UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMATICA

PROYECTO DE GRADO

SOFTWARE DE GESTIÓN Y SEGUIMIENTO ACADÉMICO
CASO: CENTRO DE CAPACITACIÓN TÉCNICA
CEINF RM 175/13

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMATICA
MENCION: INGENIERIA DE SISTEMAS INFORMATICOS

POSTULANTE: Univ. GHILMAR EMIGDIO COPA MAMANI

TUTOR METODOLOGICO: Lic. GROVER ALEX RODRIGUEZ RAMIREZ

ASESOR: Lic. JAVIER REYES PACHECO

LA PAZ – BOLIVIA

2014

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A Dios por ser mi amigo e incondicional mi guía y mi fortaleza en todo tiempo. A mis padres Jhonny e Hilda por brindarme el apoyo a lo largo de mi carrera, a mis amigos por la amistad que me brindan y por su apoyo incondicional.

AGRADECIMIENTOS

A Dios, quien siempre guía mi camino y me da la fortaleza para seguir creciendo, en el transcurrir de mi vida.

A mi familia por todo el apoyo incondicional a lo largo de mi carrera, a mis amigos y amigas de la Universidad que me dieron la fuerza necesaria para seguir adelante.

A los docentes de la carrera de informática por el excelente trabajo que realizan,.

Gracias a Todos.

RESUMEN

Dado la importancia de la incorporación de las nuevas tecnologías de información y comunicación dentro de las instituciones, las cuales no pueden quedar al margen del uso de sistemas de información automatizadas que ayuden a mejorar los procesos académicos, se ha visto la necesidad de desarrollar un software de información y seguimiento académico.

El proyecto debe adaptarse de manera adecuada al uso de las nuevas tecnologías para el tratamiento de la información, ya que con ello lograra mejorar los servicios que brinda a la sociedad.

Con este objetivo se realiza el presente Proyecto de Grado titulado software de gestión y seguimiento académico caso: centro de capacitación técnica“CEINF RM 175/13”.

En estos momentos centro de capacitación técnica CEINF, actualmente no cuenta con sistema de información que coadyuven con tareas y actividades que son importantes como ser inscripción de estudiantes, boleta de inscripción y seguimiento de académico, calificaciones, ya que son realizadas de modo manual.

Con el presente proyecto se pretende la implementación del software de gestión y seguimiento académico, para brindar información confiable, reduciendo así el trabajo excesivo.

INDICE

CAPITULO 1 MARCO REFERENCIAL

CAPITULO 2 MARCO TEORICO

CAPITULO 3 MARCO APLICATIVO

CAPITULO 4 METRICAS DE CALIDAD

CAPITULO 5 EVALUACION DE COSTOS Y BENEFICIOS

CAPITULO 6 SEGURIDAD DE SISTEMA

CAPITULO 7 CONCLUSIONES Y RECOMENDACIONES

INDICE DE FIGURAS.

INDICE DE TABLAS

CAPITULO 1 MARCO REFERENCIAL

1 INTRODUCCION.
1.1. INTRODUCCIÓN

La información va teniendo un gran desarrollo y se generan en grandes volúmenes en el tratamiento de la información, tanto así, que actualmente se ha vuelto indispensable en las actividades de las Empresas.

Aquellas instituciones que han implementado un software informático adecuado para su trabajo que realizan tienen un mayor desarrollo en el funcionamiento de sus tareas, llevando ventajas a otras instituciones u organizaciones que trabajan aun con métodos manuales.

El centro de capacitación técnica "CEINF" no está al margen de la utilización de un software informático, dado que la administración de la información es fundamental para lograr un buen desempeño en las labores académicas.
Es fundamental contar con un software que agilice la información habitual del centro de educación y lograr así, ejecutar tareas de manera óptima y precisa.
Por estas características, se debe implementar un software de información para almacenar datos, procesarlos de manera adecuada y contar con información oportuna y confiable, adecuada para las necesidades de la institución, haciendo que estos tengan mayor ventaja a la información que reciben, y más aun a la eficacia que tendrá el software.

