UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

DESARROLLO DE UN SISTEMA PARA EL CONTROL DEL PRE-REGISTRO DE LOS EGRESADOS EN LAS DIFERENTES FORMAS DE TITULACIÓN DE LA CARRERA DE INGENIERÍA EN COMPUTACIÓN EN LA FES-ARAGÓN

TESIS CONJUNTA

QUE PARA OBTENER EL TÍTULO DE:

INGENIERA EN COMPUTACIÓN

PRESENTA:

GABRIELA IVON MARTÍNEZ HERRERA
WENDY LEÓN ARROYO

ASESOR:

M. EN C. JESÚS HERNÁNDEZ CABRERA

México, 2011.

Agradecimientos

Agradezco mucho a Dios por haberme permitido estar aquí, por haberme dado la maravillosa familia que tengo, la cual me ha apoyado en cada etapa de mi vida y siempre conté con una palabra de aliento para salir adelante.
A mi madre, mi gitana que me dio la vida arriesgando la suya, siempre teniendo la verdad en su boca y dándome con afecto, amor y ternura una mano amiga.
A mi padre que siempre va un paso adelante que yo y me ha dado toda su paciencia.
A mi hermana que compartió conmigo su vida, sus experiencias y su amor, a mi cuñado que me dio su confianza y afecto.
Eric gracias por entrar a mi vida y hacerme ver el futuro diferente, eres el amor hecho hombre y un hombre hecho para mí.
A Gaby por compartir y ayudarme a concluir con este ciclo, eres una gran amiga.
A mi asesor M. en C. Jesús Hernández y sinodales por el tiempo brindado.
A todas las personas en la jefatura que nos apoyaron y confiaron en nosotras.
Y sobre todo por brindarme su amistad a pesar de que nos separamos por cosas del destino los tengo muy presente, Marcelo, Carlos, Vicky, Dra. Nelly, Ares, Gabriel.
A todos mis amigos pasados por ayudarme a crecer y madurar como persona.

"Hago de mi realidad un sueño... y de mis sueños, una realidad."

Wendy León Arroyo, Diciembre del 2010.

Agradecimientos

Gracias a Dios por darme la vida y una hermosa familia.

A mis padres Lorenzo y María por todo su cariño, apoyo y comprensión. Los admiro por su gran energía para enfrentar cualquier problema y saber que nunca se rendirán ante nada.

A mis hermanos Mauricio, Lorena Isabel, Rosa Elia, María Concepción, María del Carmen y Estela; por toda su confianza, cariño, apoyo y por estar conmigo en todos los momentos de mi vida, los seis han sido el mejor ejemplo para mí ya que son grandes seres humanos.

A mis pequeños sobrinos Andrés, Evelyn, Jazmín, Jimena, Daniel y Karen.

A mis amigos Simón, Maribel y mi compañera de tesis Wendy gracias por todo el apoyo que me han ofrecido y su valiosa amistad.

A los Profesores Marcelo Pérez Medel, Carlos Omar de la Rosa Delfín, Jesús Hernández Cabrera, Arcelia Bernal Díaz, Gabriel Ortiz Cordero y Ares Morales Juárez.

Gabriela Ivon Martínez Herrera

INTRODUCCIÓN

Actualmente la tecnología de información es requerida en cualquier lado ya que nos enfrentamos ante la necesidad de automatizar, administrar y organizar nuestra información para tener un control más eficaz de nuestros datos.

La carrera de Ingeniería en Computación de la Facultad de Estudios Superiores Aragón, actualmente no cuenta con un sistema para el control de información de los egresados que han seleccionado una modalidad para titularse, por tal motivo el proyecto está enfocado en realizar un sistema de control del pre-registro donde se pretende erradicar este problema, ya que la carrera cuenta con nuevas modalidades.

El sistema permitirá realizar consultas como: datos personales del egresado, información del asesor, el tema a desarrollar, la modalidad que escogió para titularse y la fecha que fue entregada su carta de autorización para desarrollar su tema.

El sistema de control para la carrera de ingeniería en computación se realizará con Postgresql, Netbeans y UML cómo herramienta de diseño que nos servirá para elaborar, especificar y documentar el sistema.

La Base de Datos es la que logrará la automatización, organización y control adecuado de nuestros datos la cual desarrollaremos con Postgresql.

Netbeans es un IDE (Entorno de Desarrollo Integrado) para el lenguaje java y en esta aplicación nos apoyaremos para hacer nuestra Interfaz Gráfica de Usuario (GUI) usando la herramienta de Matisse y las librerías de Swing.

La importancia del desarrollo de un sistema de control, es mantener la información de forma segura con un ambiente agradable para el usuario. Ya que realizará informes detallados del pre-registro de los egresados dichas consultas se podrán hacer de manera mensual, anual, por modalidad, así como también estadísticas.

Metodología.

En el Capítulo 1 veremos definición, historia y elementos de Ingeniería de Software, RUP, UML, Java, Netbeans y Postgresql.

En el Capítulo 2 plantearemos el problema y se propone una solución.

En el Capítulo 3 especificaremos, documentaremos, y elaboraremos diagramas del sistema que planteamos en el capítulo 2, con la ayuda de UML.

En el Capítulo 4 desarrollo de la base de datos donde se almacenará toda nuestra información.

Y en el último Capítulo 5 implementación del sistema y un manual de usuario.

Capítulo 1

CONCEPTOS BÁSICOS

Definición, historia y elementos de Ingeniería de Software, RUP, UML, Java, Netbeans y Postgresql.

1CONCEPTOS BÁSICOS.

1.1 Ingeniería de Software.

1.1.1 ¿Cómo y por qué surgió la Ingeniería de Software?

Durante las primeras décadas de la informática con el avance de la tecnología se empezó a manejar mayor cantidad de datos y cálculos en los ordenares, el problema era que se programaba sin métodos ni planificación, se trabajaba con la idea de "Codificar y corregir", no existían documentación de ningún tipo y su desarrollo era a base de prueba y error, ésto provocaba que existieran dificultades en estimar tiempos, eran elevadas las cargas de mantenimiento, se hallaban con baja calidad y fiabilidad del producto, con esto se tenía una dependencia con los realizadores y ocurrían errores costosos por fallas en el software a ésto se le llamó "Crisis del software" por estos motivos fue creada la Ingeniería del software.

El término Ingeniería de Software, fue utilizado por primera vez en Octubre de1968 por Fritz Bauer, en la conferencia sobre desarrollo del software de la OTAN(Organización del Tratado del Atlántico Norte) convocada en Garmish, Alemania, estimulando una aplicación inteligente de los aspectos técnicos, administrativos, disciplinarios y cuantificables al desarrollo y mantenimiento del software.

Dado lo anterior, el objetivo de la ingeniería de software es lograr productos de software de calidad y bajo costo (desde sus inicios hasta su forma final y durante su elaboración), mediante un proceso apoyado por métodos y herramientas.

1.1.2 ¿Qué es la Ingeniería de Software?

Es una disciplina de la Ingeniería que comprende métodos y técnicas desde las etapas iniciales hasta el mantenimiento del software, apoyándose de procesos, métodos y herramientas para que su desarrollo sea funcional y así obtener un software económico, confiable que funcione eficientemente en problemas de todo tipo.

Actualmente, los procesos de la ingeniería del software (que son muchos y variados) se aplican en todas las empresas y organismos en los que se desarrolla software de forma profesional y rigurosa, porque no hay otro modo de asegurar que el producto se va a terminar dentro de los plazos y costos previstos, y que éste va a funcionar correctamente y se va a ajustar a los niveles de calidad que el mercado exige.

1.1.3 Elementos que la componen la Ingeniería de Software.

La ingeniería de software se compone por los siguientes elementos: **métodos, herramientas, procedimientos y calidad (Figura 1.1)** éstos facilitan al administrador a controlar el proceso de desarrollo de software y ayuda a construir software de alta calidad.

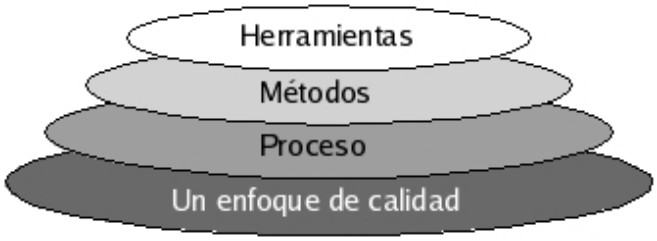


Figura 1.1 Componentes de la Ingeniería de Software

A continuación se describen:

- **HERRAMIENTAS**

Las herramientas de ingeniería de software son los métodos necesarios para desarrollar el sistema. Facilitan un soporte automático o semiautomático para los métodos y procesos. Cuando se utilizan las herramientas de forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo del software que se denomina ingeniería de software asistido por computadora (CASE, Computer Aide Software Engineering).

Case:
Es un conjunto de métodos, programas y técnicas que ayudan a la automatización del desarrollo del software, durante el ciclo de vida de un sistema completamente o en alguna de sus fases.

Sus objetivos son:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Reducir tiempo y costo de desarrollo.
- Aumentar la calidad.
- Ayudar a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.

No hay una clasificación específica de herramientas CASE y, en ocasiones, es complicado ubicarlas en una categoría.

Algunas clasificaciones pueden ser por:

- Las plataformas que soportan.
- Las fases del ciclo de vida en el desarrollo de sistemas que apoyan.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

La siguiente clasificación está basada en las fases del ciclo de desarrollo que cubren:

1. Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): Abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.

2. Herramientas de alto nivel, U-CASE (Upper CASE - CASE superior) o front-end: orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

3. Herramientas de bajo nivel, L-CASE (Lower CASE - CASE inferior) o back-end: dirigidas a las últimas fases del desarrollo: construcción e implantación.

Algunos ejemplos de herramientas CASE:

- **ASADAL**: Herramienta CASE especializada en Sistemas de Tiempo Real
- **System Architect**: herramientas CASE para Análisis y Diseño, incluye técnicas estructuradas y orientadas a objetos.
- **Win A&D**: herramientas CASE para Análisis y Diseño, incluye técnicas estructuradas y orientadas a objetos.
- **CRADLE**: conjunto de herramientas CASE integradas que dan soporte a la Planificación estratégica, Análisis y Diseño.
- **PowerDesigner 7.0**: herramienta CASE de Análisis y Diseño incluye capacidades de generación relacional y con orientación a objetos.
- **SilverRun**: Conjunto integrado de herramientas CASE para el modelado de negocios.
- **Z4-CASE**: herramienta CASE que apoya el desarrollo de aplicaciones de Software abarcando desde el análisis de procesos y requerimientos hasta la generación y pruebas del código de los sistemas.
- **ArgoUML**: herramienta CASE gratuita (licencia GPL), permite trabajar con UML 2, Soporta gran cantidad de diagramas, genera código para Java, C++ e IDL
- **Enterprise Architect**: Herramienta CASE que combina la última especificación UML 2.1. alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación.

➢ MÉTODOS

Los métodos de la ingeniería de software indican cómo construir técnicamente el software. Estos métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

Los métodos de la ingeniería de software introducen frecuentemente una notación especial orientada al lenguaje o gráfica y a un conjunto de criterios para la calidad del software.

Los métodos abarcan una gran gama de tareas que incluyen: análisis de requerimientos del sistema y del software, diseño de procedimientos algorítmicos, codificación, pruebas y mantenimiento, al conjunto de estas fases se le denomina **ciclo de vida del software (Figura 1.2)**.

El término ciclo de vida del software describe el desarrollo de software, desde la fase inicial hasta la fase terminal. El propósito de las distintas fases intermedias que se requieren es garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo.

El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en costos.

Etapas del ciclo de vida del software:

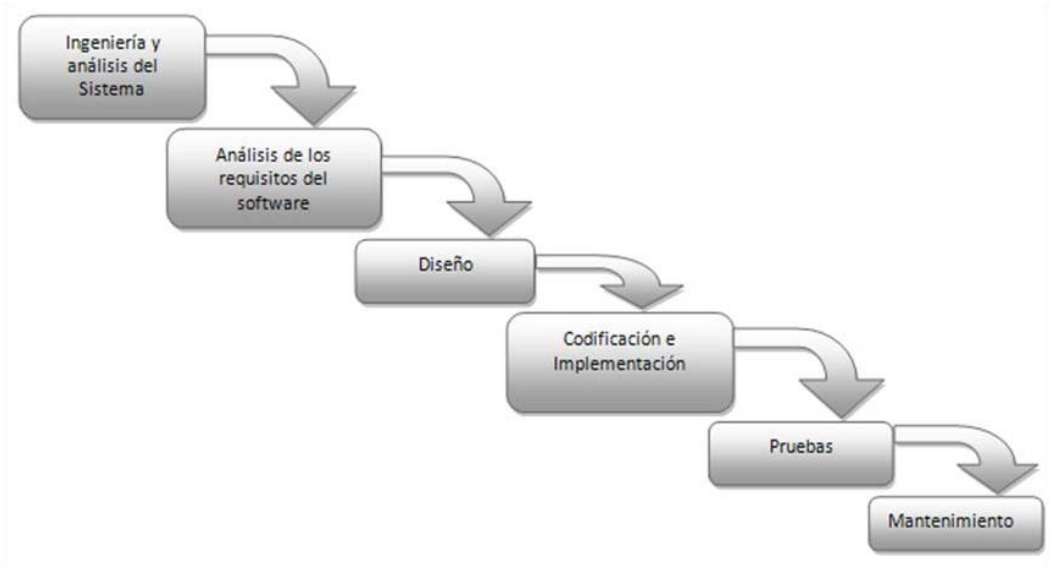


Figura 1.2 Ciclo de vida del software

1.- Ingeniería y análisis del Sistema.

- Ubicación del entorno donde se encuentra dicho problema.
- Entender y comprender la problemática.
- Identificar las necesidades del cliente.
- Establecer restricciones de costo y tiempo.
- Crear una definición del sistema que sea la base de todo el trabajo posterior.

2.- Análisis de los requisitos del software

- Obtener la información necesaria y suficiente para la solución del problema.
- Especificación precisa y completa a partir de los requisitos establecidos por el cliente.
- Indica la interfaz con otros elementos del sistema.
- El resultado de este análisis es el documento ERS (Especificación de Requisitos del Sistema).

3.- Diseño

- Conocer las relaciones entre los módulos del programa y garantizar que se cumplan.
- Se diseña la estructura general del programa.
- El problema se divide en submódulos para un mejor manejo.
- Se obtiene el SDD (Documento de diseño de software). Contiene la estructura global del sistema y la especificación de lo que debe hacer cada una de sus partes.

4.- Generación de código e Implementación

- Traducción del diseño a un lenguaje de programación.
- Se prueba individualmente cada submódulo de la aplicación.
- Se desarrolla el programa con alguna herramienta CASE.

5.- Pruebas

- Aquí garantizas que el sistema no contiene errores de diseño ni programación.
- Se ensamblan los módulos para componer el sistema.

- Se comprueba que se cumplen criterios de corrección y calidad.
- El resultado de las pruebas es el producto final listo para entregar.

6.- Mantenimiento

Gestión de cambios en el software debidos a:

- Correctivo; Errores durante el desarrollo
- Adaptativo; Adaptación a nuevos entornos. Ej. Sistema operativo
- Perfectivo; Mejoras funcionales o de rendimiento

➢ PROCEDIMIENTOS

El proceso define un marco de trabajo para un conjunto de áreas clave, las cuales forman la base del control de gestión de proyectos de software y establecen el contexto en el cual: se aplican los métodos técnicos, se producen resultados de trabajo, se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.

Los procedimientos de la ingeniería de software. Es el pegamento que une a los métodos y herramientas, facilita un desarrollo racional y oportuno del software de computadoras.

Los procedimientos definen la secuencia en la que se aplican los métodos, las entregas que se requieren y los controles que ayuden asegurar, la calidad y coordinar los cambios y las guías que facilitan a los administradores de software establecer el desarrollo.

➢ CALIDAD

Un software de calidad es aquel que cumple con los siguientes requerimientos; adaptable, confiable y aceptable.

- Adaptable: el software debe ser diseñado de tal manera, que permita ajustarse a los cambios que requiera el cliente.
- Confiable: Debe ser seguro y aprueba de fallas.
- Eficiente: tiene que ver con el uso eficaz de los recursos que necesita un sistema para su funcionamiento.

1.2 Proceso de desarrollo de software.

Un proceso de software es un conjunto de actividades y resultados asociados que tiene como propósito la producción eficaz y eficiente de un producto de software que reúna los requisitos del cliente (Figura 1.3).

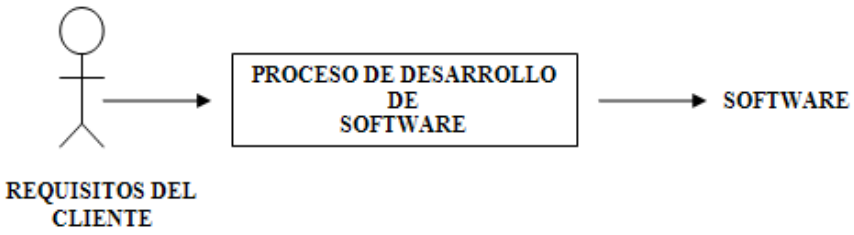


Figura 1.3 Proceso de desarrollo de software

1.2.1 Elementos de proceso de desarrollo de software

Los elementos en el proceso de desarrollo de software (Figura 1.4) son los siguientes:

- Un marco común del proceso, define un número de actividades del marco de trabajo que son aplicables a los proyectos de software, con independencia del tamaño o complejidad.
- Un conjunto de tareas, es una colección de tareas de ingeniería del software, hitos de proyectos, entregas y productos de trabajo del software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto de software.
- Las actividades de protección, tales como garantía de calidad del software, gestión de configuración del software y medición, abarcan el modelo del proceso. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

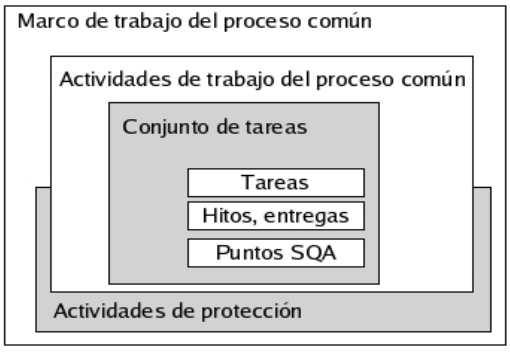


Figura 1.4: Elementos del proceso de software

1.2.2 RUP (Racional Unified Process)

Es un proceso de desarrollo de software y junto con UML (Lenguaje Unificado de Modelado), es uno de los procesos más generales, ya que está pensado en adaptarse a varios tipos de sistemas, diferentes áreas de aplicación, tipos de organizaciones y tamaños de proyectos.
RUP se caracteriza por ser: dirigido por casos de uso, centrado en la arquitectura, y es iterativo e incremental.

- **Dirigido por Casos de Uso**

RUP se basa en casos de uso para representar lo que se espera del software, apoyándose de UML como la herramienta principal.
Una característica de los casos de uso es que modelan los requerimientos funcionales del sistema.
Los casos de uso también guían el proceso de desarrollo (diseño, implementación, y prueba).

- **Centrado en la Arquitectura**

La arquitectura de un sistema software se describe mediante diferentes vistas del sistema en construcción, incluye los aspectos estáticos y dinámicos más significativos del sistema.
La arquitectura está relacionada con la toma de decisiones de cómo construir el sistema. Se basa en diseñar una arquitectura que sea fácil de modificar, comprensible y que se fundamenta en la reutilización de sus componentes.
Los casos de uso y la arquitectura están profundamente relacionados. Los casos de uso deben encajar en la arquitectura, y a su vez la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y a futuro.

