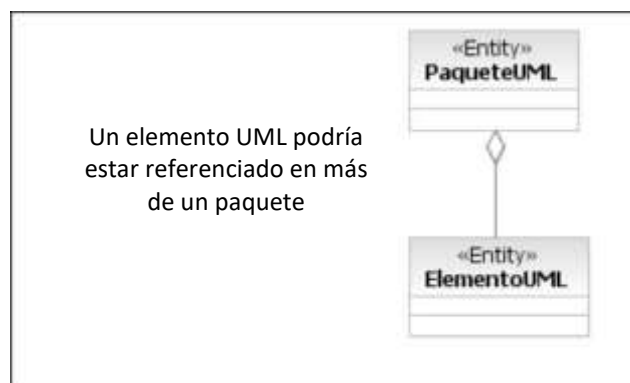


Figura 55: Diagrama de Actividades Propuesto para el CUN Venta de Productos
Elaboración propia

b. Agregación Compuesta

Es un tipo de relación utilizada para modelar la relación todo-parte entre objetos. Significa que la parte es miembro de solamente un objeto todo, es decir, la existencia de la parte depende del todo. La composición se representa con un diamante relleno. El objeto todo es el único dueño del objeto parte.

Ejemplo:

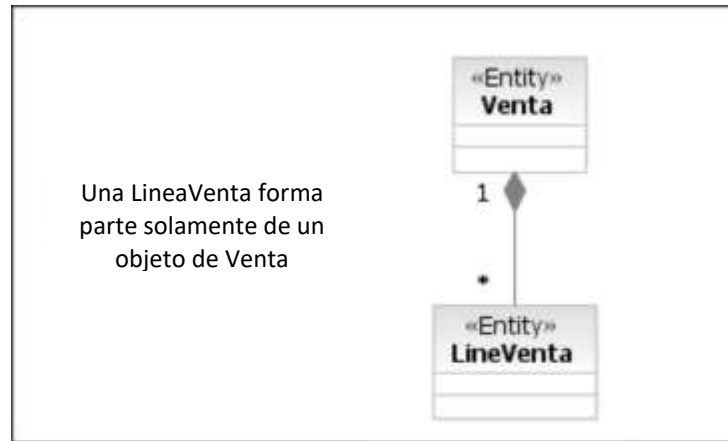


Figura 56: Diagrama de Actividades Propuesto para el CUN Venta de Productos
Elaboración propia

Paso 4. Definir jerarquías de clases. Permiten gestionar la complejidad mediante un ordenamiento lógico. Se obtiene usando los mecanismos de abstracción de Generalización y/o Especialización.

Generalización y Especialización son conceptos fundamentales en el Modelo Conceptual que permiten la reducción de la expresión. Las jerarquías de clases son a menudo las bases de inspiración para las jerarquías de las clases de software que exploten la herencia y reduzcan la duplicación de código.

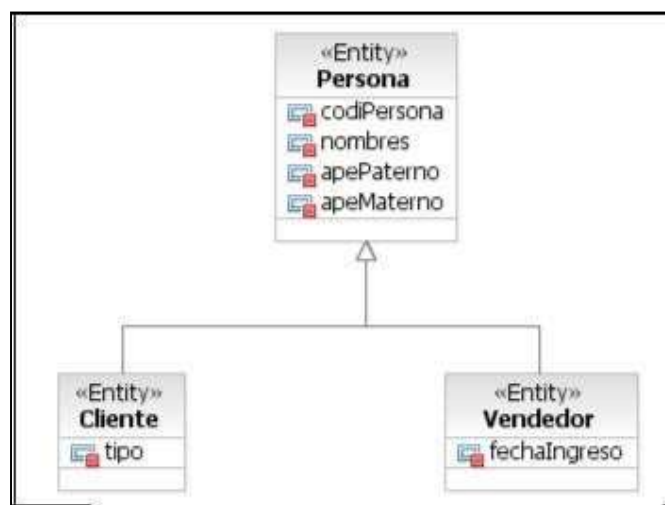


Figura 57: Jerarquía de clases
Elaboración propia

A continuación, se describen y se muestra un ejemplo de cada tipo de jerarquía de clases: Generalización y Especialización.

a. Generalización

- Consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general.
- Nombres usados: clase padre - clase hija, superclase - subclase, clase base - clase derivada
- Las subclases heredan características de sus superclases, es decir, los atributos y operaciones (y asociaciones) de la superclase están disponibles en sus subclases.

Ejemplo:



Figura 58: Generalización
Elaboración propia

b. Especialización

- Es el resultado de tomar un subconjunto de entidades de alto nivel para formar un conjunto de entidades de más bajo nivel.
- Las entidades de bajo nivel presentan atributos adicionales diferenciándolos de las otras entidades que se han creado producto de la especialización

Ejemplo:

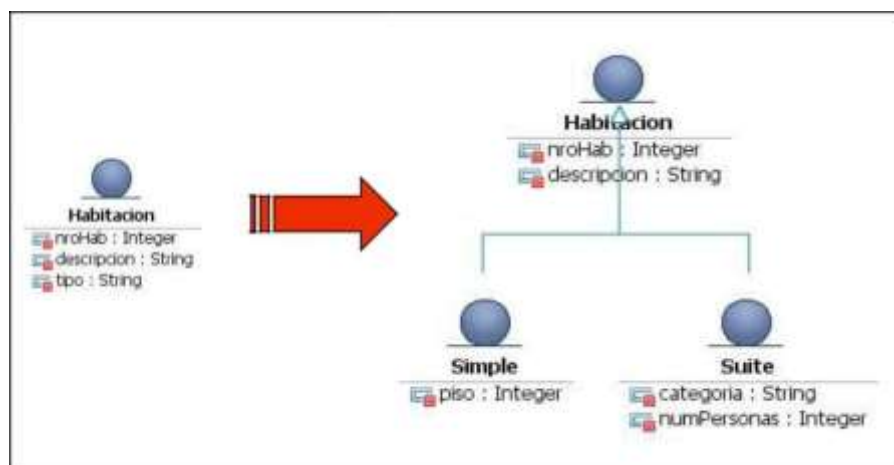


Figura 59: Especialización
Elaboración propia

3.1.2. Modelo Lógico

Un modelo lógico de datos es un modelo que no es específico de una base de datos que describe aspectos relacionados con las necesidades de una organización para recopilar datos y las relaciones entre estos aspectos.

El modelo lógico de datos es el refinamiento del modelo conceptual. En este modelo no es necesario especificar las llaves primarias y foráneas de las entidades, pues es una tarea que se recomienda realizar en el modelo físico.

Existen muchas herramientas que permiten realizar transformaciones de un modelo lógico a partir de un modelo conceptual (con notación UML). La transformación UML a modelo lógico de datos genera tipos de datos de modelo lógico de datos a partir de los tipos primitivos de UML. En la siguiente tabla, se muestra la correspondencia entre tipos primitivos de UML y tipos de datos de modelo lógico de datos que se obtienen al utilizar la herramienta InfoSphere Data Architect (IDA) de IBM:

Tipos primitivos de UML	Tipos de datos de modelo lógico de datos que la transformación genera
Boolean, boolean	BOOLEAN
Byte, byte o char	CHAR
Date	DATE
Double, double	DOUBLE
Float	FLOAT
Integer, int	INTEGER
Long, long	LONG
Short	SHORT
Cadena de caracteres	VARCHAR (32672)

Figura 60: Correlaciones entre tipos de datos del modelo conceptual con UML y el del modelo lógico de datos en IDA.

Elaboración propia

Los tipos de relaciones que se pueden crear para un modelo lógico son los siguientes:

- Relación identificada
- Relación No identificada
 - Opcional
 - Obligatoria

Relación Identificada

Las relaciones identificadas migran la llave primaria de la entidad padre a la llave primaria de la entidad hija. Su representación es una línea nítida, tal como se ilustra en la siguiente figura.

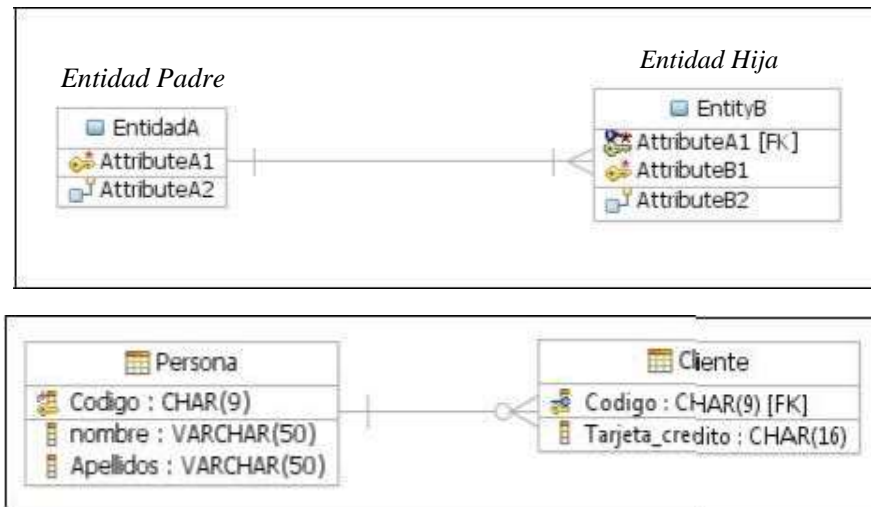


Figura 61: Relación Identificada en el Modelo lógico Vs. Modelo físico.

Elaboración propia

Relación No identificada

En una relación no identificada la llave primaria de la entidad padre, es migrada como un atributo más de la entidad hija, es decir, no formará parte de su llave primaria. Además, en este tipo de relación se representa otra característica: la existencia. La existencia describe la relación entre un par de entidades desde la perspectiva de la entidad hija. Fundamentalmente, haciendo la pregunta, ¿Es el valor de una llave foránea siempre requerida en la entidad hija? Las posibles respuestas son las siguientes:

- Opcional.** Cuando el valor de una llave foránea no es siempre requerido en la entidad hija. Sin embargo, si un valor existe, el valor de la llave foránea debe encontrarse en la llave primaria de la entidad padre.

La representación de la relación No identificada Opcional es una línea entrecortada y en el extremo de la entidad padre aparece una línea con un círculo. Así:

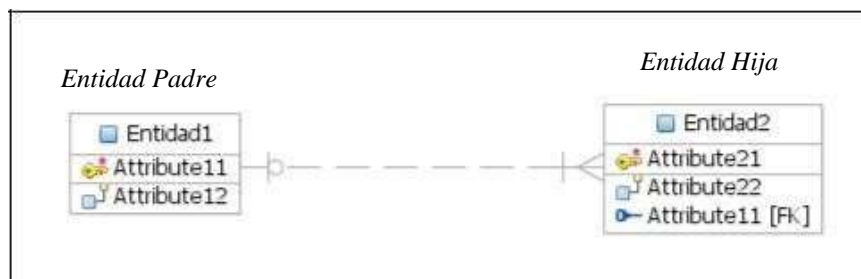


Figura 62: Relación No Identificada Opcional.

Elaboración propia

- b. **Obligatoria.** Cuando el valor de una llave foránea debe existir en la entidad hija y el valor de la llave foránea debe encontrarse en la llave primaria de la entidad padre.

La representación de la relación No identificada obligatoria es una línea entrecortada y en el extremo de la entidad padre aparece una línea. Así:

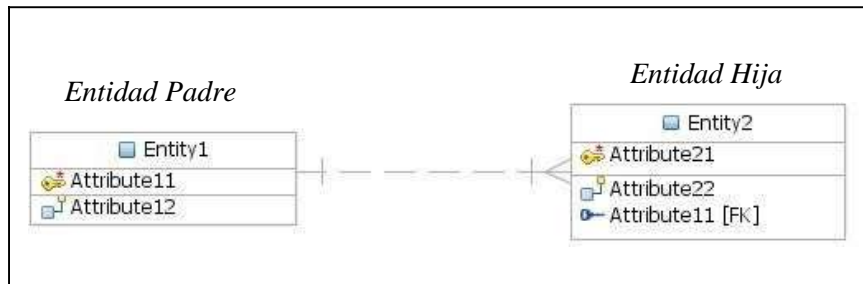


Figura 63: Relación No Identificada Obligatoria.

[Elaboración propia](#)

Un proceso de transformación del modelo conceptual (con notación UML) a modelo lógico de datos, genera tipos de relación, existencia y cardinalidad para el modelo lógico de datos de acuerdo con los tipos de asociación y multiplicidad de rol UML.

En la siguiente tabla, se muestran las correlaciones entre un tipo de asociación o multiplicidad de rol UML y el tipo de relación, existencia y cardinalidad de un modelo lógico de datos que se crean en un proceso de transformación en el entorno de IDA.

UML			Modelo lógico de datos		
Tipo de asociación	Multiplicidad de rol padre	Multiplicidad de rol hijo	Tipo de relación	Existencia	Cardinalidad
Simple o agregación	(0..1)	(0..1) /1/~/ (1..~)	No identificada	Opcional	(0..1) /1/~/ (1..~)
Simple o agregación	1	(0..1) /1/*/(1..*)	No identificada	Obligatoria	(0..1) /1/*/(1..*)
Composición	(0..1) /1	(0..1) /1/*/(1..*)	Identificada	Siempre son Obligatorias	(0..1) /1/*/(1..*)

Figura 64: Correlaciones entre asociaciones UML y relaciones de modelo lógico de datos en IDA.

[Elaboración propia](#)

3.1.3. Modelo Físico

El modelo físico es un modelo de datos de bajo nivel. Proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el ordenador.

El paso de un modelo lógico a uno físico requiere un profundo entendimiento del manejador de bases de datos que se desea emplear, incluyendo características como:

- Conocimiento a fondo de los tipos de objetos (elementos) soportados
- Detalles acerca del indexamiento, integridad referencial, restricciones, tipos de datos, etc.
- Detalles y variaciones de las versiones
- Parámetros de configuración
- Data Definition Language (DDL)
- El paso de convertir el modelo lógico de datos a tablas hace que las entidades pasen a ser tablas (más las derivadas de las relaciones) y los atributos se convierten en las columnas de dichas tablas.

El modelo físico de datos también se puede realizar directamente desde un sistema manejador de base de datos que además permite generar el script de la base de datos, o viceversa, es decir, se crea el script de la base de datos que contiene tablas y campos; luego, a partir de éste se visualiza o genera el modelo físico.

A continuación, se muestran sistemas manejadores de base de datos (SMBD) más populares.

Logo	Nombre	URL	Productos
	Microsoft	www.microsoft.com	Access, MS-SQL Server
	MySQL	www.mysql.com	MySQL
	Informix	www.informix.com	Illustra, Universal Server, Dynamic Server
	IBM	www.ibm.com	DB2
	Apache	http://db.apache.org/derby	Derby
	SQLite	http://www.sqlite.org	SQLite
	Firebird	http://firebird.sourceforge.net	Firebird

Figura 65: SMBD más populares – Parte I.
Elaboración propia

Logo	Nombre	URL	Productos
	Sybase	www.sybase.com	Adaptive Server
	Oracle	www.oracle.com	Oracle8, Oracle8i, Oracle8iEE, Oracle9i, Oracle 10g
	PostgreSQL	www.postgresql.org	PostgreSQL

Figura 66: SMD más populares – Parte II.
Elaboración propia

Caso Práctico (Modelado de datos)

A continuación, del Sistema Comercial se mostrará el modelo conceptual, y físico obtenido a partir de la ECU Registrar Pago de Facturas al Crédito.

Especificación de Caso de Uso: Registrar Pago de Facturas al Crédito

1. Breve descripción

El caso de uso permite al Producto registrar el pago de las facturas a crédito de los clientes

2. Actor(es)

Producto

3. Flujo de Eventos

3.1. Flujo Básico

1. El caso de uso comienza cuando el Producto solicita “Registrar pago de Factura” en el menú principal.
2. El sistema muestra la interfaz REGISTRAR PAGO DE FACTURAS con los siguientes campos:
Datos de Factura: NumFactura Fecha de factura, Fecha de vencimiento, Nombre del Cliente, nombre del vendedor, Monto total y una opción para Importar.
Además, incluye una cuadrícula que contiene la lista de los productos de la factura con los campos: Código, descripción, precio cantidad, total y las opciones: Registrar Pago y Salir.
3. El Producto ingresa el número de Factura
4. El Producto selecciona Importar
5. El sistema obtiene los datos de la factura: pendientes de Pago, cabecera y el detalle de la factura
6. El sistema muestra los datos de la factura: Fecha de factura, Fecha de vencimiento, Nombre del Cliente, nombre del vendedor, Monto total y agrega los datos del detalle en la cuadrícula de detalle de productos
7. El Producto selecciona “Registrar Pago”.
8. El sistema genera el número de la Hoja de Pagos, graba la Hoja de pago (Número de hoja, número de Factura, Código cliente, fecha, vendedor, Producto y monto) y actualiza el estado de la factura como “Pagada”
9. El sistema muestra el número de Hoja de Pago y el MSG “Hoja de pago generada” con el Nro. 99999”.