Este procedimiento es importante para el centro de capacitación técnica "CEINF", en cada inicio de gestión realiza las actividades respectivas para inscribir a los estudiantes, guardando la información en documentos y papeles que se van acumulando al paso de los años.
Los estudiantes solicitan información de los cursos que se imparten en la institución si desean se inscriben llenando un formulario de inscripción, la hoja después es transcrita a un archivo Excel.
Por esta razón es importante realizar el presente proyecto de grado titulado "software de seguimiento académico caso centro de capacitación técnica CEINF".
Con el software se registra al estudiante como es debido, optimizando las tareas y los procedimientos administrativos, que se realiza en toda la gestión, también se pretende mejorar el control y el seguimiento académico de los estudiantes, brindando toda la información que se requiera en tiempo real.

1.2 ANTECEDENTES

1.2.1 ANTECEDENTES DE LA INSTITUCION

El centro de "Capacitación Técnica CEINF", ha sido elaborado en base a la realidad socio-económico-cultural de nuestra sociedad actual en que vivimos, hoy en este Siglo XXI, pretende ser un instrumento técnico de capacitación para todos los niños en edad escolar, jóvenes, adultos y otros profesionales, deseosos de conocer, manejar y aprovechar el amplio campo de la informática.

En este nuevo enfoque, el Proyecto Educativo Institucional de Capacitación en Informática, requiere un compromiso, respeto mutuo, apertura de diálogo horizontal entre todos los integrantes del Instituto de referencia, donde exista una participación

Interactiva y real de todos sus integrantes, donde todos asuman responsabilidades, garantizando la capacitación y formación de los clientes en informática.
En función de los principios, fines, objetivos, la Visión, la Misión y los Valores que sustenta el Proyecto Educativo Institucional de Capacitación en Informática, la política educativa que prioriza es formar, desarrollar capacidades y competencias por la comprensión de la informática en los niños en edad escolar, adolescentes, jóvenes y adultos en el amplio campo de la informática, acorde al avance de la revolución científica y tecnológica de nuestros tiempos.

Por esta razón, teniendo conocimiento que existe la necesidad de brindar y cooperar a todos los interesados beneficiarios, sean iniciantes y/o profesionales, quiénes poseen la urgente necesidad de fortalecer su formación en el campo de la informática, se organiza el Instituto de Capacitación y Formación en Informática, en función de las reales necesidades, intereses y aspiraciones de nuestra sociedad, local, regional y nacional.

El centro de capacitación técnica "CEINF" es una institución que esta abalada por el ministerio de educación con la resolución ministerial RM 175/13.
Se encuentra ubicada en la ciudad de La Paz en el distrito COTAHUMA , en la av. Mariscal Santa Cruz Edificio, Esperanza Piso 8 of. 1
El centro brinda a los estudiantes, una formación integral en el área de informática, arquitectura, contabilidad e ingeniería civil.
En función a las necesidades de manejar y aprovechar el amplio campo de la informática.
Actualmente el sistema del instituto funciona de forma manual, el alumno solicita la información de los cursos que se enseña o se imparte, a los alumnos o interesados después si ellos desean inscribirse al curso. La secretaria los inscribe con su ci.

Y al finalizar se les otorga un certificado con la carga horaria y con la nota respectiva.

1.3 PLANTEAMIENTO DEL PROBLEMA

El software de información para el seguimiento y control académico, en el centro de capacitación "CEINF", funciona manualmente proporcionando, información desactualizada, inoportuna y lenta.

Estos inconvenientes se ven incrementados por el carácter manual con el que se desarrollan las funciones del sistema.

Por lo cual el manejo de información es ineficiente, produciendo pérdida de tiempo.

Debido a estas circunstancias mencionadas, se elabora la siguiente lista de problemas.

