UNIVERSIDAD JOSÉ CARLOS MARIÁTEGUI

VICERRECTORADO DE INVESTIGACIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

TRABAJO DE SUFICIENCIA PROFESIONAL

ANÁLISIS Y DISEÑO DE SISTEMAS ORIENTADO A OBJETOS PARA UNA BIBLIOTECA

PRESENTADO POR

BACHILLER JOAO ALFONSO MALDONADO VÁSQUEZ

ASESOR:

DR. IVAN VLADIMIR PINO TELLERIA

PARA OPTAR TÍTULO PROFESIONAL DE

INGENIERO DE SISTEMAS E INFORMÁTICA

MOQUEGUA – PERÚ

2017

CONTENIDO

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

RESUMEN

La presente investigación mantiene una teoría respecto al análisis y al diseño que puede ser tratado según el desempeño de cada Ingeniero especialista en cuanto a los sistemas, pero para el presente estudio es que trata del análisis en cuanto al funcionamiento sistemático de una biblioteca; para lo cual se ha recogido la información de diferentes fuentes , tanto primarias como secundarias, a fin de dar al interesado las condiciones para que pueda analizar y diseñar un prototipo de un sistema. El trabajo alberga un concepto del proceso y también del desarrollo del lenguaje unificado en cuanto a su modelación (UML), sin embargo, también son los factores físicos y lógicos que inciden en el dimensionamiento, así como en la creación de un prototipo de un sistema, pero para el presente estudio es enfocados al diseño de una biblioteca virtual que tiene como objetivo el control de libros, revistas, y demás otros sucedáneos que tengan semejanza a una biblioteca. Consecuentemente de la parte teórica, es que se plantea un caso práctico y seguidamente se realiza el diseño mencionado para lo cual se irá observando de cómo es que el análisis y procedimiento de un sistema comienza de forma paulatina tomando forma y consecuentemente plasmar el flujo de información, así como las actividades y eventos que ocurren en el desarrollo de este sistema.

Palabras clave: análisis, diagramas, UML, dimensionamiento.

ABSTRACT

The present research maintains a theory about the analysis and design that can be treated about the performance of each specialist Engineer in terms of systems, but for the present study is that it is an analysis in the systematic functioning of a library; to obtain detailed information about the different sources, both primary and secondary sources, and the purpose thereof. The work is based on the concept of the process and also on the development of the unified language in terms of its modeling (UML), however, it is also the physical and logical factors that influence the sizing, as well as the creation of a prototype of a system what is this? study focused on the design of a virtual library that aims to control books, magazines, and others that happen to be similar to a library. As a result of the theoretical part, it becomes a practical case and then a design is made. , as well as the activities and events that occur in the development of this system.

Keywords: analysis, diagrams, UML, dimensioning.

CAPÍTULO I

INTRODUCCIÓN

En el mundo de hoy el conocimiento y especialmente la tecnología va avanzando de manera más agresiva día a día, habiendo muchos cambios en ésta y originando que el uso de la tecnología sea un punto crítico para la mejora de una pequeña, mediana o gran empresa. Uno de estos campos es la realización de sistemas, que facilitan la vida en general de las personas ya que los pasos a seguir ya están establecidos y la factibilidad, al igual como la producción, se ven afectados de manera positiva al momento de la implementación de los sistemas de información.

En este contexto es que existe una escasez de documentación, normalmente los usuarios solo requieren el sistema mas no la parte analítica ya que no son conscientes de que este material es un punto crítico al momento de realizar un sistema, esta vendría a ser nuestro traductor para entender las funciones que realiza el sistema.

Para poder plasmar el funcionamiento de un programa de forma conceptual y mediante esquemas, es que utilizaremos el UML (Lenguaje Unificado de Modelado - Unified Modeling Language). De esta manera mostraremos diagramas que permitirán explicar cómo es que el sistema va a ser desarrollado en cada una de sus actividades, los pasos que seguirá, la lógica de negocio que usará y las personas que formarán parte de su uso.

CAPÍTULO II

OBJETIVOS

2.1. Objetivo general

Explicar cómo se analiza y diseña un sistema orientado a objetos, hacer entender la importancia de la documentación en el desarrollo de cualquier sistema sea pequeño o grande, y cómo es que ésta se aplica en el desarrollo de un sistema, en este caso el de una biblioteca.

2.2. Objetivos específicos

Entender cómo es que se aplica el análisis y diseño de sistemas a un caso real.

Observar cómo se analiza un caso práctico de uno o varios procesos en específico.

CAPÍTULO III

DESARROLLO DEL TEMA

3.1. Marco teórico

3.1.1. Metodología orientada a objetos

La metodología encaminada a objetos ha derivado de un conjunto de metodologías, pero que son anteriores a ésta. Así también los métodos en lo que respecta a los diseños estructurados que ya son realizados, en la cual estos sirven de guía a los especialista que tratan de desarrollar sistemas muy complejos usando los algoritmos como sus estratos, pero que son fundamentales en su construcción, así también los métodos de encaminados a objetos y que con el trascurrir del tiempo han evolucionado a fin de incentivar a los expertos en explotar los lenguajes de programación que son basados en objetos y encaminados a objetos, usando las clases y objetos como estratos de construcción básicos (Larman, 2005).

Hoy en día en cuanto al modelado de objetos es impactado por un conjunto de factores y no únicamente de lo que es programación orientado a Objetos, POO (Object Oriented Programming, OOP, siglas en inglés). Además, el modelado de objetos es un concepto uniforme en lo que respecta a las ciencias de la computación y que son empleados no únicamente en su formalización respecto a su programación, es que también se emplea al diseño de interfaces en sus usuarios, las bases de datos y también la arquitectura de computadoras de forma completa. El fundamento es simplemente, que un lenguaje de programación que es orientado a objetos sirve de apoyo y también hacerle a la inherente y fuerte complejidad que existe en muchos y variados sistemas (Larman, 2005).

Ahora bien, un objeto es definido como "un organismo, pero que es tangible y que además nos enseña algún comportamiento que es muy definido". Por tanto, un objeto es una cosa, real, pero también abstracta, por medio del cual guardamos datos y además existen una diversidad de métodos que las controlan " (Larman, 2005).

Los objetos tienen en cierta media "probidad" y es que no debiera de ser violada. Por tanto un objeto solo cambia el estado, la conducta, y el estar en contacto con otros objetos de forma arraigada a este objeto.

En los tiempos actuales, el Análisis que es Orientado a Objetos (AOO) va desarrollándose como formas en sus análisis de requerimiento por derecho propio

así como el complemento de también otros métodos en cuanto a sus análisis. En lugar de analizar un caso especial mediante el empleo de un modelo clásico en el cual se observan que tienen entrada, el proceso y la salida o también podría ser empleando un modelo que viene de estructuras que son jerárquicas de información, en cambio el AOO no únicamente trabaja con uno, sino que trabaja con varios conceptos nuevos y que son inusuales a muchos usuarios, sin embargo, se dice que son muy naturales.

Una condición, representa una plantilla en múltiples clases de objetos, con tendencias y elementos muy similares. Sin embargo, dichas clases contemplan tales características de un todo que único de objetos. Por tanto, cuando se formula dicha programación, pero en lenguaje orientado a objetos, no representan objetos reales, ya que las clases de objetos son las que se definen.

En un momento de una clase es otro término para otro objeto real. Si tal clase representa un objeto, en una instancia y concretamente es su representación. A menudo se usa indiferentemente el termino objeto para referirse única y precisamente, a un objeto (Larman, 2005).

En los lenguajes orientados a objetos, cada clase está compuesta de dos cualidades: atributos (estado) y métodos (comportamiento o conducta). Los atributos son las características individuales que diferencian a un objeto de otro (ambos de la misma clase) y determinan la apariencia, estado u otras cualidades de

ese objeto. Los atributos de un objeto incluyen información sobre su estado (Larman, 2005).

3.1.2. Modelado de sistemas

3.1.2.1. ¿Qué es el modelado?

La constante y compleja sistematización cada vez en cuanto a los sistemas de información, esta ha tenido enormes avances y hasta ha llegado a tal nivel que aquellas de mayor extensión se puede comparar, tanto en su dificultad y tamaño con los grandes avances de otras esferas de la Ingeniera o también la arquitectura (Booch, Rumbaugh y Jacobson, 1999).

Toda la complejidad en los sistemas informáticos, tiene dos temas importantes y fundamentales, la primera es que es muy tediosa formular y procesar un sistema de mayor capacidad y fundamentalmente si es que no se cuenta con la debida experiencia ni la debida información elemental sobre su estructura. La segunda cuestión es que al igual que la anterior, es muy tediosa concluir si es que tal sistema estar elaborada de manera correcta, lo más grave en los sistemas se refiere a los costos incurridos, así como son muy complicados de cambiar, una vez que se hayan creado tales sistemas. En cuanto a las otras ramas relacionadas a los sistemas informáticos, es muy común el empleo de modelos que permiten el análisis anticipado de sus características así como su funcionamiento (Booch et al., 1999).

El empleo de modelos en los sistemas informáticos, constituye una pieza fundamental para el desarrollo de tal complejidad, esto permitiría una réplica más

comprensible del sistema, en la cual estos eliminarían detalles que son de poca importancia, de manera que se pueda obtener un sistema orientado a objetos más sencillo de comprender, manipular y además se puedan desarrollar pronósticos en los aspectos más importantes del sistema. (Booch et al., 1999).

Un buen modelado según Booch et al. (1999), tiene varias características:

- Es necesario que permita una abstracción, de manera que se puedan ignorar detalles que no son necesarios en un determinado momento para el análisis de las propiedades importantes del sistema. Por tanto la abstracción, esta debe de ser cambiante, de manera que se pueda permitir que tal sistema tenga un alto o bajo detalle, dependiendo de la importancia del sistema.

- El sistema debe de tener notaciones que facilite al usurario, ya que si la notación es muy tediosa y difícil de que se pueda entender el modelo del sistema, será entonces de muy poca utilidad.

- El sistema creado también debe de mostrar característica similar al sistema final, desde aquellos aspectos que se planten ser estudiados o analizados antes y durante la elaboración del sistema final.

- Así también el sistema debe de contar con una base matemáticas con ciertos formalismos y permita la demostración de propiedades, afín que se pueda pronosticar el buen funcionamiento del sistema luego de ser construido.

- Finalmente, el sistema debe de ser fácil, sencillo y económico en su elaboración que el de un sistema real.

3.1.3. Necesidad del modelado

En todo sistema informático, con el transcurrir el tiempo, estos se tornan más complicados, son aplicados a distintos y más campos depositándose mayores responsabilidades, provocando de esta manera una alta exigencia en la creación de los sistemas.

No es necesario sistemas que únicamente muestren resultados correctos, sino que cuentan con otros requisitos como son la necesidad de ser sistemas muy seguros y que puedan responder de forma satisfactoria en caso de situaciones de incertidumbre.

3.1.4. Metodología de ingeniería del software

La ingeniería de software, tiene una metodología de producir el mismo software, pero de una forma organizada, empleando técnicas y convenciones predefinidas en sus notaciones.

En el momento de la creación de programación orientado a objetos, esta tuvo cierta noción de ingresar cierta percepción en la cual los objetos son entidades que son coherentes sobre la identidad, conducta y estado, tales objetos son organizados en cuanto a sus diferencias, pero también en sus similitudes, en la practica el

polimorfismo y la herencia, estas metodologías también incluyen tales conceptos, a fin de aclarar sus procedimientos, normas y guías, a fin de alcanzar un producto de calidad para satisfacer las necesidades del usuario, en sus siguientes elementos.

- Una larga vida que esto permitiría la adaptación a reglas en negocios y factibilidades tecnológicas.
- Conjunto lleno en cuanto a los modelos, así como conceptos internamente consistentes.
- Colección de normas y reglas de desarrollo.
- Estándares y estrategias de pruebas.

Es importante la utilización del modelo, así como su uso, afín de lograr cierta abstracción, por medio del cual el análisis se enfoque a la realidad para cierto nivel de diseño, así también existen detalles particulares en sus modelados. La metodología es aplicada en muchos aspectos de implementación, teniendo archivos, base de datos orientado a objetos. OMT (Object Modeling Technique) se encuentra elaborado en descripciones en sus estructuras de datos, constantes, procesos en sus transacciones. La OMT plantea énfasis en cuanto a sus especificaciones respecto a la información, así capturar los requerimientos, las imperativas especificaciones y así descender prematuramente en cuanto al diseño, las declaraciones permiten optimizar los estados, y los factores más importantes son:

- Mayor importancia respecto al análisis y nunca en el desarrollo
- Mayor importancia en sus datos que en sus modelos funcionales

El proceso metodológico OMT segmenta tal proceso en su desarrollo en componentes separados: el análisis, así también el diseño y la implantación.

3.1.4.1. Análisis.

La finalidad consiste en construir un modelo en la cual el sistema tiene que realizar. Este modelo es expresado en términos de objetos así como sus relaciones entre ellos, el fluido que es dinámico en su control así como sus transformaciones funcionales.