- **Iterativo e incremental**

Un proceso iterativo e incremental en donde se divide el desarrollo de un proyecto de software en partes más pequeñas o mini proyectos.
Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos a crecimientos en el producto.
En cada iteración los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, para implementar dichos casos de uso. Cada iteración se analiza cuando termina, si la iteración cumple sus objetivos, se continúa con la próxima.

- Reduce el riesgo de retrasos, atacando los más importantes.
- Acelera el desarrollo. Los trabajadores trabajan de manera más eficiente al obtener resultados a corto plazo.

- Tiene un enfoque más realista al reconocer que los requisitos no pueden definirse completamente al principio.

CICLO DE VIDA (RUP)

El RUP maneja un ciclo de vida (Figura 1.5), el cual divide el proceso de desarrollo en fases, teniendo un producto final al culminar cada una.

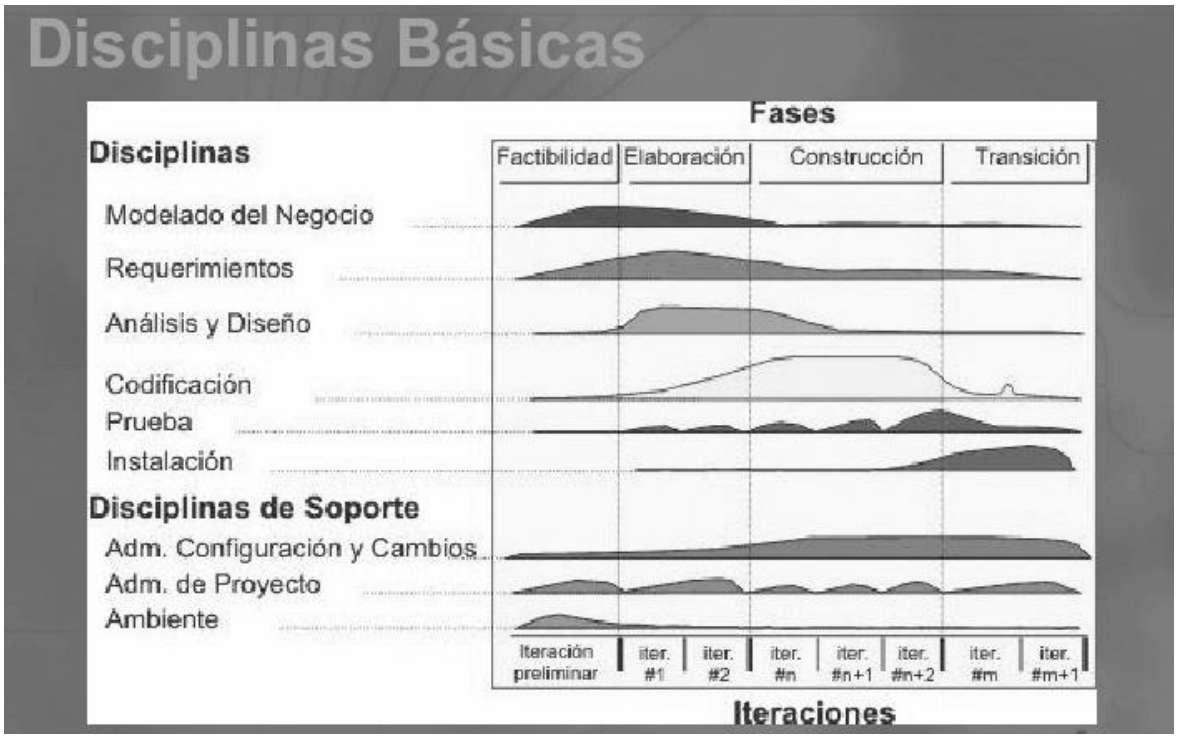


Figura 1.5 Ciclo de vida (RUP).

HITO.

Cada fase finaliza con un hito. Cada hito se determina por la disponibilidad de un conjunto de modelos o documentos que han sido desarrollados hasta alcanzar un estado predefinido. Los hitos tienen muchos objetivos. El más crítico es que los directores deben tomar ciertas decisiones antes de que el trabajo continúe con la siguiente fase.
Los hitos también permiten controlar la dirección y avance del trabajo.

- **Fase de Inicio**

Durante la fase de inicio se desarrolla una descripción del producto final, y se presenta el análisis del negocio.
En esta fase se identifican y anticipan los riesgos más importantes. Se identifican todos los casos de uso.
La fase de inicio finaliza con el **Hito de Objetivos del Ciclo de Vida,** que se obtiene cuando:

- Cuál es el conjunto de necesidades del negocio, y que conjunto de funciones satisfacen estas necesidades.
- Una planificación preliminar de iteraciones.
- Una arquitectura preliminar.

- **Fase de elaboración**

Establecen una firme comprensión del problema a solucionar, elimina los mayores riesgos, se complementan los casos de uso que describen la funcionalidad del sistema y se diseña una arquitectura.

La fase de elaboración finaliza con el **hito de la Arquitectura del Ciclo de Vida**, se obtiene cuando:

- Los casos de uso describen la funcionalidad del sistema.
- La arquitectura es estable.
- Los mayores riesgos han sido moderados

- **Fase de Construcción**

Durante la fase de construcción todos los componentes, características y requisitos deben de ser implementados, integrados y probados para obtener un producto aceptable. La línea base de la arquitectura crece hasta convertirse en el sistema completo.
Al final de esta fase, el producto contiene todos los casos de uso implementados, sin embargo puede que no esté libre de defectos.

La fase de construcción finaliza con el **hito de Capacidad Operativa Inicial**, se obtiene cuando:

- El producto es estable para ser usado
- Todas las partes están listas para comenzar la transición

- **Fase de Transición**

La fase de transición es poner el producto en las manos de los usuarios finales, preparar al usuario en el manejo del producto.

La fase de transición finaliza con el **hito de Lanzamiento del Producto,** se obtiene cuando:

- Se han alcanzado los objetivos fijados en la fase de Inicio.
- El usuario está satisfecho.

1.2.3 PROCESO DE SOFTWARE PERSONAL (PSP).

El proceso de software personal (PSP, Personal Software Process) es un modelo para la mejora del proceso de desarrollo de software, está basado en la creencia de que la calidad del software depende del trabajo de cada uno de los ingenieros; por lo cual, el proceso debe ayudar a controlar, manejar y mejorar el trabajo de éstos. El objetivo de PSP es mejorar la planeación del trabajo, conocer con precisión el desempeño, medir la calidad de los productos y mejorar las técnicas para su desarrollo. La instrumentación de esta tecnología consiste en lo que se conoce como la evolución del PSP. Se siguen ciertos pasos comenzando con las líneas base PSP0 y PSP0.1 el proceso personal de planeación PSP1 y PSP1.1, el manejo personal de calidad PSP2 y PSP2.1, y por último, el proceso personal cíclico PSP3 (Figura 1.6)

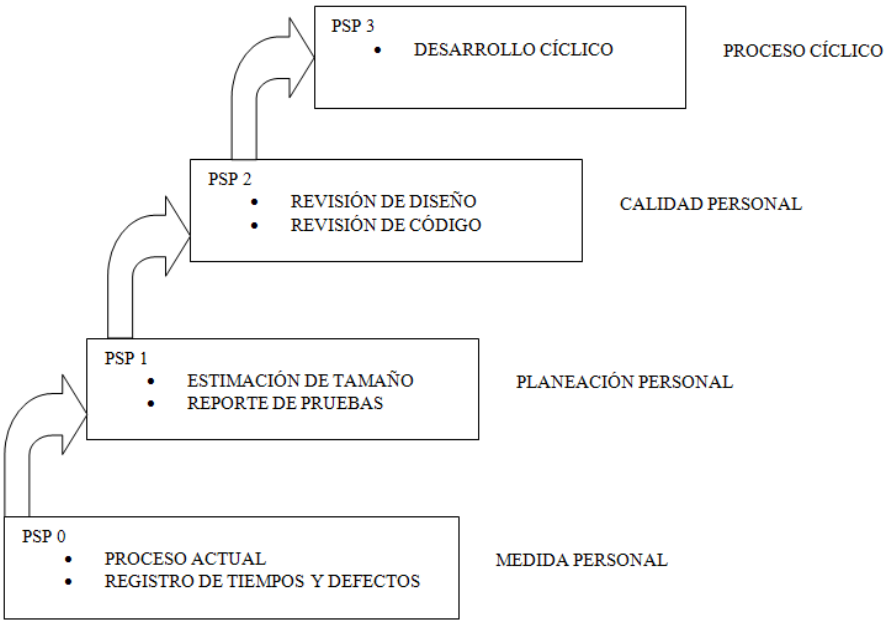


Figura 1.6 Niveles de PSP

1.2.4 PROCESO DE SOFTWARE EN EQUIPOS (TSP)

El proceso de software en equipos, (TSP, Team Software Process) extiende el modelo PSP e integra los aspectos del desarrollo de software realizados por equipos de trabajo, definiendo aspectos como la asignación y control de taréas para los diversos miembros del equipo (Figura 1.7). TSP define un marco de trabajo en equipos con los objetivos de:

1. Desarrollar productos en varios ciclos.
2. Proporcionar métricas para equipos.
3. Evaluar roles y equipos.
4. Ofrecer guías para la solución de problemas de equipo.

Para lograr estos objetivos:

- Se establece el producto y objetivos que se tienen que alcanzar.
- Se designan roles definidos y objetivos de equipo.
- Definen la estrategia de desarrollo.
- Se elabora un plan general y otro de calidad para identificar, encontrar y prevenir problemas.
- Diseñarán reportes para la administración.
- Cada cierto tiempo se tiene una revisión con la administración.

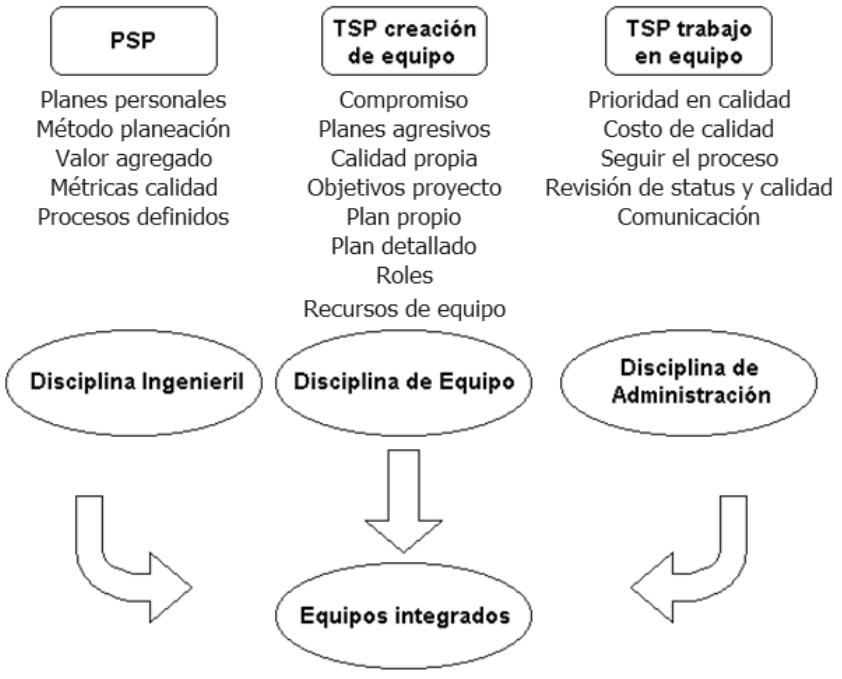


Figura 1.7 Proceso de software en equipos (TSP)

1.3 TECNOLOGÍA ORIENTADA A OBJETOS.

La tecnología orientada a objetos es una metodología de diseño de software que modela las características de objetos reales o abstractos por medio del uso de clase y objetos

1.3.1. HISTORIA

En la tecnología tradicional fue hecha en una manera secuencial o lineal, es decir una serie de pasos consecutivos con estructuras consecutivas y bifurcaciones.

Los lenguajes basados en esta forma de programación ofrecían ventajas al principio, después la tendencia de crear programas cada vez más grandes y complejos.

En los lenguajes basados en la programación estructurada la idea principal es separar las partes complejas del programa en módulos o segmentos que sean ejecutados conforme se requieran. Así obtenemos un diseño modular, compuesto por módulos independientes que se comunican entre sí (Figura 1.8).

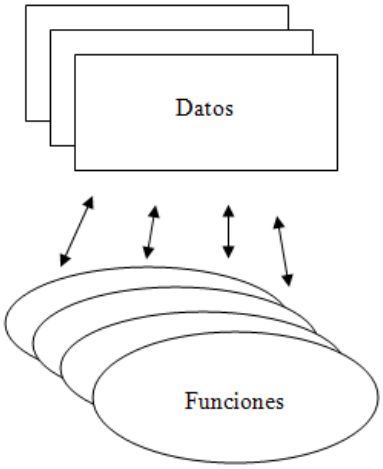


Figura 1.8 Programación Estructurada

La evolución que se iba dando en la programación conducía siempre a ir descomponiendo más el programa, esto conduce directamente a la programación orientada a objetos.

Es así como aparece la programación orientada a objetos (POO). La POO viene de la evolución de la programación estructurada; básicamente la POO simplifica la programación con la nueva filosofía y nuevos conceptos que tiene. La POO se basa en dividir el programa en pequeñas unidades lógicas de código llamada objetos. Los objetos son unidades independientes que se comunican entre ellos mediante mensajes.

La tecnología orientada a objetos ya no se aplica solamente a los lenguajes de programación, también se aplica en el análisis, diseño y en las bases de datos. Para obtener una adecuada programación orientada a objetos hay que desarrollar todo el sistema aplicando esta tecnología.

La programación orientada a objetos es actualmente una de las formas más populares de programar para el desarrollo de proyectos de software y se debe a sus grandes capacidades y ventajas como:

- Permitir un modelado más natural al mundo real.
- Facilitar la reutilización, extensión del código, creación de programas visuales, trabajo en equipo, y mantenimiento de software.
- Ofrecer mecanismos de abstracción para mantener controlable de sistemas complejos.

El modelamiento visual es la clave para realizar el análisis Orientado a Objetos. UML (Unified Modelling Languaje) se ha convertido en el estándar para el diseño de software orientado a objetos.

1.3.2 ELEMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS

A diferencia de la programación tradicional, la programación orientada a objetos define una estructura de más alto nivel llamado objeto, que ofrece dos ventajas sobre la programación tradicional.

a) La primera es permitir al programador que organice su programa de acuerdo con abstracciones de más alto nivel, siendo éstas más cercanas a la manera de pensar de la gente. En otras palabras, los objetos son las unidades de representación de las aplicaciones, por ejemplo, cuentas de banco, reservación de vuelos etcétera.

b) La segunda es que los datos globales desaparecen, siendo éstos junto con las funciones parte interna de los objetos. Por lo tanto, cualquier cambio en la estructura de alguno de los datos sólo debiera afectar las funciones definidas en ese mismo objeto y no en los demás.

En general, un programa orientado a objetos se define exclusivamente en términos de objetos y sus relaciones (Figura 1.9).

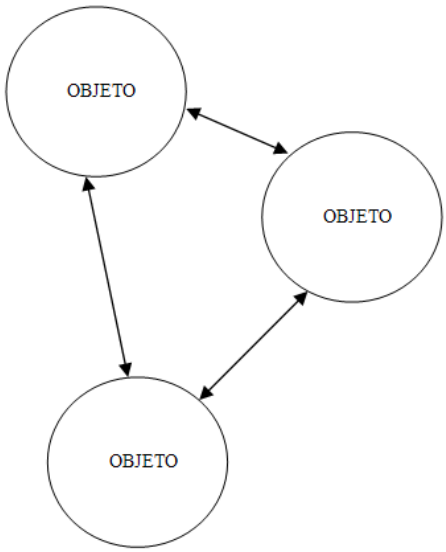


Figura 1.9 Objetos y sus relaciones

Los datos y funciones se guardan dentro de objetos. Los datos están ubicados en el centro del objeto (un concepto puramente ilustrativo), lo cual hace que un cambio en su estructura sólo afecte, las funciones del mismo objeto, pero no al resto de la aplicación (Figura 1.10).

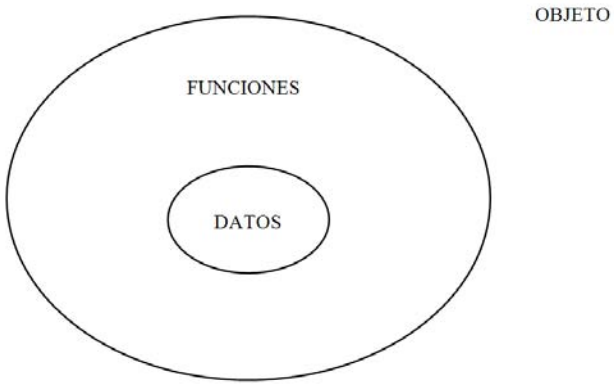


Figura 1.10 Programación orientada a objetos

Los elementos principales de Programación Orientada a Objetos son:

- Objeto
- Clase
- Herencia
- Envió de mensaje

Objeto

Entender que es un objeto es fundamental para cualquier lenguaje orientado a objetos. Un objeto es lo mismo que en la vida real, es cualquier cosa que vemos a nuestro alrededor. Un objeto poseé atributos y métodos (o funciones).

- Atributos son los que representan los datos asociados al objeto, que serían sus propiedades o características.
- Un método es una función o subrutina asociada a un objeto la cual es llamada para una determinada tarea.

Un objeto mantiene sus características en una o más variables(o datos), e implementa su comportamiento con métodos.

Por ejemplo el modelado de una bicicleta[1] (Figura 1.11):

Bicicleta modelada como un objeto:

Atributos:
- **Velocidad**
- **Cadencia**
- **Marcha**

Métodos:
- **Cambiar marcha**
- **Frenar**
- **Cambiar cadencia**

Figura 1.11 Modelado de una bicicleta

Clase:

Es una plantilla que describe un conjunto de objetos con atributos y métodos similares. Como en el mundo real existen varios objetos del mismo tipo, esto es denominado clase.

[1]El ejemplo fue sacado de http://java.sun.com/docs/books/tutorial

Ejemplo, la clase bicicleta (Figura 1.13) especificaría:

Atributos: Velocidad actual
Cadencia actual
Color
Marcha actual
Métodos: Cambiar marcha
Frenar
Cambiarla cadencia de pedaleo

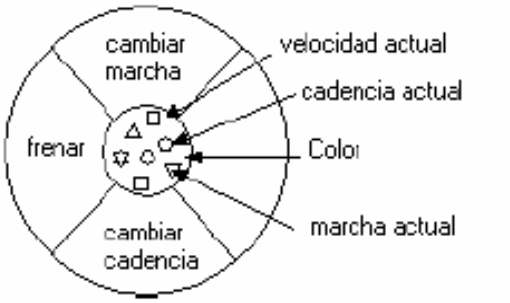


Figura 1.13 Clase bicicleta

Cuando en un objeto existe una representación concreta y específica de una clase, esto se le llama instancia de la clase (Figura 1.14).