- El proceso de inscripción se realiza manualmente
- Gran cantidad de información que es manipulada de forma manual en papeles y posteriormente se las transcribe a un archivo Excel.
- Deficiencia en la elaboración de reportes, las mismas se encuentran documentadas en papel.
- Gasto de tiempo y recursos, en el proceso de búsqueda de información de algún estudiante.

1.3.2 FORMULACIÓN DEL PROBLEMA

Es así, planteando el problema y habiendo tomado en cuenta las consideraciones y puntos de vista tratados, el presente proyecto de grado va de acuerdo a la misión del instituto, se propone responder lo siguiente.

¿El software de información de seguimiento académico para el centro de capacitación "CEINF", será capaz de mejorar los procesos de control y seguimiento académico reduciendo así el costo, tiempo, para obtener mejores resultados?

El software de información y seguimiento académico para el centro de capacitación Ceinf, propuesto y desarrollado hará posible la solución de todos los problemas y ayudara a la solución de tareas, procedimientos que demanda la institución.

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Construir un software de información computarizada para el seguimiento académico de estudiantes, que permita un adecuado seguimiento, organizar eficientemente la información y optimizar el tiempo, en los procesos de administración de información para el centro de capacitación técnica "CEINF".

Actualizando, registro de actividades facilitando el manejo de información y reduciendo el tiempo de elaboración de los reportes.

1.4.2 OBJETIVOS ESPECÍFICOS.

Como solución a los problemas, basados en los requerimientos de la institución, proponemos los siguientes puntos en el presente proyecto de grado.

- Construir el módulo de datos de inscripción personales y académicos de los estudiantes.
- Construir la base de datos del sistema, implementar procesos de protección de la información de la Base de datos.
- Garantizar la seguridad de la información, estableciendo restricciones de acceso al sistema.
- Optimizar la búsqueda de información que sea requerida por los estudiantes, administrativos, etc.
- Elaborar interfaz de usuario, que tenga fácil acceso a la información obtener reportes, ofreciendo un resguardo de seguridad criptográfica a la información de la base de datos.

1.5 ALCANCES Y LIMITES

1.5.1 ALCANCES.

El alcance de este proyecto, se pretende desarrollar e implementar un software informático en el control y seguimiento académico de los estudiantes.

El software se dedicara a la inscripción de los estudiantes, realizando el seguimiento de notas y asignación materias.

Dando cumplimiento a los objetivos planteados en el control y seguimiento académico de los estudiantes.

1.5.2LIMITES

Se desarrollara la automatización de los procesos de información académica de filiación

al registro de los estudiantes, docentes, asignación de materias, etc.

El desarrollo del software propuesto está dirigido al centro de capacitación, dando cumplimiento a los objetivos planteados en el proyecto.

1.6 JUSTIFICACIÓN

De acuerdo al presente proyecto de grado en primera instancia, ayudara a la institución a contar con un nuevo software de información académica, aplicando métodos y técnicas para su desarrollo, para que contribuyan a una adecuada información que permita lograr una funcionalidad más eficiente en el centro de capacitación.

Para que el manejo de información y búsqueda de datos sea óptima y eficiente para que contribuya a una adecuada información.

El software de seguimiento académico es un avance de soluciones para el manejo y procesamiento de información académica, cumpliendo así con las necesidades de la institución, a fin de facilitar y simplificar el trabajo manual.

El software facilitara el trabajo al personal de la institución, en los diferentes procesos mostrando la calidad del servicio a quienes requieran la información acerca de calificaciones, instructores, etc.

También con el desarrollo e implementación del software ayudara en el desempeño eficiente del seguimiento académico, permitiendo a la institución brindar un mejor servicio a los estudiantes.

El software también reducirá los costos de materiales e insumos necesarios utilizados para el procesamiento de la información, lo que permite a la institución a minimizar costos, pero el beneficio también será de forma intangible, se refleja a través de los resultados en tiempo y esfuerzo de trabajo.