3.1.4.2. Diseño.

Representa el componente de mayor nivel a fin de desarrollar la solución y resolver el problema. En ella es definida la arquitectura del sistema para tomar las decisiones importantes.

3.1.4.3. Implementación.

Acá el diseño de objetos se transforma en códigos. Y en cada una de las faces, se segmenta en sub tareas, como los modelos de objetos, el dinámico y el funcional; los análisis del sistema y los objetos del sistema.

3.1.5. Llegada del lenguaje UML

El lenguaje UML (Lenguaje unificado de modelado) tuvo sus inicios en los años 1994, en la que Rumbaugh se integró a la corporación Rational, creada por Booch (dos investigadores en el campo de construcción de software).

El fin era la unificación de ambos métodos que se han desarrollado: la metodología de Booch y el OMT (Object Modelling Tool). Los primeros intentos se iniciaron en octubre de 1995. En esa misma etapa, otro importante investigador, Jacobson, se integró a Rational y tomaron en cuenta sus ideas. Conocidos como los “tres amigos”. Posteriormente, este lenguaje se abrió a la colaboración de otras empresas a fin de aportar ideas innovadoras. Todos estos apoyos llevaron a la creación de la primera edición y versión de UML.

Constituye un lenguaje de modelo visual utilizado para la especificación, visualización, la construcción y la documentación en un sistema. Y esta se utiliza para el entendimiento, el diseño, la configuración, el mantenimiento y el control de la información en la construcción de los sistemas

UML (Lenguaje unificado de Modelado) decepciona la información en cuanto a su estructura muy estática y el dinamismo en su comportamiento del sistema. El sistema se modela como la organización de objetos discretos, pero que interactúan, afín de realizar el trabajo que beneficia al usuario.

Un modelado cuenta con un lenguaje que unifica la pasada experiencia en cuanto a sus técnicas de modelado e incorpora las prácticas reales en un estándar acercamiento.

La UML, no representa algún lenguaje de programación. Sus herramientas ofrecen códigos del propio UML, esto en una variedad de lenguaje en su

programación, así también desarrollar modelos inversos en la ingeniería en los ya existentes programas.

La UML, tiene derivación y unificación en los tres métodos de análisis y los diseños de mayor extensión.

- Metodología de Booch para la descripción de conjuntos de objetos y sus relaciones.

- Técnica de modelado orientada a objetos de James Rumbaugh (OMT: Object - Modelling Technique).

- Aproximación de Ivar Jacobson (OOSE: Object- Oriented Software Engineering) mediante la metodología de casos de uso (use case).

El proceso de UML se inició a finales de 1994, cuando en 1995 Booch y Rumbaugh, sus métodos fueron unificaron, así también es Ivar Jacobson y su corporación Objectory se integraron a Rational, aportando de esta manera la metodología de OOSE.

Las metodologías que fueron centradas en objetos, fueron tres que son las de Booch y Rumbaugh, en la cual sus aproximaciones tienen un enfoque de modelado hacia los objetos que componen el sistema, así como su colaboración y su relación.

En cuanto al método de Jacobson se enfoca más al usuario, y es que toda la metodología viene de los escenarios centrados en el usuario. La UML se ha fomentado como un estándar en el momento de la OMG, que representa el origen de COBRA, que es la estándar pionera en la corporación en cuanto a su programación de distribución de objetos.

La OMG, aprobó a la UML en el año de 1997, llegando a convertirse en la notación estándar de facto para el diseño y análisis que es el de orientado a objetos.

El primer modelo fue la UML, en la publicación de un modelo-meta, con su propia notación, que incluya la notación en la mayor parte de información en sus requisitos, en su análisis y en su diseño. Dicho modelo que es auto-referencial (un lenguaje de modelo cuyo propósito es general y de ser capaz de realizar modelo a si mismo).

En el proceso de la UML sus representantes tomaron en cuenta lo siguiente:

- Conceder semánticas y notaciones a fin de alcanzar cierta cantidad en los aspectos del modelado contemporáneo, pero de una forma directiva y también económica.

- Dar las suficientes semánticas para alcanzar temas del modelado que representan las predicciones y los pronósticos, como son la tecnología de componentes, la distribución de cómputo, etc.

- Dar los mecanismos en sus extensiones de tal manera que los proyectos que son concretos se puedan extender el llamado meta-modelo, pero a un precio bajo.

La UML, no es en sí una metodología completa en su desarrollo. Es un lenguaje universal así como cualquier otro lenguaje muy general.

El UML debe entenderse como un estándar para modelado y no como un estándar de proceso software. Aunque UML debe ser aplicado en el contexto de un proceso, la experiencia ha mostrado que organizaciones diferentes y dominios del problema diferentes requieren diferentes procesos. Por ello se han centrado los esfuerzos en un meta-modelo común (que unifica las semánticas) y una notación común que proporcione una representación de esas semánticas. De todas formas, los autores de UML fomentan un proceso guiado por casos de uso, centrado en la arquitectura, iterativo e incremental (Booch et al., 1999).

La UML, tanto en sus modelos como en sus conceptos, se asocian en las áreas conceptuales:

3.1.5.1. Estructura estática.

En un modelo para la informática, primero es necesaria la definición de su universo, esto representa los claves conceptos para la aplicación, las internas propiedades y la existencia de relación entre ellas. Esto representa la estructura estática. Los modelados como clases, son los conceptos de aplicación y cada una de las cuales describe muchos objetos, en la cual almacenan la información y se

implementan a través de la comunicación. La información almacenada se modela como atributos. En cuanto a la estructura estática es expresada con diagramas en clases en las cuales se usan en la generación de las mayorías de las declaraciones en sus estructuras para determinado programa.

3.1.5.2. Comportamiento dinámico.

Existen dos maneras en la modelación del comportamiento, la primera representa la vida de un determinado objeto y de cómo se interactúa con el resto del mundo, en cambio la otra representa los patrones en su comunicación en un conjunto de conexión de objetos, vale decir la forma de interactuar. Una máquina de estados representa la visión en sus estados y muestran la manera en que el objeto responde a los eventos, pero todo en función de su estado actual. En cuanto a la interacción de los objetos, representa los enlaces entre los objetos con el flujo de los mensajes, así como ciertos enlaces que hay entre ellos. Todo ellos lo que hace es unificar la estructura de los datos, tanto en sus controles como en el flujo de sus datos.

3.1.5.3. Construcciones de implementación.

Así también un UML significa bastante en el análisis lógico, así como en la implementación tangible. Por tanto un componente representa la parte física pero que es reemplazable en un sistema, si como es capaz de dar respuestas a los pedidos descritos en un conjunto de interfaces. Así entonces un nodo, representa un recurso computacional y define la localización en la ejecución de un sistema, esta puede contener componente, pero también objetos.

3.1.5.4. Organización del modelo.

Toda información que contenga el modelo, esta debiera de ser segmentada en coherentes piezas, a fin que puedan trabajar los equipos en diversas partes de manera concurrente. La mente humana necesita la organización y el contenido del modelo en paquetes de un tamaño modesto. Pero los paquetes son organizados de forma jerárquica y de un propósito general de los modelos de un UML. Así también se usa para su almacenamiento, la gestión de la configuración y el control del acceso, para finalmente la creación de bibliotecas que tengan ciertos códigos y que estos sean reutilizables.

3.1.5.5. Elementos de anotación.

Las partes explicativas, representan los elementos de anotación en los modelos UML. Que se pueden aplicar a fin de describir como clasificar, así como realizar observaciones en cualquier componente de un modelo. La nota, representa el tipo principal de la anotación y es el símbolo para demostrar todas las restricciones y todos los comentarios en un conjunto de elementos.

3.1.5.6. Relaciones.

Hay cuatro relaciones que conforman los elementos del modelado en UML. La dependencia, la asociación, la generalización y la realización, que a continuación detallamos:

- La dependencia: acá se relacionan dos elementos de forma semántica en la que un cambio a un elemento afecta a otro elemento en su semántica. Esta se

representa en forma de línea pero de manera discontinua y que a veces incorpora cierta etiqueta.

- La asociación: representa a una relación pero de forma estructural y describe ciertos enlaces que son las conexiones entre los objetos. La agregación representa una relación pero estructural y representa un tipo muy especial, dicha relación en entre un todo y las partes. Esta también es una línea continua, así mismo tiene una línea continua, con su respectiva etiqueta. Generalmente hay otros adornos para dar la multiplicidad y los roles entre los objetos involucrados.

- La generalización: Acá la relación consiste en la especialización y la generalización, por medio del cual los objetos de la especialización es la que sustituye a los objetos que componen el elemento general. De manera que el hijo es el que comparte la estructura junto al compartimiento del padre y se representa con la línea que tiene punta de flecha.

- La realización: Representa la relación entre los que clasifican que especifica cierto contrato y de otro clasificador que es el que garantiza su cumplimiento. Acá hay relaciones en dos tipos de sitios: estas representan los interfaces además entre todos los casos de usos y sus colaboraciones que ellas la realizan. Esta es una mezcla entre lo que es la dependencia y la generalización y es una línea discontinua que tiene cierta punta en la flecha pero vacía. Según Booch et al., (1999).

3.1.6. Diagramas

Estos se utilizan en la representación de distintas perspectivas en un sistema de manera que un diagrama representa la proyección del mismo. UML es el que muestra un conjunto amplio de diagramas y son utilizados en pequeños subconjuntos, pero para representar las cinco vistas importantes en la arquitectura de un sistema. A continuación explicaremos cada diagrama.

3.1.7. Diagrama de clases

Los diagramas de clases visualizan las relaciones de todas las clases que tienen el sistema, estas podrían ser asociadas, tanto de herencia y de uso como de contenimiento. Este diagrama contiene lo siguiente:

- En cuanto a clase: los atributos, los métodos y la visibilidad.
- En cuanto a las relaciones: la herencia, la composición, la agregación, la asociación y usos.

3.1.7.1. Elementos de diagramas de clases.

a. Clase.

Representa la unidad básica que encapsula en la información dirigida a un objeto. Por medio de ello se puede modelar el contexto a través de un estudio (un carro, una tienda, cuenta corriente) en la UML, la clase se representa en forma de rectángulo y tiene tres divisiones:

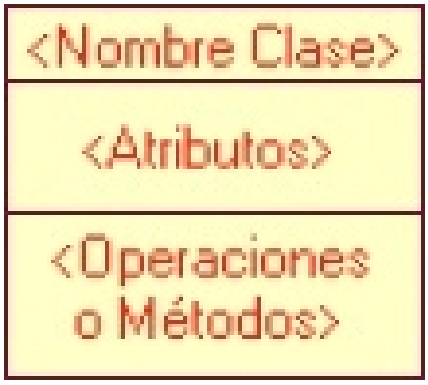

Figura 1. Modelo de clases
Fuente: Salinas, 2012

Donde:

- El superior: Indica la connotación de la clase.
- El intermedio: Es el que tiene todos los atributos que representan a la clase.
- El inferior: Que representa a las operaciones, de los cuales indican la forma como hay la interacción con el objeto de su entorno.

Ejemplo:

Una cuenta en el banco de tipo corriente que posee como característica: balance y que se puede realizar las transacciones y las operaciones como: el de depositar, el de girar y el balance, pero el diseño asociado es:

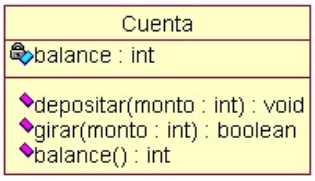

Figura 2. Ejemplo de una clase
Fuente: Salinas, 2012

Las características de una clase son llamadas atributos, estos suelen ser de tres tipos, aquellos que muestran el grado de comunicación como la visibilidad en sus contextos, estos representan lo siguiente:

- Public (+, ): Demuestra que la característica es visible de adentro como de afuera de la clase.

- Private (-, ): Indica la característica y sólo es accesible internamente de la clase (únicamente los métodos lo pueden accesar).

- Protected (#, ): Muestra que la característica no es accesible externamente de la clase, pero si puede ser accesado por métodos de la clase y de las subclases que de ella se deriven.

Por otra parte los métodos de una clase representan la manera de cómo se interactúan su entorno, éstos pueden tener también atributos.

- Public (+, ): Muestra que será visible el método tanto interna como externamente, vale decir que es accesible desde todos sus lados.

- Private (-, ): Muestra que tales atributos serán únicamente accesible internamente (únicamente otras metodologías de clase serán accesados.

- Protected (#, ): Muestra que este atributo no es accesible externamente de la clase, pero si son accesados por diferentes métodos de la clase y por otros métodos de la subclase que de ella se deriven.