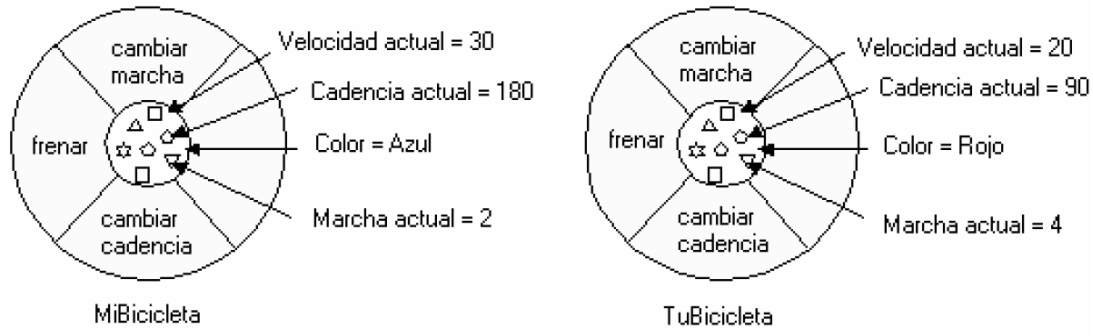


Figura 1.14 Objeto o instancia de la clase bicicletas

Herencia: Consiste en permitir crear nuevas clases (subclase) partiendo de otras ya existentes, estas heredan automáticamente sus atributos y métodos de la clase base (superclase). Una subclase se puede personalizar con atributos y métodos propios. No se está limitado a un solo nivel de herencia, se puede tener todos los que se consideren necesarios.

Ejemplo, la superclase es bicicleta y las subclases son las bicicletas de montaña, las de carretera y las de tándem, estas comparten algunos métodos: frenar y cambiar la cadencia de pedaleo entre otros (Figura 1.15).

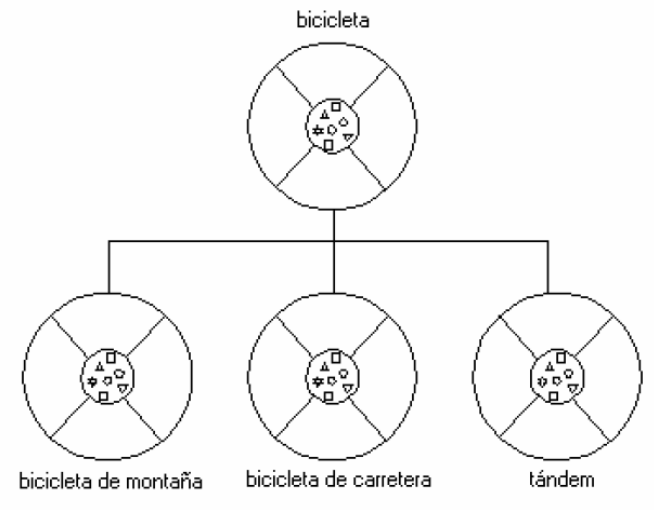


Figura 1.15 Herencia en la clase y subclases bicicletas

Envió de mensajes: La interacción entre objetos se produce mediante mensajes. Los mensajes son llamados a métodos de un objeto en particular.

Un mensaje está compuesto por los siguientes tres elementos:

1. El objeto destino, hacia el cual el mensaje es enviado
2. El nombre del método a llamar
3. Los parámetros solicitados por el método

Como todo lo que un objeto puede hacer está expresado mediante métodos, el envío de mensajes soporta todas las posibles interacciones entre objetos.
Para enviar o recibir mensajes, los objetos no necesitan formar parte del mismo proceso, ni siquiera de la misma máquina.

Ejemplo (Figura 1.16): 1. El objeto al cual se manda el mensaje (Tu Bicicleta).
2. El método o función miembro que debe ejecutar (Cambiar De Marcha).
3. Los parámetros que necesita ese método (Marcha)

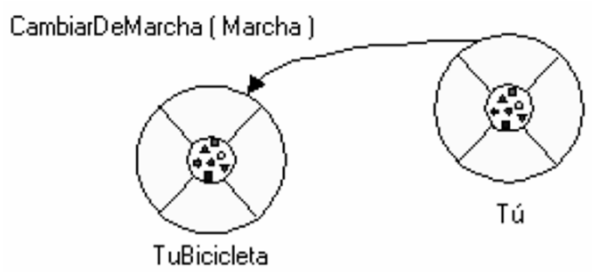


Figura 1.16 Envió de mensajes

1.4 UML

1.4.1 Que es UML?

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientada a objetos. Tal como indica su nombre, UML es un lenguaje de modelado. Un modelo es una simplificación de la realidad. Los objetivos principales del modelado de un sistema es capturar las partes esenciales del sistema, especificando o describiendo métodos y procesos, para obtener de manera más sencilla una solución al problema.

Algunas funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: permite especificar las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas.
- Documentar: Los elementos gráficos sirven como documentación del sistema desarrollado, por lo cual pueden servir para su futura revisión ya que los diagramas están detallados.

1.4.2 DIAGRAMAS DE UML.

Un diagrama es la representación gráfica de un conjunto de elementos y las relaciones entre ellos, ofrece una vista del sistema a modelar, para poder representar correctamente un sistema.

En UML existen 13 tipos diferentes de diagramas y se categorizan en (Figura 1. 17):

- **Diagramas de Estructura**: Resaltan los elementos que deben existir en el sistema.
- **Diagramas de Comportamiento**: Resaltan lo que debe suceder en el sistema.
- **Diagramas de Interacción**: Son un subtipo de diagramas de comportamiento, que resaltan el flujo de control y de datos entre los elementos del sistema.

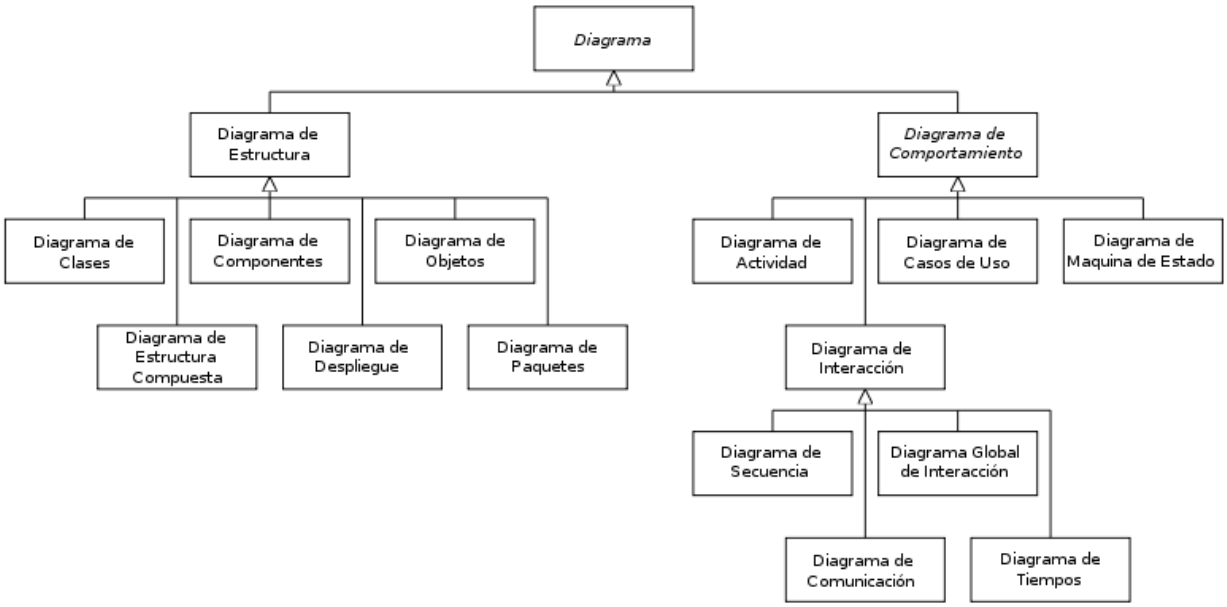


Figura 1.17 Diagramas UML

Los principales diagramas UML utilizados son:

• **Diagrama de Casos de Uso.**

Un diagrama de casos de uso ayuda a ver el comportamiento que tendrá nuestro sistema, observando como será la interacción con el usuario en un escenario cercano al real (Figura 1.18).

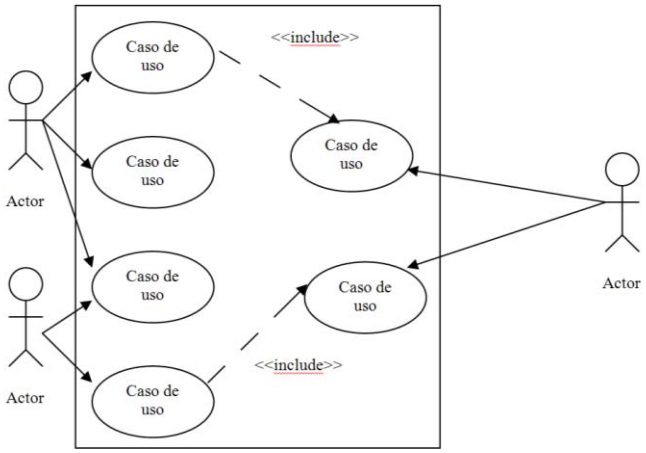


Figura 1.18 Caso de Uso

Los casos de uso tienen como componentes principales Actor, Casos de uso, y arcos de comunicación (Figura 1.19).

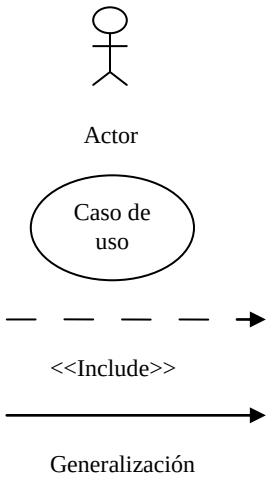


Figura 1.19 Compontes de los Casos de usos

• **Diagrama de Secuencia.**

El diagrama de secuencia describe cómo interactúan los objetos entre sí, proporcionándonos una visión de forma secuencial de los envíos de mensajes entre ellos (Figura 1.20).

Línea de vida muestra las acciones y el periodo desde que encuentra activa una instancia hasta el término de esta.

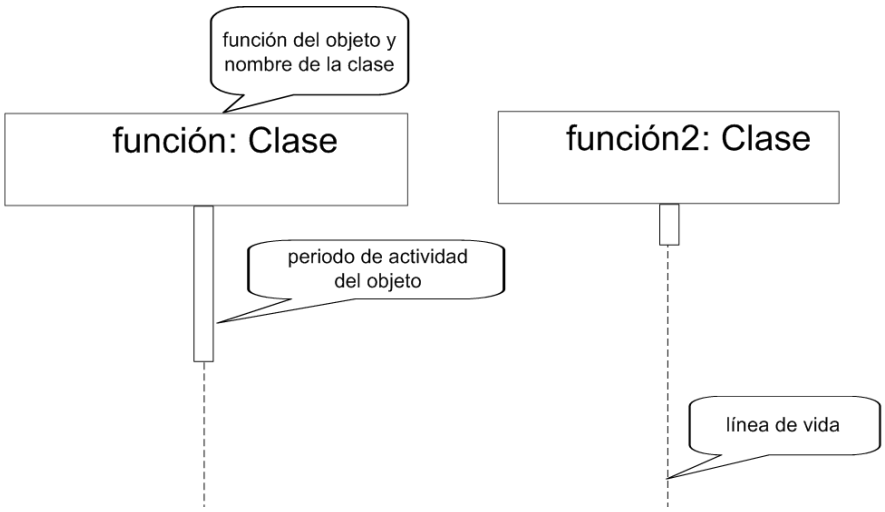


Figura 1.20 Diagrama de secuencia.

TIPOS DE MENSAJES.

Los envíos de mensajes se representan mediante flechas horizontales que unen la línea de vida del objeto emisor con la línea de vida del objeto destino (Figura 1.21).

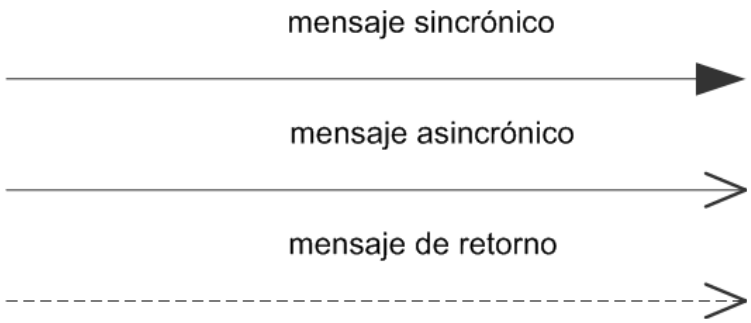


Figura 1.21 Tipos de mensajes

Mensaje sincrónico se usa para que el destinatario inicie un método no continua su actividad hasta que el destinatario termina (Figura 1.22).

Mensaje asincrónico se utiliza cuando no es necesario que termine un proceso para continuar con el anterior, éstos pueden activarse en paralelo.

Mensaje de retorno puede o no ser usado ya que algunos procesos no requieren informar su término para continuar con los siguientes.

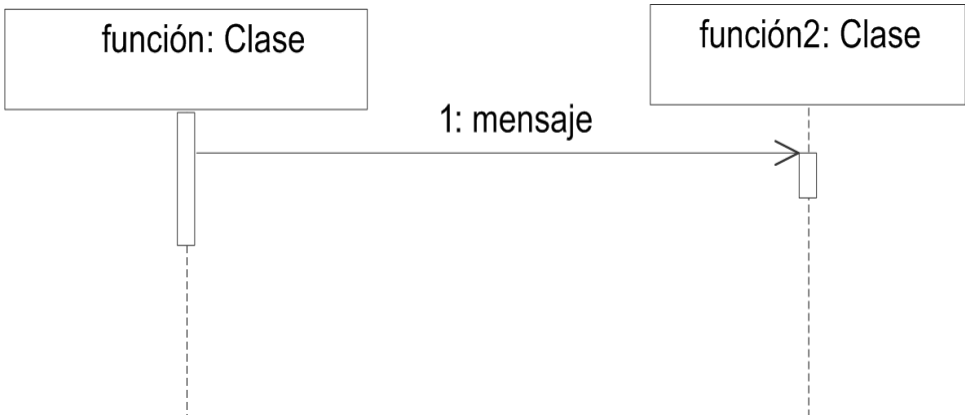


Figura 1.22 Mensaje sincrónico

• **Diagrama de Clase.**

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema (Figura 1.23).

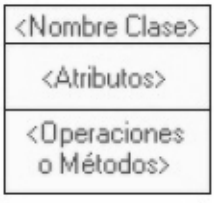


Figura 1.23 Diagrama de clases.

Ejemplo del diagrama de clases (Figura 1.24).

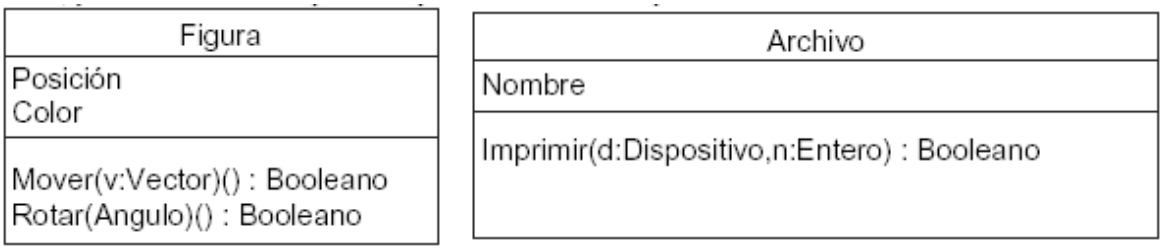


Figura 1.24 Ejemplo de diagrama de clases

• **Diagrama de Estado.**

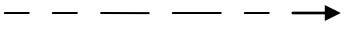
El diagrama de estado muestra los estados que adquiere un objeto durante su ciclo de vida, y como afectan estos en los demás (Figura 1.25).

Figura 1.25 Diagrama de Estado

• Diagrama de Paquetes.

El diagrama de paquetes nos permite visualizar nuestro sistema en una jerarquía lógica, además de mostrarnos las dependencias entre estas agrupaciones (Figura 1.26).

Se pueden indicar relaciones de dependencia entre paquetes mediante una flecha con la línea a trazos.



Si el paquete A depende del paquete B, entonces irá una flecha de A a B.

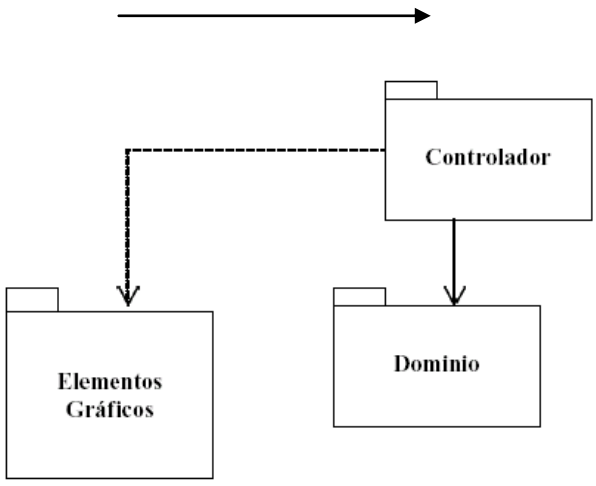


Figura 1.26 Diagramas de Paquetes

1.5 Java y sus características.

Java es un lenguaje de programación orientado a objetos ya que está formado de una colección de clases, y accedes a ellas por medio de métodos. La compañía Sun describe el lenguaje Java como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”. El objetivo principal del lenguaje Java es conectar a los usuarios con la información, estando situada en un ordenador local, en un servidor de Web, en una base de datos o en cualquier otro lugar.

Las características principales son:

- Es un lenguaje libre, se puede utilizar el compilador y la máquina virtual de forma gratuita.

- Está diseñado para ser multiplataforma y ser empleado el mismo programa en diferentes sistemas operativos.

- La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad

- Crea e inserta applets en paginas HTML,o mediante servlets y paginas jsp, genera código HTML dinámico. Con la capacidad de acceder a la base de datos. Un applet es un componente de una aplicación que se ejecuta en un navegador o browser (por ejemplo Netscape Navigator o Internet Explorer) y no requiere instalación en el servidor donde se encuentra el browser. Un servlet es una aplicación sin interface gráfica que se ejecuta en un servidor de Internet. La ejecución como aplicación independiente es análoga a los programas desarrollados con otros lenguajes.

- Toda su funcionalidad se encuentra en clases que forma parte del API de java.API (Application Programming Interface) provee de un conjunto de paquetes de clases para efectuar toda clase de tareas necesarias dentro de un programa.

- Conserva los tipos básicos de datos, int, float, double, char, etc, similares a los del lenguaje C/C++.

- Los programas Java se compilan y traducen a un formato denominado bytecodes –un formato intermedio de representación de las aplicaciones, que es independiente de la arquitectura del ordenador y de su sistema operativo. Para ejecutar aplicaciones Java en cualquier sistema, es

necesario disponer de un software que interprete los bytecodes (un runtime de Java); sin embargo, el código de las aplicaciones es siempre el mismo.

1.5.1. Historia de Java

Java surgió en 1991 por un grupo de ingenieros de Sun Microsystems como un lenguaje de programación destinado a electrodomésticos. Por la reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño pequeño.

Era la JVM (Maquina Virtual de Java) quien interpretaba el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: “Write Once, Run Everywhere”. Java se introdujo a finales de 1995 en computadoras. La clave fue la incorporación de un intérprete Java en la versión 2.0 del programa Netscape Navigator, produciendo una verdadera revolución en Internet. Java 1.1 apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. Java 1.2, más tarde rebautizado como Java 2, nació a finales de 1998

1.6 Neatbeans

Es un Entorno de Desarrollo Integrado (IDE) lo cual significa que integra todas las herramientas necesarias que permite editar, compilar, ejecutar, depurar, y desarrollar aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, pero se puede utilizar para cualquier otro lenguaje de programación. Estas aplicaciones son desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs (Interfaz de Programación de Aplicaciones) de NetBeans y un archivo especial (manifest file) que lo identifica como módulo.