10. El Producto cierra la interfaz “REGISTRAR PAGO DE FACTURAS” y regresa a la interfaz del menú principal del sistema y finaliza el caso de uso.

3.2. Flujos Alternativos

<Salir>

Si el Producto solicita “Salir” antes de registrar el pago, el sistema cierra la interfaz y el caso de uso finaliza.

4. Requerimientos Especiales

No presenta.

5. Pre Condiciones

1. El Producto está logeado en el sistema.
2. Facturas pendientes de pago

6. Post Condiciones

Se graba la hoja de pago y actualiza las facturas como pagadas

7. Puntos de extensión

Ninguno

8. Prototipo

Registrar Pago de Facturas

Datos de la Cliente

Num Factura Fecha de Factura

Cliente Fecha de Vencimiento

Nombre

Vendedor Monto Total

Detalle de productos

CODIGO	DESCRIPCION	PRECIO	CANTIDAD	TOTAL

Figura 67: Interfaz Registrar Pago
Fuente.- Tomado del manual de ADSII 2018

Solución:

El modelo conceptual nos muestra las siguientes relaciones:

- Asociación de clase: Factura-Producto
- Asociaciones bidireccionales (con navegabilidad):
 - Factura-Cliente
 - Factura-Vendedor

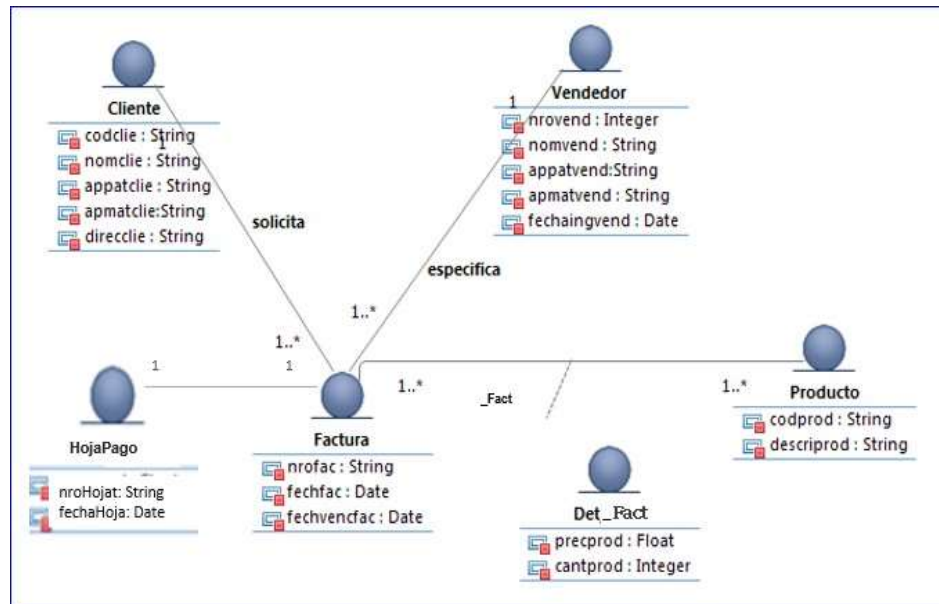


Figura 68: Modeo Conceptual para Registrar Pago

Fuente.- Tomado del manual de ADSII 2018

A continuación, se muestra el modelo físico del caso.

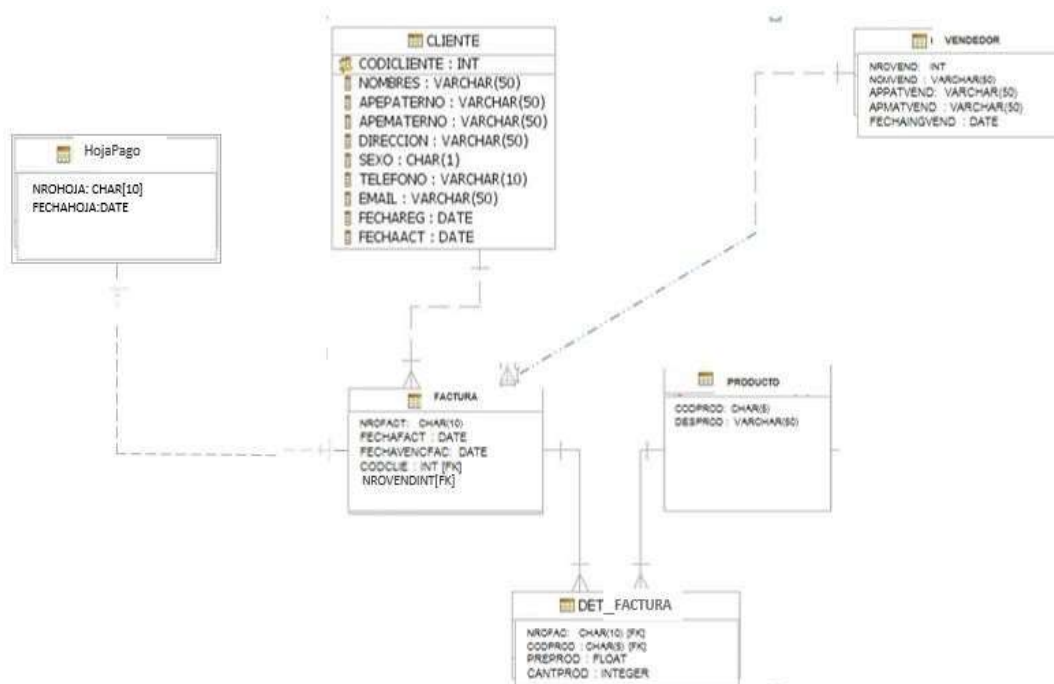


Figura 69: Modeo Físico para Registrar Pago

Fuente.- Tomado del manual de ADSII 2018

ACTIVIDAD PROPUESTA

A partir de la Especificación del Caso de Uso Mantener Productos (pag.36) y Evaluar Pedido al Crédito (pag.38), elabore el Modelo Conceptual y el Modelo Físico para el caso de estudio Fashion Importaciones.

Resumen

1. El modelo conceptual es un modelo de datos de alto nivel en el que disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos.
2. El modelo conceptual orientado a objetos beneficiará a dos equipos de trabajo: equipo de desarrolladores porque requieren identificar el modelo de dominio de datos y equipo de base de datos debido a que es el input para obtener finalmente el modelo físico.
3. Un modelo lógico de datos es un modelo que no es específico de una base de datos que describe aspectos relacionados con las necesidades de una organización para recopilar datos y las relaciones entre estos aspectos. Es el refinamiento del modelo conceptual; no es necesario especificar las llaves primarias y foráneas de las entidades, pues es trabajo que se recomienda realizar en el modelo físico.
4. El modelo físico es un modelo de datos de bajo nivel. Proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el ordenador.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- http://www.michael-richardson.com/processes/rup_classic/#extend.bus_model/disciplines/rup_business_modeling_discipline_553E20F6.html
- https://www.ibm.com/support/knowledgecenter/SSB2MU_8.3.0/com.ibm.rhp.uml.diagrams.doc/topics/rhp_t_dm_creating_rhp_projects.html

3.2. AUDITORÍA EN BASE DE DATOS

La información que se trata y almacena en una base de datos cada día cobra más y más importancia, así como se han debido implementar políticas y ver más detalladamente el manejo de la información, la transparencia, fiabilidad y seguridad en una base de datos. Un punto importante en donde hay que detenerse y aplicar políticas de control, seguimiento de cambios de datos que alteran la base de datos, así respaldar y encontrar el hilo de la trayectoria desde que se conecta un usuario a una base de datos determinada, los cambios que realice y el termino de su sesión como tal usuario. Tratando así en resguardar el 100% de la integridad de la información tratada en las bases de datos, es aquí en donde entra en marcha la auditoría de base de datos.

La auditoría de las bases de datos consiste en el control de acceso, de actualización, de integridad y calidad de los datos.

3.2.1. Definición, Objetivo e Importancia

Definición

Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información almacenada en la base de datos, incluyendo la capacidad de determinar:

- Quién accede a los datos.
- Cuándo se accedió a los datos
- Desde qué tipo de dispositivo/aplicación
- Desde qué ubicación en la Red
- Cuál fue la sentencia SQL ejecutada
- Cuál fue el efecto del acceso a la base de datos

Objetivos Generales

Disponer de mecanismos que permitan tener trazas de auditoría completas y automáticas relacionadas con el acceso a las bases de datos incluyendo la capacidad de generar alertas con el objetivo de:

- Mitigar los riesgos asociados con el manejo inadecuado de los datos
- Apoyar el cumplimiento regulatorio.
- Satisfacer los requerimientos de los auditores
- Evitar acciones criminales
- Evitar multas por incumplimiento

Importancia

- La importancia de la auditoría del entorno de bases de datos radica en que es el punto de partida para poder realizar la auditoría de las aplicaciones que utiliza esta tecnología.
- Toda la información financiera de la organización reside en bases de datos y deben existir controles relacionados con el acceso a las mismas.
- Se debe poder demostrar la integridad de la información almacenada en las bases de datos.
- Las organizaciones deben mitigar los riesgos asociados a la pérdida de datos y a la fuga de información.
- La información confidencial de los clientes son responsabilidad de las organizaciones.
- Los datos convertidos en información a través de bases de datos y procesos de negocios representan el negocio.

- Las organizaciones deben tomar medidas mucho más allá de asegurar sus datos. Deben monitorearse perfectamente a fin de conocer quién o qué les hizo exactamente: qué, cuándo y cómo.

3.2.2. Modelo de Datos con tablas de auditoría

Para poder realizar una auditoría de la base de datos, tenemos 4 técnicas que nos ayudarán a solucionar nuestro trabajo, estas técnicas se pueden combinar para tener una solución adecuada en nuestro modelado de datos.

1. Auditoría nativa de las bases de datos

Todos los motores de datos actuales (SQL Server, Oracle, Informix, Sybase, etc.) tienen herramientas para realizar la auditoría de la base de datos o la creación de logs de acceso de usuarios a la base de datos. Esta técnica es muy buena, pero ocupa mucho espacio de datos.

2. Añadir campos a una tabla

El modo más simple de auditar una tabla es registrando cada cambio fila por fila; sin embargo, también es la menos recomendable. Se basa en añadir uno o dos campos testigos que registren la fecha en que se ejecuten cambios y el usuario que los efectúe, sin embargo, cambios en cada columna no son registrados ni auditados, solo se registra el último usuario que haya hecho un cambio. Por ejemplo, si Pilar hace un cambio en la columna "Precio" y luego Juan cambia el valor de la columna "Cantidad" solo quedará registrado que Juan hizo un cambio. Este tipo de auditoría es algo ingenuo, pero que puede servir en algunos casos.

3. Triggers y Tablas Espejo

Todos los motores tienen en los triggers (desencadenadores o disparadores) una ayuda para llevar a cabo registros de auditoría. Esta forma tiene el siguiente procedimiento:

- Crear una tabla espejo con los mismos campos que la tabla auditada, pero con campos adicionales, como el usuario que haya hecho el cambio, la fecha y la operación realizada que puede ser fila agregada, fila modificada o fila eliminada.
- Añadir triggers INSERT, UPDATE y DELETE a la tabla origen, de modo que al efectuarse cualquiera de las operaciones indicadas, grabe el mismo registro en la tabla de auditoría, incluyendo los campos de auditoría (usuario, fecha, tipo de operación).

Esta forma de auditoría, si bien es mejor que la anterior, ya que registra cada cambio realizado y almacena un historial completo para cada fila, impone una sobrecarga de operaciones por cada fila, debido a que se vuelve a registrar la misma fila con datos adicionales en caso sea insertada, actualizada o eliminada. Y a la vez aumenta el espacio a ocupar.

4. Creación de tablas de movimiento

Es un modo simple de auditar una tabla, se lleva a cabo registrando cada cambio fila por fila, en una tabla de operación o movimiento esta tabla permitirá reconstruir la información.

Resumen

1. Generar data de auditoría en una base de datos es una ventaja y, a la vez, un problema, ya que acarrea costos en espacio de almacenamiento y tiempo en manejarla. Pero aun así es necesaria en ciertas actividades, especialmente, la financiera. Como recomendación, la auditoría debería realizarse a ciertas tablas de una base de datos y no a todas, ya que complicaría aún más el proceso de administración.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- http://www.michael-richardson.com/processes/rup_classic/#extend.bus_model/disciplines/rup_business_modeling_discipline_553E20F6.html
- https://www.ibm.com/support/knowledgecenter/SSB2MU_8.3.0/com.ibm.rhp.uml.diagrams.doc/topics/rhp_t_dm_creating_rhap_projects.html



DISEÑO DE SISTEMAS

LOGRO DE LA UNIDAD DE APRENDIZAJE

Al término de la unidad, el alumno sustenta su proyecto final relacionado a la elaboración de la documentación del Análisis y Diseño de Sistema de un caso de estudio utilizando una Herramienta CASE.

TEMARIO

4.1 Tema 10 : Diseño de Sistemas

- 4.1.1 : Diseño de sistemas según AUP
- 4.1.2 : Artefactos
- 4.1.3 : Reglas básicas de nomenclaturas
- 4.1.4 : Actividades para identificar los artefactos del modelo de diseño de sistema

4.2 Tema 11 : Desarrollo del Modelo de Diseño de sistemas para un caso

- 4.2.1 : Configuración de perfiles para el Modelo de Diseño de Sistemas - Caso: Venta de Productos de Fashion Importaciones

4.3 Tema 12 : Patrones de Diseño

- 4.3.1 : Importancia de aplicar patrones de diseño
- 4.3.2 : Catálogo de Patrones de Diseño más utilizados en proyectos de sistemas

4.4 Tema 13 : Diseño de la Arquitectura

- 4.4.1 : Diseño de la Arquitectura de sistemas
- 4.4.2 : Diseño de la Arquitectura para el caso Fashion Importaciones

4.5 Tema 14 : Diseño de Casos de Uso

- 4.5.1 : Diseño de Casos de Uso para un dato maestro
- 4.5.2 : Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte I
- 4.5.3 : Diseño de Casos de Uso para un dato transaccional
- 4.5.4 : Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte II

4.1. DISEÑO DE SISTEMAS

En este apartado se describirá cómo se desarrolla el diseño de sistemas según AUP. Además, se mencionarán los artefactos relevantes que se representan así como las reglas básicas de nomenclaturas. Finalmente, se describirá las actividades para identificar los artefactos del modelo de diseño de sistema.

4.1.1. Diseño de sistemas según AUP

Debido a que AUP toma como base el enfoque de RUP, Scott Ambler sugiere que el equipo de desarrolladores no debe realizar todos los artefactos que se definen en la disciplina de diseño de RUP, sino solo los que realmente brinden apoyo durante la tarea de diseño para obtener una solución viable definiendo la tecnología a aplicar en el sistema software.

4.1.2. Artefactos

En la siguiente tabla se muestra la lista de artefactos del Modelo de Diseño con una breve descripción. Estos artefactos serán los que se desarrollarán en el curso.



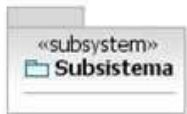


Artefacto	Descripción
 Modelo de Diseño	Representa la vista interna del sistema. El modelo de diseño se convertirá en la materia prima que nuestra disciplina de implementación transformará en código ejecutable.
 Capa	Representan un medio para organizar los artefactos del modelo de diseño. Dependiendo del estilo arquitectónico, una capa agrupa un conjunto de subsistemas junto con sus clases de diseño.
 Subsistema	Un subsistema contiene las clases de diseño de un grupo de casos de uso. correspondencia directa con los paquetes de análisis.
 Librería	Una librería contiene clases utilitarias, adicionales a las del API del lenguaje de Programación, implementadas por el equipo de desarrollo.
 Clases de diseño	Son abstracciones de clase directamente utilizadas en la implementación, es decir, estas clases junto con sus atributos y operaciones se mapean directamente en el lenguaje de programación.