3.1.8. Relaciones entre clases

Conocido el concepto de clase, es importante explicar cómo se da la interrelación entre dos o más clases (con objetivos y características diferentes). Ahora es importante explicar de lo que se conoce como cardinalidad entre relaciones: en la UML, las relaciones referidas a la cardinalidad, indican el nivel y grado de la dependencia y se notifican en cada extremo de dicha relación y son:

- Pueden ser uno o varios: 1..* (1..n)
- Cero o varios: 0..* (0..n)
- Cantidad de número fijo: m (m representa al número)

3.1.9. La herencia (especialización/generalización)

Nos muestra que la subclase proviene de los atributos y los métodos y son especificados en una súper clase y por tanto la subclase luego de tener sus mismos métodos y también atributos, tiene los atributos y las características que son visibles de la superclase (protected y public). Ejemplo:

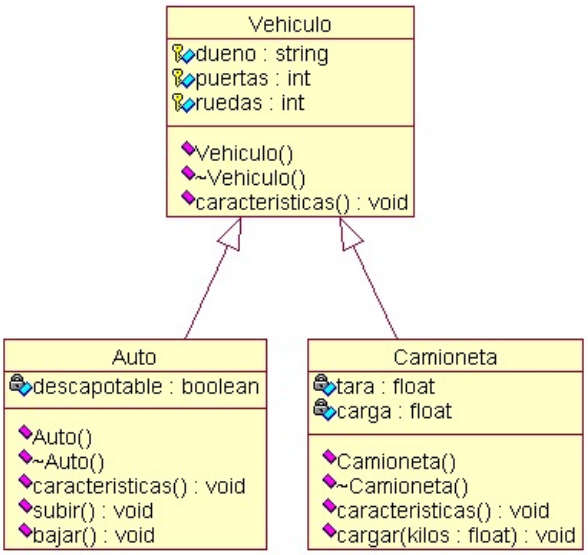

Figura 3. Herencia en una clase
Fuente: Salinas, 2012

La figura, muestra en que auto y en que camión heredan en vehículos, vale decir que en que auto se tienen los atributos del vehículo, (velmax, precio), luego es que tiene algo particular que representa a descapotable, luego el camión es que también hereda todas las características del vehículo, pero tiene particularmente el propio acoplado, la carga y tara (Cockburn, 2001).

Debemos indicar que representa este entorno, lo único observable, es la metodología de atributos que son aplicables en las instancias del vehículo, es que se tiene la definición publica, por tanto los atributos como son los descapotables no representan la visibilidad pro representar como privados.

3.1.10. Agregación

No son necesarios los datos básicos en el caso de modelar todo objeto en su complejidad y que estos provean lenguajes; como las secuencias de caracteres, enteros y reales. Donde se necesita tener objetos que representan las instancias definidas de clase debido al desarrollador en la aplicación y es que hay dos posibilidades:

- Para el caso de valor, que representa a una relación pero estática, en la que el tiempo de vida del objeto, incluye el condicionamiento debido al tiempo de vida en la que se incluye. Esta relación se llama la composición, en la que el objeto de base, se forma por medio del objeto incluido.

- En cuanto a la referencia: representa una relación pero dinámica, en la cual es el tiempo de vida del propio objeto no es dependiente del que lo incluye.

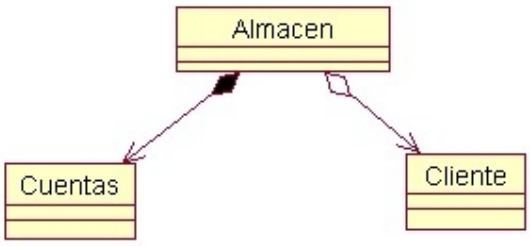

Figura 4. Ejemplo de agregación
Fuente: Salinas, 2012

- El lugar tiene cuentas y clientes (la figura de rombo va hacia el objeto que tiene en las referencias)
- Si se elimina el objeto del almacén, igualmente se eliminan los objetos de la cuenta que son asociados, pero no son influidos en los objetos de clientes asociados.
- La confección (de valor) en un rombo, pero relleno.
- En la aglomeración es la que sobresale un rombo pero transparente.

Las flechas de estas relaciones muestran la trayectoria del objeto que es referenciado. En cambio la flecha es eliminada en el caso de no existir esta particularidad.

3.1.11. Asociación

Esta relación de clases se conoce también como una asociación, que permite relacionar los objetos que entre si se colaboran. Ya que no es una asociación fuerte, vale decir que el tiempo de vivencia del objeto no tiene dependencia del otro.

Ejemplo:

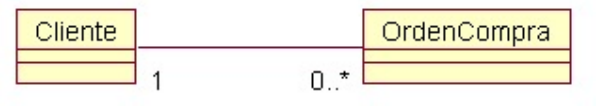

Figura 5. Ejemplo de asociación
Fuente: Salinas, 2012

3.1.12. Dependencia o instanciación

Es una forma de relación especial, pero en la cual una clase se instancia es decir que depende de otro objeto/clase. Y es denotado por alguna flecha que es punteada.

La aplicación particular de este tipo representa para mostrar la relación de dependencia que hay entre una clase y la otra, como podría ser la aplicación de forma gráfica que tiene una instancia en una ventana. Esta creación es condicionada en la instancia que viene del objeto de aplicación.

Figura 6. Ejemplo de dependencia
Fuente: Salinas, 2012

Cabe destacar que el objeto creado (en este caso la ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la aplicación).

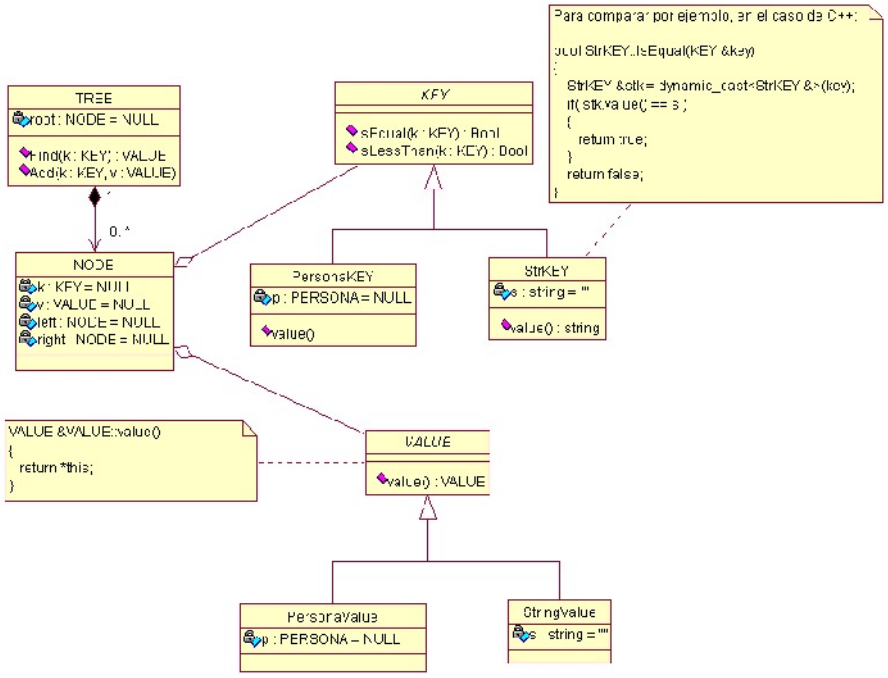

Figura 7. Diagrama de clases
Fuente: Salinas, 2012

3.1.13. Diagrama de objetos

Muestran un conjunto de objetos y sus relaciones, son como fotos instantáneas de los diagramas de clases y cubren la vista de diseño estática o la vista de procesos estática desde la perspectiva de casos reales o prototípicos. Los diagramas de objetos pueden ayudar a explicar las clases y su herencia, ayuda a partes interesadas para quienes los diagramas de clases sean demasiado abstractos (Larman, 2005).

Un objeto cuenta con una estructura, es decir con unos atributos y acciones. Se representa en un rectángulo con tres compartimientos, en el primero va el nombre del objeto, en el segundo sus atributos y en el tercero sus operaciones, el ultimo pudiendo ser omitido si así se prefiere (Larman, 2005).

Definición de objeto: Un objeto en programación orientada a objetos (POO) representa alguna entidad de la vida real, es decir, alguno de los objetos que pertenecen al negocio con que estamos trabajando o al problema con el que nos estamos enfrentando, y con los que podemos interactuar. A través del estudio de ellos se adquiere el conocimiento necesario para, mediante la abstracción y la generalización, agruparlos según sus características en conjuntos.

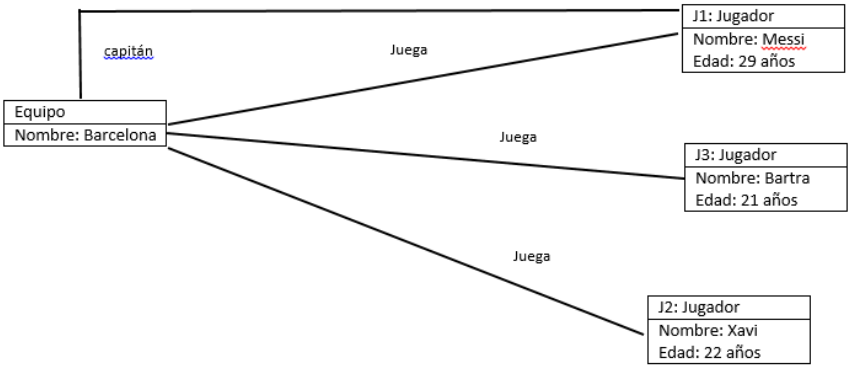

Figura 8. Diagrama de objetos

Características de los diagramas de objetos:

- La clase define las reglas, los objetos expresan los hechos.
- La clase define que puede ser, el objeto describe que es.
- Se considera un caso especial de diagrama de clases.
- Puede construirse junto con el de clases.
- El objeto o instancia se encuentra a la izquierda.
- Los nombres de los objetos están subrayados.

3.1.14. Los diagrama de casos de uso

Las figuras que son diagramas, muestra una serie de casos en su aplicación y los actores de cada tipo especial de clase, pero al mismo tiempo sus asociaciones tapan la vista estática en los usos pero son importantes para la modelación y para la organización de todo el comportamiento

El diagrama en los casos de los usos representa la forma en como un cliente (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).

Un diagrama de casos de uso consta de los siguientes elementos:

- Actor
- Casos de uso
- Relaciones de uso, herencia y comunicación

3.1.14.1. Elementos.

a. Actor.

Una definición previa, es que un actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Como ejemplo a la definición anterior, tenemos el caso de un sistema de ventas en que el rol de vendedor con respecto al sistema puede ser realizado por un vendedor o bien por el jefe de local.

Figura 9. Ejemplo de actor

b. Caso de uso.

Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

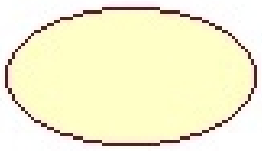

Figura 10. Ejemplo de caso

3.1.14.2. Relaciones

a. Asociación.

Representa a la asociación más elemental que muestra el llamado desde un caso de uso hacia otra operación, pero que tal relación es representada por una flecha simple.

b. La dependencia o la instanciación.

Representa la forma especial de la relación entre las clases, en donde únicamente una clase, es la que depende de la otra, vale decir que se crea, pero tal relación es una flecha punteada.

c. La generalización.

Representa a una relación, pero la más aplicada y es que tiene una función doble que depende del estereotipo y es que puede ser de uso o también puede ser de herencia. Esta relación es dirigida en los casos de usos y no es para actores

d. La extends.

Es aconsejable usar en un caso de uso y esta es similar al otro (atributos)

e. Users.

Es aconsejable usar en la que hay un conjunto de atributos que son parecidos en más de solo un caso de su uso y no es necesario tener copiada la descripción del atributo.

3.1.15. Los diagramas de secuencia

El diagrama de secuencia en UML muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo.

El diagrama de secuencia nos muestra los objetos participando en la interacción o los eventos que originan los actores dentro de un sistema y como se comunican o interactúan entre sí a lo largo del tiempo, esta descripción es importante porque puede dar detalle a los casos de uso, este diagrama es más adecuado para observar la perspectiva cronológica de las interacciones, muestra la secuencia explícita de mensajes y son mejores para especificaciones de tiempo real y para escenarios complejos. La creación de los diagramas de secuencia forma parte de la investigación para conocer el sistema, por lo que es parte del análisis mismo.

Las características de los diagramas de secuencia son las siguientes:

- Mostrar la secuencia de mensajes entre objetos durante un escenario concreto
- Cada objeto viene dado por una barra vertical
- El tiempo transcurre de arriba abajo
- Cuando existe demora entre el envío y la atención se puede indicar usando una línea oblicua.

3.1.15.1. Elementos de los diagramas de secuencia.

a. Objetos.

Se obtienen de los diagramas de casos de uso, y se representan con dos componentes: el nombre del objeto y la clase a la que pertenece.

Figura 11. Ejemplo de objetos
Fuente: García, 2008

b. Mensajes.

Representa a la comunicación de los objetos que dan información a fin de crear una acción. En la percepción del mensaje, normalmente se considera en un evento. Esta es representada con una flecha pero horizontal que parte de la línea de vida del objeto que dio el mensaje, hasta la línea del objeto que ha recibido el mensaje.

Los tipos de mensajes son:

- **El simple:** Que transfiere el control de objeto hacia el otro.
- **Los síncronos:** Representan a los más usados, que el emisor del mensaje espera al cual el destinatario termine el método en mención luego de seguir la actividad normal.
- **Asíncrono:** Acá el emisor realiza de forma directa sin esperar al destinatario en realizar sus acciones.

Figura 12. Tipos de mensaje
Fuente: García, 2008

c. Tiempo.