1.7 APORTES

La implementación del presente proyecto para la institución, contara con el software de información especializada en el control y seguimiento académico.

El software será elaborado bajo factores de fiabilidad y facilidad de uso, permitiendo así la satisfacción del usuario que opera el software.

Coadyuvará al manejo de información académica del instituto, facilitando al acceso a grandes cantidades de información de manera rápida y sencilla.

1.8 METODOLOGIA

Los métodos y estándares de software, van proporcionando las guías para poder conocer todo el camino que se debe recorrer, desde su inicio hasta su implementación, con lo cual se asegura el cumplimiento y la entrega del producto final.

Para el proyecto de grado, se utiliza la metodología RUP (Rational Unified Process). El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.
La metodología RUP se encuentran las guías para documentar e implementar de manera fácil el desarrollo del software.

Ingeniería de software

Ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software.1 Es la aplicación de la ingeniería al software, ya que integra matemáticas, ciencias de la computación y prácticas cuyos orígenes se encuentran en la ingeniería.
Se pueden citar otras definiciones enunciadas por prestigiosos autores:

- Ingeniería de software es el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978)
- Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976).
- Ingeniería de software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable, que sea fiable y trabaje en máquinas reales (Bauer, 1972).

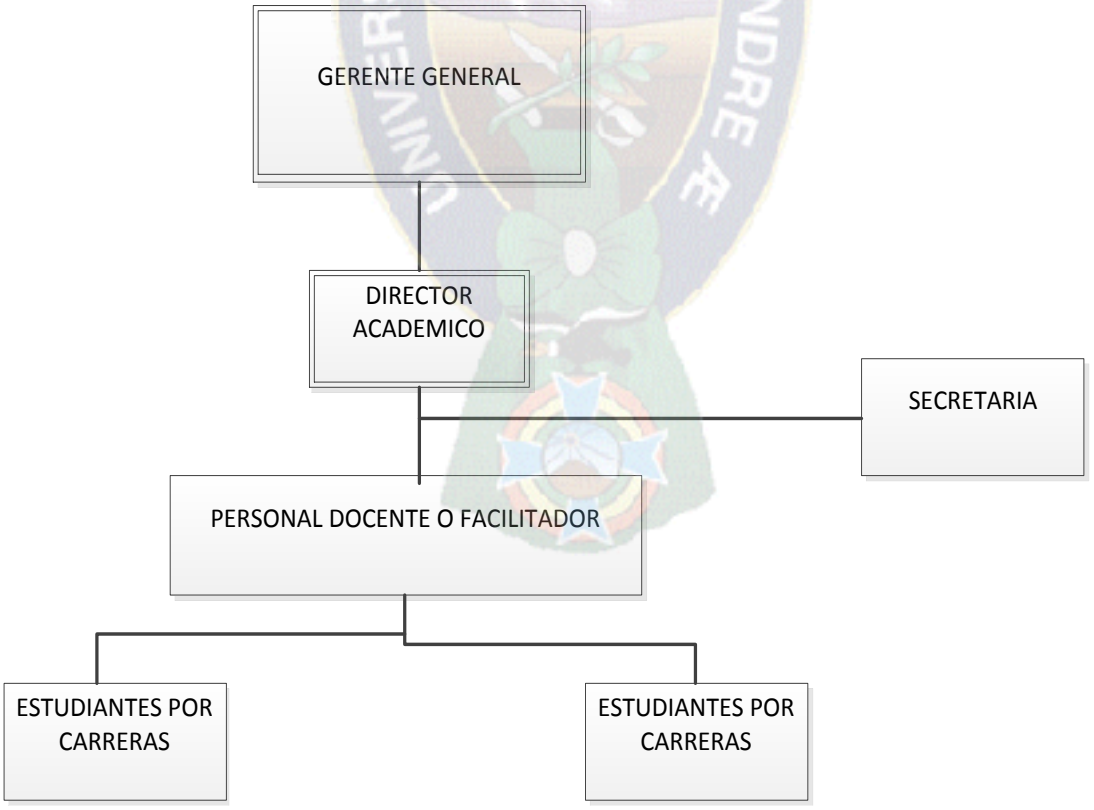
2 MARCO TEORICO

2.1 MARCO INSTITUCIONAL.

El instituto de capacitación en informática "CEINF "ha sido elaborado en base a la realidad socio-económico-cultural, pretende ser un instrumento técnico de capacitación para todos los niños en edad escolar, jóvenes, adultos y otros profesionales .El instituto inicio sus servicios el 2004 con la finalidad de poner los diferentes cursos que ofrecen ya sean de informática de ingeniería o de arquitectura.

En función de los principios, fines, objetivos, la Visión, la Misión y los Valores que sustenta el Proyecto Educativo Institucional de Capacitación en Informática, la política educativa que prioriza es formar, desarrollar capacidades y competencias por la comprensión de la informática en los niños en edad escolar, adolescentes, jóvenes y adultos en el amplio campo de la informática

Actualmente a la hora de realizar tareas y actividades que son necesarias, como la inscripción de estudiantes, boleta de inscripción seguimiento académico y entrega de certificado de aprobación son de forma manual



GERENTE GENERAL.

Es la máxima autoridad de la institución, ya que es el dueño de la misma. Tiene la finalidad de planear y desarrollar, metas a corto y largo plazo.