Figura 70: Tabla de artefactos de Diseño – Parte I

Fuente. – Elaboración propia

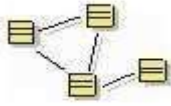
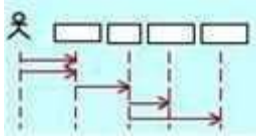
 <p>Diagrama de Clases</p>	<p>El diagrama de clases describe la estructura de un caso de uso. Contiene las clases que participan en el caso de uso, aunque algunas de ellas puedan participar en varios.</p>
 <p>Diagramas de Secuencia</p>	<p>Muestra una secuencia detallada de interacción entre los objetos de diseño. Visualizan el intercambio de mensajes entre objetos. Se crea un diagrama de secuencia por cada flujo de trabajo del caso de uso: flujo básico, subflujos y flujos alternativos.</p>

Figura 71: Tabla de artefactos de Diseño – Parte II

Fuente. – Elaboración propia

4.1.3. Reglas básicas de nomenclaturas

La siguiente tabla muestra los artefactos del modelo de análisis que requiere atender cuantas reglas en su nomenclatura:




Artefacto	Descripción
 <p>Capas, Subsistemas y Librerías</p>	<p>Agrupan las clases de diseño y deben seguir la nomenclatura de paquete tal como se realiza en el código fuente: empezar con minúscula y si el nombre es compuesto, a partir de la segunda palabra, cada palabra empieza con mayúscula.</p>
 <p>HttpServlet</p>	<p>Una clase de diseño sigue la misma nomenclatura de las clases que se implementan en un programa: empiezan con mayúscula y si el nombre es compuesto, a partir de la segunda palabra, cada palabra empieza con mayúscula.</p>
 <p>JavaServerPage</p>	<p>Esta clase de diseño representa ea jsp que se crea en una aplicación web Java. Su nombre puede empezar con minúscula y si el nombre es compuesto, a partir de la segunda palabra, cada palabra empieza con mayúscula o separados con un guión bajo. Ejemplos: jspCita o jsp_cita</p>

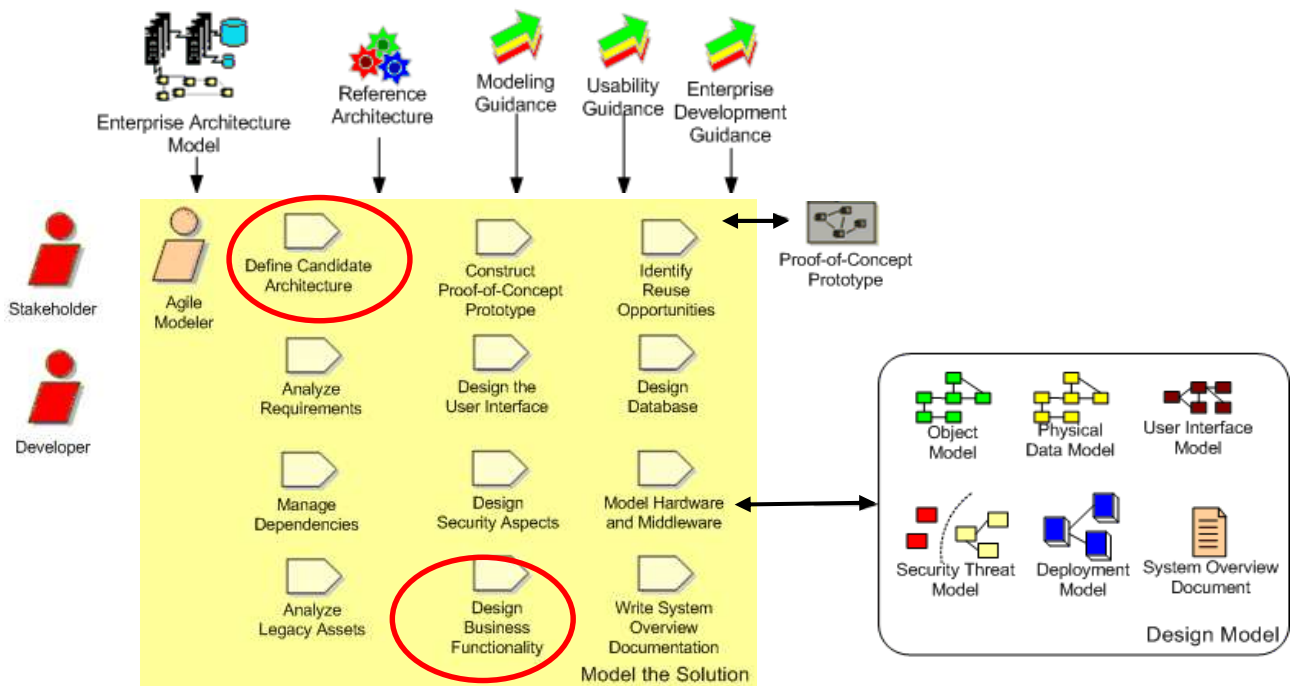
Figura 72: Tabla de nomenclaturas de artefactos de Diseño

Fuente. – Elaboración propia

4.1.4. Actividades para indetificar los artefactos del modelo de diseño de sistemas

La siguiente figura resalta las actividades que se desarrollarán en el curso para elaborar el modelo de diseño:

- **Definir una arquitectura** del sistema donde se realiza el Diseño de la Arquitectura.
- **Diseñar las funcionalidades**, específicamente en esta etapa se realizará el Diseño de los Casos de Uso.



Copyright 2005 Scott W. Ambler

Figura 73: Modelado de la solución – Diseño según AUP

Fuente. - Tomado de <http://www.ambysoft.com/unifiedprocess/aup11/html/model.html>

Resumen

1. El diseño se traduce en el modelo de diseño, el cual es usado para representar la estructura más detallada de las funcionalidades o casos de uso atendiendo tecnologías de implementación o programación.
2. Los artefactos claves para representar la funcionalidad o caso de uso del sistema a nivel de diseño son los siguientes: paquetes de diseño, y clases de diseño.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>

4.2. DESARROLLO DEL MODELO DE DISEÑO DE SISTEMAS PARA UN CASO

En esta sección se configurará el perfil de diseño y se creará el paquete que incluirá el modelo de diseño para el caso de estudio Fashion Importaciones.

4.2.1. Configuración de perfiles para el Modelo de Diseño de Sistemas - Caso: Venta de Productos de Fashion Importaciones

Paso 1: Importe el proyecto FashionImportaciones:

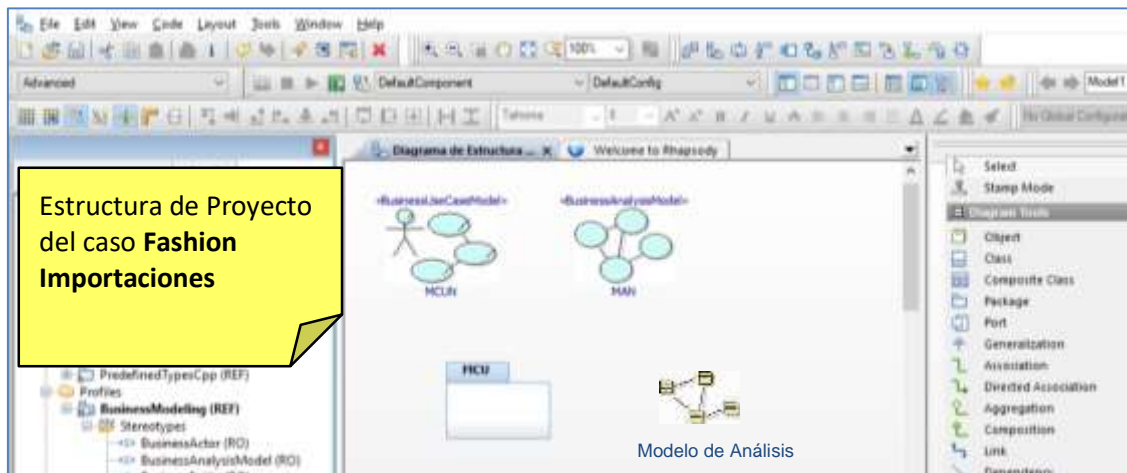


Figura 74: Estructura de Proyecto del caso "Fashion Importaciones".
Elaboración propia

Paso 2: Siga los pasos de la Unidad de Aprendizaje 1 para agregar el perfil de diseño en el proyecto Fashion Importaciones. Luego, cree el paquete del modelo de diseño con su estereotipo correspondiente "designModel". Coloque "MD" en *Name* y "Modelo de Diseño" en *Label*.

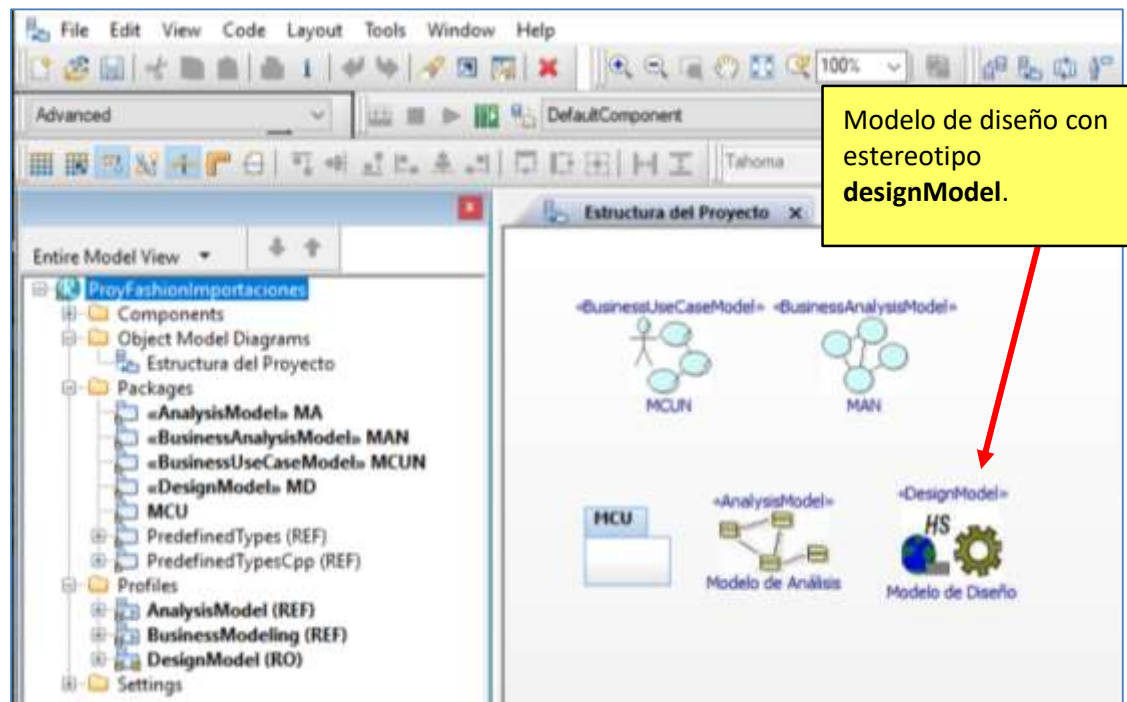


Figura 75: Estructura de Proyecto del caso "Fashion Importaciones" con el modelo de diseño
Elaboración propia

Resumen

1. El diseño se traduce en el modelo de diseño, el cual es usado para representar la estructura más detallada de las funcionalidades o casos de uso atendiendo tecnologías de implementación o programación.
2. Los artefactos claves para representar la funcionalidad o caso de uso del sistema a nivel de diseño son los siguientes: paquetes de diseño, clases de diseño.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>

4.3. PATRONES DE DISEÑO

Como analistas y programadores vamos desarrollando a diario nuestras habilidades para resolver problemas usuales que se presentan en el desarrollo del software. Por cada problema que se nos presenta pensamos distintas formas de resolverlo, incluyendo soluciones exitosas que ya hemos usado anteriormente en problemas similares. Es así como a mayor experiencia que tengamos, nuestro abanico de posibilidades para resolver un problema crece, pero al final siempre habrá una sola solución que mejor se adapte a nuestra aplicación. Si documentamos esta solución, podemos reutilizarla y compartir esa información que hemos aprendido para resolver de la mejor manera un problema específico.

4.3.1. Importancia de aplicar patrones de diseño

Los patrones del diseño tratan los problemas del diseño que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes. Asimismo, son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema, y para describirlo debemos especificar:

- Nombre
- Propósito o finalidad
- Sinónimos (otros nombres por los que puede ser conocido)
- Problema al que es aplicable
- Estructura (diagrama de clases)
- Participantes (responsabilidad de cada clase)
- Colaboraciones (diagrama de interacciones)
- Implementación (consejos, notas y ejemplos)
- Otros patrones con los que está relacionado

4.3.2. Catálogo de Patrones de diseño más utilizados en proyectos de sistemas

El catálogo más famoso de patrones se encuentra en “Design Patterns: Elements of Reusable Object-Oriented Software”, de Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, 1995, Addison-Wesley, también conocido como el libro GOF (Gang-Of-Four), quienes recopilaron 23 patrones que son utilizados por muchos diseñadores.

Patrones de diseño				
Propósito				
Ámbito	Clase	De Creación	Estructurales	De Comportamiento
	Objeto			
		<ul style="list-style-type: none"> • Factory Method 	<ul style="list-style-type: none"> • Adapter (de clases) 	<ul style="list-style-type: none"> • Interpreter • Template Method
		<ul style="list-style-type: none"> • Abstract Factory • Builder • Prototype • Singleton 	<ul style="list-style-type: none"> • Adapter (de objetos) • Bridge • Composite • Decorator • Façade • Flyweight • Proxy 	<ul style="list-style-type: none"> • Chain of Responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

Figura 76: Patrones de diseño GoF

Fuente.- Tomado de <http://ares.cnice.mec.es/informes/21/contenidos/11.htm>.

En la figura anterior, se muestra la clasificación de los patrones GOF considerando dos aspectos: el **propósito** para el que han sido definidos y el **ámbito**.

Según el propósito para el que han sido definidos, existen 3 tipos:

- **Creacionales:** Solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.
- **Estructurales:** Solucionan problemas de composición (agregación) de clases y objetos.
- **De Comportamiento:** Dan soluciones respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan.

Según el ámbito, se clasifica en patrones de clase y de objeto, es así como se cuenta con 6 tipos:

Creacionales

- Creacional de la Clase: Los patrones creacionales de Clases usan la herencia como un mecanismo para lograr la instanciación de la Clase. Por ejemplo, el método Factoría.
- Creacional del objeto: Los patrones creacionales de objetos son más escalables y dinámicos comparados de los patrones creacionales de Clases. Por ejemplo, la Factoría abstracta y el patrón Singleton.

Estructurales

- Estructural de la Clase: Los patrones estructurales de Clases usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo, el patrón Adaptador (Clase).
- Estructural de Objetos: Los patrones estructurales de objetos crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición del patrón estructural del objeto puede ser cambiado en tiempo de ejecución, el cual nos da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite.

Comportamiento

- Comportamiento de Clase: Los patrones de comportamiento de Clases usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo, Interpreter.
- Comportamiento de Objeto: Los patrones de comportamiento de objetos nos permiten analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor.

La siguiente figura muestra la plantilla que utilizan los autores de los patrones GOF para describir un patrón.

<p>Nombre del patrón: Nombre estándar del patrón.</p> <p>Contexto: ¿Cuál es el alcance del patrón?</p> <p>Problema: ¿Qué pretende resolver?</p> <p>Solución</p> <p>Estructura: Diagramas de clases oportunos para describir las clases que intervienen en el patrón.</p> <p>Estrategia: Como funciona.</p> <p>Consecuencias: Consecuencias positivas y negativas en el diseño derivadas de la aplicación del patrón.</p> <p>Patrones relacionados: Referencias cruzadas con otros patrones.</p>

Figura 77: Plantilla de Patrones GoF

Fuente.- Tomado de <http://ares.cnice.mec.es/informes/21/contenidos/11.htm>

Con la aparición del J2EE, todo un nuevo catálogo de patrones de diseño apareció. Desde que J2EE es una arquitectura por sí misma que involucra otras arquitecturas, incluyendo servlets, JavaServer Pages, Enterprise JavaBeans, y más, merece su propio conjunto de patrones específicos para diferentes aplicaciones empresariales.

De acuerdo con el libro "J2EE PATTERNS Best Practices and Design Strategies", existen 5 capas en la arquitectura J2EE:

- **Cliente:** Esta capa corresponde a lo que se encuentra en el computador del cliente. Es la interfaz gráfica del sistema y se encarga de interactuar con el usuario. J2EE tiene soporte para diferentes tipos de clientes incluyendo clientes HTML, applets Java y aplicaciones Java.
- **Presentación:** La capa de presentación es la que ve el usuario, comunica y captura la información proveniente de él.
- **Negocio:** Es donde reside el núcleo del sistema. Se comunica con la capa de presentación para recibir y enviar los datos al usuario.
- **Integración:** La capa de integración es donde residen los datos y es la encargada de acceder a los mismos. Contiene uno o más gestores de base de datos. Recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.
- **Recurso:** Es la capa donde reside la base de datos y sistemas legados.