El diagrama representa al tiempo en dirección vertical. El tiempo se inicia en la parte superior y avanza hacia la parte inferior. Un mensaje que esté más cerca de la parte superior ocurrida antes que uno que este cerca la parte inferior.

Con ello el diagrama de secuencias tiene dos dimensiones, La dimensión horizontal es la disposición de los objetos y la dimensión vertical que muestra los tiempos.

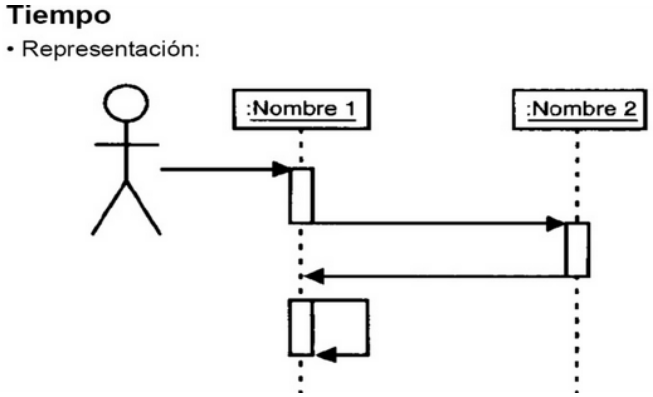

Figura 13. Tiempo
Fuente: García, 2008

d. Recursividad

En ocasiones un objeto posee una operación que se invoca a sí misma. A esto se le conoce como recursividad y es una característica fundamental de varios lenguajes de programación.

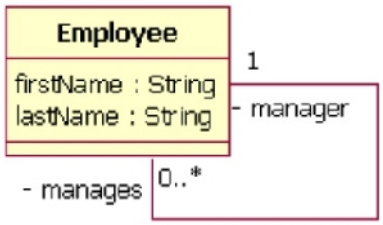

Figura 14. Ejemplo de recursividad

3.1.16. Diagrama de colaboración

Los diagramas de colaboración son otro tipo de diagramas de interacción, que contiene la misma información que los diagramas de secuencia, sólo que se centran en las responsabilidades de cada objeto, en lugar del tiempo en que los mensajes son enviados. Un diagrama de colaboración describe en forma de un grafo el comportamiento de sistemas, subsistemas y operaciones, representando los objetos que intervienen, así como los mensajes que intercambian, enumerados en el tiempo.

Lo que se quiere lograr con este diagrama es manejar la comunicación entre los elementos del sistema, mostrar cómo será implementada una operación, indicar cómo deben colaborar los objetos del sistema para llevar a cabo una operación.

Las características del diagrama de colaboración son:

- Muestra cómo las instancias específicas de las clases trabajan juntas para conseguir un objetivo común.
- Implementa las asociaciones del diagrama de clases mediante el paso de mensajes de un objeto a otro. Dicha implementación es llamada "enlace".

Los elementos del diagrama de colaboración son:

- Objetos o roles: nodos del grafo
- Enlaces o comunicaciones: arcos del grafo
- Mensajes: llevan número de secuencia y flecha dirigida
- Anidamiento: se utiliza la numeración decimal.
- Iteración: colocar un * antes del número de secuencia y una cláusula de condición, si es necesario.
- Bifurcación: los caminos alternativos tendrán el mismo número de secuencia, seguido del número de subsecuencia, y se deben distinguir por una condición.

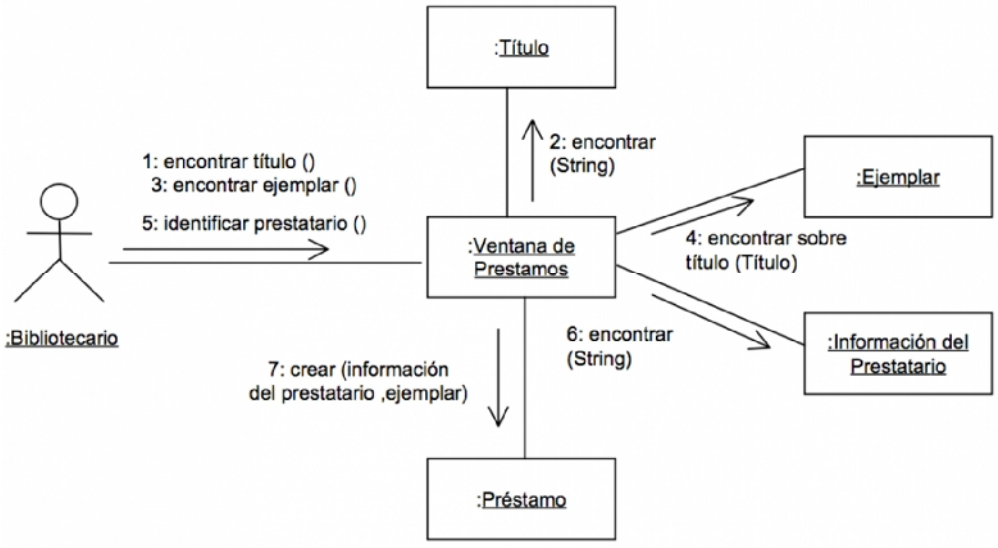

Figura 15. *Ejemplo de colaboración*

Fuente: Cillero, 2015

3.1.17. Diagrama de estados

Muestran una máquina de estados compuesta por estados, transiciones, eventos y actividades. Estos diagramas cubren la vista dinámica de un sistema y son muy importantes a la hora de modelar el comportamiento de una interfaz, clase o colaboración.

Las características de los diagramas de estado son:

- Son autómatas jerárquicos que permiten expresar concurrencia
- Son causantes
- El inicio entre los estados es al instante y su causa es el suceso de cierto evento.

3.1.17.1. Elementos del diagrama de un estado.

a. El estado.

Muestra en un intervalo de tiempo del objeto (que no es instantáneo) por medio del cual es el objeto que espera una operación, y es que tiene cierto estado característico y recibe estímulos.

b. Los eventos.

Representa a una ocurrencia que causa la transición en un objeto de un estado hacia otro.

c. El envío de mensajes.

Es la que se representa el momento, por medio del cual envía el mensaje hacia otros objetos. Esto es ejecutado por medio de una línea punteada que va al diagrama de estados del propio objeto receptor del mismo mensaje. Ya que no únicamente muestra la transición de estados a través de eventos.

d. La transición simple.

Esta representa a la relación de dos estados en la que muestran que un objeto del primer estado, ingresa al segundo estado y realizar ciertas operaciones, si un evento sucede en donde las condiciones son satisfactorias. Se da como una lineal pero sólida en dos estados, y podrían estar acompañadas de un texto.

e. La transición interna.

Esta es la que sigue el mismo estado, en vez de dar en dos estados diferentes. Muestra un evento que no influye en cambio de estado. Se da como una cadena adicional en compartimiento de las acciones de estado.

f. Los subestados.

Se separa en varios subestados y con ciertas transiciones entre ellos y las conexiones de nivel superior. Pero también las conexiones ven a nivel inferior como inicio o como fin, que están conectados en los ingresos y salidas del nivel que es el superior.

g. La transición compleja.

Es la que asocia entre tres o más estados, dados en una transición de variadas fuentes y/o variados destinos. Es la que representa la subdivisión en los conflictos en el control de una sincronización u objeto. Por tanto es la que se da de como una línea vertical en la que entran y salen múltiples líneas de estado en transición.

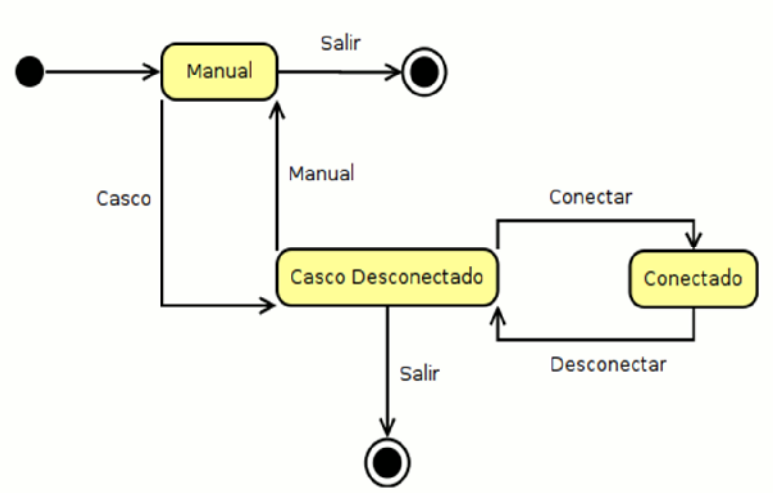

Figura 16. *Diagrama de estados*
Fuente: Stolfi, 2010

3.1.18. Diagrama de actividades

Son un tipo especial de diagramas de estados que se centra en mostrar el flujo de actividades dentro de un sistema. Los diagramas de actividades cubren la parte dinámica de un sistema y se utilizan para modelar el funcionamiento de un sistema resaltando el flujo de control entre objetos.

En ella misma muestra las rutas posibles que podrían desencadenar para el caso de uso y también sirve en la modelación de la lógica de forma detallada en toda regla de negocios. Esta apoya al auxilio a todos los miembros en el desarrollo y a entender de cómo se usa el propio sistema y cuál es la reacción en eventos

determinados. En varios sentidos, los diagramas UML de una actividad que es el equivalente orientado a los objetos en los diagramas de flujos y también los diagramas de flujos pero de datos (DFD), de un desarrollo estructurado, en la cual se puede mencionar que es un diagrama de varias actividades que describe el problema, en tanto que un diagrama de flujo únicamente describe la solución.

3.1.18.1. Elementos de un diagrama de actividades.

a. El inicio.

Esta es representada mediante un círculo cuyo color es de negro pero sólido, dedo en el inicio de un diagrama.

b. La actividad.

Está representada por la acción que será ejecutada por el sistema y es representada en un ovalo, después de la realización, es la que se continua con la actividad siguiente.

c. La transición.

Esta transición sucede en el momento de llevar a cabo la variación de una actividad hacia otra, esta transición se representa únicamente por una línea pero con una flecha en la terminación para indicar la dirección.

SÍMBOLO	SIGNIFICADO
	Punto de inicio del proceso
	Actividad
	Condicional
	Flujo de secuencia
	Bifurcación o entrada
	Punto final del proceso
	Swinlanes ("Calles")

Figura 17. Elementos de diagrama de actividades
Fuente: Maravillas, 2015

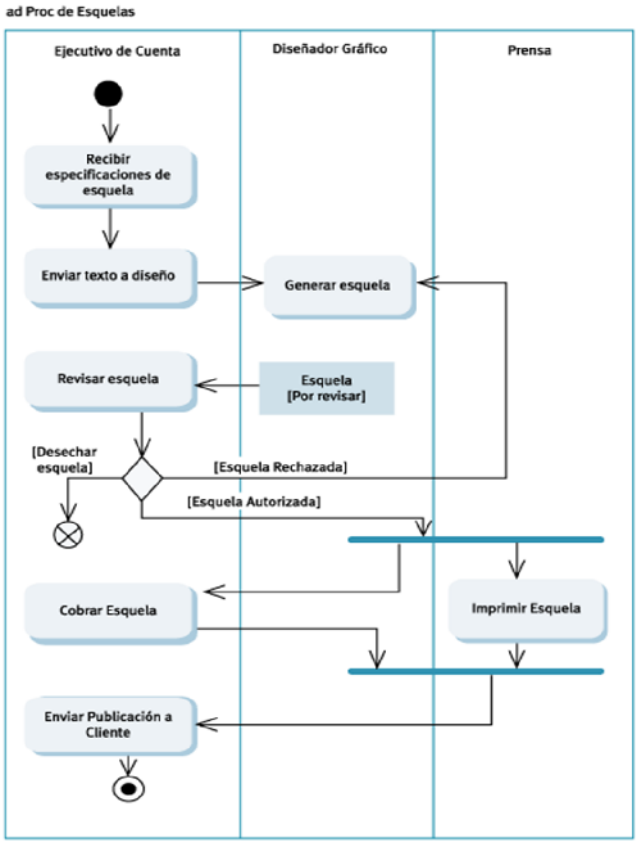

Figura 18. Diagrama de actividades
Fuente: Orozco, 2012

3.1.19. Los diagramas de componentes

Indica las dependencias y la organización, pero entre varios componentes.

Tapan la observación de la implementación de forma estática y está relacionado con diagramas de clase, pues en un componente puede que tenga una o varias clases, colaboraciones o también interfaces.

- Estos diagramas dan una descripción de los elementos físicos de todo el sistema y sus dependencias.
- Indican las formas de realización así también los códigos fuentes, como binarios y también ejecutable.
- Por otro lado, los componentes reemplazan a todos los tipos, pero de elementos del software que ingresan para la fabricación en las aplicaciones, pero informáticas.
- También podrían ser únicamente archivos simples, los paquetes, sus bibliotecas que están cargadas de forma dinámica.

3.1.19.1. Los elementos de un diagrama de componentes.

a. Componente.