Características de NetBeans:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)

- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)
- Elaborar aplicaciones de diferentes tipos (escritorio, web, móvil, Enterprise)
- Soporta varios lenguajes. (JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++.)
- Multiplataforma (JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++.)

1.7 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS), publicados bajo la licencia BSD. Como otros proyectos de código libre, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*). Postgres fue el pionero de varios conceptos en el sistema de objeto-relacional, ya que incluye algunas características de la orientación a objetos, como: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Pero esto no quiere decir que PostgreSQL sea un sistema de gestión de bases de datos puramente orientado a objetos.

1.7.1 Evolución del proyecto PostgreSQL:

1977-1985 Ingres (INteractive Graphics REtrieval System): Este proyecto inicio en la Universidad de Berkeley de California, como un ejercicio de aplicación sobre las bases de datos relacionales (RDBMS).

1986-1994 Postgres (POSTerior a inGRES): Él director de proyecto Michael Stonebraker con el objetivo de aplicar los conceptos de Objetos Relacionales.

1995 Postgres95: Andrew Yu & Jolly Chen incluyeron SQL en Postgres para posteriormente liberar su código en la web con el nombre de Postgres95. El proyecto incluía múltiples cambios al código original que mejoraban su rendimiento y legibilidad.

1996- en la actualidad PostrgreSQL: Se crearon nuevas mejoras y modificaciones, que repercutieron en un 20-40% más de eficiencia, así como la incorporación del estándar SQL92.

El objetivo básico, luego de la capacitación SQL y PostgreSQL, es la interacción de la base de datos con PHP y Java. Este entrenamiento implica el desarrollo de aplicaciones web, por un lado, las JSP (Java Server Pages) y Postgres proponen el desarrollo de aplicaciones seguras, para un ámbito cada vez menos seguro; la misma consideración es oportuna para la simbiosis Postgres y PHP.

Estar capacitado para desarrollar aplicaciones web con Postgres, Java, JSP, PHP genera posibilidades profesionales de excepción.

1.7.2 Características de PostgreSQL

- Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

- Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transactions, optimización de consultas, herencia, y arrays.

- Tiene varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin , pgAccess) y para hacer diseño de bases de datos (Tora , Data Architect).

- Es una base de datos ACID (Atomicity, Consistency, Isolation and Durability)

  - ❖ Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
  - ❖ Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
  - ❖ Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
  - ❖ Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

- Soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

- Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

- Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

- La flexibilidad del API de PostgreSQL soporta al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

- Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Capítulo 2

ANÁLISIS DEL SISTEMA DE LAS DIFERENTES FORMAS DE TITULACIÓN.

Análisis del problema y solución para erradicarlo

2. ANÁLISIS DEL SISTEMA DE LAS DIFERENTES FORMAS DE TITULACIÓN.

2.1 Análisis del Problema.

La carrera de ingeniería de computación de la FES Aragón no cuenta con un sistema, que maneje la información de los egresados que ya seleccionaron alguna de las nuevas formas de titulación, las cuales son:

1. **Tesis**
2. **Examen General de Conocimientos.**
3. **Desarrollo de un Caso Práctico.**
4. **Alto Nivel Académico.**
5. **Seminario y Diplomados de actualización y capacitación profesional.**
6. **Informe de Ejercicio Profesional.**
7. **Memoria de desempeño de servicio social.**

Los egresados deben de cubrir algunos requerimientos para el pre-registro de titulación como es la entrega de algunos documentos en la jefatura de la carrera, los cuales dependen de la modalidad que hayan seleccionado para titularse.

En todas las modalidades es indispensable presentar Examen Profesional, un Trabajo Escrito (Tesis o Tesina para las nuevas modalidades) y elegir un Asesor con mínimo tres años de experiencia docente en la UNAM.

1. **Tesis**
   - ✓ Anteproyecto escrito de tesis.
   - ✓ Historia académica de licenciatura
   - ✓ Formatos de registro de titulación firmados por el Asesor
   - ✓ Constancia de 100% de Créditos de licenciatura (original y copia - se tramita en Servicios Escolares).
   - ✓ Carta de término de Servicio Social (original y copia).
   - ✓ Carta de liberación de Servicio Social (original y copia).
   - ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).

2. **Examen General de Conocimientos.**
   - ✓ Informe detallado del trabajo monográfico.
   - ✓ Historia académica de licenciatura
   - ✓ Formatos de registro de titulación firmados por el Asesor
   - ✓ Carta de término de Servicio Social (original y copia).
   - ✓ Carta de liberación de Servicio Social (original y copia).

- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad de modalidad de examen general de conocimientos, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-mail.
  - Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

3. Desarrollo de un Caso Práctico.

- ✓ Informe detallado del proyecto
- ✓ Historia académica de licenciatura
- ✓ Formatos de registro de titulación firmados por el Asesor
- ✓ Carta de término de Servicio Social (original y copia).
- ✓ Carta de liberación de Servicio Social (original y copia).
- ✓ Constancia de 100% de Créditos de licenciatura (original y copia - se tramita en Servicios Escolares).
- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad del desarrollo de un caso práctico, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-mail.
  - Área de desarrollo
  - Institución donde se desarrolló el caso práctico.
  - Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

4. Alto Nivel Académico.

- ✓ Informe detallado de un trabajo monográfico o reporte detallado de servicio social.
- ✓ Historia académica de licenciatura.
- ✓ Formatos de registro de titulación firmados por el Asesor.
- ✓ Carta de término de Servicio Social (original y copia).
- ✓ Carta de liberación de Servicio Social (original y copia).
- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad de modalidad de examen general de conocimientos, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-

mail.
- Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

5. Seminario y Diplomados de actualización y capacitación profesional.

- ✓ Informe de las actividades desarrolladas.
- ✓ Copia de la constancia o diploma que lo avale.
- ✓ Historia académica de licenciatura
- ✓ Formatos de registro de titulación firmados por el Asesor
- ✓ Carta de término de Servicio Social (original y copia).
- ✓ Carta de liberación de Servicio Social (original y copia).
- ✓ Constancia de 100% de Créditos de licenciatura (original y copia - se tramita en Servicios Escolares).
- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad de seminarios y cursos de actualización y capacitación profesional, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-mail.
  - Nombre del diplomado
  - Duración y fecha (240 horas mínimo)
  - Institución donde se realizó
  - Módulos o temario.
  - Área y actividades en que se desarrolla profesionalmente.
  - Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

6. Informe de Ejercicio Profesional.

- ✓ Comprobantes que avalen la experiencia profesional, por ejemplo: Carta membretada mencionando la participación, proyectos y antigüedad, recibos de nómina, recibos de honorarios, comprobantes de alta de I.M.S.S. o I.S.S.S.T.E, etc.
- ✓ Breve informe del ejercicio profesional de por lo menos los últimos tres años.
- ✓ Historia académica de licenciatura
- ✓ Formatos de registro de titulación firmados por el Asesor
- ✓ Carta de término de Servicio Social (original y copia).
- ✓ Carta de liberación de Servicio Social (original y copia).
- ✓ Constancia de 100% de Créditos de licenciatura (original y copia - se tramita en Servicios Escolares).

- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad del ejercicio profesional, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-mail.
  - Área del ejercicio profesional
  - Tiempo de experiencia
  - Institución o empresas donde se ha desarrollado
  - Proyectos en los que ha participado
  - Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

7. Memoria de desempeño de servicio social.

- ✓ Informe de las actividades desarrolladas.
- ✓ Informe de los proyectos en los que se participó.
- ✓ Historia académica de licenciatura
- ✓ Formatos de registro de titulación firmados por el Asesor
- ✓ Carta de término de Servicio Social (original y copia).
- ✓ Carta de liberación de Servicio Social (original y copia).
- ✓ Constancia de 100% de Créditos de licenciatura (original y copia - se tramita en Servicios Escolares).
- ✓ Comprobante de pago por concepto de registro de titulación (pagar en cajas de la FES).
- ✓ Carta de autorización de modalidad de memoria de desempeño de servicio social, dirigida a la Jefa de Carrera de Ingeniería en Computación, mencionando los siguientes datos:
  - Datos personales: nombre, generación, dirección, teléfono, e-mail.
  - Institución donde se realizó el Servicio Social.
  - Área de desarrollo
  - Actividades desarrolladas
  - Período
  - Proyectos en los que se participó.
  - Nombre del asesor, quién deberá firmar la solicitud con Vo. Bo.

Si la jefatura de carrera acepta la modalidad de titulación, los documentos se recaudan, almacenan y administran en carpetas. Y al egresado se le entregara una carta de pre-registro para poder continuar con sus trámites.

Esta información también se tiene en una hoja de cálculo, en la cual las búsquedas son muy limitadas ya que esta información se tiene por varios años sin

tener seguridad.

En esta hoja se guardan los siguientes datos:

- Número de cuenta.
- Nombre completo.
- Modalidad.
- Tema.
- Nombre del asesor de tesis.
- El día, mes y año que se registro.

Por este motivo, no hay control del porcentaje de cada una de las modalidades y se tiene que invertir más tiempo al momento que son requeridas ciertas consultas como: la cantidad de alumnos que se registraron ya sea por día, semana, mes, año o un periodo de tiempo o modalidad

2.2 Sistema Propuesto.

En actualidad, es de suma importancia tener una base de datos la cual respalde y organice de una forma más rápida para su localización o consulta de las diferentes formas de titulación, para tener un control de los egresados que están en trámites de titulación.

Este sistema ayudará a que el pre-registro de titulación se lleve de una forma más organizada y segura, esto beneficiara a la jefatura de carrera. Para erradicar los problemas de tiempo en búsqueda, consulta, modificación, y actualización implementaremos un sistema el cual va a contener:

- Datos personales del egresado:
  - ✓ Número de Cuenta.
  - ✓ Apellido Paterno.
  - ✓ Apellido Materno.
  - ✓ Nombre.
  - ✓ Domicilio.
  - ✓ Teléfono.
  - ✓ Correo.
  - ✓ Fecha de registro.
- Datos de su forma de titulación:
  - ✓ Opción de Titulación.
  - ✓ Tema.
- Datos de su asesor:

- ✓ R.F.C.
- ✓ Número de Trabajador.
- ✓ Apellido Paterno.
- ✓ Apellido Materno.
- ✓ Nombre.
- ✓ Grado.

En el sistema existirán dos formas de acceder una de ella será como Administrador que tendrá a su cargo los siguientes privilegios:

- Registro de egresados.
- Registro de asesor.
- Buscar egresado.
- Modificar egresado.
- Modificar asesor.
- Estadísticas.

Y el Usuario que sólo podrá:

- Buscar egresado.
- Estadísticas.

En la parte de Estadísticas podremos obtener un balance de personas que se registraron ya sea por:

- Modalidad.
- Asesor.
- Periodo de tiempo.

2.3 Requerimientos de Software y Hardware.

El sistema será instalado únicamente en la computadora personal que ocupa la secretaría de la jefatura de carrera ya que ella es la encargada de los registros, la recaudación de los datos de egresado y asesores.

La secretaría de la jefatura de carrera contará con la contraseña de Administrador para mantener segura la información que se recaude en el sistema.

Los Usuarios que serán el personal que laboran en la jefatura, contará con un acceso libre al sistema ya que en ocasiones se requiere alguna información de

algún egresado o las estadísticas de los egresados que se han registrado ya sea por modalidad, asesor o periodo de tiempo.

El sistema será ocupado únicamente los días laborales de la jefatura, ya que tan solo en estos días los egresados pueden entregar documentos y solicitar el pre-registro de titulación.

HARDWARE

- Pentium III en adelante.
- Memoria RAM 2GB.
- Procesador INTEL.
- Monitor.
- Disco duro 200GB.

SOFTWARE

- El sistema se desarrollo bajo el ambiente Windows ya que en la jefatura es el sistema con la que trabaja la computadora personal.
- La Base de Datos estará hecha en PostgreSQL espacio disponible en disco 70MB (mínimo).
- La plataforma de desarrollo que será Java, Mínimo Java Standard Edition Runtime Environment.

Capítulo 3

DISEÑO DEL SISTEMA DE LAS DIFERENTES FORMAS DE TITULACIÓN CON UML.

Elaboración de los diagramas del sistema con ayuda de UML.

3 DISEÑO DEL SISTEMA DE LAS DIFERENTES FORMAS DE TITULACIÓN CON UML.

3.2 Diagrama de casos de uso.

En estos diagramas lo principal es identificar los actores, son aquellas personas que interactúan con el sistema en nuestro sistema existen dos actores (Figura3.1):

Figura3.1 Actores

Los casos de uso son las secuencias de interacción por las cuales él o los actores utilizan el sistema.

Diagrama de caso de de uso para el sistema de pre-registro (Figura 3.2).

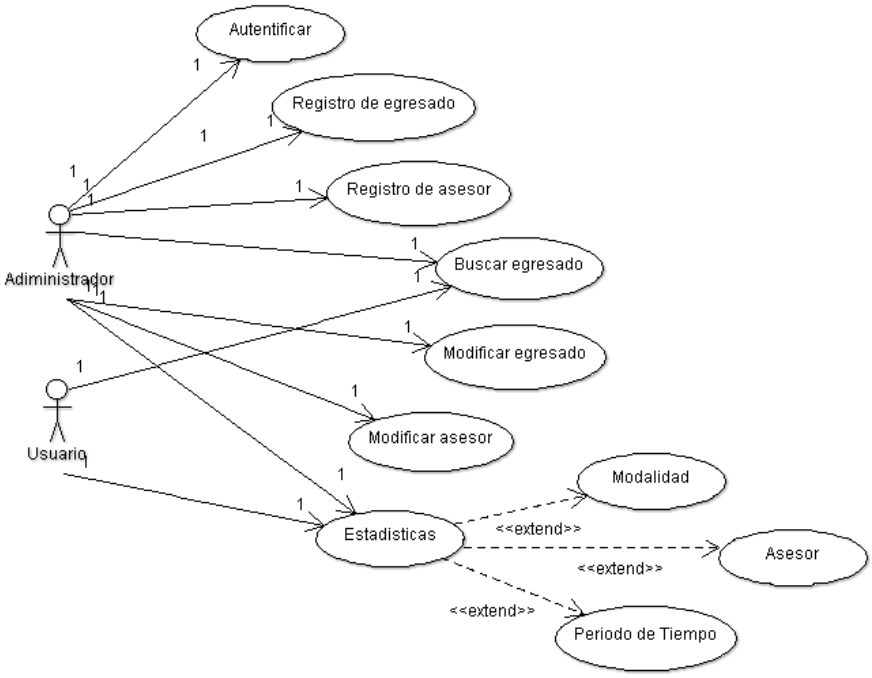
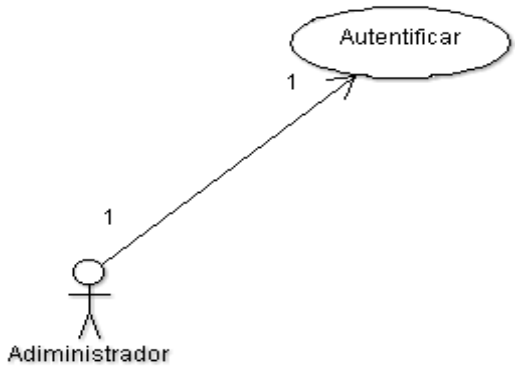


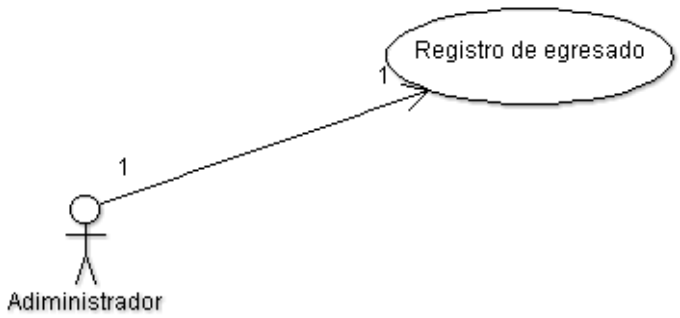
Figura 3.2 Diagrama de caso de uso

Diagrama de caso de uso


Definición	
Caso de uso:	Autentificar.
Tipo:	Básico.
Actor principal:	***Administrador (AD)***
Descripción:	Este caso de uso describe como el ***(AD)*** se autentifica ante el sistema de pre-registro para posteriormente permitirle llevar a cabo otros casos de uso.
Precondiciones:	El ***(AD)*** debe iniciar el sistema de pre-registro.
Poscondiciones:	El ***(AD)*** deberá estar autentificado ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
1.	Ingresa su contraseña y pulsa el botón ingresar.	2.	Verifica la contraseña en la base de datos.	E1
			Si la contraseña es correcta el sistema muestra un pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	

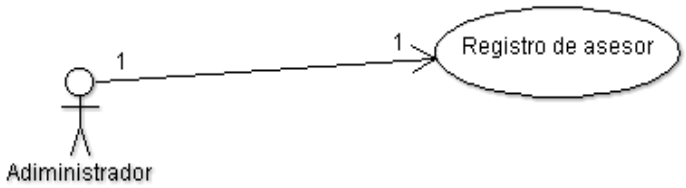
Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Contraseña incorrecta.	Muestra un cuadro de diálogo “La contraseña no es correcta”.

Diagrama de caso de uso	
	
Definición	
Caso de uso:	Registro de egresado.
Tipo:	Básico.
Actor principal:	***Administrador (AD)***
Descripción:	Después de que fue aceptada la modalidad de titulación, que escogió el egresado, deberá ser registrado con sus datos. El ***(AD)*** llevará el control de todos los egresados, por medio de formatos se tendrán que llenar.
Precondiciones:	➢ El ***(AD)*** debe de haber ingresado al sistema y autentificado. ➢ El ***(AD)*** deberá contar con toda la información necesaria del egresado.
Poscondiciones:	➢ El ***(AD)*** podrá registrar uno o más egresados.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	

2.	Selecciona la opción de Registro de egresados.	3.	Muestra el formato de Registro de egresado donde pedirá estos datos: • Número de Cuenta. • Apellido Paterno. • Apellido Materno. • Nombre. • Domicilio. • Teléfono. • Correo. • Opción de Titulación. • Tema. • Nombre del asesor. • Fecha de registro.	
4.	Llena el formato con los datos del egresado y pulsa el botón Registrar.	5.	Los datos se almacenan y el sistema muestra un cuadro de diálogo “El egresado ha sido registrado”.	E1
		6.	Regresa a la pantalla de opciones.	

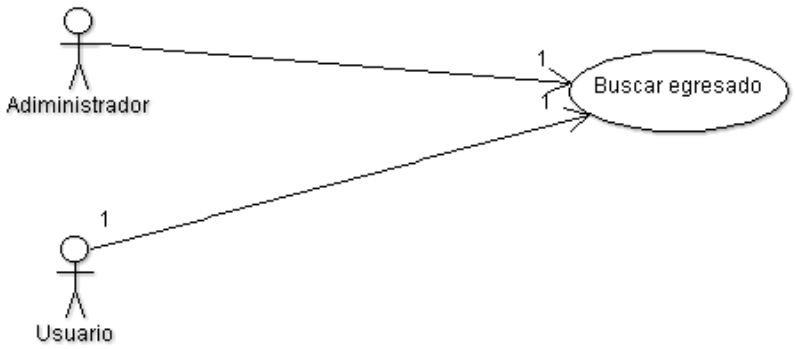
Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Campo vacio.	Muestra un cuadro de diálogo “Faltan campos por llenar”.