Nombre	Quiénes la componen	Dónde se ubica
Capa Cliente	Aplicaciones cliente, applets, aplicaciones y otras GUIs	PC Cliente
Capa de Presentación	JSP, Servlet y otras UIs	Servidor J2EE
Capa de Negocios	EJBs y otros objetos de negocios	Servidor J2EE
Capa de Integración	JMS, JDBC	Servidor J2EE
Capa de Recursos	Bases de Datos, Sistemas Legados	Servidor BD

Figura 78: Capas de la Arquitectura J2EE

Fuente.- Tomado de <http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes1.htm>

Existen 15 patrones J2EE que están divididos en 3 de las capas descritas: presentación, negocio e integración. Cabe señalar que todos estos patrones poseen tanto características de diseño como de arquitectura.

A continuación, se presentan los patrones de cada estas 3 capas: presentación, negocio e integración.

Patrones de la Capa de Presentación

Patrón	Descripción
Decorating Filter / Intercepting Filter	Un objeto que está entre el cliente y los componentes Web. Este procesa las peticiones y las respuestas.
Front Controller/ FrontComponent	Un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en 2 diferentes objetos: el Front Controller y el <i>Dispatcher</i> . En ese caso, El Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el <i>Dispatcher</i> direcciona los requerimientos a manejadores apropiada.
View Helper	Un objeto helper que encapsula la lógica de acceso a datos en beneficio de los componentes de la presentación. Por ejemplo, los JavaBeans pueden ser usados como patrón View Helper para las páginas JSP.
Composite view	Un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View.
Service To Worker	El Controlador actúa como Front Controller, pero con un factor importante: aquí el Dispatcher (el cual es parte del Front Controller) usa View Helpers a gran escala y ayuda en el manejo de la vista.
Dispatcher View	Es con el controlador actuando como Front Controller, pero con un asunto importante: aquí el Dispatcher (el cual es parte del Front Controller) no usa View Helpers y realiza muy poco trabajo en el manejo de la vista. El manejo de la vista es manejado por los mismos componentes de la Vista.

Figura 79: Capa Presentación J2EE

Fuente.- Tomado de <http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes1.htm>

Patrones de la Capa de Negocio

Patrón	Descripción
Business Delegate	Un objeto que reside en la capa de presentación y en beneficio de los otros componentes de la capa de presentación llama a métodos remotos en los objetos de la capa de negocios.
Value Object/ Data Transfer Object/ Replicate Object	Un objeto serializable para la transferencia de datos sobre la red.
Session Façade/ Session Entity Façade/ Distributed Façade	El uso de un bean de sesión como una fachada (facade) para encapsular la complejidad de las interacciones entre los objetos de negocio y participantes en un flujo de trabajo. El Session Façade maneja los objetos de negocio y proporciona un servicio de acceso uniforme a los clientes.
Aggregate Entity	Un bean entidad que es construido o es agregado a otros beans de entidad.
Value Object Assembler	Un objeto que reside en la capa de negocios y crea Value Objects cuando es requerido.
Value List Handler/ Page-by-Page Iterator/ Paged List	Es un objeto que maneja la ejecución de consultas SQL, caché y procesamiento del resultado. Usualmente implementado como beans de sesión.
Service Locator	Consiste en utilizar un objeto Service Locator para abstraer toda la utilización JNDI y para ocultar las complejidades de la creación del contexto inicial, de búsqueda de objetos home EJB y recreación de objetos EJB. Varios clientes pueden reutilizar el objeto Service Locator para reducir la complejidad del código, proporcionando un punto de control.

Figura 80: Capa Negocio J2EE

Fuente.- Tomado de <http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes1.htm>

Patrones de la Capa de Integración

Patrón	Descripción
Data Access Object	Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.
Service Activator	Se utiliza para recibir peticiones y mensajes asíncronos de los clientes. Cuando se recibe un mensaje, el Service Activator localiza e invoca a los métodos de los componentes de negocio necesarios para cumplir la petición de forma asíncrona.

Figura 81: Capa Integración J2EE

Fuente.- Tomado de <http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes1.htm>

Resumen

1. Los patrones del diseño tratan los problemas del diseño que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes. Asimismo, son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema.
2. Los patrones de diseño mayormente utilizados en proyectos de desarrollo de software son los siguientes: del catálogo de patrones GoF, patrones J2EE, .

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>
- <http://ares.cnice.mec.es/informes/21/contenidos/11.htm>
- <http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes1.htm>

4.4. DISEÑO DE LA ARQUITECTURA

En esta sección se explicará sobre los patrones arquitectónicos que se considerarán realizar en los proyectos del curso.

4.4.1. Diseño de la Arquitectura de sistemas

A continuación, se mencionan las capas arquitectónicas que se diseñaran para crear los componentes internos de las realizaciones de los casos de uso.


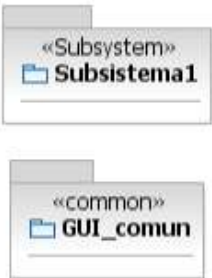

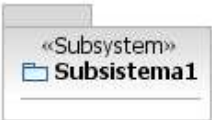

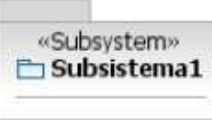

Capa	Subsistema/Librerías	Componentes
		<p>Clases estereotipadas:</p> <ul style="list-style-type: none"> • Páginas HTML: <<Client Page>> y <<HTML Form>> • Páginas JSP: <<Server Page>>, <<Client Page>> y <<HTML Form>>
		<p>Clase estereotipada para servlets: <<Http Servlet>></p>
		<p>Clases de diseño: <i>beans</i>.</p>
		<p>Clases de diseño: clases utilitarias.</p>

Figura 82: Capas, subsistemas, librerías y elementos de diseño
Elaboración propia


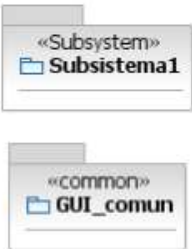

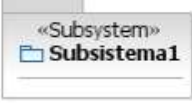

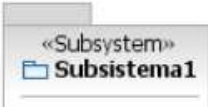


Capa	Subsistema/Librerías	Componentes
		<p>Clases estereotipadas:</p> <ul style="list-style-type: none"> • Páginas HTML: <<Client Page>> y <<HTML Form>> • Páginas JSP: <<Server Page>>, <<Client Page>> y <<HTML Form>>
		<p>Clase estereotipada para servlets: <<Http Servlet>></p>
		<ul style="list-style-type: none"> • Clases de diseño: servicios, <i>beans</i> y clases DAO. • Interfaces que presentan las operaciones de acceso a una tabla.
		<p>Clases de diseño: clase abstracta <i>DAOFactory</i> y sus clases hijas.</p>
		<p>Clases de diseño: clases utilitarias.</p>

Figura 83: Capas, subsistemas, librerías y elementos de diseño según patrón arquitectónico MVC y patrón DAO
Elaboración propia

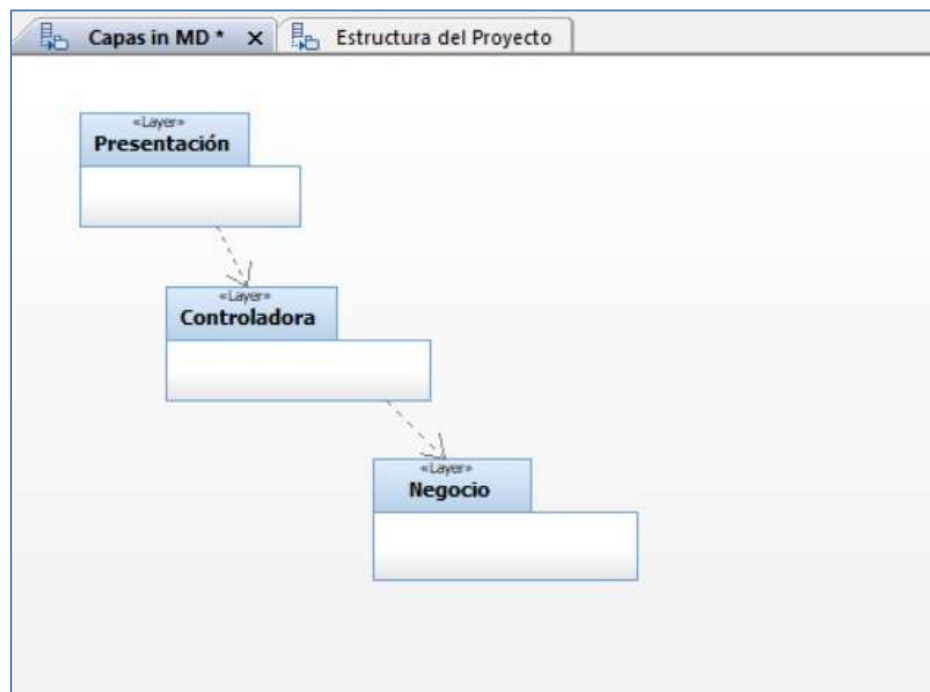


Figura 84: Capas arquitectónicas y sus dependencias
Elaboración propia

4.4.2. Diseño de la Arquitectura para el caso Fashion Importaciones

Ahora, usted debe crear los paquetes que representan la capa arquitectónica en el modelo de diseño para el caso Fashion Importaciones. Para ello, cree las capas del patrón MVC: presentación, negocio y controladora.

PASO 1: Cree el diagrama de modelo de objetos “Capas”, desde el Browser de Proyecto. Para ello, clic derecho sobre el modelo de Diseño (MD) y seleccione **Add/Diagram.../Object Model Diagrams**.

PASO 2: Cree los 3 paquetes desde el Browser de Proyecto. Para ello, clic derecho sobre el modelo de Diseño (MD) y seleccione **Add Package**. Luego, arrastre los 3 paquetes al Diagrama de “Capas”.

PASO 3: Desde el diagrama “Capas” seleccione los 3 paquetes y asigñarle el estereotipo “Layer” del perfil **DesignModel**.

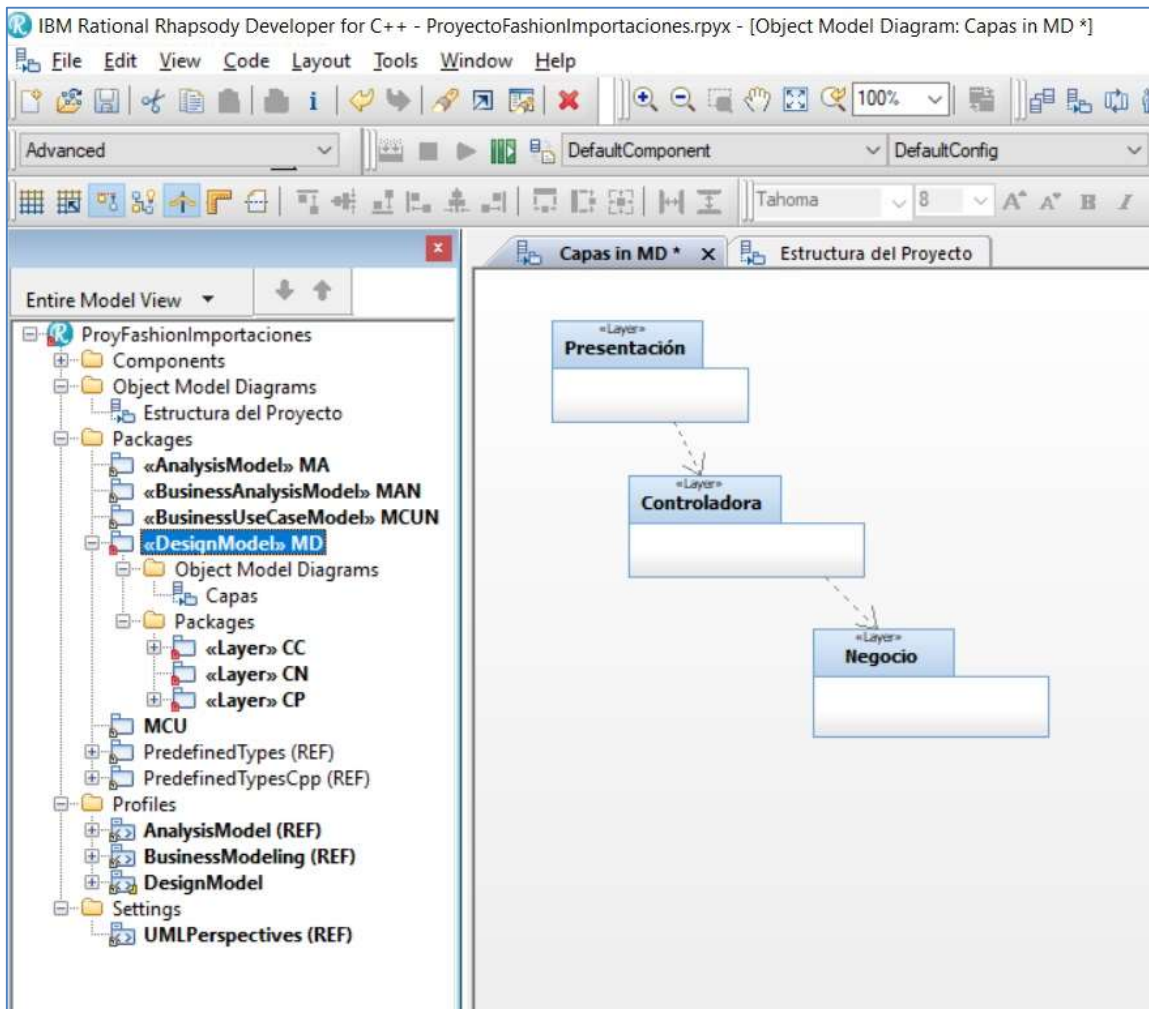


Figura 85: Capas arquitectónicas y sus dependencias
Elaboración propia

Resumen

1. En el diseño de la arquitectura se consideran patrones. En el curso utilizaremos la organización de los componentes con los patrones arquitectónicos MVC (Model View Controller) y los componentes de diseño de las capas de DAO (Data Access Objet).

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>

4.5. DISEÑO DE CASOS DE USO

En la siguiente tabla, se muestra la organización de las clases de diseño e interfaces en capas, subsistemas y librerías que utilizaremos según patrón arquitectónico MVC y patrón DAO:




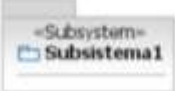




Capa	Subsistema/Librerías	Elementos de diseño
 «Layer» Presentación	 «Subsystem» Subsistema1	Clases estereotipadas: <ul style="list-style-type: none"> • Páginas HTML: «Client Page» • Páginas JSP: «Server Page», «Client Page» y «HTML Form»
 «Layer» Controladora	 «Subsystem» Subsistema1	Clase estereotipada para servlets: «Http Servlet»
 «Layer» Negocio	 «Subsystem» Subsistema1	<ul style="list-style-type: none"> • Clases de diseño: servicios, <i>beans</i> y clases DAO. • Interfaces que presentan las operaciones de acceso a una tabla.
	 «Common» factorias	Clases de diseño: clase abstracta <i>DAOFactory</i> y sus clases hijas.
	 «Library» util	Clases de diseño: clases utilitarias.

Figura 86: Capas, subsistemas, librerías y elementos de diseño

[Elaboración propia](#)

Para las realizaciones de diseño de un caso de uso se crearán diagramas de clases, y de secuencia. A continuación se describirán con detalle estos diagramas y sus elementos.

Diagrama de clases: Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

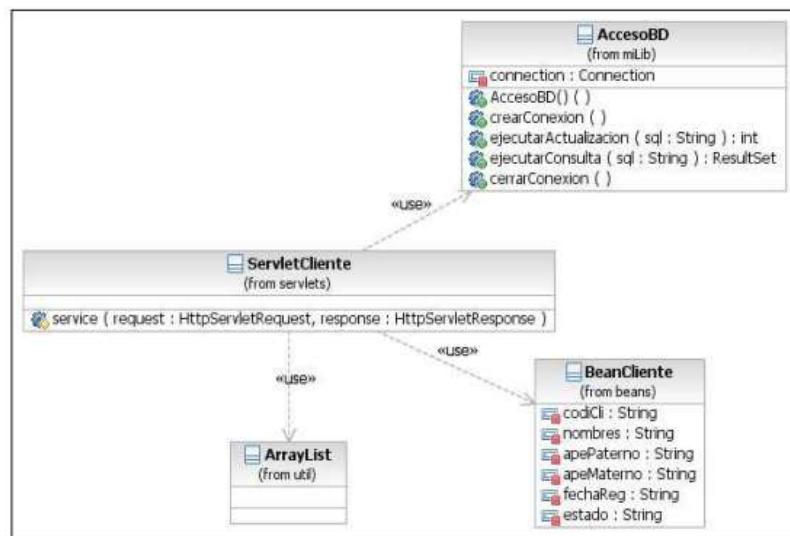


Figura 87: Clases de diseño y sus relaciones de dependencia estereotipadas

[Elaboración propia](#)

Tipos de relaciones entre clases de diseño: En las siguientes tablas se muestran las relaciones que pueden existir entre clases. La descripción de cada una permitirá entender la estructura de clases diseñada para una funcionalidad que será implementada en JAVA:




Tipo de relación	UML	Java
Herencia		<pre>public class ClaseA { //Más código } public class ClaseB extends ClaseA { //Más código }</pre>
Implementación	 	<pre>public interface InterfazX { //Más código } public class ClaseY implements InterfazX { //Más código }</pre>

Figura 88: La relación de herencia e implementación

[Elaboración propia](#)



Tipo de dependencia	UML	Descripción
<<use>> (De uso)		El funcionamiento del origen depende del funcionamiento del destino.
<<instantiate>> (De instancia)		El origen solo crea instancias del destino.