Es una parte física de un sistema (módulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los

componentes. Se representa como un rectángulo en el que se escribe su nombre y en él se muestran dos pequeños rectángulos al lado izquierdo.

Los componentes se pueden agrupar en paquetes, así como los objetos en clases, además puede haber de ellos relaciones de dependencia como generalización, asociación, agregación, realización.

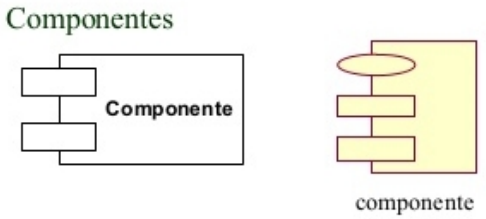

Figura 19. Ejemplo de componente

UML define 5 estereotipos estándar que se aplican en los componentes:

- Executable: componente que se puede ejecutar.
- Library: biblioteca de objetos estática o dinámica.
- Table: componentes que representan una tabla de base de datos.
- File: componente que representa un documento que contiene código, fuente o dato.
- Document: componente que representa un documento.

b. Interfaces.

Es el lazo de unión entre varios componentes, se pueden representar de dos maneras: forma icónica o expandida.

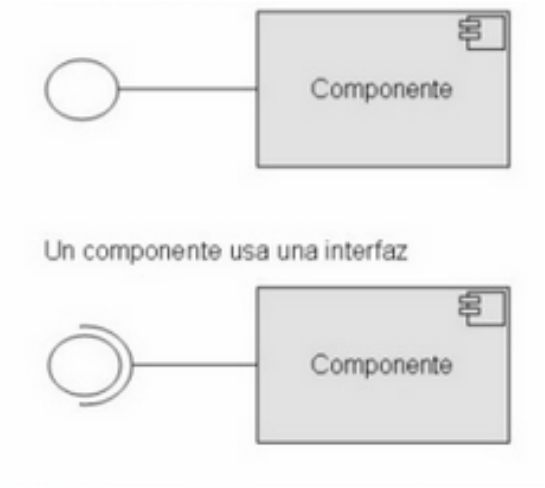

Figura 20. Muestra de interfaz
Fuente: Cruz, 2011

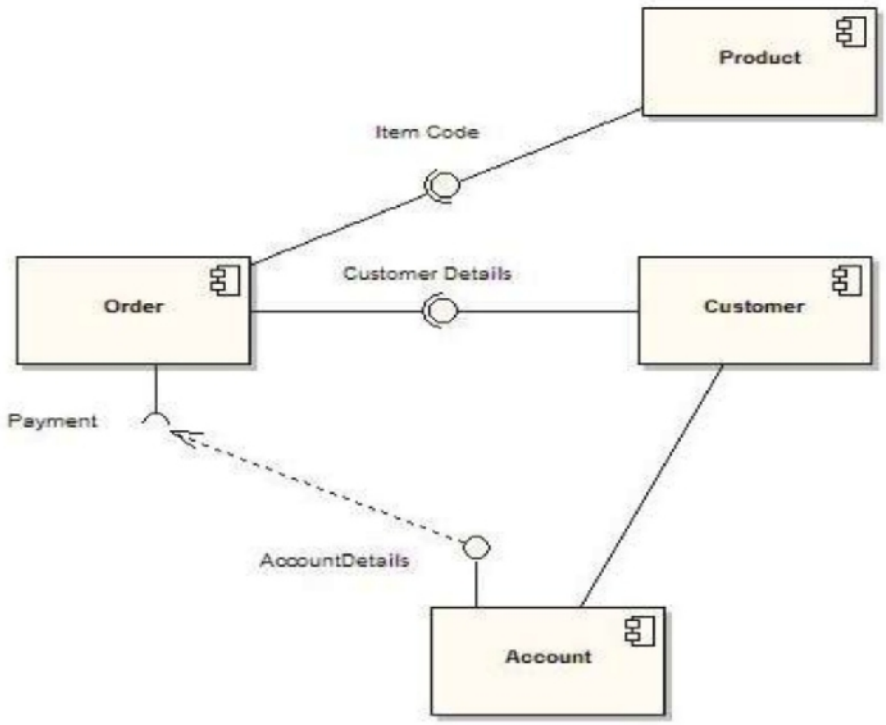

Figura 21. Diagrama de componentes
Fuente: Huerta, 2012

3.1.20. Diagrama de despliegue

Representan la configuración de los nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos. Muestran la vista de despliegue estática de una arquitectura y se relacionan con los componentes ya que, por lo común, los nodos contienen uno o más componentes.

Las características de los diagramas de despliegue son:

- Describe la arquitectura de un modelado mediante tres caminos: los procesadores, los dispositivos y el componente de un software.

- También da una descripción de la topología dado en el sistema en el momento de la acción

3.1.20.1. Elementos del diagrama de despliegue.

- Nodos: Son objetos físicos que existen en tiempo de ejecución, y que representan algún tipo de recurso computacional (capacidad de memoria y procesamiento).

- Artefactos: Un artefacto es producto del proceso de desarrollo de software, que puede incluir los modelos del proceso (archivos fuente, ejecutables, documentos de diseño, reportes de prueba, prototipos, manuales y más).

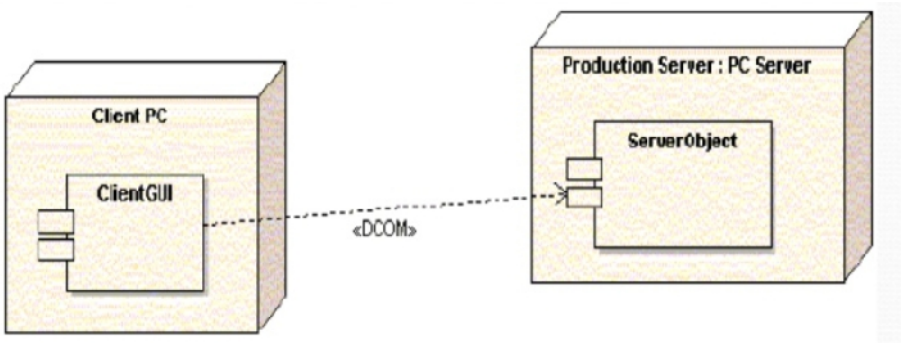

Figura 22. Diagrama de despliegue

3.2. Caso práctico

3.2.1. Procedimiento

El trabajo presentado contiene los datos que se generaron en todo el proceso de un modelado, pero de biblioteca de forma virtual. Tal proyecto es un análisis sobre la aplicación y accesible, pero por medio de la Internet, en ella los usuarios son los que consultan cierto catálogo de todos los libros, pero disponibles, los solicita, hace las búsquedas muy específicas, seguidamente presenta todas las opciones en la administración del stock de toda la bibliografía y también sus entregas y préstamos. Sin embargo, que para llevar a cabo el desarrollo se usan todos los conocimientos que fueron adquiridos a todo lo largo de la especialidad en la Ingeniería en Informática y en Sistemas.

3.2.2. Descripción del proyecto

Este proyecto de libros pero virtual es especialmente a bibliotecas pero de barrios que están muy interesadas en dar los servicios por medio de la Internet. Y es que

tiene por finalidad el facilitar toda la consulta del catálogo que se encuentra disponible y también permite los ejemplares solicitados y la renovación de los préstamos a través de una interfaz del interesado de forma amigable y sencilla.

En el momento del accedo de la web en la biblioteca, los interesados pueden consultar toda la bibliografía que es agregada y también la bibliografía más recomendada que sean por otros usuarios. Así también pueden las búsquedas de los libros y mirar el detalle del libro. En ella se observará una foto de la portada, su título y los autores, así también se encuentra la fecha de su publicación, el ISBN, su editorial y las categorías en las que pertenecen, la descripción de los libros y también todos sus comentarios.

Los interesados pueden ser socios de tal plataforma en la que completan cierto formulario de todo el registro, pero disponible que este en la web. Luego realizaran la identificación y la solicitud de cierto libro para determinada fecha que es de 30 días y la realización de hasta tres renovaciones en los préstamos, pero siempre que el ejemplar este en poder del socio y este no se encuentre solicitado por el otro interesado.

Sin embargo, en cada solicitud o la renovación se dará en el correo electrónico de aviso al socio y también para cierta cuenta de biblioteca que es habilitada. Pero estas renovaciones no se realizarán hasta cinco días, pero antes que culmine dicho préstamo. Y cada socio no puede tener más de tres libros de forma simultánea.

Además, se puede consultar la bibliografía que se tiene en préstamo actualmente y todas las solicitudes que estén pendientes. Así también se puede cancelar cierta solicitud que no fuera satisfactoria. Otra función importante del sistema permite votar y opinar en la bibliografía en la que el socio ya leyó de forma independiente en la que ya obtuvo dicho libro.

Esta biblioteca tiene un administrador que gestiona el stock y el catalogo que se encuentra disponible para cada bibliografía, e informa si el libro es entregado a su correspondiente socio, así como cuando es devuelto. Y si es que el libro es devuelto, pero fuera del plazo, dicho sistema penaliza al propio usuario de manera que ya no pueda solicitar préstamo, hasta que tenga la autorización del administrador, el propio encargado, también consultara todas las solicitudes que están pendientes, todos los préstamos y rechazar la solicitud de préstamo.

3.2.3. Las tareas a desarrollar

Pero luego de realizar toda la descripción de tal proyecto, es necesario conocer los requisitos que no cambian en la realización del trabajo. Así que nos dispondremos a realizar un cronograma de trabajo.

Tabla 1

Ejemplo de tabla de desarrollo de actividades

Documento para la entrega	La fase	La tarea	Duración estimada
Plan de trabajo	Todo el plan de trabajo	Definir a los requerimientos	2 días
		Definir a los objetivos	1 día
		Planificar para el proyecto	1 día
	Análisis	Descripción de casos de uso, identificación de actores	4 días
		El prototipo	3 días
		y el glosario	1 día
		La revisión de todo el análisis	3 días
Paquete 1	Diseño	Los diagramas de las clases	5 días
		Los diagramas de estado, secuencia y actividad	6 días
		El diseño de la base de los datos	4 días
		El diseño en la arquitectura y la	11 días
	Implementación	Instalación del lugar de	5 días
		programación	23 días
Paquete 2		Pruebas	5 días
		manual de instalación	2 días

3.2.3.1. Análisis.

En esta parte es necesario avanzar todo el análisis, describiendo en los actores y los casos más importantes de su uso.

a. Reconocimiento de los actores.

Todos los actores pero que interactúan en la biblioteca es el:

- **El invitado:** Representa a todos los usuarios que hacen la consulta del catálogo sin la identificación previa. Pero puede consultar todo material disponible en la que no hay préstamo.

- **El socio:** Al igual que en los invitados, estos también realizan la consulta de catálogo de los libros. Sin embargo estos si están habilitados, previa identificación, en la que se solicita los préstamos, para renovarlos y también la cancelación de solicitudes no satisfechas.

- **El administrador:** Representan a los usuarios cuyos permisos son especiales para poder manejar todo el sistema en general.

3.3. Representación de resultados

Para el siguiente proyecto se utilizó la metodología UML como ya lo habíamos mencionado anteriormente, a continuación a partir de la descripción

empezaremos a realizar los diagramas correspondientes utilizando la metodología RUP.

3.3.1. Fase de inicio

En esta fase ejecutaremos las actividades que se deben realizar en la metodología RUP y también se desarrollará los planes que se harán en el sistema. Luego de ver los distintos problemas existentes en las bibliotecas públicas.

3.3.1.1. Visión del proyecto.

El propósito del presente proyecto es para mejorar la administración de préstamos y devoluciones de libros mediante un sistema. Con el mismo a la vez se quiere dar la facilidad al bibliotecario de realizar un control eficiente y dar una buena atención a los diferentes usuarios que acuden en solicitud de algún libro.

3.3.1.2. Misión del proyecto.

La misión del proyecto es de dar un sistema de control en las bibliotecas públicas locales dando una mayor eficiencia al préstamo y devolución de libros y de esta manera poder llevar un control óptimo por parte del bibliotecario.

3.3.2. Planificación

3.3.2.1. Fase de inicio.

Tabla 2

Tabla de planificación de actividades

Actividad	Días	Iteración
Elaboración de misión y visión del proyecto	1	1
Generación de la planificación del proyecto	3	1
Generación del modelo de negocio	5	1
Generación de estudios de requisitos	7	1
Elaboración del diagrama de caso de uso	2	2
Elaboración diagrama de secuencias	2	1
Elaboración diagrama de estado	3	2
Elaboración diagrama de clases y demás diagramas	6	3

3.3.2.2. Modelado del negocio.

En el modelado del negocio identificaremos todas las condiciones que podremos tener en todo el sistema ya sea por parte del usuario o del bibliotecario, podemos observar los siguientes requisitos.

a. Préstamo.

- El usuario puede realizar la solicitud del libro, pero según la disponibilidad del propio material.