Diagrama de caso de uso	
	
Definición	
Caso de uso:	Registro de asesor
Tipo:	Básico.
Actor principal:	***Administrador (AD)***
Descripción:	➢ El ***(AD)*** deberá contar con toda la información necesaria del asesor desde el tiempo en que lleva laborando que debe de ser como mínimo 3 años y sus datos laborales.
Precondiciones:	➢ El ***(AD)*** debe de haber ingresado al sistema y autentificado. ➢ El ***(AD)*** deberá contar con toda la información necesaria del asesor.
Poscondiciones:	➢ El ***(AD)*** podrá registrar uno o más asesores.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	
2.	Selecciona la opción de Registro de asesor.	3.	Muestra el formato de Registro de asesor que contiene: • R.F.C. • Número de Trabajador. • Apellido Paterno. • Apellido Materno. • Nombre. • Grado.	

4.	Llena el formato con los datos del asesor y pulsa el botón Registrar.	5.	Los datos se almacenan y el sistema muestra un cuadro de diálogo “El asesor ha sido registrado”.	E1
		6.	Regresa a la pantalla de opciones.	

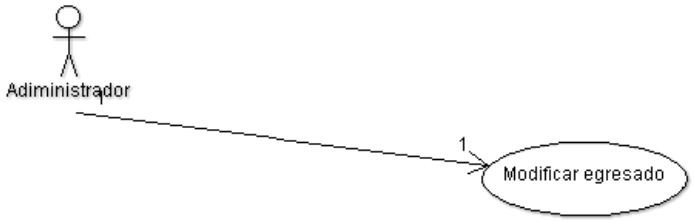
Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Campo vacio.	Muestra un cuadro de diálogo “Faltan campos por llenar”.

Diagrama de caso de uso	
	
Definición	
Caso de uso:	Buscar egresado
Tipo:	Básico.
Actor principal:	***Administrador (AD), Usuario (Us)***
Descripción:	El ***(AD)*** o ***(Us)*** podrán buscar a los egresados que hayan registrado una modalidad de tesis.
Precondiciones:	➢ El ***(AD)*** o ***(Us)*** deberán de haber ingresado al sistema. ➢ El ***(AD) o (Us)*** deberán contar con la información necesaria del egresado.
Poscondiciones:	➢ El ***(AD)*** o ***(Us)*** deberán estar autentificados ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Si entra como ***(AD)*** muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado.	

			• Modificar asesor. • Estadísticas. Si entra como ***(Us)*** muestra una pantalla de opciones que contendrá: • Buscar egresado. • Estadísticas.	
2.	Selecciona la opción de Buscar egresado.	3.	Muestra el formato de Buscar de egresado que contiene: • Número de Cuenta. • Apellido Paterno. • Apellido Materno. • Nombre.	
4.	Captura los datos del egresado y pulsa el botón Buscar.	5.	Muestra la pantalla con la información del egresado.	E1

Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Egresado no registrado.	Muestra un cuadro de diálogo “No se encontró al egresado”.

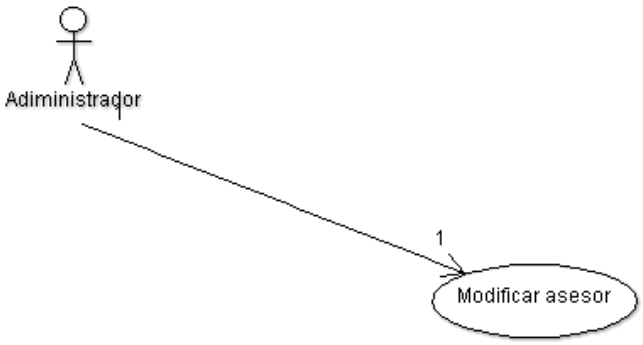
Diagrama de caso de uso


Definición	
Caso de uso:	Modificar egresado
Tipo:	Básico.
Actor principal:	***Administrador (AD)***
Descripción:	El ***(AD)*** llevará el control de todos los egresados, podrá modificar a los egresados que hayan registrado una modalidad de tesis.
Precondiciones:	➢ El ***(AD)*** debe de haber ingresado al sistema y autentificado. ➢ El ***(AD)*** deberá contar con toda la información necesaria del egresado.
Poscondiciones:	➢ El ***(AD)*** deberá estar autentificado ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	
2.	Selecciona la opción de Modificar egresado.	3.	Muestra un formato donde pedirá estos datos: • Número de Cuenta. • Apellido Paterno. • Apellido Materno. • Nombre.	E1

4.	Captura los datos del egresado y pulsa el botón Siguiente.	5.	Muestra el formato de Modificar egresado que contiene los datos: • Número de Cuenta. • Apellido Paterno. • Apellido Materno. • Nombre. • Domicilio. • Teléfono. • Correo. • Opción de Titulación. • Tema. • Nombre del asesor. • Fecha de registro.	
6.	Modifica el o los datos del egresado y pulsa el botón de modificar.	7.	Los datos se almacenan y el sistema muestra un cuadro de diálogo “El egresado ha sido modificado”.	
		8.	Regresa a la pantalla de opciones.	

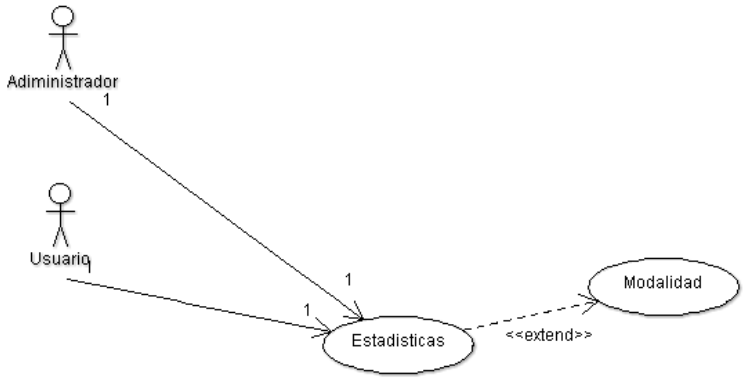
Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Egresado no registrado.	Muestra un cuadro de diálogo “No se encontró al egresado”.

Diagrama de caso de uso	
	
Definición	
Caso de uso:	Modificar asesor
Tipo:	Básico.
Actor principal:	***Administrador (AD)***
Descripción:	El ***(AD)*** llevará el control de todos los asesores, podrá modificar a los asesores que estén registrados.
Precondiciones:	➢ El ***(AD)*** debe de haber ingresado al sistema y autentificado. ➢ El ***(AD)*** deberá contar con toda la información necesaria del asesor.
Poscondiciones:	➢ El ***(AD)*** deberá estar autentificado ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	

2.	Selecciona la opción de Modificar asesor.	3.	Muestra un formato donde pedirá estos datos: • R.F.C. • Apellido Paterno. • Apellido Materno. • Nombre.	E1
4.	Captura los datos del asesor y pulsa el botón Siguiente.	5.	Muestra el formato de Modificar asesor que contiene los datos: • R.F.C. • Número de Trabajador. • Apellido Paterno. • Apellido Materno. • Nombre.	
6.	Modifica los datos del asesor y pulsa el botón de Modificar.	7.	Los datos se almacenan y el sistema muestra un cuadro de diálogo “El asesor ha sido modificado”.	
		8.	Regresa a la pantalla de opciones.	

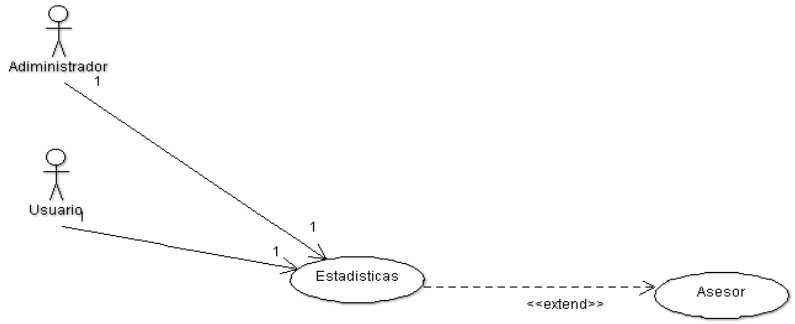
Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Asesor no registrado.	Muestra un cuadro de diálogo “No se encontró al asesor”.

Diagrama de caso de uso


Definición	
Caso de uso:	Modalidad
Tipo:	Básico.
Actor principal:	***Administrador (AD) y Usuario (Us)***
Descripción:	El ***(AD)*** llevará el control de todos los egresados que se registraron a las nuevas formas de titulación. El ***(AD) o (US)*** podrán ver un control de las personas registradas en las diferentes modalidades.
Precondiciones:	➢ El ***(AD) o (Us)*** deben de haber ingresado al sistema.
Poscondiciones:	➢ El ***(AD) o (Us)*** deberán estar autentificados ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Muestra un pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	

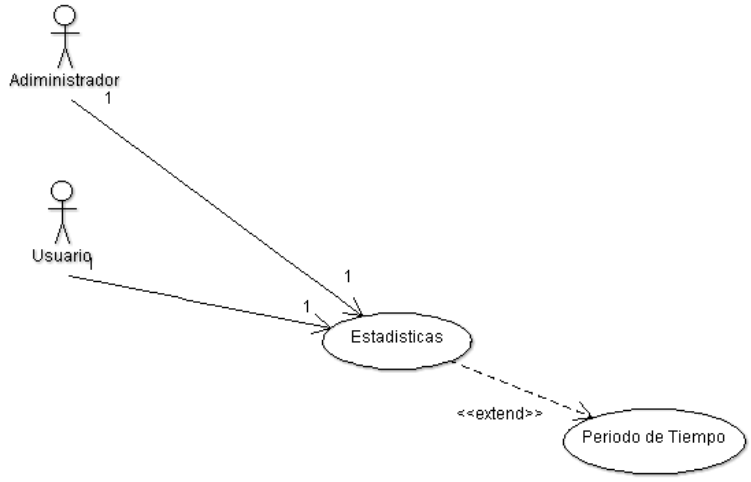
2.	Selecciona la opción de Estadísticas.	3.	Muestra una pantalla para seleccionar: • Modalidad. • Asesor • Periodo de tiempo.	
4.	Selecciona la opción de Modalidad.	5.	Muestra una pantalla de selección con las modalidades: • Tesis. • Examen General de Conocimientos. • Desarrollo de un Caso Práctico. • Alto Nivel Académico. • Seminario y Diplomados de actualización y capacitación profesional. • Informe de Ejercicio Profesional. • Memoria de desempeño de servicio social.	
6.	Selecciona una de las modalidades y pulsa el botón de Estadísticas.	7.	Muestra una tabla con los campos: • Modalidad. • No. De Cuenta. • Nombre. • Apellido Paterno. • Apellido Materno. • Asesor. • Mes. • Año. • Y el Total de egresados que se registraron.	

Diagrama de caso de uso	
	
Definición	
Caso de uso:	Asesor
Tipo:	Básico.
Actor principal:	***Administrador (AD) y Usuario (Us)***
Descripción:	El ***(AD)*** llevará el control de todos los egresados que se registraron a las nuevas formas de titulación. El ***(AD) o (US)*** podrán ver un control de las personas registradas con los diferentes asesores.
Precondiciones:	➢ El ***(AD) o (Us)*** deben de haber ingresado al sistema.
Poscondiciones:	➢ El ***(AD) o (Us)*** deberán estar autentificados ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1.	Si entra como ***(AD)*** muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas. Si entra como ***(Us)*** muestra una pantalla de opciones que contendrá: • Buscar egresado. • Estadísticas.	

2.	Selecciona la opción de Estadísticas.	3,	Muestra una pantalla para seleccionar: • Modalidad. • Asesor • Periodo de tiempo.	
4.	Selecciona la opción de Asesor.	5.	Muestra un formato donde pedirá estos datos: • R.F.C. • Apellido Paterno. • Apellido Materno. • Nombre.	E1
6.	Captura los datos del asesor y pulsa el botón de Estadísticas.	7.	Muestra una tabla con los campos: • Modalidad. • No. De Cuenta. • Nombre. • Apellido Paterno. • Apellido Materno. • Asesor. • Mes. • Año. • Y el Total de egresados que se registraron.	

Excepciones		
Identificador	**Nombre**	**Respuesta del Sistema**
E1	Asesor no registrado.	Muestra un cuadro de diálogo “No se encontró al asesor”.

Diagrama de caso de uso


Definición	
Caso de uso:	Periodo de Tiempo
Tipo:	Básico.
Actor principal:	***Administrador (AD) y Usuario (Us)***
Descripción:	El ***(AD)*** llevará el control de todos los egresados que se registraron a las nuevas formas de titulación. El ***(AD) o (US)*** podrán ver un control de las personas registradas en un periodo de tiempo.
Precondiciones:	➢ El ***(AD) o (Us)*** deben de haber ingresado al sistema.
Poscondiciones:	➢ El ***(AD) o (Us)*** deberán estar autentificados ante el sistema.

Flujos				
Actor		**Sistema**		
Paso	**Acciones**	**Paso**	**Acciones**	**Excepción**
		1,	Si entra como ***(AD)*** muestra una pantalla de opciones que contendrá: • Registro de egresados. • Registro de asesor. • Buscar egresado. • Modificar egresado. • Modificar asesor. • Estadísticas.	

			Si entra como ***(Us)*** muestra una pantalla de opciones que contendrá: • Buscar egresado. • Estadísticas.	
2,	Selecciona la opción de Estadísticas.	3,	Muestra una pantalla para seleccionar: • Modalidad. • Asesor • Periodo de tiempo.	
4.	Selecciona la opción de Periodo de Tiempo.	5.	Muestra un formato donde pedirá estos datos: • Mes y Año de Inicio • Mes y Año de Fin	
6.	Captura los datos de periodo de tiempo y pulsa el botón de Estadísticas.	7.	Muestra una tabla con los campos: • Modalidad. • No. De Cuenta. • Nombre. • Apellido Paterno. • Apellido Materno. • Asesor. • Mes. • Año. • Y el Total de egresados que se registraron.	

3.3 Diagrama de clase.

En estos diagramas describiremos la estructura de nuestro sistema mostrando las clases, atributos y las relaciones entre ellos (Figura 3.3).

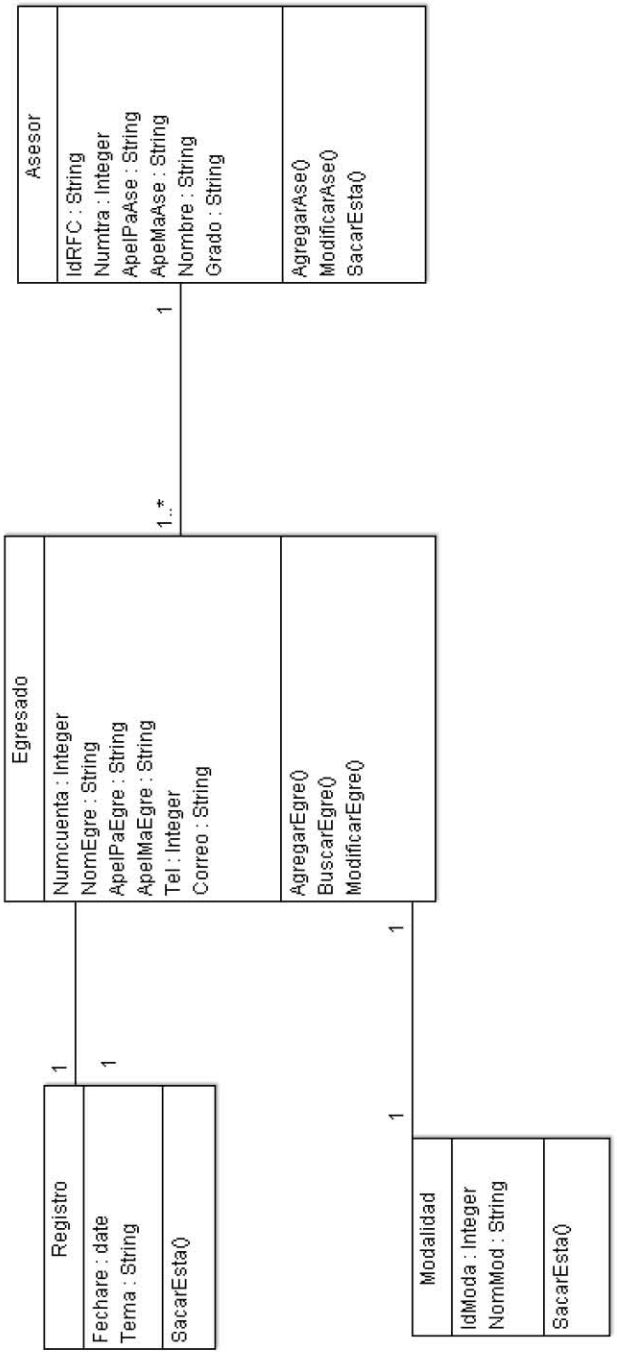


Figura 3.3 Diagrama de Clases.

3.4 Diagrama de estado.

En este diagrama identificamos cada una de los eventos, estados y sus transiciones por las que pasaran los objetos.