Figura 89: La relación de dependencia

[Elaboración propia](#)

Diagramas de secuencia: Describe la dinámica del sistema, describiendo las interacciones entre un grupo de objetos mostrando de forma secuencial los envíos de mensajes entre objetos. El diagrama puede asimismo mostrar los flujos de datos intercambiados durante el envío de mensajes.

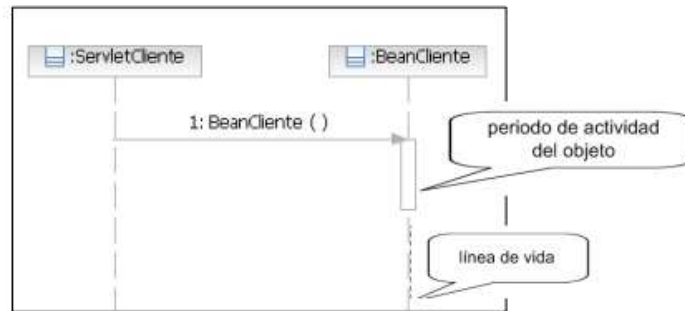


Figura 90: Diagrama de secuencia con mensaje sincrónico
Elaboración propia

Línea de vida de un objeto: Dado que representa la dinámica del sistema, el diagrama de secuencia hace entrar en acción las instancias de clases que intervienen en la realización de la subfunción a la que está vinculado. A cada instancia se asocia una línea de vida que muestra las acciones y reacciones de la misma, así como los periodos durante los cuales ésta está activa, es decir, durante los que ejecuta uno de sus métodos.

Mensajes: Para interactuar entre sí, los objetos se envían mensajes. Durante la recepción de un mensaje, los objetos se vuelven activos y ejecutan el método del mismo nombre. Un envío de mensaje es, por tanto, una llamada a un método y se representan mediante flechas horizontales que unen la línea de vida del objeto emisor con la línea de vida del objeto destinatario. Existen diferentes tipos de mensajes:

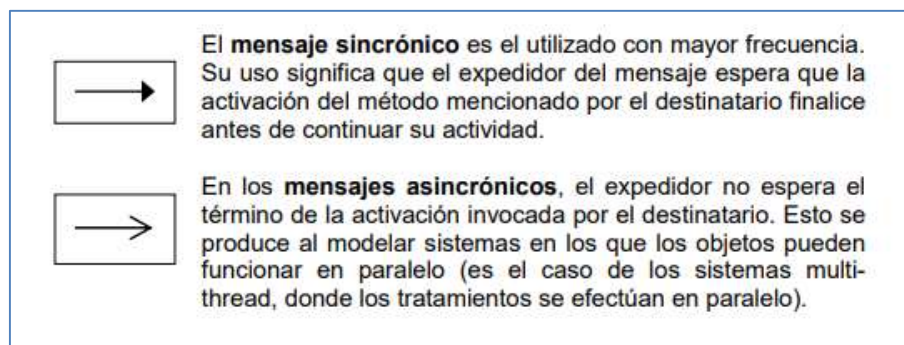


Figura 91: Tipos de mensajes entre instancias de una clase
Elaboración propia

Fragmentos combinados: Para un diagrama de secuencia que representa procedimientos complejos hay un número de mecanismos que permiten agregar un grado de lógicas de procedimientos a los diagramas y que a la vez vienen bajo el encabezado de fragmentos combinados. Un fragmento combinado es una o más secuencias de procesos incluidas en un marco y ejecutadas bajo circunstancias nombradas específicas. Los fragmentos disponibles son los siguientes:

1. El fragmento **Alternative** (denotado "alt") modela estructuras if...else.
2. El fragmento **Option** (denotado "opt") modela estructuras switch.
3. El fragmento **Break** modela una secuencia alternativa de eventos que se procesa en lugar de todo del resto del diagrama.
4. El fragmento **Parallel** (denotado "par") modela procesos concurrentes.
5. El fragmento de secuenciado **Weak** (denotado "seq") incluye un número de secuencias para las cuales todos los mensajes se deben procesar en un segmento anterior, antes de que el siguiente segmento pueda comenzar, pero que no impone ningún secuenciado en los mensajes que no comparten una línea de vida.
6. El fragmento de secuenciado **Strict** (denotado "strict") incluye una serie de mensajes que se deben procesar en el orden proporcionado.
7. El fragmento **Negative** (denotado "neg") incluye una serie de mensajes inválidos.
8. El fragmento **Critical** incluye una sección crítica.
9. El fragmento **Ignore** declara un mensaje o mensajes que no son de ningún interés si este aparece en el contexto actual.
10. El fragmento **Consider** es el opuesto del fragmento **Ignore**: cualquier mensaje que no se incluya en el fragmento **Consider** se debería ignorar.
11. El fragmento **Assertion** (denotado "assert") designa que cualquier secuencia que no se muestra como un operando de la aserción es inválida.
12. El fragmento **Loop** incluye una serie de mensajes que están repetidos.

Figura 92: Tipos de fragmentos combinados
Elaboración propia

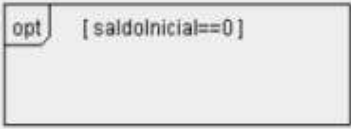
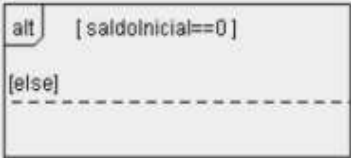
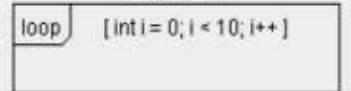

Fragmento OPT: Estructura de control alternativa 'if' que no tiene secuencia 'else'.	
Condición	Resultado
if (saldoInicial == 0) System.out.println("Sin fondos");	
Fragmento ALT: Estructura de control alternativa 'if-else'.	
Condición	Resultado
if (saldoInicial == 0) System.out.println("Sin fondos"); else System.out.println("Con fondos");	
Fragmento LOOP: Estructuras de repetición o bucles 'for' y 'while'.	
Condición	Resultado
for(int i = 0; i < 10; i++) { System.out.println("contador."+i); }	
while(i < 10) { System.out.println("contador."+i); i++; }	

Figura 93: Fragmentos combinados más utilizados con su correspondiente estructura de control en Java
Elaboración propia

4.5.1. Diseño de Casos de Uso para un dato maestro

De la Especificación del caso de uso que manipula un dato maestro, se crea los diagramas de la realización de diseño del caso de uso.

Por ejemplo, los diagramas que describen la funcionalidad de un mantenimiento son los siguientes:

- Diagrama de clases de diseño
- Diagramas de secuencia por cada flujo descrito en la especificación:
 - Diagrama de secuencia para el flujo básico
 - Diagrama de secuencia para el subflujo agregar
 - Diagrama de secuencia para el subflujo actualizar
 - Diagrama de secuencia para el subflujo desactivar

Además, es opcional, realizar los diagramas de secuencia de cada una de las operaciones que implementan la lógica de la aplicación, tales como: agregar, actualizar y desactivar.

A continuación, se muestra la ECU del CU Mantener Cajeros y la realización del CU, la cual contiene el diagrama de clases y el diagrama de secuencia del flujo básico.

Importante: Tener en cuenta que la creación de que los paquetes y las clases de diseño se crean dentro de las capas arquitectónicas:

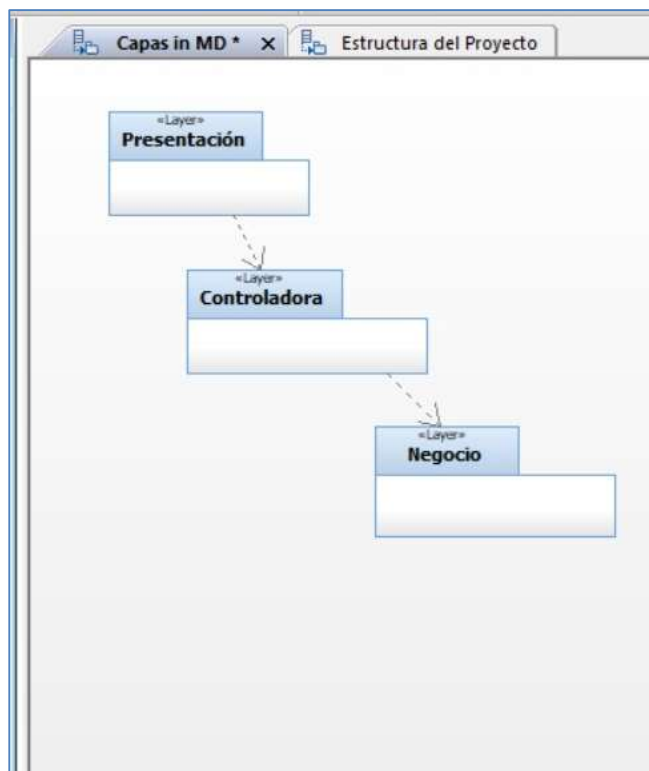


Figura 94: Capas de diseño
Elaboración propia

ESPECIFICACIÓN DE CASO DE USO: Mantener Cajero**1. Descripción**

El caso de uso permite mantener actualizado el registro de los cajeros de la clínica. De acuerdo a su necesidad, el Administrador de la Clínica puede agregar, actualizar y desactivar un cajero.

2. Actor(es)

Administrador.

3. Flujo de Eventos**3.1. Flujo Básico**

1. El caso de uso se inicia cuando el Administrador selecciona la opción "Cajeros" en la interfaz del menú principal.
2. El sistema muestra la interfaz "MANTENER CAJERO" con la lista de cajeros con los campos: código, nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro, fecha de actualización y estado. Además, muestra las opciones: **Agregar Cajero, Actualizar Cajero y Desactivar Cajero.**
3. Si el Administrador elige un cajero
 - a. Si elige "Actualizar" ver el Subflujo Actualizar Cajero.
 - b. Si elige "Desactivar" ver el Subflujo Desactivar Cajero.
4. Si el Administrador NO elige un cajero
 - a. **Si** elige "Agregar" ver el Subflujo Agregar Cajero.
5. El Administrador selecciona "Salir" y el caso de uso finaliza.

3.2. Subflujos**3.2.1. Agregar Cajero**

1. El sistema muestra la interfaz CAJERO con los siguientes campos: código (**solo** lectura), nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro (**sólo** lectura) y fecha de actualización (**solo** lectura). Además, muestra las opciones: **Aceptar y Cancelar.**
2. El Administrador ingresa los datos del Cajero.
3. El Administrador selecciona la opción Aceptar.
4. El sistema valida los datos ingresados.
5. El sistema genera un nuevo código de cajero y obtiene la fecha del sistema para la fecha de registro y la fecha de actualización
6. El sistema graba un nuevo registro de cajero y muestra el MSG "Cajero creado con código Nro. 999999".
7. El Administrador cierra la interfaz CAJERO y regresa a la interfaz MANTENER CAJERO con la lista de cajeros actualizada y el subflujo finaliza.

3.2.2. Actualizar Cajero

1. El sistema muestra los datos del cajero seleccionada en la interfaz CAJERO: código (**sólo** lectura), nombres, apellido paterno, apellido materno, teléfono, correo, dirección, fecha de registro (**sólo** lectura) y

fecha de actualización (solo lectura). Además muestra las opciones:
Aceptar y Cancelar.

2. El Administrador actualiza los datos del cajero.
3. El Administrador selecciona la opción Aceptar.
4. El sistema valida los datos ingresados del cajero.
5. El sistema obtiene la fecha del sistema para la fecha de actualización, actualiza el registro de cajero y muestra el MSG "Cajero actualizado satisfactoriamente".
6. El Administrador cierra la interfaz CAJERO y regresa a la interfaz MANTENER CAJERO con la lista de cajeros actualizada y el subflujo finaliza.

3.2.3. Desactivar Cajero

1. El sistema muestra el MSG: "¿Está seguro que desea desactivar el(los) cajero(s) seleccionado(s)?".
2. El Administrador selecciona la opción YES para confirmar la desactivación.
3. El sistema actualiza el registro del(los) cajero(s) en estado "Desactivado".
4. El sistema muestra la interfaz MANTENER CAJERO con la lista de cajeros actualizada y termina el subflujo.

3.3. Flujos Alternativos

1. Datos del Cajero Inválidos

Si los datos ingresados son nulos o inválidos, tanto en los subflujos Agregar como en Actualizar Cajero, el sistema muestra el MSG: "Se han encontrado datos inválidos" y los subflujos continúan en el paso 2.

2. Cajero ya existe

Si el sistema detecta que el cajero ya existe en el paso 4 del subflujo Agregar Cajero, muestra el MSG: "Cajero ya existe" y el subflujo finaliza.

3. No confirma Desactivación

Si el Administrador selecciona NO en el paso 2 del subflujo Desactivar Cajero, finaliza el subflujo.

4. Precondiciones

1. El Administrador está identificado en el sistema.
2. Lista disponible de Cajeros.

5. Poscondiciones

1. En el sistema quedará registrado el nuevo Cajero.
2. En el sistema quedará actualizado el registro del Cajero.
3. En el sistema quedará desactivado el Cajero.

6. Puntos de Extensión

Ninguno.

7. Requisitos Especiales

Ninguno.