También realiza evaluaciones periódicas acerca del cumplimiento de las funciones de los diferentes departamentos.

DIRECTOR ACADEMICO.

Esta encargado de manejar la parte académica de la institución, tiene como objetivo organizar y planificar los cursos programados en las áreas de informática, arquitectura, e ingeniería.

Asigna aulas horarios y paralelos.

SECRETARIA.

De acuerdo al cronograma de los cursos programados por la dirección academica, se realiza la inscripción de los estudiantes.

El proceso comienza cuando el estudiante solicita inscribirse a dicho curso, por lo cual brinda los datos requeridos, la información es llenada en un formulario por la secretaria. Una ves terminada la operación se hace la entrega de la boleta de inscripción y su debida factura.

PERSONAL DOCENTE.

Es el encargado de capacitar y enseñar a los estudiantes para que tengan un conocimiento mas amplio sobre la materia también aclarara las preguntas que tengan los estudiantes.

Actualmente el sistema del instituto funciona de forma manual, el alumno solicita los información de los cursos que se enseña o se imparte en la institución, después si desean pueden inscribirse al curso regular o aun curso programado. La secretaria solicita los datos requeridos del estudiante, la información es llenada en un formulario en forma manual una ves terminada entrega la boleta de inscripción con la fecha de inicio del curso y su respectiva factura.
Después todos los datos de los inscritos son transcritos a un archivo Excel, para así tener información manual y en la computadora.
Al terminar el curso se otorga un certificado a los estudiantes con la carga horaria y con la respectiva nota.

2.2 METODOLOGIA.

2.2.1 RUP (PROCESO UNIFICADO DE RATIONAL):

Es el proceso unificado de desarrollo, es un proceso para el desarrollo de un proyecto de un software que define claramente quien,como,cuando y que debe hacerse en el proyecto de desarrollo de software[JACOBSON 1999]

El Proceso Unificado de Rational (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

También se conoce por este nombre al software, también desarrollado por Rational, que incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo con las necesidades.

El proceso unificado esta basado en componentes. Utiliza el nuevo estándar de modelo visual, el lenguaje unificado de modelado (UML), se sostiene sobre tres ideas básicas casos de usos, arquitectura y desarrollo iterativo incremental .Para hacer que las tres ideas básicas funciones se necesita un proceso polifacético, que tenga en cuenta ciclos, fases, flujos de trabajo, gestión del riesgo, control de calidad y control de configuración .El proceso unificado ha establecido un marco de trabajo que integra todas estas facetas

[IBoochRumb, 1999].