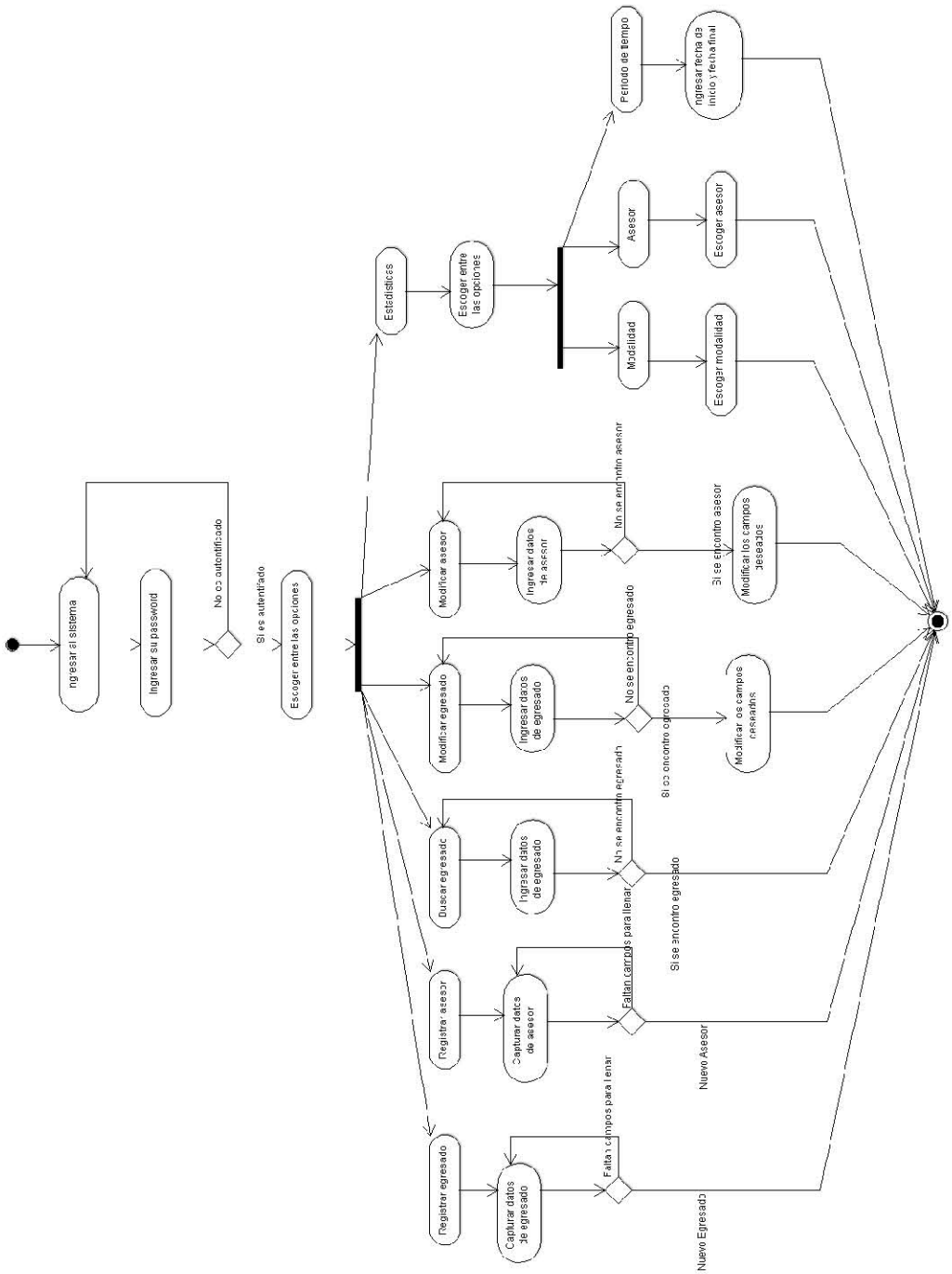


Figura 3.4 Diagrama de Estado para el Administrador

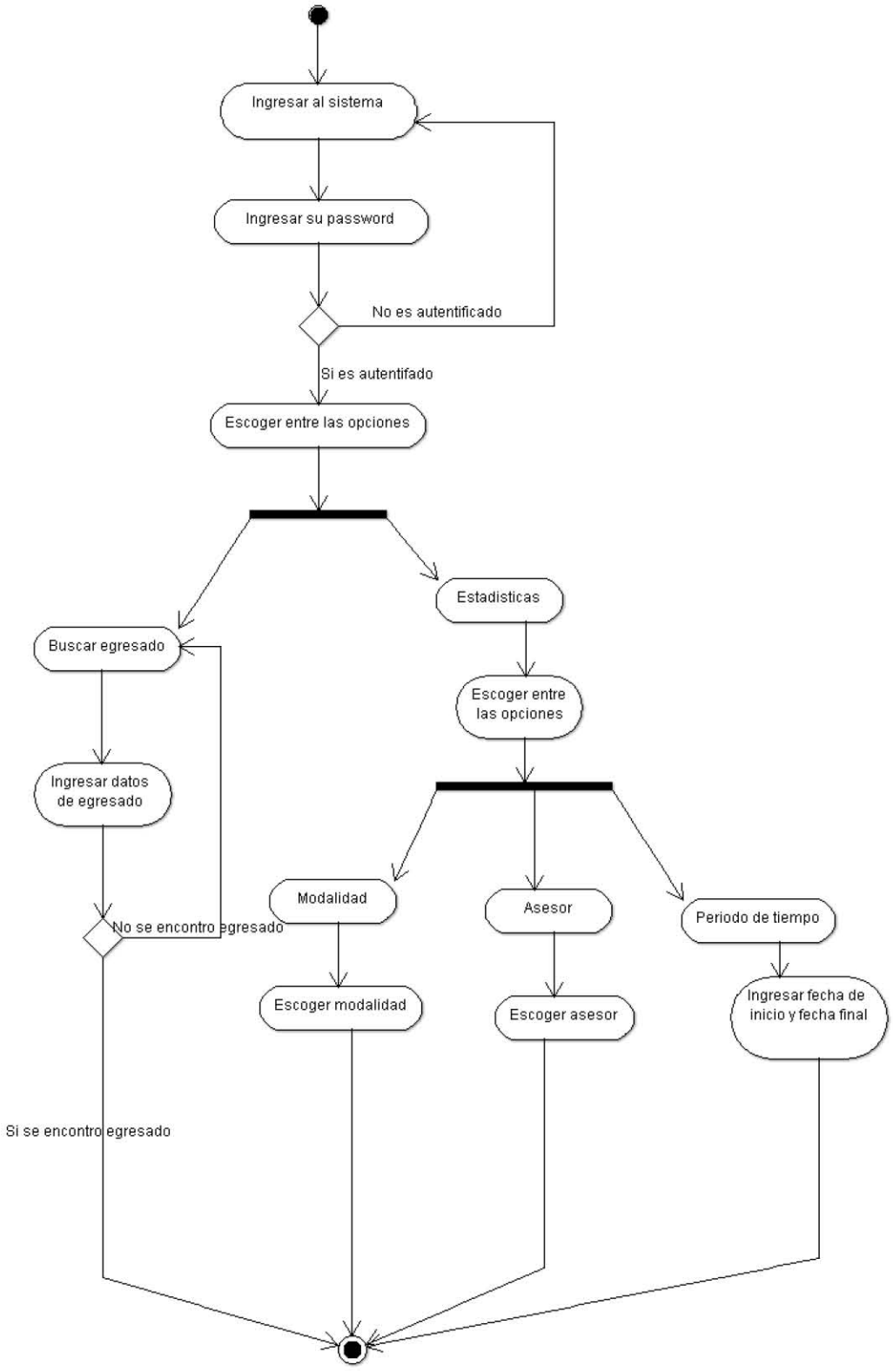


Figura 3.4 Diagrama de Estado para el Usuario

3.5 Diagrama de secuencia.

En estos diagramas mostramos la interacción entre los objetos al transcurrir el tiempo.

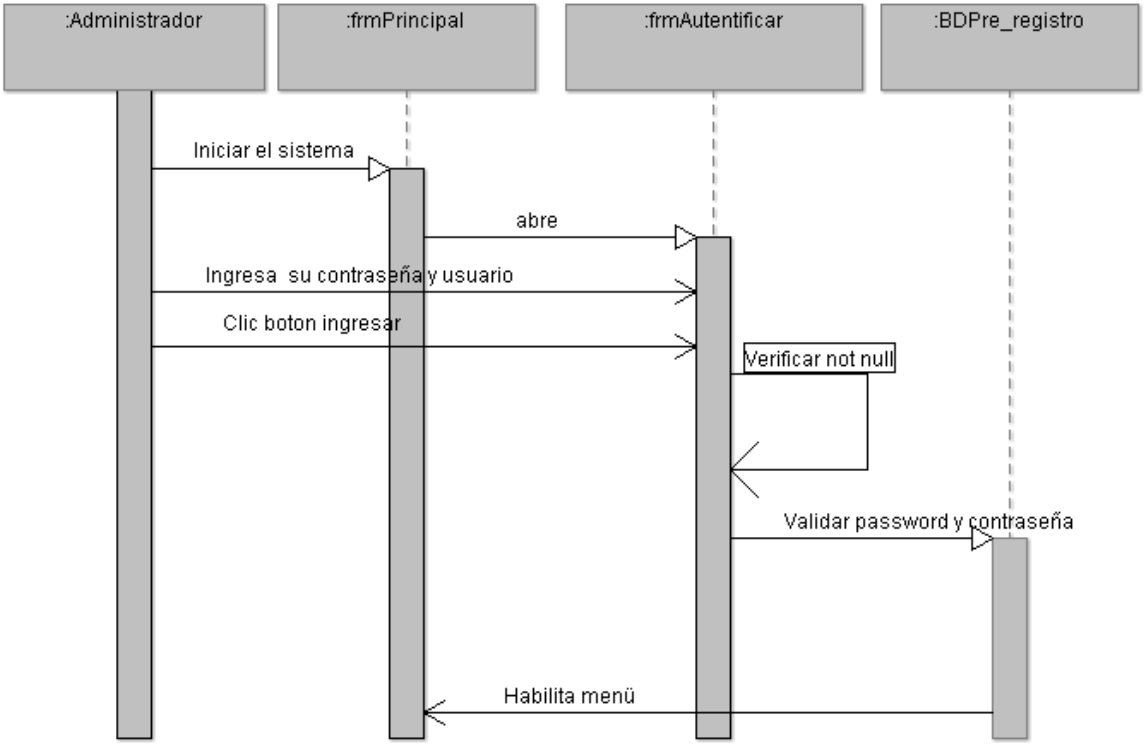


Figura 3.5 Diagrama de Secuencia de Autentificación

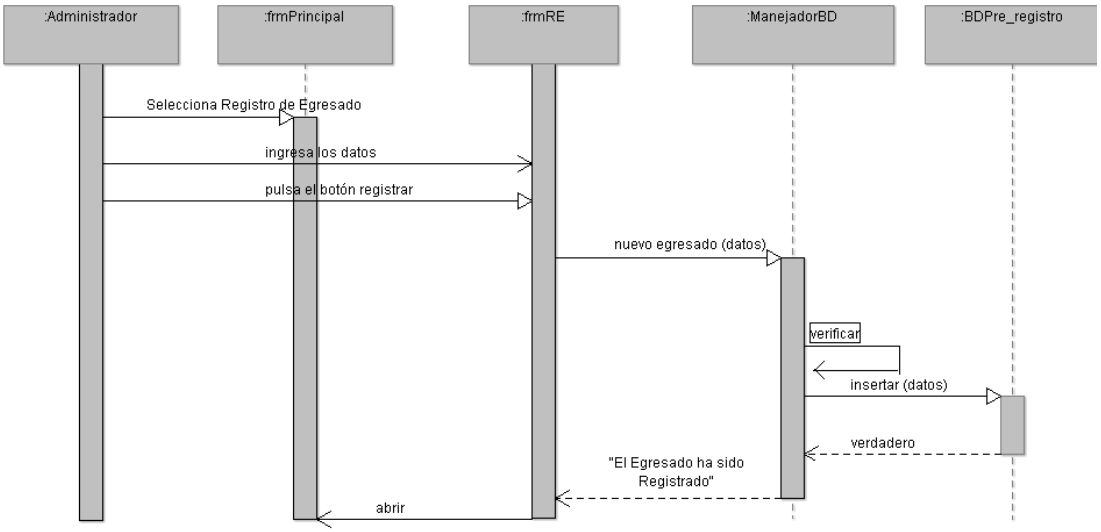


Figura 3.6 Diagrama de Secuencia de Registro de egresado

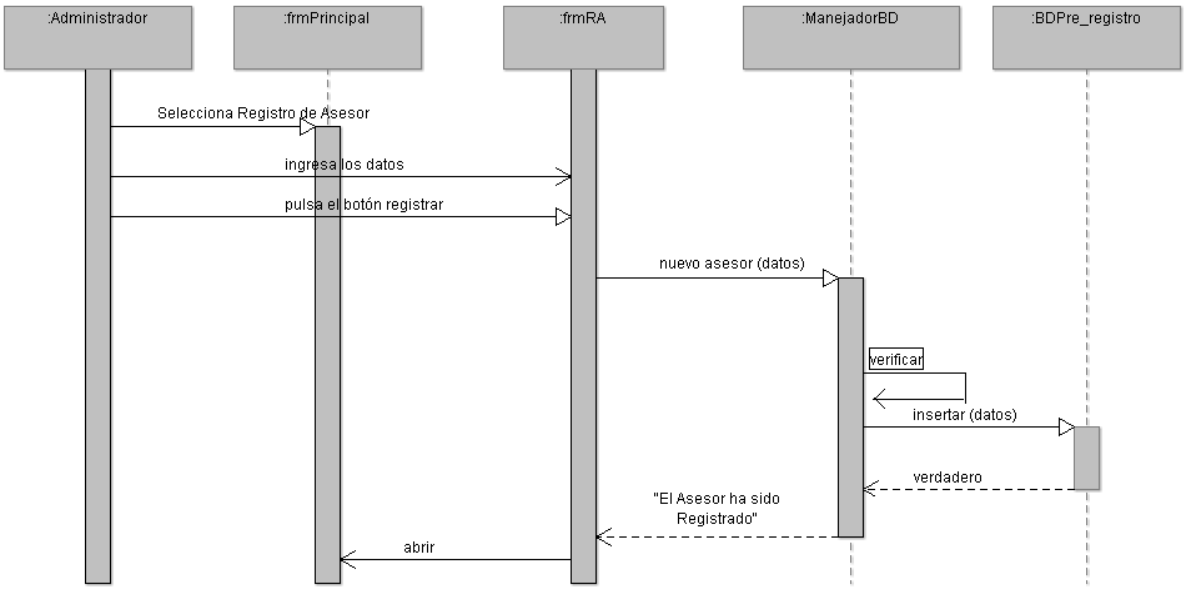


Figura 3.7 Diagrama de Secuencia de Registro de Asesor

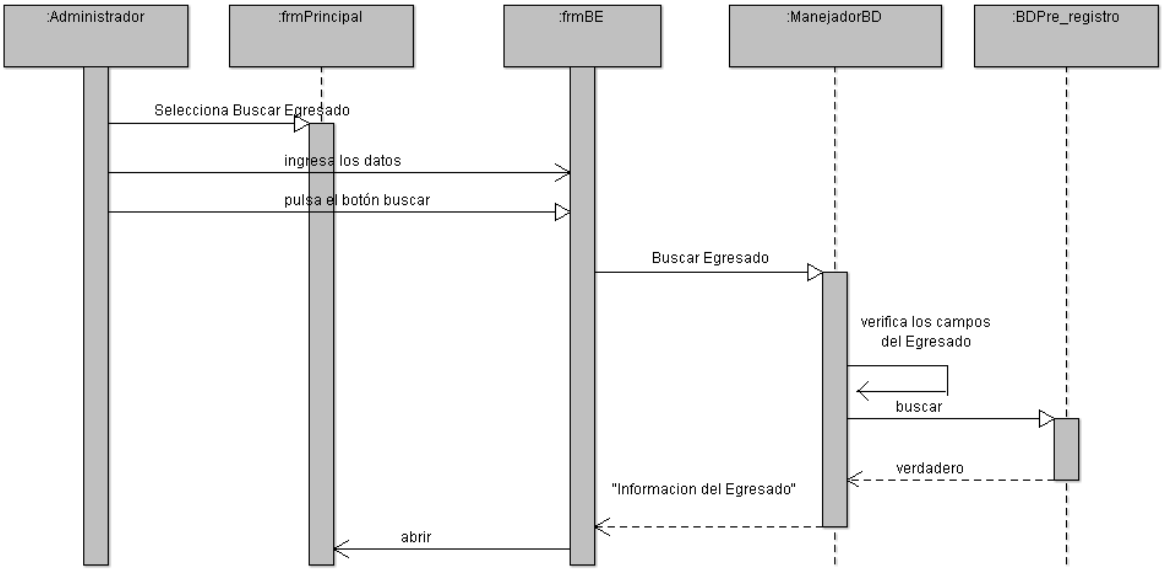


Figura 3.8 Diagrama de Secuencia de Buscar Egresado

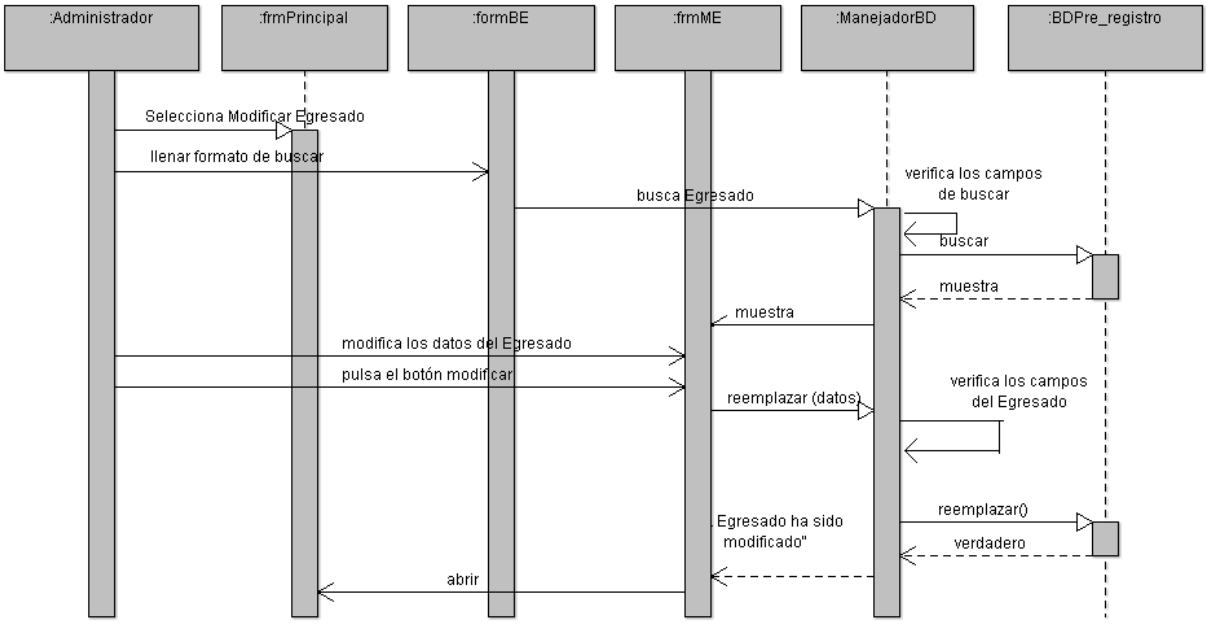


Figura 3.9 Diagrama de Secuencia de Modificar Egresado

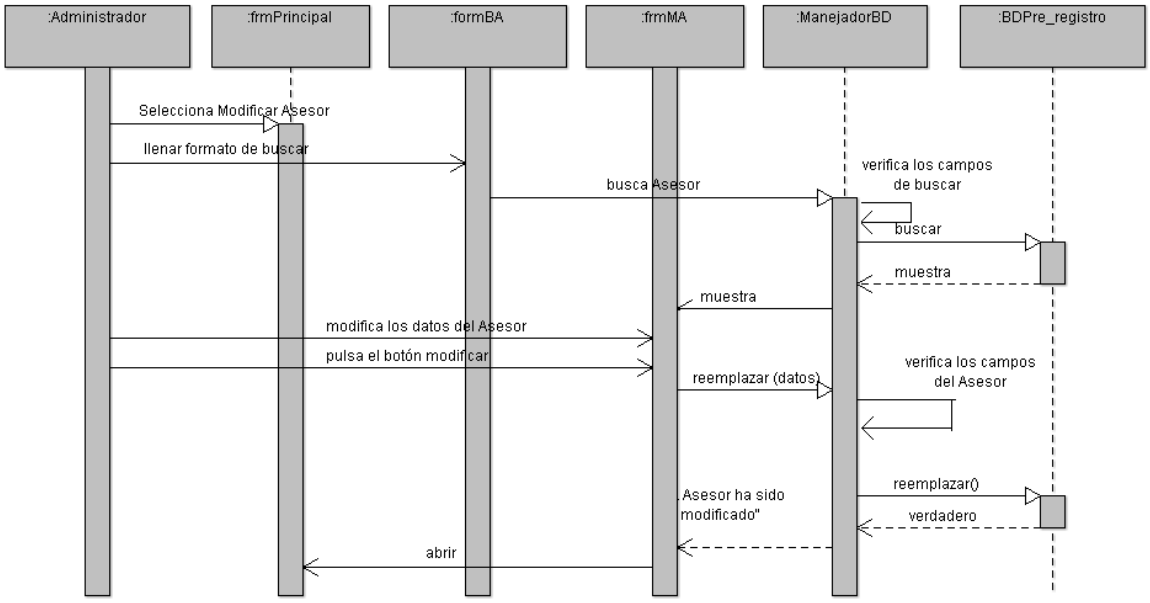


Figura 3.10 Diagrama de Secuencia de Modificar Asesor

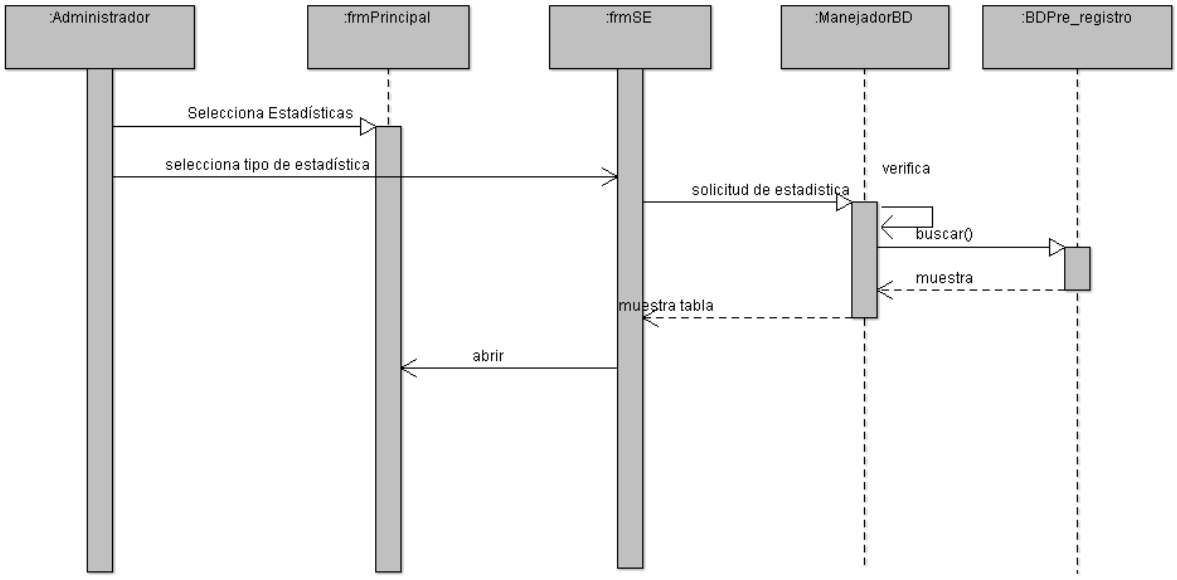


Figura 3.11 Diagrama de Estadísticas

Capítulo 4

DESARROLLO

Elaboración de la base de datos donde se almacenara la información de los egresados y asesores.