Diagrama de clases de Diseño del CU Mantener Cajeros

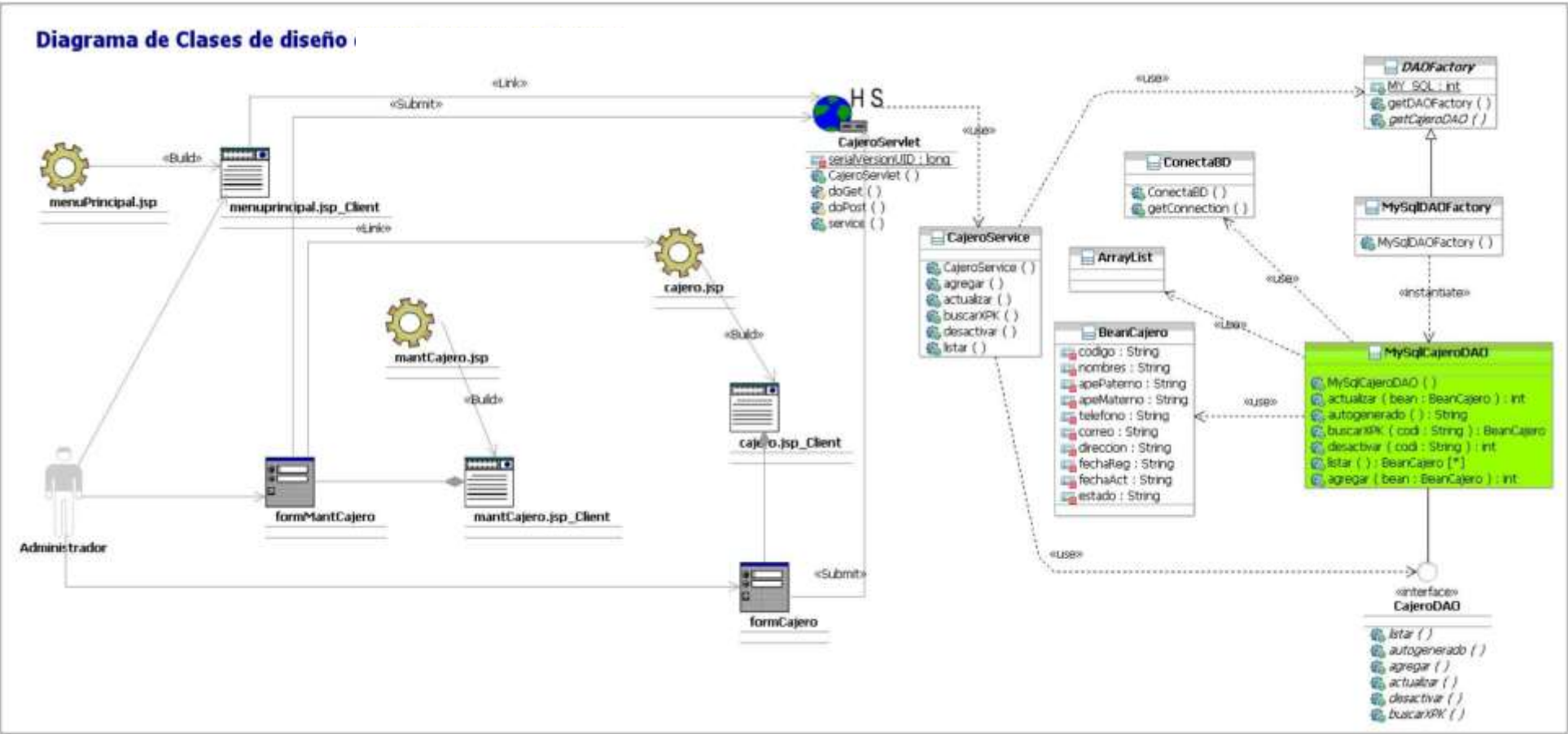


Figura 95: Diagrama de clases de diseño
Elaboración propia

Diagrama de secuencia de Diseño del flujo básico del CU Mantener Cajeros

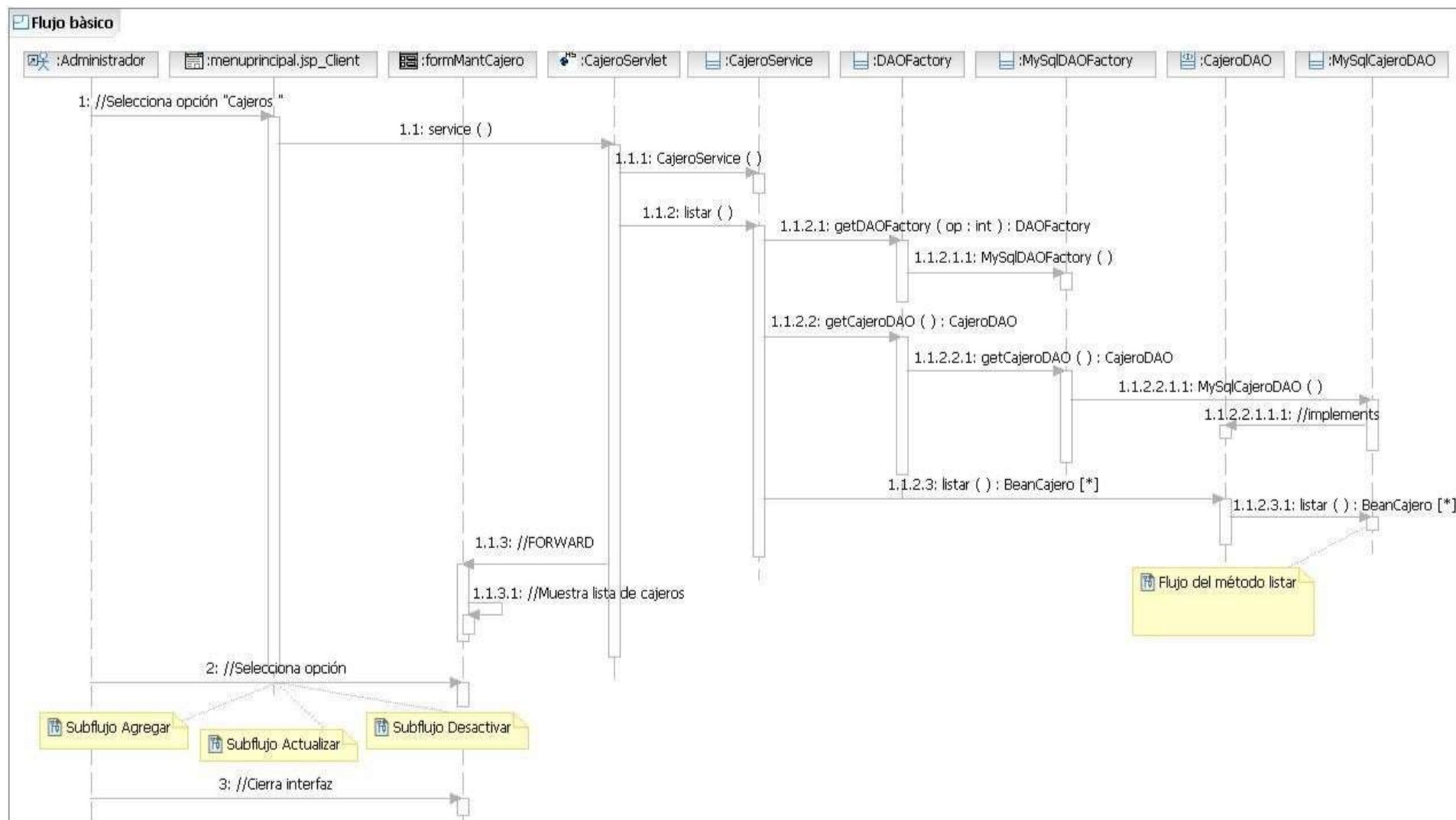


Figura 96: Diagrama de secuencia de diseño del flujo básico

Elaboración propia

4.5.2. Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte I

Ahora, usted desarrollará la solución del diseño del caso de uso “Mantener Productos” del caso Fashion Importaciones (página 36), el cual manipula un dato maestro. Para ello, confeccione los siguientes diagramas:

- Diagrama de clases
- Diagramas de secuencia por cada flujo descrito en la especificación:
 - Diagrama de secuencia para el flujo básico
 - Diagrama de secuencia para el subflujo agregar
 - Diagrama de secuencia para el subflujo actualizar
 - Diagrama de secuencia para el subflujo desactivar
 - Diagrama de secuencia de sus operaciones: agregar, actualizar y desactivar

4.5.3. Diseño de Casos de Uso para un dato transaccional

De la Especificación del caso de uso que manipula un dato transaccional, se crea los diagramas de la realización de diseño del caso de uso.

Los diagramas que describen la funcionalidad serán los siguientes:

- Diagrama de clases de diseño
- Diagramas de secuencia por cada flujo descrito en la especificación
- Diagramas de secuencia para sus operaciones de la lógica de la aplicación

A continuación, se muestra la ECU del CU Generar Cita y la realización del CU, la cual contiene el diagrama de clases de diseño, el diagrama de secuencia del flujo básico y de algunas de sus operaciones.

Especificación de caso de uso: Generar Cita

1. Descripción:

El caso de uso permite a la recepcionista de la clínica, registrar una cita médica para consultas externas por especialidad.

2. Actor(es)

Recepcionista

3. Flujo de Eventos

3.1. Flujo Básico

13. El caso de uso comienza cuando la recepcionista selecciona la opción "Generar Citas" de la interfaz del menú principal.
14. El sistema muestra la interfaz "GENERAR CITAS" con la fecha y hora de registro cargado y los siguientes campos:
 - Datos de la HC: número, nombre y apellidos del paciente.
 - Datos del médico: nombres, apellidos y especialidad.
 - Datos de la cita : fecha y hora de la cita y consultorio.
 - Además, presenta las opciones: Buscar HC, Buscar Horarios de Médico y Grabar Cita.
15. La recepcionista selecciona "Buscar HC".
16. El sistema **incluye el caso de uso Buscar Historia Clínica**.
17. El sistema muestra los datos de la historia clínica del paciente.
18. La recepcionista selecciona "Buscar Médico".
19. El sistema **incluye el caso de uso Buscar Horarios de Médico**.
20. El sistema muestra los datos del médico y de la cita.
21. La recepcionista selecciona "Grabar Cita".
22. El sistema valida los datos.
23. El sistema genera el número de cita y registra la cita con estado pendiente.
24. El sistema muestra el mensaje "Cita generada". El caso de uso termina.

3.2. Flujos Alternativos

1. No existe HC

Si en el paso 6 el sistema detecta que no existe la HC del paciente, muestra el MSG "No existe HC" y ofrece la posibilidad de registrar la HC del paciente.

2. No hay médicos disponibles

Si en el paso 8 el sistema detecta que no hay médicos disponibles, muestra el MSG "No hay médicos disponibles" y el caso de uso finaliza.

3. Campos vacíos

Si en el paso 10 el sistema detecta que alguno de los campos está vacío, muestra el MSG "Alguno de los campos está vacío" y el caso de uso continúa.

12. Precondiciones

4. La recepcionista está identificada en el sistema.
5. Lista disponible de historias clínica.
6. Lista disponible de médicos.

13. Poscondiciones

2. En el sistema quedará registrada la cita en estado pendiente.

14. Puntos de Extensión

En el paso 6, el sistema extiende al caso de uso Mantener HC – subflujo "Registrar HC".

15. Requisitos Especiales

Ninguno.

16. Prototipos

Generar Citas

Fecha: 08/03/2010
Hora: 15:32

Datos de la HC del paciente

Nº HC:

Paciente:

Datos del médico

Nombre y apellidos:

Especialidad:

Datos de la cita

Fecha:

Hora:

Consultorio:

Figura 97: Prototipo del CU Generar Cita
Elaboración propia

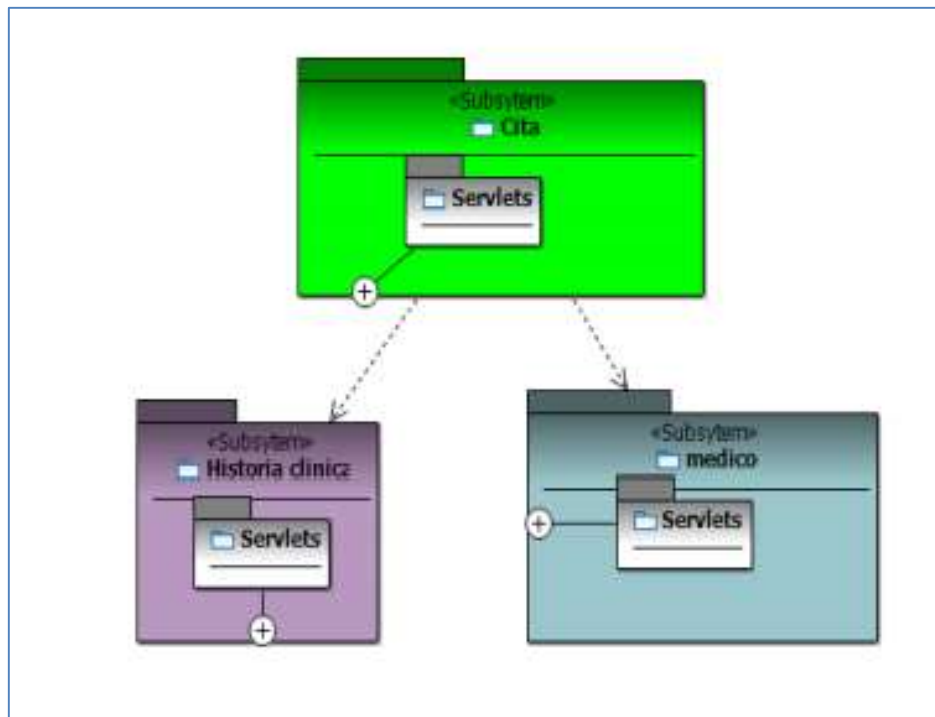


Figura 98: Capa Presentación
Elaboración propia

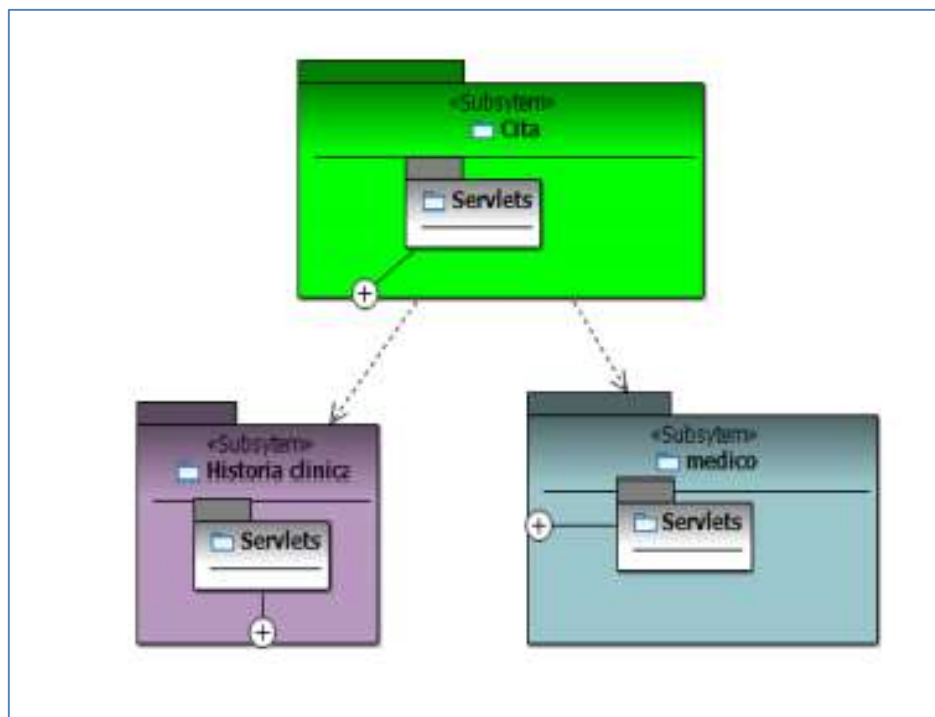


Figura 99: Capa Controladora
Elaboración propia

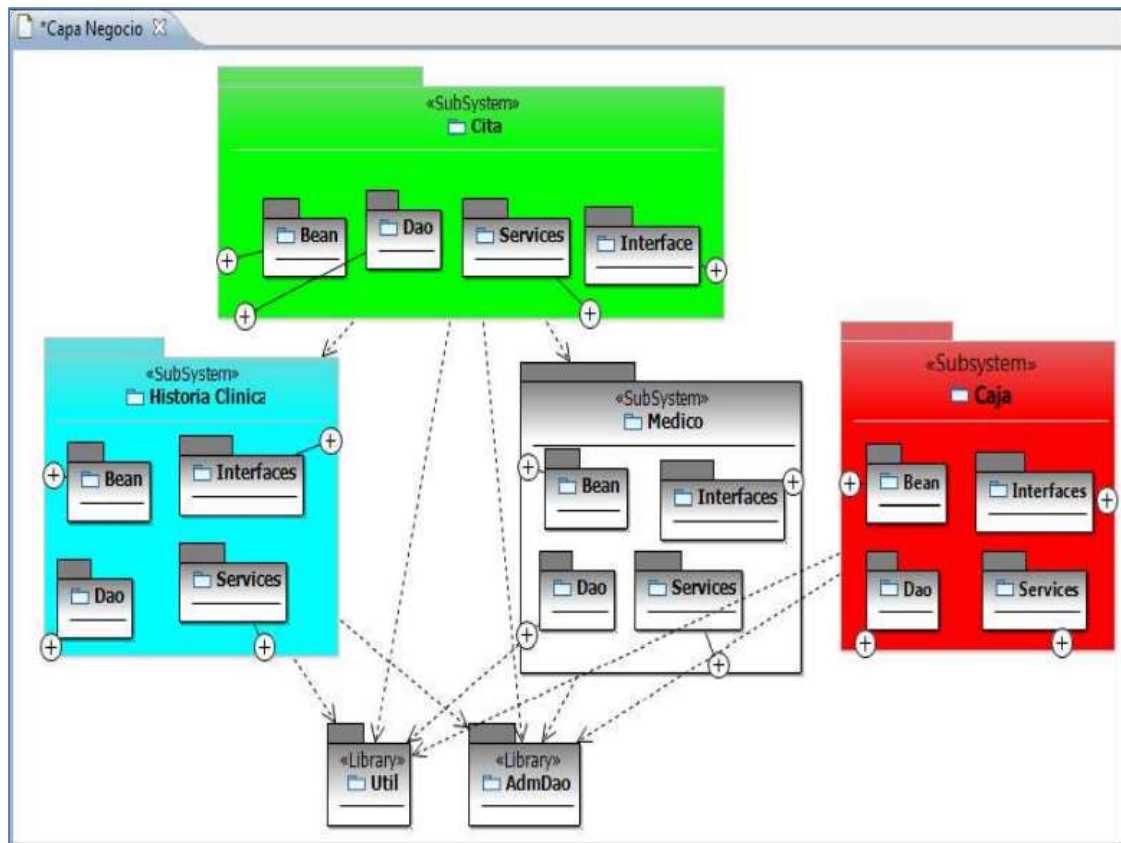
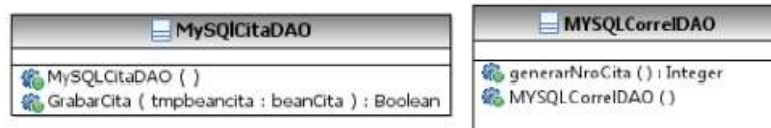