- El interesado puede solicitar cierta cantidad máxima de dos libros.
- El periodo máximo de préstamo será de una semana calendario.
- Los usuarios que tengan alguna sanción vigente no podrán realizar préstamos.
- Para realizar un préstamo es necesario estar registrado en el sistema ya existente en la biblioteca.
- En caso de que el usuario sea nuevo se realizará un registro único cada principio de semestre.
- Cada usuario sabrá la fecha límite de devolución del material que tiene en su poder.
- El interesado debe de contar CI y la matrícula de la universidad para realizar el préstamo del material que solicito.

b. La devolución.

- El interesado debe dejar el libro antes de los siete días calendario estipulado.
- En caso de exceder la fecha límite de devolución de los siete días calendarios se le emitirá una sanción al usuario infractor.
- En caso de que el día de devolución sea en un día festivo (feriado) o por algún motivo de no atención de la biblioteca la devolución será al día siguiente de la fecha estipulada en el momento del préstamo.
- El sistema se implementará en las máquinas con las que se cuenta en las bibliotecas, lo que se trata de implementar es que el sistema sea más eficiente que el actual existente.

3.3.2.3. Fase de elaboración.

En esta parte identificaremos las problemáticas existentes en el modelamiento, en la que se especifica todos los casos de la aplicación del sistema y también se realiza su arquitectura para bajar los riesgos construidos en el proyecto.

3.3.2.4. El modelo de un negocio.

Este modelamiento es donde se capturan los requisitos a través de los flujos en el trabajo que se hace para la biblioteca en donde se encuentran los actores de la que son los bibliotecarios y los usuarios.

a. Actores del negocio.

Tabla 3

Actores que formarán parte del sistema

Actor	Descripción
Invitado	Es el usuario que consulta el catálogo sin identificarse antes. Puede realizar consultas del libro disponible y que no puede pedir el préstamo.
Socio	Los socios, al igual que los invitados son los que consultan los catálogos, sin embargo estos se encuentran habilitados para pedir el préstamo, de manera que se pueda renovar o cancelar las solicitudes que no están satisfechas.
Administrador (Bibliotecario)	Realiza todas las acciones en sistema como préstamos, devoluciones y sanciones pertinentes.

b. Diagrama de paquetes.

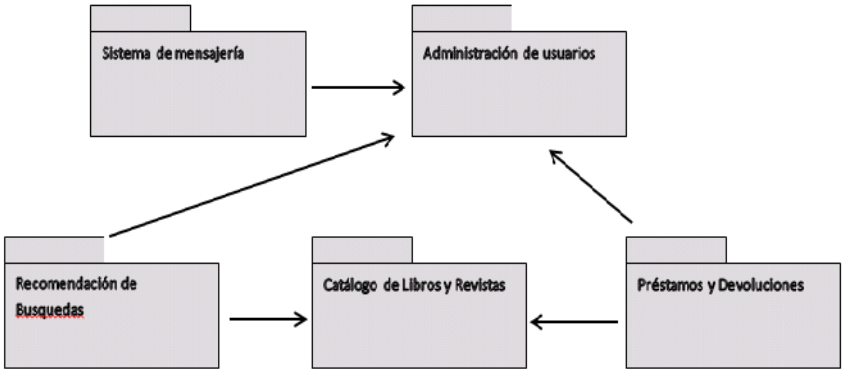

Figura 23. Diagrama de paquetes - biblioteca

En la figura 23 se muestra como el sistema está dividido en agrupaciones lógicas denominadas paquetes, estos son dependientes unos de otros, exceptuando "Catalogo libros y revistas" y "Administración de usuarios" que se pueden trabajar por separado al ser totalmente independientes de los demás paquetes.

c. Diagrama de componentes.

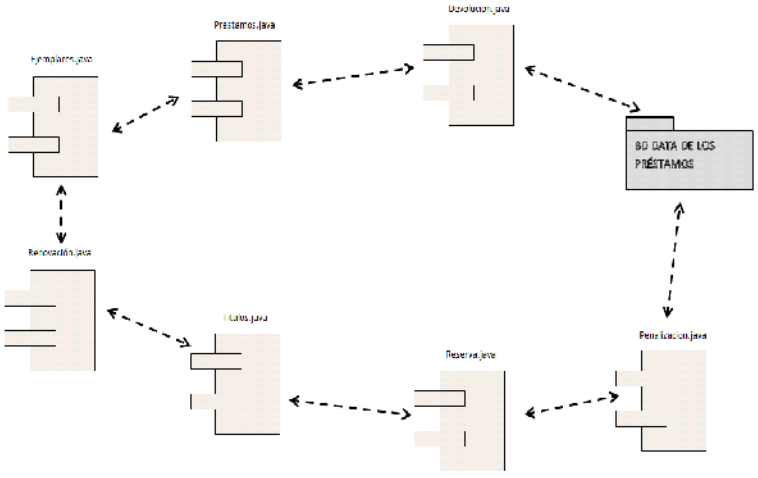

Figura 24. Diagrama de componentes - biblioteca

Sistema encargado de la gestión de los préstamos, reservas, devoluciones, sanciones de libros en una biblioteca. El lenguaje de desarrollo será Java, y los accesos a la información del prestatario serán mediante un paquete de base de datos.

d. El diagrama del despliegue.

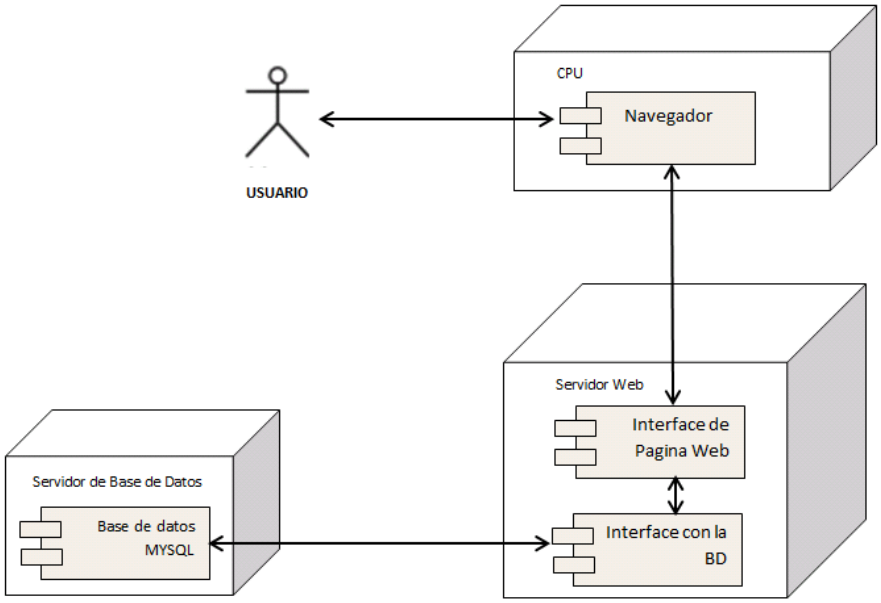

Figura 25. El Diagrama del despliegue - biblioteca

Diagrama encargado de explicarnos la arquitectura física mediante la cual el usuario y/o administrador se podrá comunicar con el sistema.

e. Los casos de uso.

En cuanto a la descripción de un proyecto es necesario identificar lo siguiente:

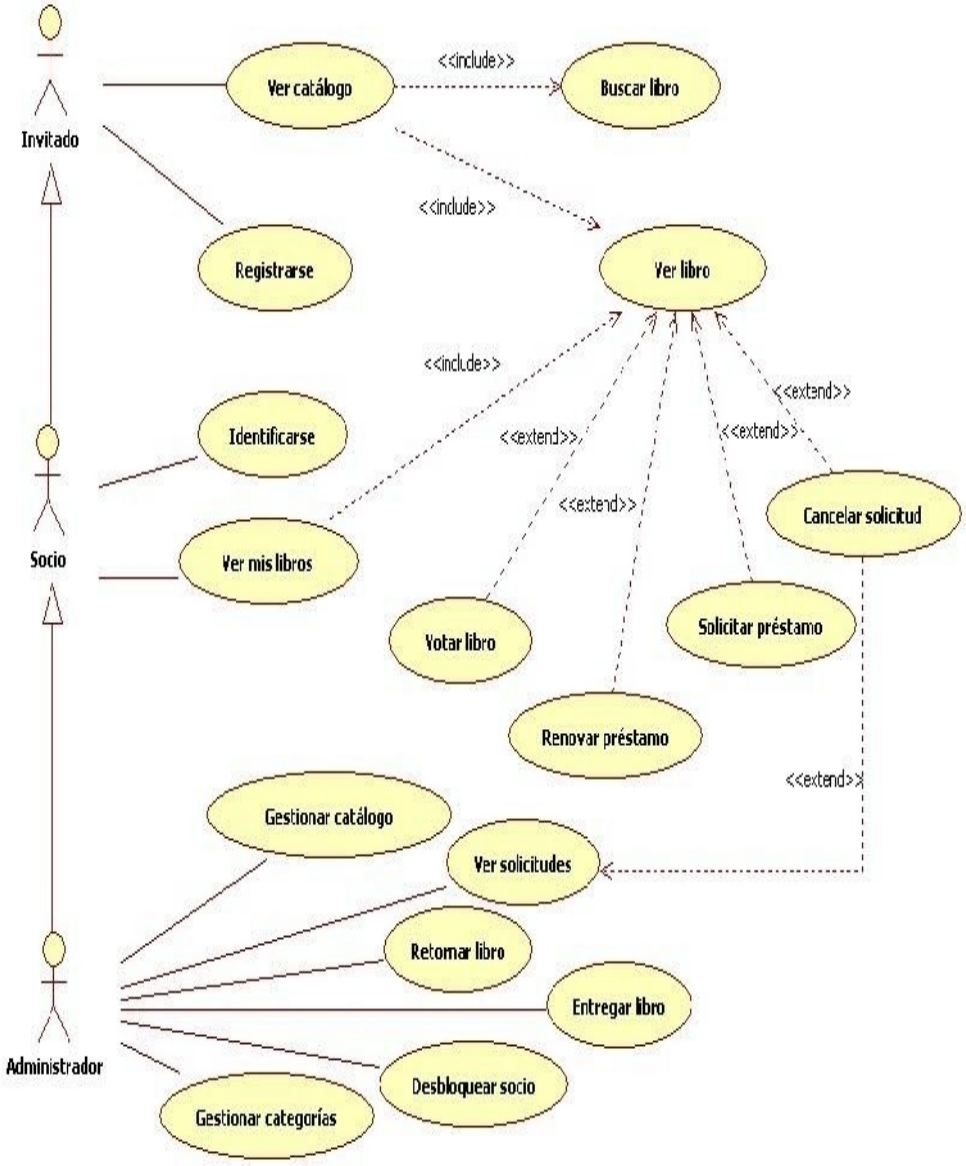

Figura 26. Diagramas de casos de uso de biblioteca

- *Caso de uso expandido: Solicitud de préstamo de libro.*

Tabla 4

Tabla en la cual explicaremos el caso de uso de solicitud de préstamo

El caso de uso	**Solicitud de préstamo de libro**
Los actores	E usuario
El propósito	Es el que realizar las solicitudes de todo el material de la biblioteca (desde los terminales de las bibliotecas)
Resumen	El interesado llega a la biblioteca y es el que accede en una de la terminales que es de consulta, acá es donde se detalla el formulario, ya que se solicita el título y también el autor del libro, su tipo de material y de forma opcional su código del propio libro.
Tipo	Primario

Referencia cruzada		
Nº Ref.	**Condición**	**Categoría**
1	El usuario visualiza en la consola otro formulario de llenado con los datos precisos del material bibliográfico.	visible
2	El usuario llena el formulario con todos los datos del material bibliográfico (título, autor y tipo del material bibliográfico) y de manera opcional el código del material bibliográfico.	visible
3	El usurario “ENVÍA” el formulario llenado.	visible

Curso de eventos	
Acción del actor	**Respuesta del sistema**
2.- El usuario introduce datos del material	1.- Genera la solicitud donde se registran los datos del usuario
4.- El usuario visualiza el estado de existencia	3.- Envía reporte a usuario mostrando estado de existencia

- *Caso de uso expandido: Realizar préstamo.*

Tabla 5

Caso de uso expandido del préstamo.