4. DESARROLLO

4.1 Base de datos

La base de datos es un conjunto de información que se ubica agrupada o estrucuturada(Figura 4.1).
Se elabora la base de datos para manejar, almacenar e implementar la seguridad de las grandes cantidades de información,

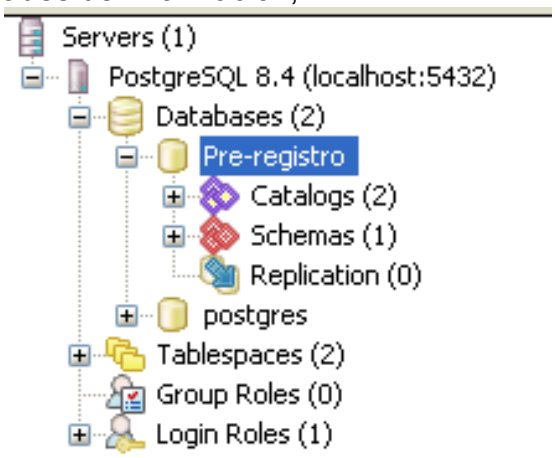


Figura 4.1 Base de Datos Pre-registro

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a traves de estas conexiones se relacionan los datos de ambas tablas.

Las tablas es una forma de representar la información más compacta y se puede acceder a la información contenida en dos o más tablas.

4.2 Diseño de la base de datos

Para el diseño de la base de datos definimos la información que se requiere almacenar, después diseñamos las tablas describiendo cada uno de los campos que componen el registro y los valores o datos de cada uno de los campos. Este es un proceso complicado ya que abarca decisiones a muy distintos niveles, pero si el problema se divide en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se divide en diseño conceptual (requisitos de usuario), diseño lógico (descripción de la estructura de la base de datos) y diseño físico (descripción de la implementación de una base de datos).

4.3 Modelo Entidad-Relación (E/R)

Este modelo es basado en una apreciación del mundo real, la cual está formada por objetos llamados entidades y las relaciones entre estos objetos y características de estos objetos llamados atributos.

Entidad

Es un objeto que se pude distinguir de otros objetos.

- EGRESADO
- DATOS_ASESOR
- GRADOS_ASESOR
- FORMAS_TITULACION
- EGRESADOSXFORMAS_TITULACION

Relaciones

Una relación es la asociación entre dos o más entidades.

- EGRESADOSXFORMAS TITULACION tiene un EGRESADO
- EGRESADOSXFORMAS TITULACION tiene una FORMAS TITULACION
- DATOS ASESOR contiene un GRADOS ASESOR

Atributos

Son las características de las Entidades

Tipos de Atributos:

- Valores
  - ✓ Monovaluados (ej.: edad)
  - ✓ Multivaluados (ej.: teléfonos)
- Almacenados o derivados
  - ✓ Ej.: la edad de una persona es casi siempre un atributo derivado de la fecha de nacimiento
- Posiblemente nulos
  - ✓ Cuando un atributo se puede dejar “en blanco”
- Claves
  - ✓ Permiten localizar una entidad, son únicos

4.3.1 Diagrama Entidad-Relación

Este modelo representado por un esquema gráfico (Figura 4.2).

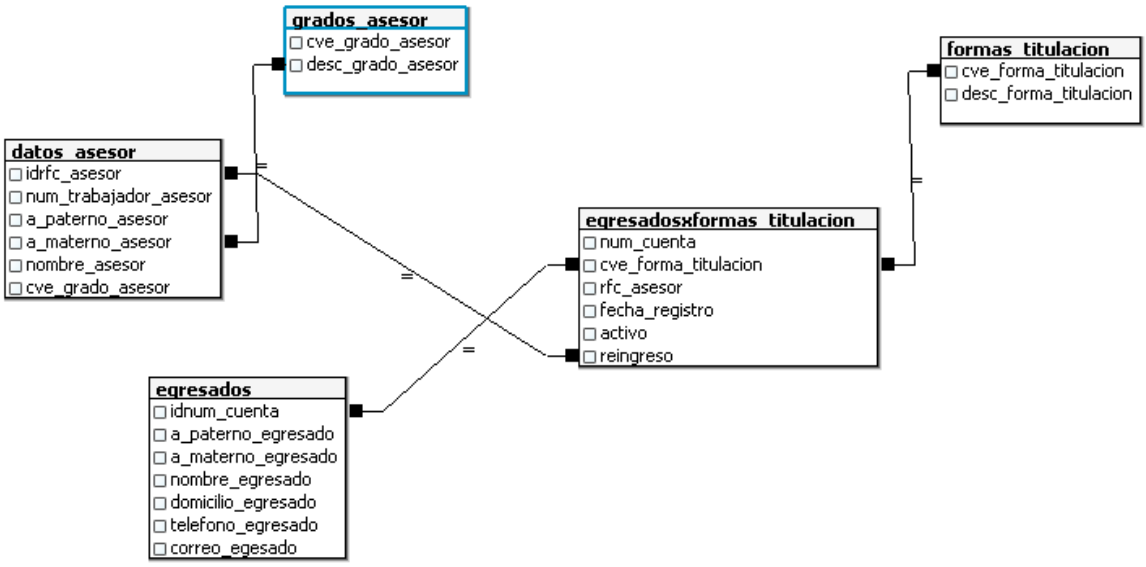


Figura 4.2 Diagrama entidad-relacion

4.4 Creación de la base de datos en PostreSQL.

A continuación mostraremos las tablas que fueron creadas en la base de datos Pre-registro.

```
-- Database: "Pre-registro"

-- DROP DATABASE "Pre-registro";

-- Database: pre_registro
-- DROP DATABASE pre_registro;
CREATE DATABASE pre_registro
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       LC_COLLATE = 'Spanish_Mexico.1252'
       LC_CTYPE = 'Spanish_Mexico.1252'
       CONNECTION LIMIT = -1;
CREATE TABLE EGRESADOS(
IDNUM_CUENTA   VARCHAR(10),
A_PATERNO_EGRESADO VARCHAR(50),
A_MATERNO_EGRESADO VARCHAR(50),
NOMBRE_EGRESADO  VARCHAR(50),
DOMICILIO_EGRESADO VARCHAR(200),
TELEFONO_EGRESADO VARCHAR(15),
CORREO_EGESADO  VARCHAR(150),
CONSTRAINT PK_IDNUM_CUENTA PRIMARY KEY(IDNUM_CUENTA)
)

CREATE TABLE FORMAS_TITULACION(
CVE_FORMA_TITULACION INT,
DESC_FORMA_TITULACION VARCHAR(200),
CONSTRAINT PK_CVE_FORMA_TITULACION PRIMARY KEY(CVE_FORMA_TITULACION)
)
CREATE TABLE GRADOS_ASESOR(
CVE_GRADO_ASESOR INT,
DESC_GRADO_ASESOR VARCHAR(50),
CONSTRAINT PK_CVE_GRADO_ASESOR PRIMARY KEY(CVE_GRADO_ASESOR)
)
CREATE TABLE DATOS_ASESOR(
IDRFC_ASESOR  VARCHAR(18),
NUM_TRABAJADOR_ASESOR INT,
A_PATERNO_ASESOR VARCHAR(50),
A_MATERNO_ASESOR VARCHAR(50),
NOMBRE_ASESOR  VARCHAR(50),
CVE_GRADO_ASESOR INT,
CONSTRAINT PK_IDRFC_ASESOR PRIMARY KEY(IDRFC_ASESOR)
)
```

```
CREATE TABLE EGRESADOSxFORMAS_TITULACION(
NUM_CUENTA VARCHAR(10),
CVE_FORMA_TITULACION INT,
RFC_ASESOR VARCHAR(18),
FECHA_REGISTRO DATE,
ACTIVO INT,
REINGRESO INT
)

ALTER TABLE EGRESADOSxFORMAS_TITULACION ADD CONSTRAINT FK_CAT_EGRESADOS FOREIGN KEY(NUM_CUENTA) REFERENCES CAT_EGRESADOS (NUM_CUENTA)
ALTER TABLE EGRESADOSxFORMAS_TITULACION ADD CONSTRAINT FK_CAT_FORMAS_TITULACION FOREIGN KEY(CVE_FORMA_TITULACION) REFERENCES CAT_FORMAS_TITULACION (CVE_FORMA_TITULACION)
ALTER TABLE EGRESADOSxFORMAS_TITULACION ADD CONSTRAINT FK_CAT_DATOS_ASESOR FOREIGN KEY(RFC_ASESOR) REFERENCES CAT_DATOS_ASESOR (RFC_ASESOR)


SELECT * FROM CAT_EGRESADOS
DROP TABLE CAT_EGRESADOS
```

4.4.1 Diccionario de datos.

Tabla de EGRESADO

Columna	Tipo de dato	Descripción egresado
IDNUM_CUENTA	VARCHAR(10)	Llave primaria de la tabla, identificador del alumno
A_PATERNO_EGRESADO	VARCHAR(50)	Apellido paterno del egresado
A_MATERNO_EGRESADO	VARCHAR(50)	Apellido materno del egresado
NOMBRE_EGRESADO	VARCHAR(50),	Nombre del egresado
DOMICILIO_EGRESADO	VARCHAR(200),	Domicilio completé del egresado
TELEFONO_EGRESADO	VARCHAR(15)	Teléfono de casa o celular del egresado
CORREO_EGESADO	VARCHAR(150),	Correo del egresado

Tabla FORMAS_TITULACION.

Columna	Tipo de dato	Descripción egresado
CVE_FORMA_TITULACION	INT	Llave primaria de la tabla, campo incrementable, identificador para la forma de titulación
DESC_FORMA_TITULACION	VARCHAR(200)	Nombre de la forma de titulación.

Tabla GRADOS_ASESOR.

Columna	Tipo de dato	Descripción egresado
CVE_GRADO_ASESOR	INT	Llave primaria de la tabla, campo incrementable, identificador para el grado de asesor
DESC_GRADO_ASESOR	VARCHAR(50)	Título del asesor.

Tabla de DATOS_ASESOR

Columna	Tipo de dato	Descripción egresado
IDRFC_ASESOR	VARCHAR(18),	Llave primaria de la tabla, campo incrementable, identificador para el grado de asesor
NUM_TRABAJADOR	INT	Número que otorga la UNAM a los maestros.
A_PATERNO_ASESOR	VARCHAR(50),	Apellido paterno del asesor
A_MATERNO_ASESOR	VARCHAR(50),	Apellido materno del asesor
NOMBRE_ASESOR	VARCHAR(50),	Nombre del asesor
CVE_GRADO_ASESOR	INT	Título del asesor.

Tabla EGRESADOSXFORMAS_TITULACION

Columna	Tipo de dato	Descripción egresado
IDNUM_CUENTA	VARCHAR(10),	Llave foránea de la tabla EGRESADO
CVE_FORMA_TITULACION	VARCHAR(50),	Llave foránea de la tabla FORMAS_TITULACION.
IDRFC_ASESOR	VARCHAR(50),	Llave foránea de la tabla DATOS_ASESOR
FECHA_REGISTRO	DATE	Día, mes, año en que se tramitó el registro de la titulación
ACTIVO	VARCHAR(50),	Si no existen los datos en la base de datos de egresados se da de alta
REINGRESO	VARCHAR(200),	Si ya existen los datos del egresado no es necesario volver a ingresarlos.

Capítulo 5

IMPLEMENTACIÓN Y PRUEBAS

Interfaz y manual de usuario

5. IMPLEMENTACIÓN Y PRUEBAS.

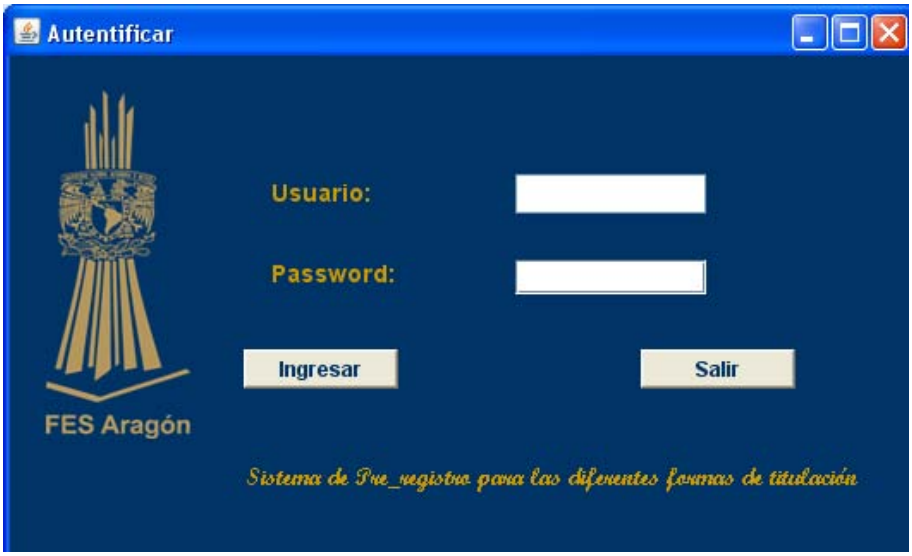
5.1 Interfaz y Manual de Usuario

Al iniciar el sistema se mostrará la pantalla **Inicio** (Figura 5.1) en la cual tendremos 2 formas de ingresar al sistema ya sea por: **Administrador*(Ad)*** e **Usuario *(Us)***

Figura 5.1 Pantalla Inicio.

```
@SuppressWarnings("unchecked")
Generated Code

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    autentificar nuevaVentana = new autentificar();
    nuevaVentana.setVisible(true);
    inicio.this.dispose();



}

private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {...}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    opcion2 OpcionUsuario = new opcion2();
    OpcionUsuario.setVisible(true);
    inicio.this.dispose();
}

/**...*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new inicio().setVisible(true);
        }
    });
}
```

Si ingresamos por el modo de ***(Ad)*** el sistema nos muestra la pantalla de **Autentificar** (Figura 5.2) aquí ingresaremos el nombre de usuario y password, si los datos no son válidos nos mostrará una alerta de “La contraseña no es correcta”.



5.2 Pantalla Autentificar.

```
public autentificar() {
    initComponents();
}

/**...*/
@SuppressWarnings("unchecked")
Generated Code

private void button2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    inicio regresar = new inicio();
    regresar.setVisible(true);
    this.dispose();
}

private void button1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     String usuario=textField1.getText();
     String password=jPasswordField1.getText();

     if ((usuario.equals("admin"))&&(password.equals("ico")) )
         {
         opcion1 validado = new opcion1();
         validado.setVisible(true);
         this.dispose();
         }
      else
         {
         JOptionPane.showMessageDialog(this, "Usuario no valido", "error",JOptionPane.ERROR_MESSAGE);
         }
```

Si el nombre de usuario y password son válidos el sistema muestra la pantalla de **Opciones** (Figura 5.3).

Figura 5.3 Pantalla de Opciones

```
package pegistro;

/**...*/
public class opcion1 extends javax.swing.JFrame {

    /** Creates new form opcion1 */
    public opcion1() {
        initComponents();
    }

    /**...*/
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        registroegre regegre = new registroegre();
        regegre.setVisible(true);
        opcion1.this.dispose();
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        registroase regase = new registroase();
        regase.setVisible(true);
        opcion1.this.dispose();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {



        opcion1.this.dispose();
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        buscaregre busegre = new buscaregre();
        busegre.setVisible(true);
        opcion1.this.dispose();

    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         buscarase busase = new buscarase();
        busase.setVisible(true);
        opcion1.this.dispose();
    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        estadistica1 estadis = new estadistica1();
        estadis.setVisible(true);
        opcion1.this.dispose();
    }

    /**...*/
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new opcion1().setVisible(true);
```

Si el ***(Ad)*** selecciona la opción de **Registro de Egresado**, el sistema nos muestra la pantalla de **Registro de Egresado** (Figura5.4) la cual llenaremos con los datos proporcionados por él egresado y pulsaremos el botón de Registrar, si faltaron campos de llenar nos mostrará un alerta “Falta campos por llenar”, si está completo el registro el sistema regresará a la pantalla de **Opciones,** y si damos clic en el botón de **Nuevo** podremos ingresar otro **Egresado** (Figura 5.4).

Figura 5.4 Pantalla de Registro de Egresado

```
package pegistro;
import java.sql.*;
//import javax.swing.*;
//import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.DateFormat;
import javax.swing.JOptionPane;

/**...*/
public class registroegre extends javax.swing.JFrame {
Egresado p = new Egresado();
Egresado prue = new Egresado();
Conectate conn = new Conectate();
Statement st = null;
 ResultSet rs= null;
    public registroegre() {
        initComponents();

        Combo1.addItem("selecciona");

    try {

 st=conn.getConnection().createStatement();
        rs=st.executeQuery("select * from formas_titulacion order by desc_forma_titulacion");
while(rs.next())


     Combo1.addItem(rs.getString(2));

     }
     } catch (SQLException e) {e.printStackTrace();}

             Combo2.addItem("selecciona");

         try {

      st=conn.getConnection().createStatement();
             rs=st.executeQuery("select * from asesor order by appaterno");
     while(rs.next())
     {

     Combo2.addItem(rs.getString(3)+ " "+rs.getString(4)+ " "+ rs.getString(5));

     }
     } catch (SQLException e) {e.printStackTrace();}



         }
     private void nuevo(){
     txtcuenta.setText("");
     txtnombre.setText("");
     txtpaterno.setText("");
     txtmaterno.setText("");
     txtdomicilio.setText("");
     txttel.setText("");
```

Si el ***(Ad)*** selecciona la opción de **Registro de Asesor**, el sistema nos muestra la pantalla de **Registro de Asesor** (Figura 5.5) la cual llenaremos con los datos personales del maestro y pulsaremos el botón de Registrar, si faltaron campos de llenar nos mostrará un alerta “Falta campos por llenar” y si esta completo el registro el sistema regresará a la pantalla de **Opciones** (Figura 5.3).