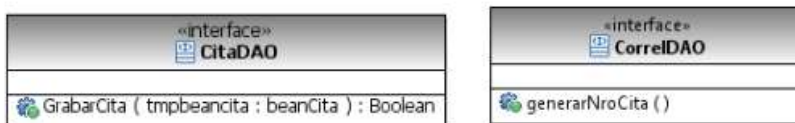
Figura 100: Capa Negocio

Elaboración propia

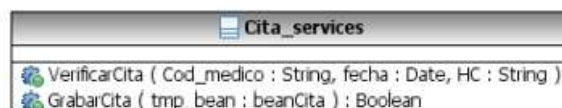
Crear las clases DAO MySQLCitaDAO y MySQLCorrelDAO y BeanCita



Crear las clases Interfaces CitaDAO y CorrelDAO



Crear la clase cita_services



Agregar los métodos al DAOFactory

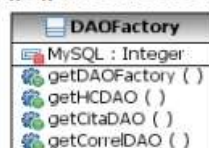


Figura 101: Clases de Diseño de Generar Cita

Elaboración propia

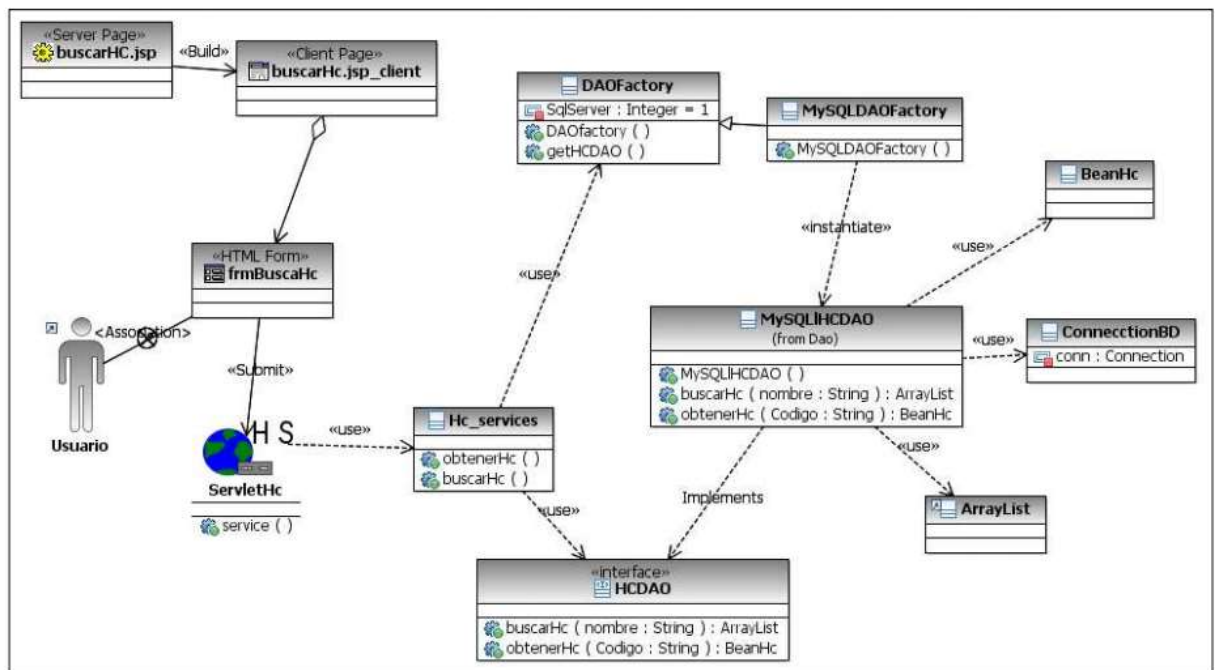


Figura 102:Diagrama de Clases de Diseño del CU Generar Cita
Elaboración propia

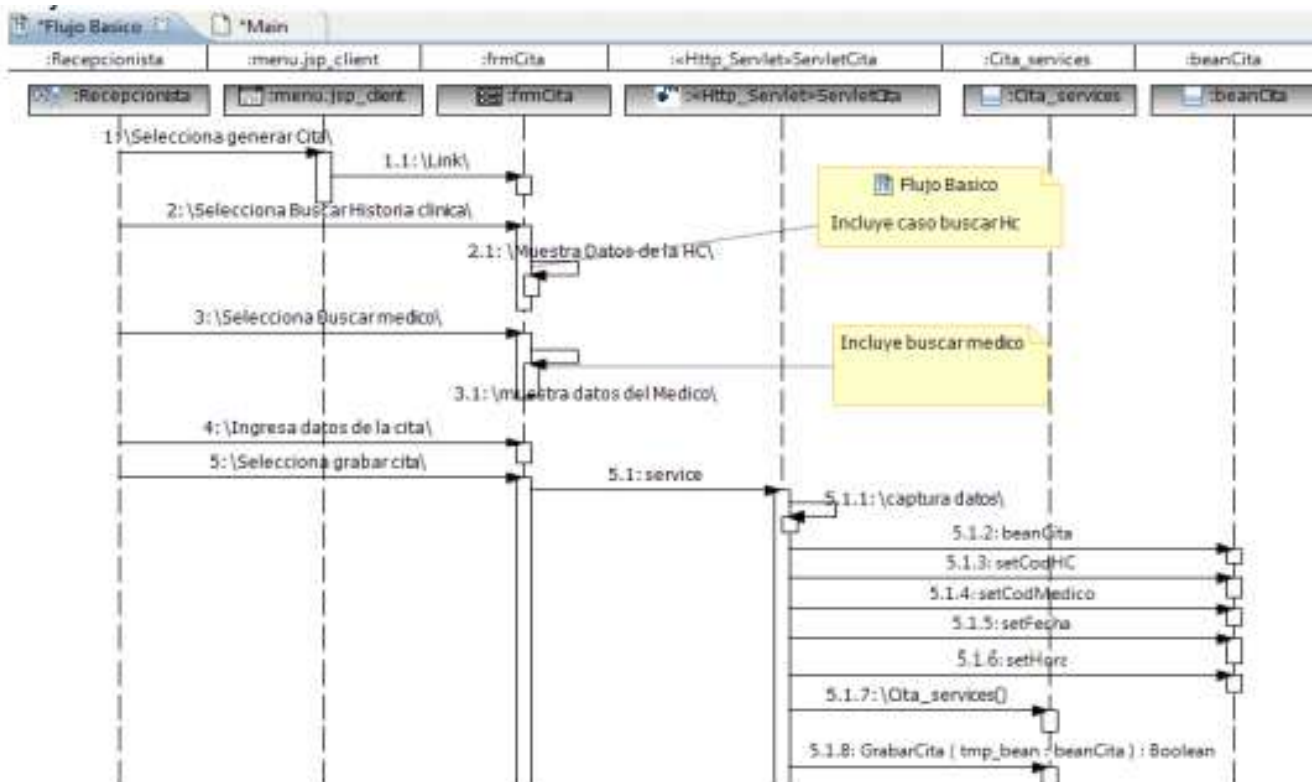


Figura 103: Diagrama de secuencia del flujo básico del CU Generar Cita – Parte I
Elaboración propia

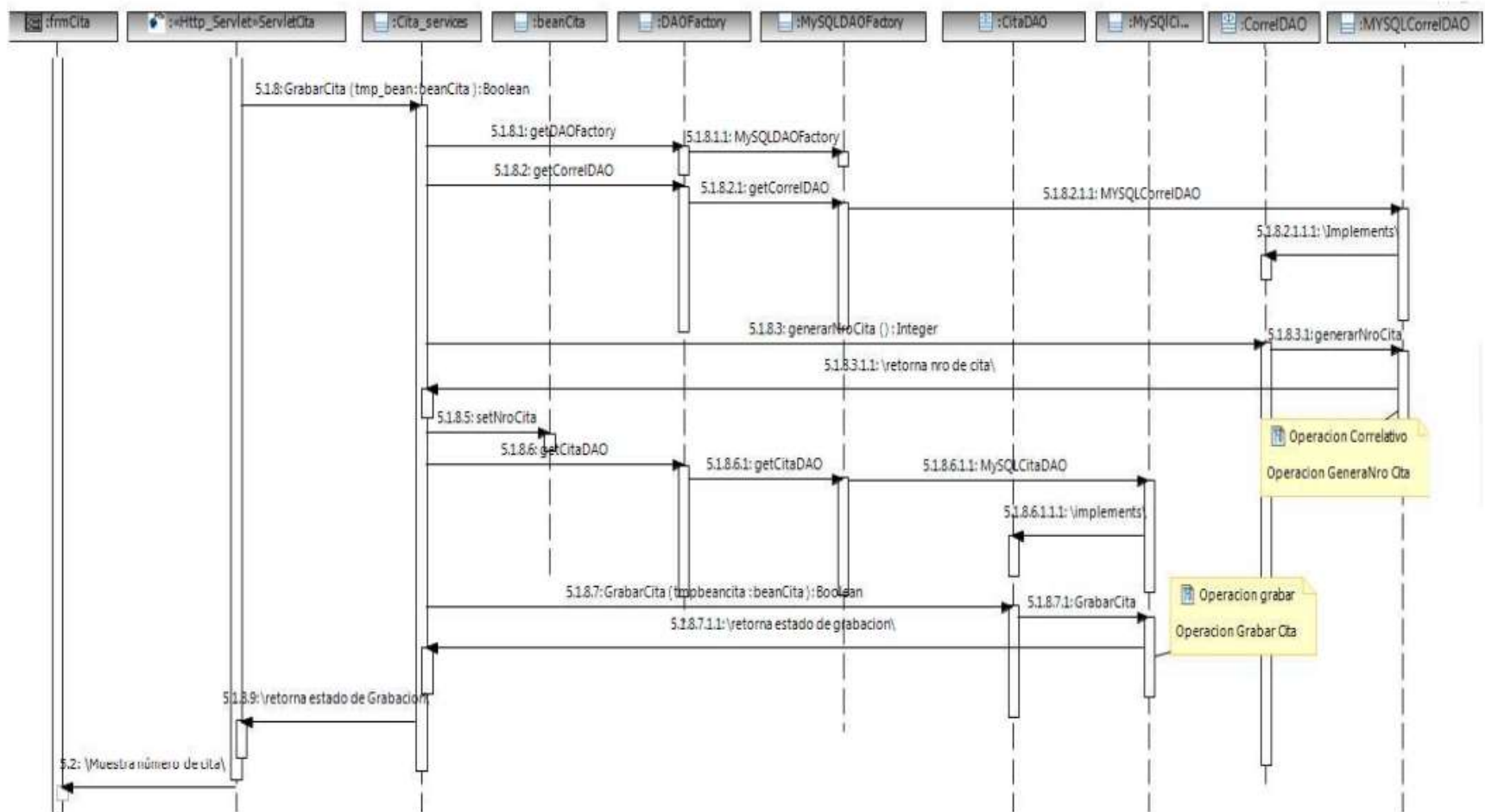
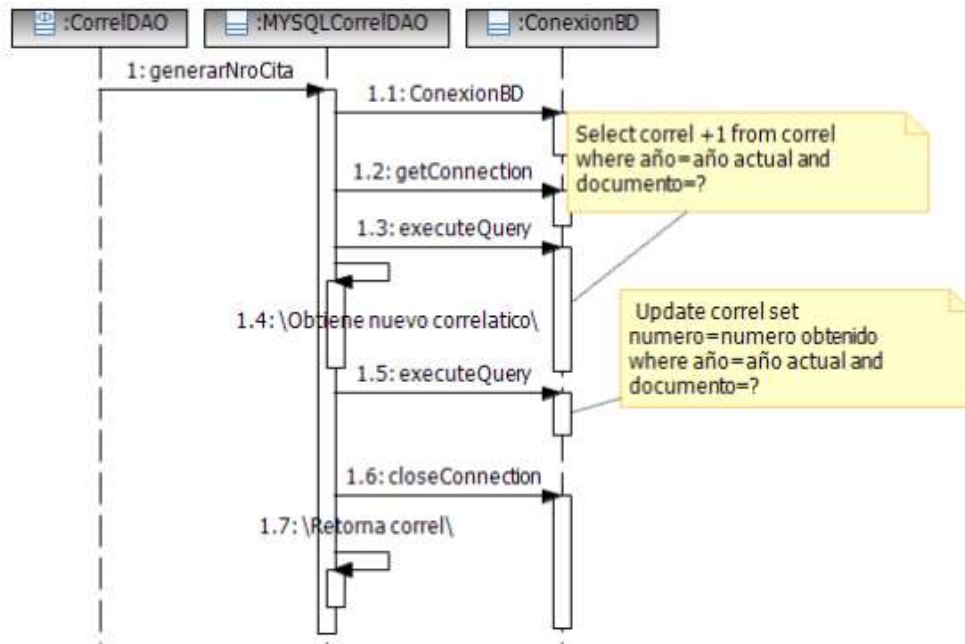


Figura 104: Diagrama de secuencia del flujo básico del CU Generar Cita – Parte II

Elaboración propia

Operación generaNroCita



Operación grabar

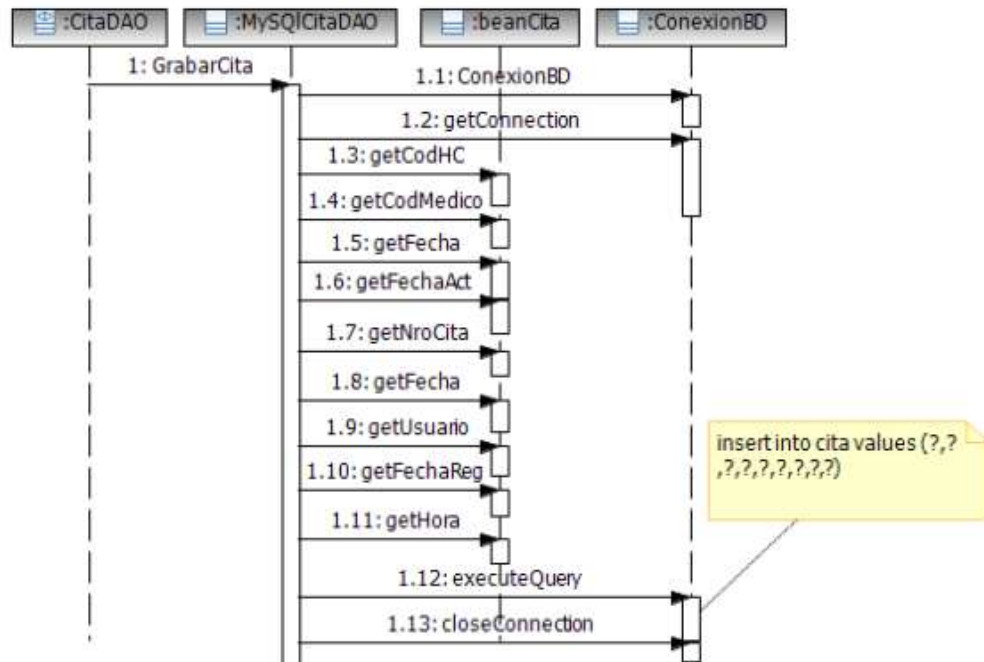


Figura 105: Diagrama de secuencia de operaciones del flujo básico del CU Generar Cita
Elaboración propia

Especificación de caso de uso: Buscar Historia Clínica

1. Descripción:

El caso de uso permite buscar la historia clínica de un paciente por número de historia clínica o nombres y/o apellidos.

2. Actor(es)

Enfermera.
Recepcionista.

3. Flujo de Eventos

3.1. Flujo Básico

1. El caso de uso comienza cuando es invocado por otro caso de uso base.
2. El sistema muestra la interfaz "BUSCAR HISTORIA CLINICA" con los campos: número de historia clínica, nombres, apellido paterno y apellido materno, y una lista con los datos del resultado de consulta: número de historia clínica, nombres, apellido paterno y apellido materno del paciente. Además, incluye las opciones: **Buscar**, **Aceptar** y **Cancelar**.
3. El actor ingresa el criterio de búsqueda (número de historia clínica o nombres y/o apellidos).
4. El actor selecciona "Buscar".
5. El sistema muestra la relación de historias clínica de pacientes que coinciden con el criterio de búsqueda.
6. El actor selecciona una Historia Clínica.
7. El actor selecciona "Aceptar".
8. El sistema carga los datos en la interfaz del caso de uso base que lo invocó y finaliza el caso de uso.