Caso de uso	**Realizar préstamo del material**
Actores	Bibliotecario y usuario
Propósito	Generar el préstamo del material
Resumen	El bibliotecario ubica el material solicitado, verifica documentos, datos del usuario, y realiza el préstamo entregando el material bibliográfico a los usuarios. Seguidamente el usuario se retira.
Tipo	Primario

Referencia cruzada

Nº Ref.	**Condición**	**Categoría**
1	Bibliotecario realiza la búsqueda del material.	visible
2	Bibliotecario pide los documentos al usuario.	visible
3	Usuario hace entrega de los documentos.	visible
4	Bibliotecario recepciona documentos.	visible
5	Bibliotecario hace entrega del material bibliográfico.	visible
6	Bibliotecario ingresa los datos del material bibliográfico prestado.	visible

Curso de eventos

Acción del actor	**Respuesta del sistema**
1.- Usuario ingresa datos de material solicitado por tipo	2.- Búsqueda de material por tipo
4.- El usuario verifica el estado de existencia	3.- Envía reporte a usuario mostrando estado de existencia, código de libro y título

- *Caso de uso expandido: Realizar devolución.*

Tabla 6

Devolución de libro

Caso de Uso	**Realizar devolución**
Actores:	Usuario, bibliotecario
Propósito:	Finalizar el proceso de circulación
Resumen:	El usuario llega con el material bibliográfico prestado a la biblioteca, el bibliotecario verifica el préstamo y registra la devolución en el sistema, el sistema verifica las reglas de préstamo para el usuario, si existen faltas genera un reporte automáticamente, dependiendo de esta acción el bibliotecario devuelve las garantías o no.
Tipo:	Primario

Referencia cruzada

Nº Ref.	**Condición**	**Categoría**
1	El usuario entra a la biblioteca con el material bibliográfico prestado	visible
2	El Bibliotecario recibe el material bibliográfico	visible
3	El Bibliotecario verifica la fecha de préstamo y la fecha devolución anteriormente asignada.	visible
4	Dependiendo de la fecha de devolución asignada y la actual, se realiza la entrega de los documentos dejados como garantía.	superfluo
5	El bibliotecario ingresa los datos del material bibliográfico devuelto.	visible

Curso de eventos

Acción del actor	**Respuesta del sistema**
1.-El bibliotecario realiza la entrega palpable del material bibliográfico solicitado. 3.- El bibliotecario hace la entrega del reporte impreso al usuario	2.- El sistema imprime un reporte donde se encuentra nombre de usuario, código de libro, titulo, fecha de préstamo y fecha de devolución

3.3.2.5. Fase de construcción.

Una vez realizada la fase de análisis, continuamos con el diseño de la aplicación presentando los diagramas de clases, estado, actividad y secuencia más significativos.

3.3.2.6. Vista lógica.

a. Diagrama de clases.

A continuación presentamos el diagrama de clases de entidad junto con sus atributos principales.

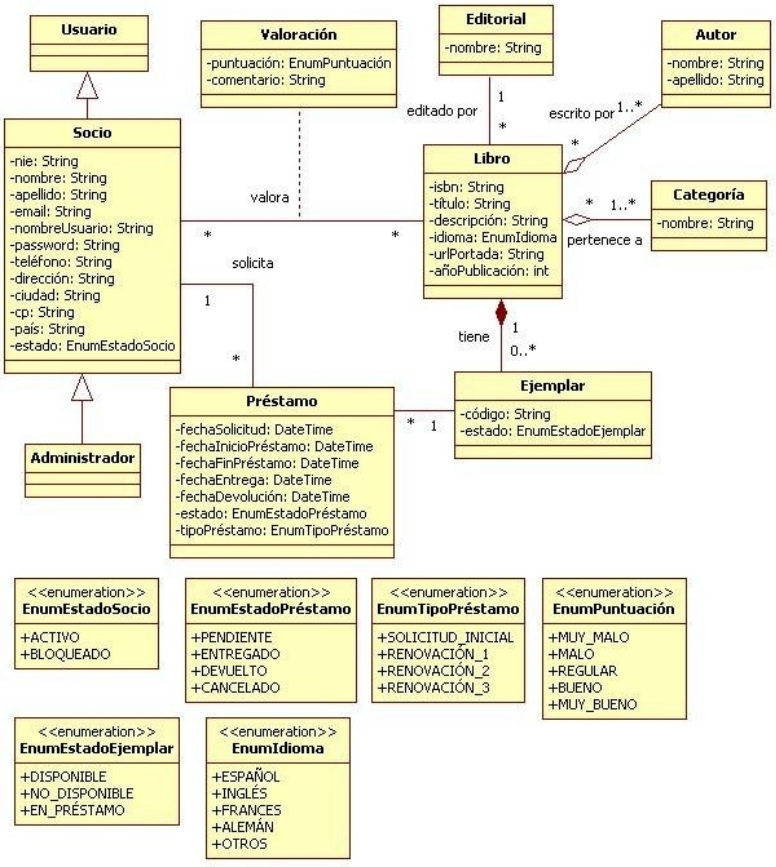

Figura 27. Diagrama de clases biblioteca

3.3.2.7. Vista de implementación.

a. Diagrama de estado.

Como parte del modelado de los aspectos dinámicos del sistema, se presentan los diagramas de estado más interesantes.

- *Estados de un socio.*

Figura 28. Diagrama de estado – estados de un socio

- *Tipos de préstamo.*

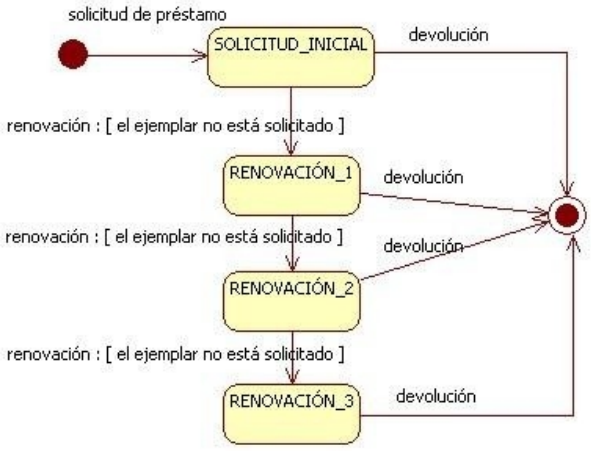

Figura 29. Diagrama de estado - tipos de préstamo

- *Estados del préstamo.*

Figura 30. Diagrama de estado - estados del préstamo

- *Estados de un ejemplar.*

Figura 31. Diagrama de estado - estados de un ejemplar

b. Diagramas de actividad.

La actividad principal de la biblioteca virtual es la solicitud, entrega, renovación y devolución de ejemplares. A continuación, modelamos el flujo de tareas de estas actividades.

- *Solicitud de préstamo.*

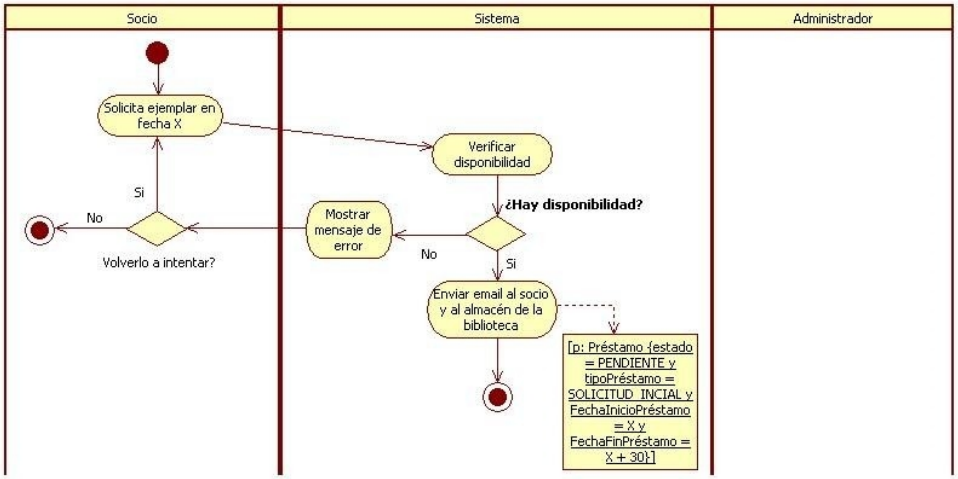

Figura 32. Diagrama de actividad - solicitud de préstamo

- *Entrega de ejemplar.*

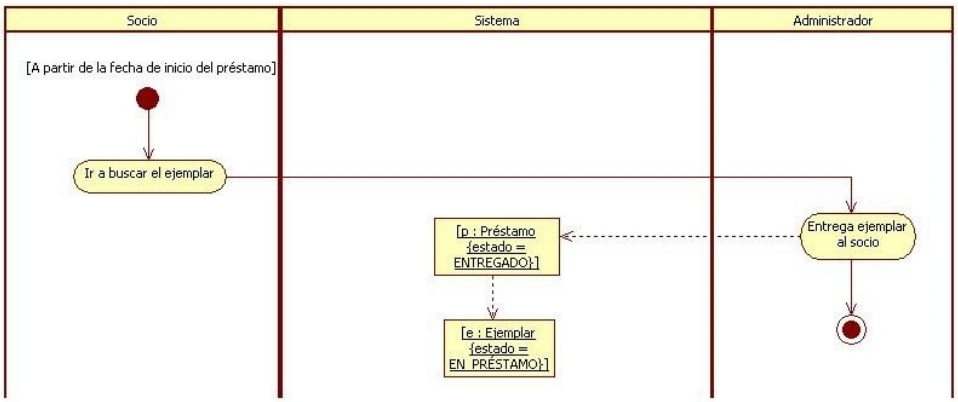

Figura 33. Diagrama de actividad - entrega de ejemplar

- *Renovación y devolución de un ejemplar.*

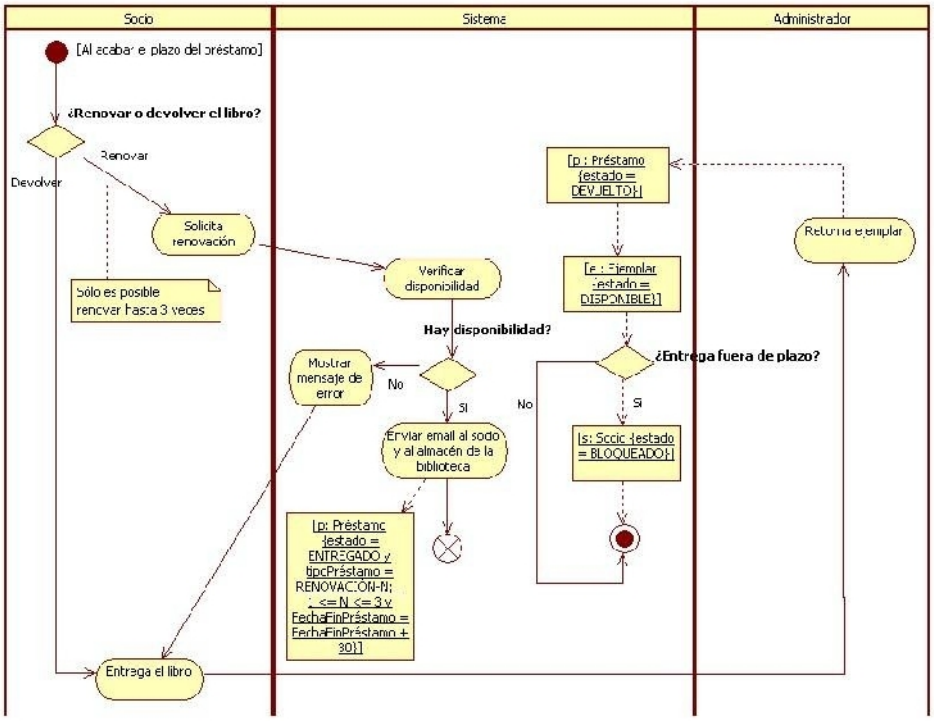

Figura 34. Diagrama de actividad – renovación y devolución de un ejemplar

c. Diagramas de secuencia

Continuando con el modelado de los aspectos dinámicos del sistema, se presentan los diagramas de secuencia para algunos de los casos de uso más representativos de la aplicación. Hay que tener en cuenta que es posible que en la etapa de implementación aparezcan otras colaboraciones de clase no representadas por estos diagramas. Sin embargo, éstos siguen siendo válidos en tanto presentan una primera visión de las interacciones entre clases que sucederán dentro de la aplicación.