Figura 5.5 Pantalla de Registro de Asesor

```
package pegistro;
import java.sql.*;
import javax.swing.*;
//import java.sql.PreparedStatement;
import java.sql.SQLException;


/**...*/
public class registroase extends javax.swing.JFrame {

     Statement st = null;
 ResultSet rs= null;
 asesor a= new asesor();
Conectate conn = new Conectate();
DefaultListModel modelo1= new DefaultListModel ();
    /** Creates new form registroase */
    public registroase() {
        initComponents();

        grado.addItem("selecciona");

    try {


 st=conn.getConnection().createStatement();
        rs=st.executeQuery("select * from grados_asesor order by desc_grado_asesor");
while(rs.next())

grado.addItem(rs.getString(2));

}
} catch (SQLException e) {e.printStackTrace();}

}


    /**...*/
    @SuppressWarnings("unchecked")
    Generated Code
private void nuevo(){
txtrfc.setText("");
txtapa.setText("");
txtama.setText("");
txtnumt.setText("");
txtnombre.setText("");
grado.setSelectedIndex(0);

}

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        opcion1 regresar = new opcion1();
        regresar.setVisible(true);
        this.dispose();
}
```

Si el ***(Ad)*** selecciona la opción de **Buscar Egresado**, el sistema nos muestra la pantalla de **Buscar Egresado** (Figura 5.25) la cual llenaremos con algunos de los datos del egresado, si el sistema no encuentra al egresado manda una alerta “No se encontró al egresado” y nos mostrará la pantalla de Información del egresado.

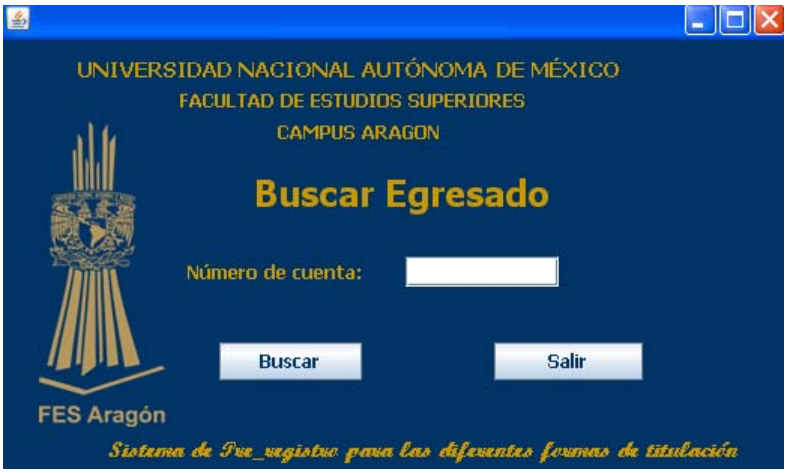


Figura 5.6 Pantalla de Buscar Egresado

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
   try {


        String b;
        b=txtnumcue.getText();
        ResultSet rs= null;
         st=con.getConnection().createStatement();
         rs=st.executeQuery("select * from prueba WHERE cuenta='"+b+"'");

         boolean encuentra=false;

         while(rs.next())
         {

         if(b.equals(rs.getString(2)));
         {
         modiegre modificar=new modiegre();
         modificar.setVisible(true);
         buscaregre.this.dispose();

         modificar.txtcuenta.setText((String)rs.getString(1));

         modificar.txtnoe.setText((String)rs.getString(2));
         modificar.txtape.setText((String)rs.getString(3));
         modificar.txtame.setText((String)rs.getString(4));
```

Si el ***(Ad)*** selecciona la opción de **Modificar Egresado** (Figura 5.7), el sistema nos muestra la pantalla de **Buscar Egresado** (Figura5.6) la cual llenaremos con el número de cuenta del egresado, si el sistema no encuentra al egresado manda una alerta "No se encontró al egresado" y si está en la base de datos nos mostrará la pantalla de Información del egresado.

En esta pantalla podremos modificar los datos del egresado y después pulsaremos el botón de Modificar y nos saldrá un cuadro de dialogo "El egresado ha sido modificado".

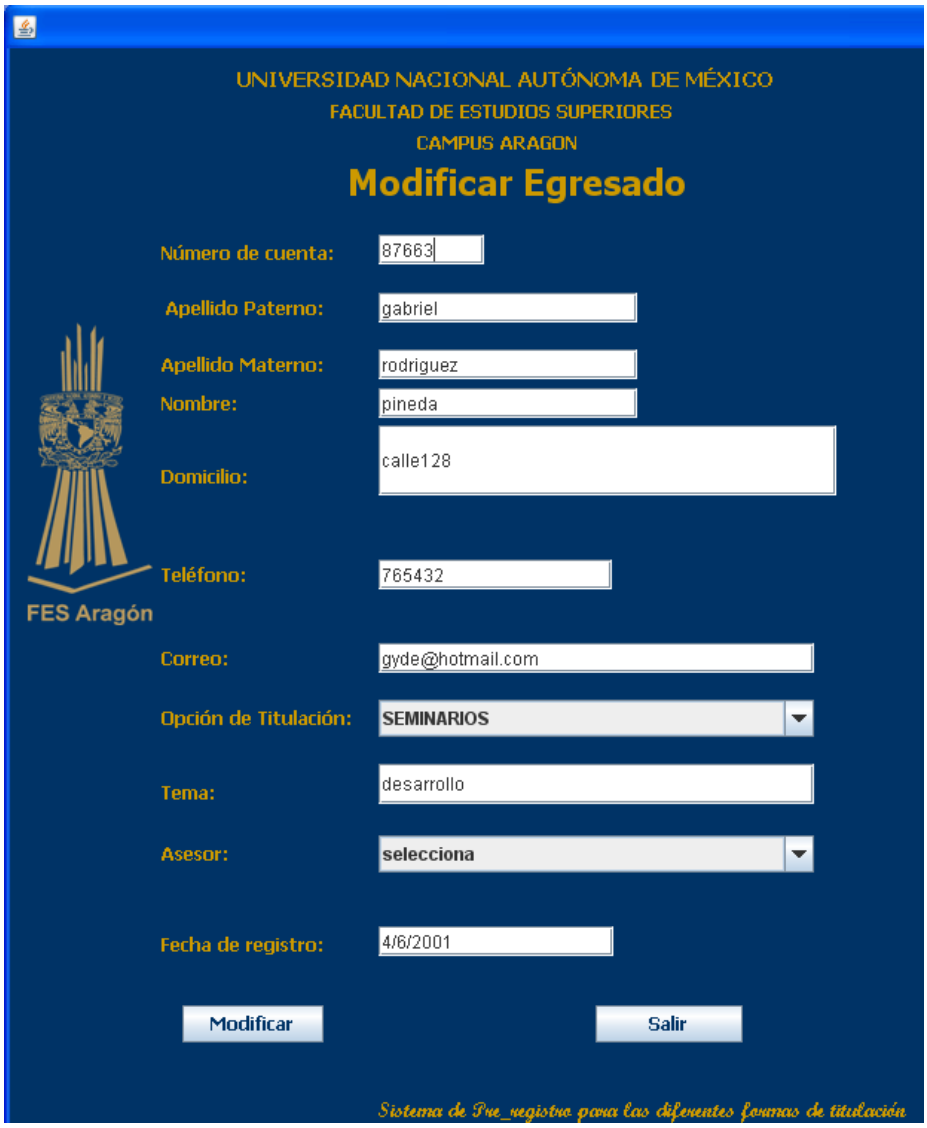


Figura 5.7 Pantalla Modificar egresado

```
Conectate con = new Conectate();
Statement st = null;
 ResultSet rs= null;

    /** Creates new form modiegre */
    public modiegre() {
        initComponents();
      combo1.addItem("selecciona");

    try {

 st=con.getConnection().createStatement();
        rs=st.executeQuery("select * from formas_titulacion order by desc_forma_titulacion");
while(rs.next())
{

combo1.addItem(rs.getString(2));

}
} catch (SQLException e) {e.printStackTrace();}

        combo2.addItem("selecciona");

    try {

 st=con.getConnection().createStatement();
        rs=st.executeQuery("select * from asesor order by appaterno");
while(rs.next())
{
```

```
              String correo, moda,tema,asesor,fecha,tel,cuenta;


              cuenta=txtcuenta.getText();
              nomb = txtnoe.getText();
              apa = txtape.getText();
             ama = txtame.getText();
             doc = txtdoe.getText();
             correo = txtcoe.getText();
             asesor=combo2.getSelectedItem().toString();
             moda=combo1.getSelectedItem().toString();
             tema =txttem.getText();
              fecha=txtfer.getText();
              tel= txttee.getText();
              ResultSet rs= null;
               st=con.getConnection().createStatement();
               rs=st.executeQuery("update prueba"
                       + " set nombre='"+nomb+"'"
                       + ", apppaterno='"+apa+"'"
                       + ",appmaterno='"+ama+"'"
                       + ",domicilio='"+doc+"'"
                       +",correo = '"+correo+"'"
                       +",modalidad = '"+moda+"'"
                       +",tema = '"+tema+"'"
                       +",asesor = '"+asesor+"'"
                       +",fecha = '"+fecha+"'"
                       +",telefono = '"+tel+"'"
                       + " where cuenta='"+cuenta+"' ");
                   //JOptionPane.showMessageDialog(this,"se guardao","acep
    }
```

Si el ***(Ad)*** selecciona la opción de **Modificar Asesor** (Figura 5.9), el sistema nos muestra la pantalla de **Buscar Asesor** (Figura5.8) la cual llenaremos con algunos de los datos del asesor, si el sistema no encuentra al asesor manda una alerta “No se encontró el asesor” y si está en la base de datos nos mostrará la pantalla de Información del asesor.

Figura 5.8 Pantalla de Buscar Asesor

```
private void buscarAsActionPerformed(java.awt.event.ActionEvent evt) {

        try {

       String b;
       b=txtnumt.getText();
       ResultSet rs= null;
        st=con.getConnection().createStatement();
        rs=st.executeQuery("select * from asesor WHERE numtrabajador='"+b+"'");

        boolean encuentra=false;

        while(rs.next())
        {

        if(b.equals(rs.getString(2)));
        {
        modiase modificar=new modiase();
        modificar.setVisible(true);
        buscarase.this.dispose();
        modificar.txtrfcm.setText((String)rs.getString(1));
        modificar.txtnumm.setText((String)rs.getString(2));
        modificar.txtapm.setText((String)rs.getString(3));
        modificar.txtamm.setText((String)rs.getString(4));
        modificar.txtnomm.setText((String)rs.getString(5));
        modificar.grado1.setSelectedItem((String)rs.getString(6));
```

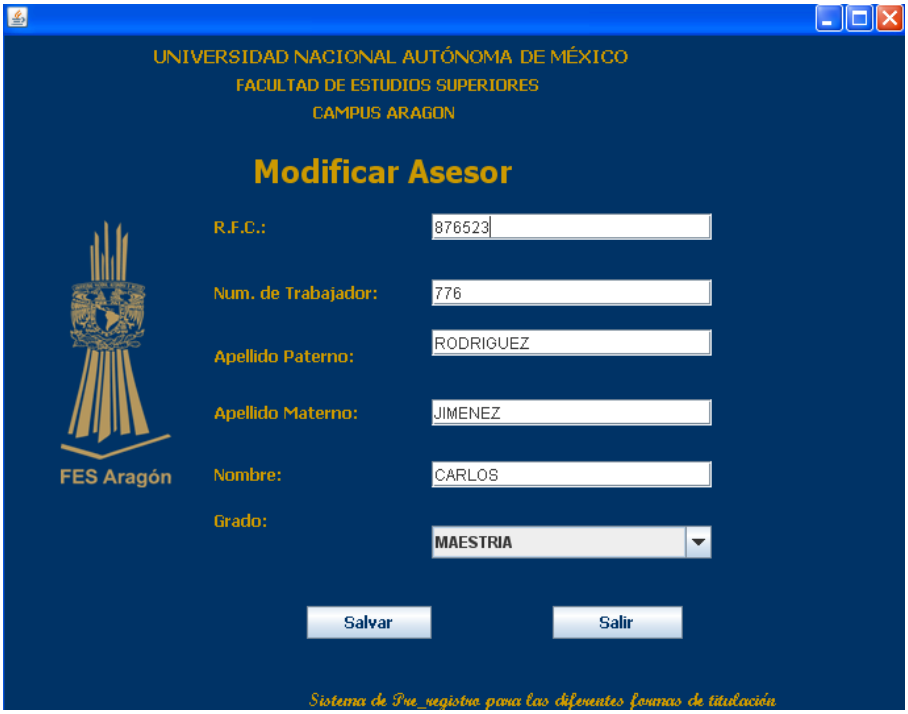


Figura 5.9 Pantalla de Buscar Asesor

```
private void modificarActionPerformed(java.awt.event.ActionEvent evt) {

        try {
        String rfc;
        String nombre;
        String ap;
        String  am;
        String numtra, grado;



        rfc = txtrfcm.getText();
            nombre = txtnomm.getText();
       numtra = txtnumm.getText();
       ap = txtapm.getText();
       am = txtamm.getText();
       grado=grado1.getSelectedItem().toString();

        ResultSet rs= null;
         st=con.getConnection().createStatement();
         rs=st.executeQuery("update asesor"
                 + " set rfc='"+rfc+"'"
                 + ", appaterno='"+ap+"'"
                 + ",apmaterno='"+am+"'"
                 + ",nombre='"+nombre+"'"
                 +",grado = '"+grado+"'"
                 + " where numtrabajador='"+numtra+"' ");
    }
```

Si el ***(Ad)*** selecciona la opción de Estadísticas, el sistema nos muestra la pantalla de **Tipos de Estadísticas** (Figura5.10).

Figura 5.10 Pantalla Estadísticas

```
        // TODO add your handling code here:
        opcion1 regresar = new opcion1();
        regresar.setVisible(true);
        this.dispose();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
estadistica regresar = new estadistica();
        regresar.setVisible(true);
        this.dispose();         // TODO add your handling code here:
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
estadisticaAse regresar = new estadisticaAse();
        regresar.setVisible(true);
        this.dispose();         // TODO add your handling code here:
         // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
estadisticaperiodo regresar = new estadisticaperiodo();
        regresar.setVisible(true);
        this.dispose();         // TODO add your handling code here:
        // TODO add your handling code here:
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
estadisticaMyT regresar1 = new estadisticaMyT();
        regresar1.setVisible(true);
        this.dispose();         // TODO add your handling code here:
```

Si el ***(Ad)*** selecciona la opción de **Modalidad**, el sistema mostrará la pantalla de **Modalidad** (Figura5.11) donde nos mostrará las diferentes formas de Titulación y en la parte inferior de nuestra pantalla se muestra la tabla de Estadísticas.

Figura 5.11 Pantalla Estadísticas modalidad

```
public class estadistica extends javax.swing.JFrame {
     Statement st = null;
 ResultSet rs= null;
 //asesor a= new asesor();
Conectate conn = new Conectate();
//DefaultListModel modelo1= new DefaultListModel ();
Object[][] dtPer;
int fila = -1;

    /** Creates new form estadistica1 */
    public estadistica() {
        initComponents();
            combomoda.addItem("selecciona");

    try {


 st=conn.getConnection().createStatement();
        rs=st.executeQuery("select * from formas_titulacion order by desc_forma_titulacion");
while(rs.next())
{

combomoda.addItem(rs.getString(2));

}
} catch (SQLException e) {e.printStackTrace();}
```

Conclusiones

CONCLUSIONES

El desarrollo del sistema se realizó bajo varias técnicas y bajo la metodología del diseño del software, apoyándonos en herramientas del modelado (UML, ArgoUML), entornos de desarrollo integrado (Netbeans) y herramientas de diseño de base de datos (Postgresql). Buscamos que todo el software fuera libre para no tener problemas de licencia.

El sistema tiene el control de los datos de los egresados que han registrado una forma de titilación, con esto se tiene de forma segura los datos y las consultas se realizan en un menor tiempo de forma eficaz, en comparación de cómo se almacenaban los datos (hoja de calculó) anteriormente.

Aprendimos el manejo de herramientas que nos ayudaron a agilizar la elaboración del sistema y así como también tener un programa más estable, además pusimos en práctica varios conocimientos adquiridos de la carrera de Ingeniería en Computación: programación orientada a objetos, manejadores de base de datos, Proceso Unificado de Rational (RUP) este proceso nos ayudo a desarrollar, diseñar e implementar el sistema. UML nos ayudo al entendimiento total del sistema por medio de diagramas donde describimos los métodos y procesos del sistema y a su vez nos ayudo a enfocarnos claramente en como actuará nuestro sistema con el usuario.

Las principales dificultades en tecnologías fueron el determinar que software utilizamos ya que en algunos casos es necesario contar con las licencias para las instalaciones como en el caso de Microsoft SQL Server 2005, Visual Studio; por lo cual se optó por utilizar software que no contara con este impedimento, como es el caso de PostgreSQL y java con su IDE Netbeans.

La educación que se nos dio en la FES Aragón como ingenieras nos prepara para encontrar la manera de resolver problemas que nos sean planteados y tener una visión más amplia de las formas de atacar un problema, en este caso esta preparación nos sirvió de guía para realizar el análisis del problema y posteriormente un diseño del sistema. En algunas de las materias de la carrera de Ingeniería en Computación y a los cursos ínter-semestrales adquirimos los conocimientos y principios de las bases de datos y programación para poder desarrollar nuestro sistema

Bibliografía

BIBLIOGRAFÍA

1. Ian Sommerville 1995. Ingeniería de Software, 5ta. Edición. Addison Wesley.
2. Rogers. Pressman 1997. Ingeniería del Software, Un enfoque práctico 4ta. Edición. McGraw Hill.
3. Richard E. Fairley 1993. Ingeniería del Software 3ra Edición. Mc Graw Hill.
4. Javier García de Jalón 2000. Aprenda Java como si estuviera en primero. Escuela Superior de Ingenieros.
5. Joseph Schmuller 2000. Aprendiendo UML en 24 horas 2da Edición. Prentice Hall.
6. James Rumbaugh, Ivar Jacobs, Grady Booch 1999. The Unifed Modeling Language Reference Manual. Addison-Wesley.
7. Bruce Momjian 2001. PostgreSQL: Introduction and Concepts 2da Edición. Pearson Education
8. Thomas Lockhart 2008. Manual del usuario de Postgresql: El equipo de desarrollo de Postgresql. Postgrsql Global Development Group.

Referencias de internet

Última consulta.

1. Tutoriales de java básicos y como crear la GUI.

   http://java.sun.com/docs/books/tutorial

   Septiembre del 2009.

2. Conceptos de Ingeniería del software.

   http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

   Noviembre del 2009.

3. Conceptos de Ingeniería del Software.

   http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software

   Diciembre del 2009.

4. Conceptos de RUP.

   http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

   Diciembre del 2009.

5. Conceptos de UML.

   http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

   Julio del 2010.

6. Manual de Postgresql.

   http://www.youblisher.com/p/46270-Manual-de-Postgress/

   Septiembre del 2010.

7. Tutoriales de Java con ejemplos.

   http://www.javadabbadoo.org/cursos/infosintesis.net/javaseav/paqsql/altabajamod/paso02operativa.html

   Octubre del 2010.

8. Conexión de Postgresql con Netbeans.

   http://migue.pezweb.com/archives/11

   Octubre del 2010.

9. Tutoriales de Netbeans.

   http://wiki.netbeans.org/Avbravo_TutorialesEspanol

   Octubre del 2010.