3.2. Flujos Alternativos

1. Clientes no encontrados

En el paso 5, si el sistema no muestra ninguna Historia Clínica por el criterio de búsqueda muestra el MSG: "No se encuentra la historia clínica para el criterio ingresado". El caso de uso continúa en el paso 3 o si el actor selecciona "Cancelar", finaliza el caso de uso.

4. Pre Condiciones

1. El actor se ha identificado en el sistema.
2. Lista de historias clínicas disponibles.

5. Post Condiciones

Ninguna.

6. Puntos de Extensión

Ninguno.

7. Requisitos Especiales

Ninguno.

8. Prototipos

Buscar Historia Clínica

Ingrese aquí los datos de la historia clínica de un paciente que desea buscar.

Criterios de búsqueda

Nota: Para buscar historias clínicas ingresando solo un fragmento de su número, nombres y/o apellidos use el carácter "%".

Nº HC :	<input type="text"/>	Nombre :	<input type="text"/>
Apellido paterno :	<input type="text"/>	Apellido materno :	<input type="text"/>

Buscar

Figura 106: Prototipo Buscar Historia Clínica
Elaboración propia

Diagrama de clases de diseño

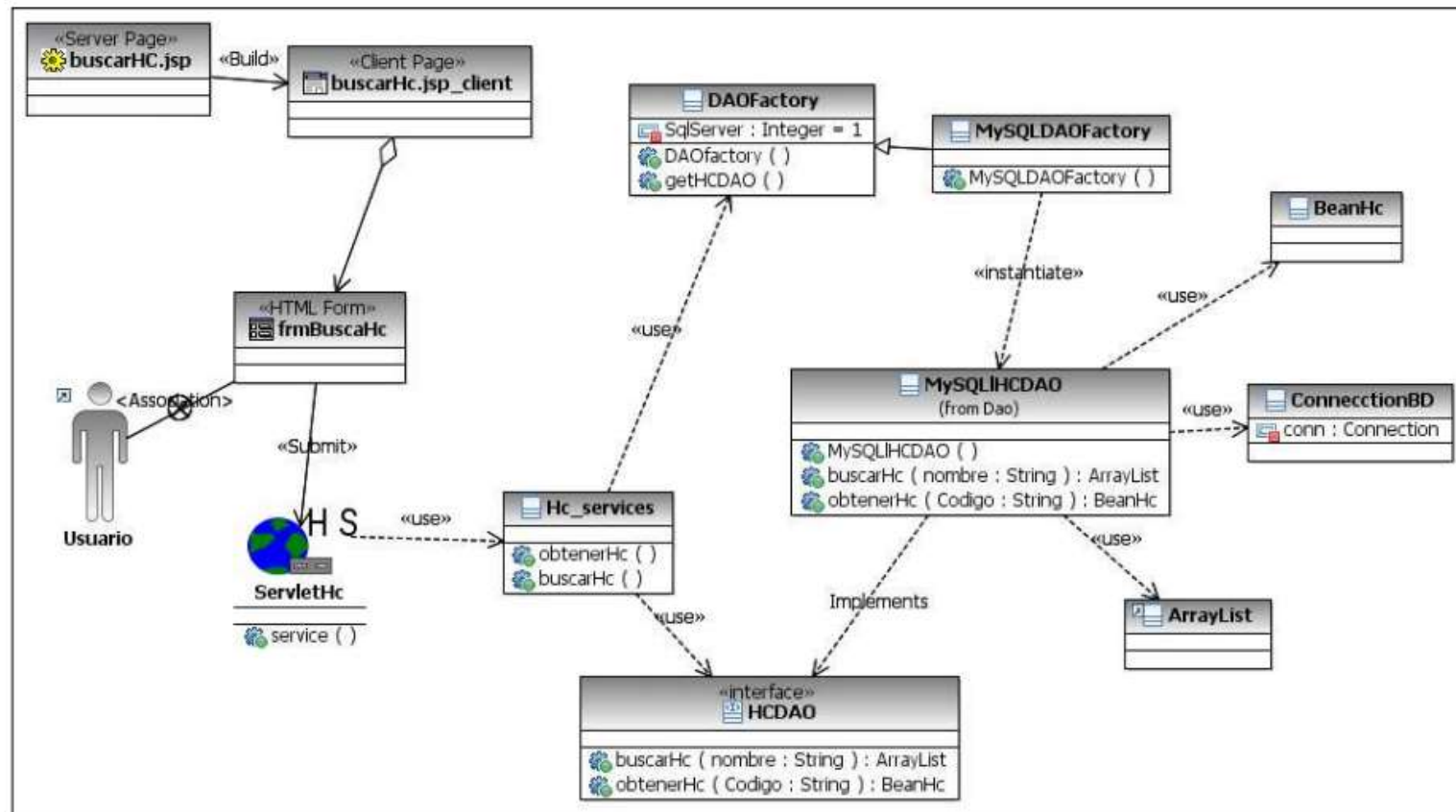


Figura 107: Diagrama de Clases de Diseño del CU Buscar Historia Clínica
Elaboración propia

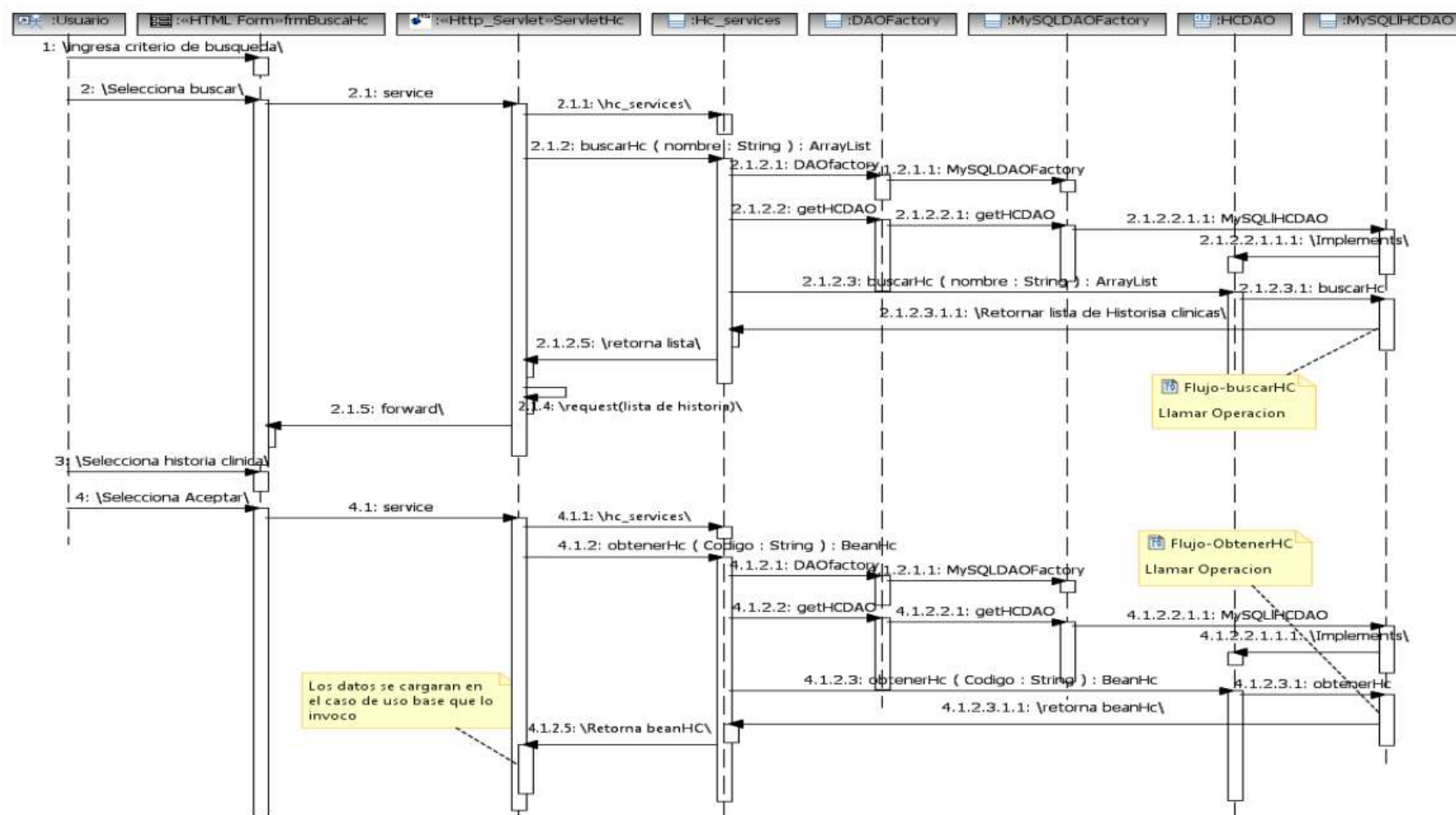
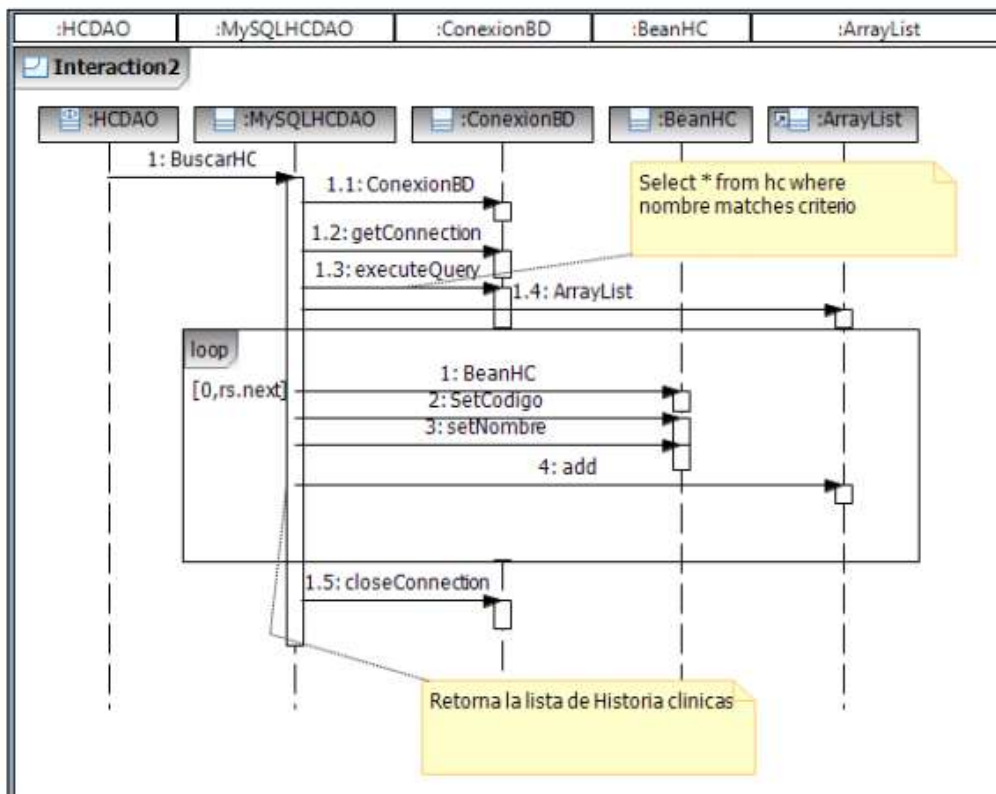


Figura 108:Diagrama de secuencia del flujo básico del CU Buscar Historia Clínica

Elaboración propia

:Paso 14 Flujo Buscar Buscar_HC



:Paso 14 Flujo Buscar Obtener_HC

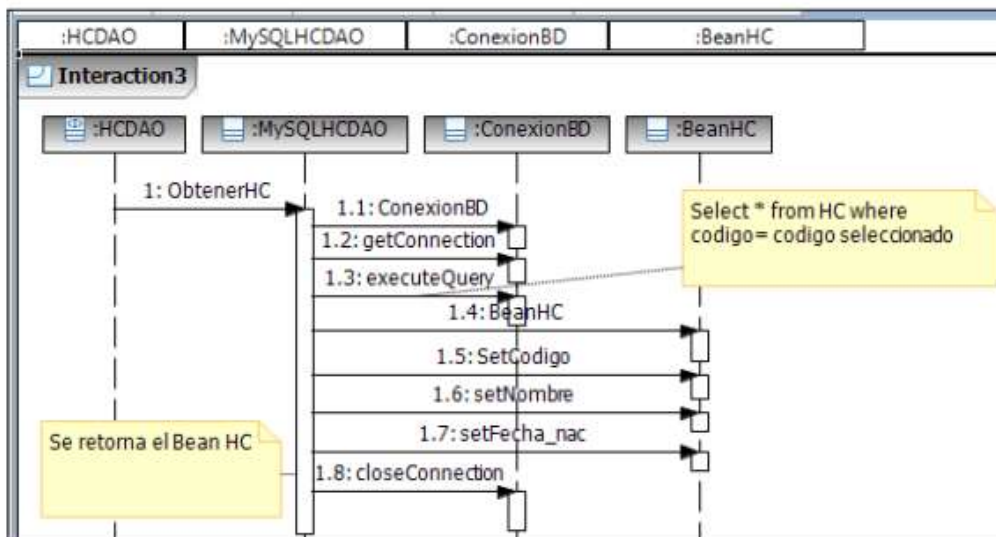


Figura 109:Diagrama de secuencia del flujo de las operaciones del CU Buscar Historia Clínica

Elaboración propia

4.5.4. Diseño de los Casos de Uso para el caso Fashion Importaciones – Parte II

Ahora, usted desarrollará la solución del diseño del caso de uso “Evaluar Pedidos” del caso Fashion Importaciones (página 38), el cual manipula un dato transaccional. Para ello, confeccione los siguientes diagramas:

- Diagrama de clases de diseño
- Diagramas de secuencia por cada flujo descrito en la especificación
- Diagramas de secuencia por de algunas de sus operaciones

Resumen

1. De la Especificación del caso de uso, se crea los diagramas de la realización de diseño del caso de uso. Los diagramas que describen la funcionalidad a través de las realizaciones de diseño son los siguientes:

Diagrama de clases de diseño

Diagramas de secuencia por cada flujo descrito en la especificación:

Además, es opcional, realizar los diagramas de secuencia de cada una de las operaciones que implementan la lógica de la aplicación.

2. Los artefactos claves para representar la funcionalidad o caso de uso del sistema a nivel de diseño son los siguientes: paquetes de diseño, clases de diseño.

Recursos

Pueden revisar los siguientes enlaces para ampliar los conceptos vistos en esta unidad:

- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=overview-uml-design-in-rational-rhapsody>
- <https://www.ibm.com/docs/en/rhapsody/8.4.0?topic=examining-rational-rhapsody-samples>

Bibliografía

- Ambler, Scott W. (13 de mayo de 2006) The Agile Unified Process (AUP). Recuperado el 15 de noviembre de 2020 de www.ambysoft.com/unifiedprocess/agileUP.html
- Báez Pérez, Carmen Inés (2014) *Proceso de desarrollo de software: basado en la articulación de RUP y CMMI priorizando su calidad*. Tunja: Universidad de Boyacá. Centro de Información: Código 005.14 BAEZ.
- Debrauwer, Laurent (2016) *UML 2.5: iniciación, ejemplos y ejercicios corregidos*. 4a ed. Barcelona: ENI. Centro de Información: Código 005.117 DEBR 2016
- Kendall, Kenneth E. y Kendall, Julie E. (2011) *Análisis y Diseño de Sistemas*. 8a ed. México: Pearson Educación. Centro de Información: Código 004.2 KEND 2011.
- Lasa Gómez, Carmen (2017) *Metodologías ágiles: Scrum, Kanban, Lean*. Madrid: Anaya Multimedia. Centro de Información: Código 005.1 LASA 2017.
- Pressman, Roger S. (2010) *Ingeniería del Software: un enfoque práctico*. 3a ed. México, D.F.: McGraw-Hill. Centro de Información: Código 005.1 PRES 2010.
- Sommerville, Ian. (2011) *Ingeniería de Software*. 9a ed. México: Pearson Educación. Centro de Información: Código 005.1 SOMM/ES 2011.
- Watson, Andrew (23 de marzo de 2016) *Whitepaper - Visual Modeling: past, present and future*. Recuperado el 15 de noviembre de 2020 de <https://www.uml.org/news.htm>
- Manrique Grisales, J. (14 de noviembre de 2010) *La bestia que se tragó Armero*. *Diario El Espectador*, pp. 16-17.