- *Identificación en el sistema.*

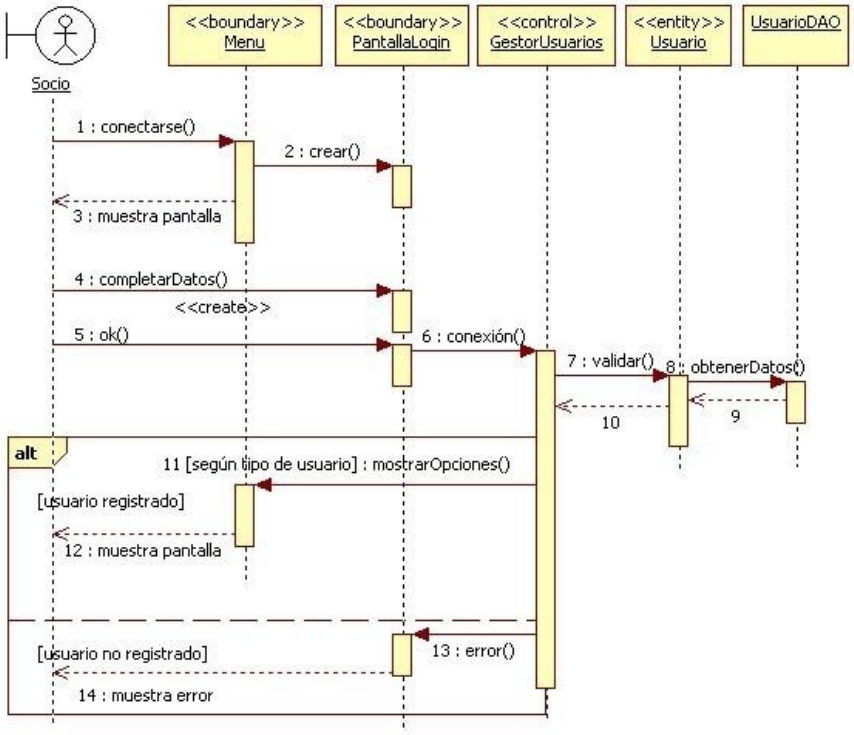

Figura 35. Diagrama de secuencia - identificación en el sistema

- *Búsqueda de libro.*

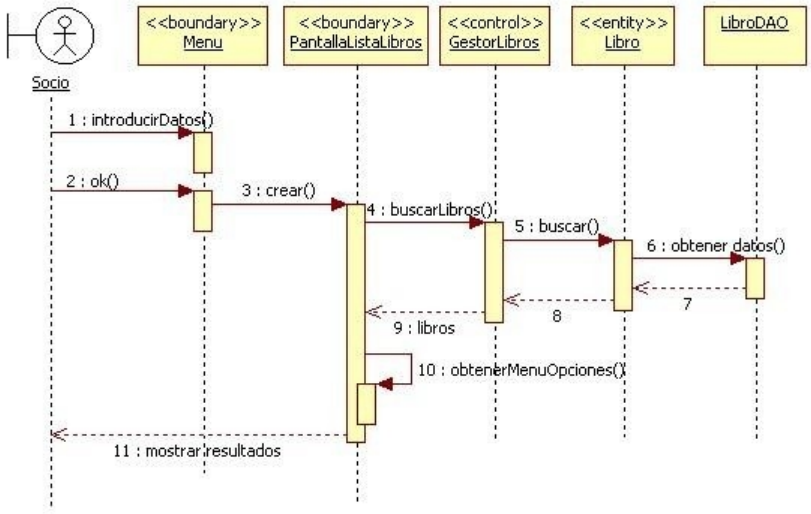

Figura 36. Diagrama de secuencia – búsqueda de libros

- *Registro de un nuevo socio.*

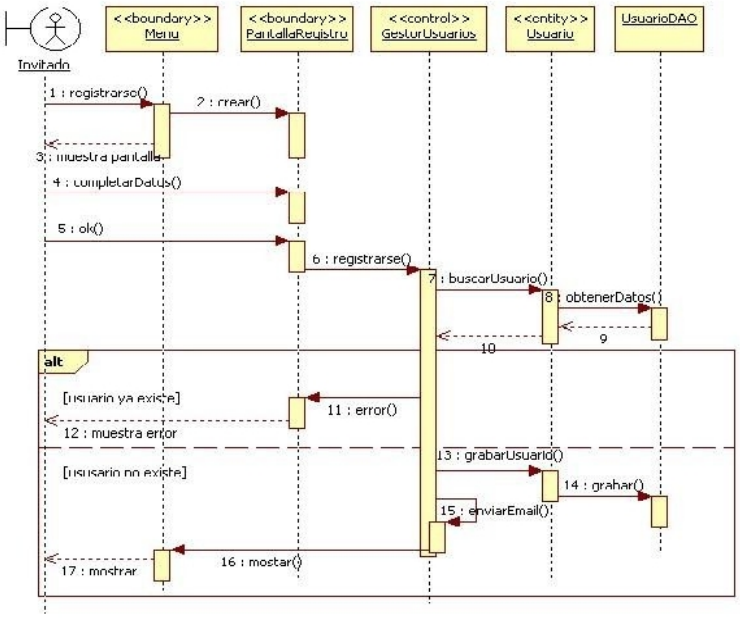

Figura 37. Diagrama de secuencia - registro de un nuevo socio

- *Ver libro.*

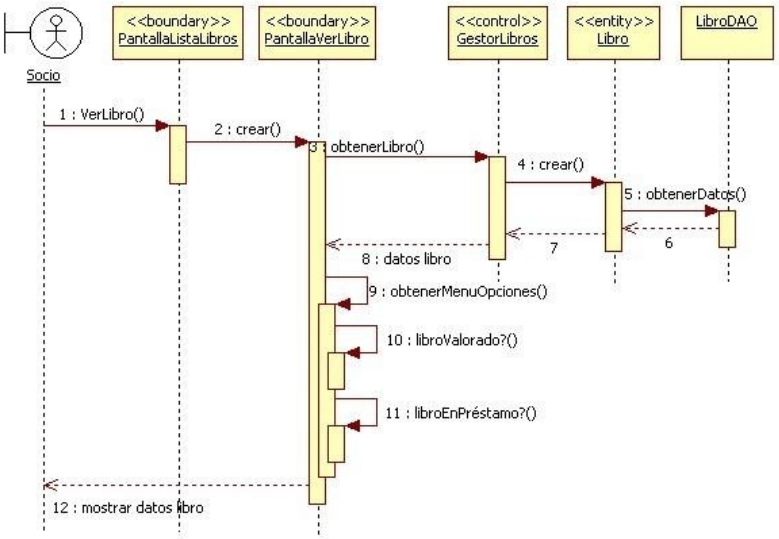

Figura 38. Diagrama de secuencia – ver libro

- *Solicitar préstamo.*

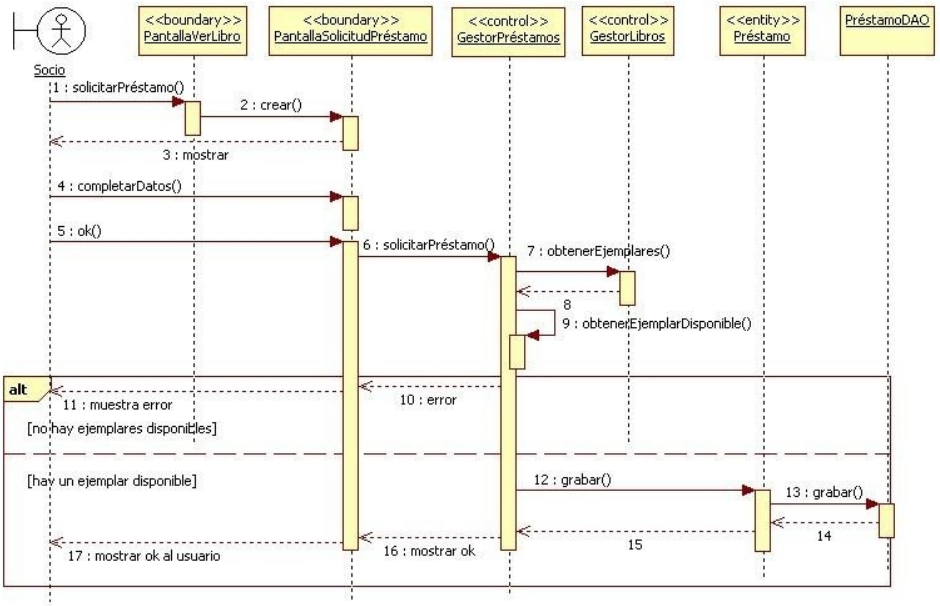

Figura 39. Diagrama de secuencia – solicitar préstamo

Los casos de uso renovar préstamo y cancelación de solicitud pueden representarse con diagramas de secuencia similares al de solicitar préstamo.

- *Entregar libro.*

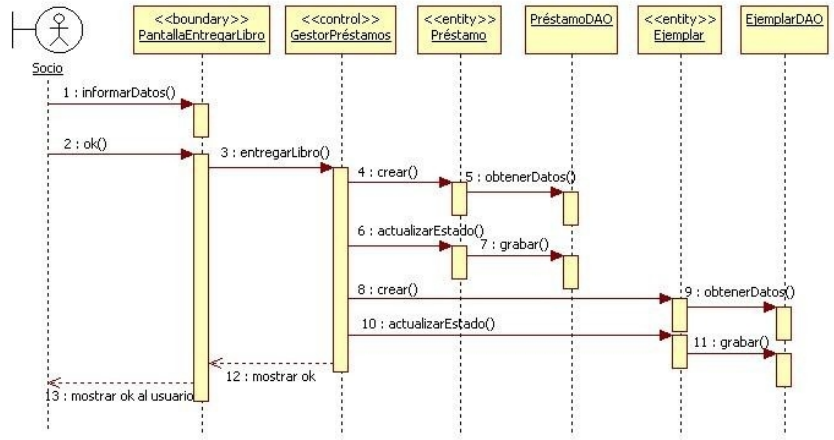

Figura 40. Diagrama de secuencia - entregar libro

De esta manera es que vemos como para este proyecto sacamos los diagramas más importantes y la lógica que va a cumplir nuestro sistema para poder satisfacer los requerimientos de uso del usuario final.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

Primera. Sin una planificación clara y realista hubiera sido imposible llevar el trabajo a buen puerto. A su vez, hemos hecho visible que utilizar una metodología rigurosa de análisis y diseño orientado a objetos permite enfocarse en gran parte del lenguaje utilizado y facilita potencialmente la posterior implementación.

Segunda. En la actualidad se continúa mejorando el análisis y desarrollo de sistemas de manera constante para poder satisfacer la demanda de las nuevas tecnologías emergentes con el análisis, debido a que tienen que tener al momento de la implementación en el campo de desarrollo de sistemas.

Tercera. Cualquier lenguaje de programación orientado a objetos se adapta de manera eficaz en el análisis y diseño en el desarrollo de sistemas.

4.2. Recomendaciones

Primera. El análisis nos permite observar de manera más profunda las posibles vulnerabilidades que podría tener nuestro sistema.

Segunda. A través de estas vulnerabilidades podemos observar la importancia que tiene la metodología orientada a objetos en el desarrollo de sistemas.

Tercera. Sin el uso de los diagramas de objetos y la tecnología UML, el desarrollo de los sistemas seria desordenado e ineficiente ya que no se podría analizar de manera correcta la real necesidad del cliente al momento de solicitar un sistema.

REFERENCIAS BIBLIOGRÁFICAS

Bauer, C. y King, G. (2005). *Hibernate in action.* Manning.

Berrio, K. y Salazar, J. (2014). *Diagrama de estados.* Recuperado de http://es.slideshare.net/still01/diagramas-de-estados-16815255

Booch, G., Rumbaugh, J. y Jacobson, I. (1999). *El lenguaje unificado de modelado*. Addison Wesley.

Cillero, M. (2015). *Metrica 3.* Recuperado de https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-interaccion/

Cockburn, A. (2001). *Writing effective use cases.* Addison-Wesley.

Connolly, T. y Begg, C. (2005). *Database Systems: A practical approach to design, implementation, and management* (4ª edición). Addison Wesley.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2011). *Patrones de Diseño*. Lugar: Addison Wesley.

García, F. (2008). *Diagramas de secuencia*. Recuperado de https://es.slideshare.net/FABIANGARCIA/diagramas-de-secuencia-presentation

Huerta, D. (2012). *Diagrama de componentes.* Recuperado de https://es.slideshare.net/dhuertacruz/diagrama-de-componentes-15705604

Larman, C. (2005). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development (3ª edición).* Lugar: Prentice Hall.

Mahmoud, Q. (2004). *"Developing web applications with java server faces"*. Sun developer network. Recuperado de http://java.sun.com/developer/technicalArticles/GUI/JavaServerFaces

Mann, K. (2004). "*Getting around JSF: The role of JSP*". Javaworld. Recuperado de https://www.javaworld.com/article/2073098/java-web-development/getting-around-jsf--the-role-of-jsp.html

Maravillas, G. (2015). *Diagramas de estado*. Recuperado de https://lalomaravillas.wordpress.com/2015/06/09/diagramas-de-estado/

Murach, J. y Steelman, A. (2008). *Java servlets and JSP (2ª edición)*. Mike Murach & Associates Inc.

Orozco, S. (2012). *Diagrama de actividad.* Recuperado de https://www.taringa.net/posts/ciencia-educacion/12484172/Diagrama-de-Actividad.html

Pérez, A. (2005*). "JSF - JavaServer Faces (y comparación con Struts)"*. Recuperado de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jsf.

Prakash, J. (2005). "*Design with the JSF architecture*". IBM DeveloperWorks. Recuperado de http://www.ibm.com/developerworks/java/library/wadsgnpatjsf.html?ca=drs-

Pressman, R. (2006). *Ingeniería del software: Un enfoque práctico (6ª edición)*. Mcgraw-Hill / Interamericana De Mexico.

Rocha, A. (2013). *Diagrama de objetos.* Recuperado de https://es.slideshare.net/still01/diagramas-de-objetos-16815266

Roman, E., Patel Sriganesh, R., Brose, G. (2005). *Mastering enterprise javabeans (3ª edición)*. Wiley.

Salinas, P. (2012). *Modelo de clases*. Recuperado de http://users.dcc.uchile.cl/~psalinas/UML/modelo.html

Slideshare. (2011). *Diagrama de colaboración.* Recuperado de http://es.slideshare.net/d-draem/diagramas-de-colaboracion-8052167

Stolfi, D. (2010). *Desarrollo de software.* Recuperado de http://www.danielstolfi.com/comvet/desarrollo-software.php

Wikispaces. (2015). *Diagrama de colaboración.* Recuperado de https://wikiUML.wikispaces.com/Diagrama+de+Colaboraci%C3%B3n