2.2.1.1 HISTORIA

Los orígenes de RUP se remontan al modelo espiral original de Barry Boehm. Ken Hartman, uno de los contribuidores claves de RUP colaboró con Boehm en la investigación. En 1995 Rational Software compró una compañía sueca llamada Objectory AB, fundada por Ivar Jacobson, famoso por haber incorporado los casos de uso a los métodos de desarrollo orientados a objetos. El Rational Unified Process fue el resultado de una convergencia de Rational Approach y Objectory (el proceso de la empresa Objectory AB). El primer resultado de esta fusión fue el Rational Objectory Process, la primera versión de RUP, fue puesta en el mercado en 1998, siendo el arquitecto en jefe Philippe Kruchten.

El primer libro para describir el proceso fue titulado "The Unified Software Development Process (ISBN 0-201-57169-2)" El Proceso Unificado de Desarrollo de Software (ISBN 0-201-57169-2), y publicado en [1999 por Ivar Jacobson, Grady Booch y James Rumbaugh]

2.2.1.2 FASES DEL RUP:

RUP se divide en 4 fases, dentro de las cuales se realizan varias iteraciones según el proyecto y en las que se hace mayor o menos esfuerzo en las distintas actividades. En las iteraciones de cada fase se hacen diferentes esfuerzos en diferentes actividades:

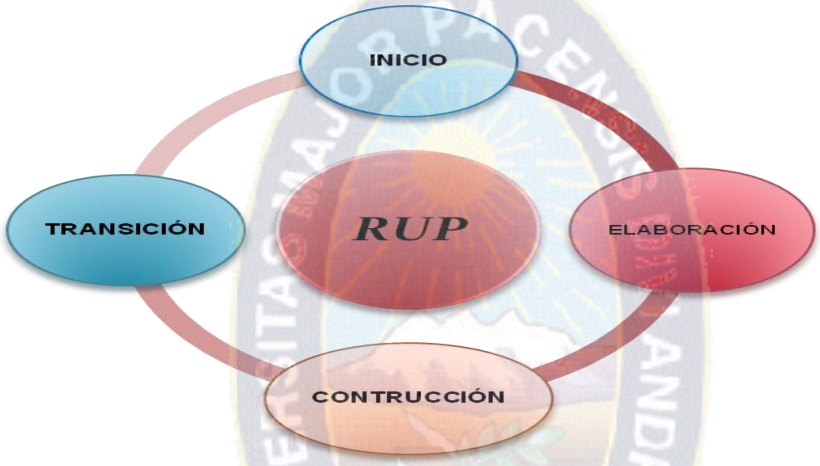


FIGURA 1 (FASES DE RUP**)**

FUENTE(METODOLOGIA RUP**)**

a) FASE DE INICIO

En esta fase se establece el caso del negocio que incluye el contexto del negocio, factores del éxito y pronostico financiero .Para complementar la caja del negocio de debe generar, un modelo básico del caso de uso, un plan del proyecto, el gravamen de riesgo inicial y la descripción del proyecto. Después de que se terminen estos, el proyecto se comprueba contra los criterios siguientes:

- Requisitos que entienden según lo evidenciado por la fidelidad de los casos primarios del uso

- Credibilidad de las estimaciones de costo/beneficio, prioridades, riesgos, y del proceso de desarrollo.

- Profundidad y anchura de cualquier prototipo arquitectónico que fuera desarrollado.

- Gastos efectivos contra gastos previstos.

b) FASE DE ELABORACIÓN:

La fase de elaboración comprende el análisis del dominio del problema y la arquitectura del proyecto consigue su forma básica. Esto implica vistas arquitectónicas del modelo de caso de uso, del modelo de análisis, del modelo del diseño, del modelo de implementación y modelo de despliegue. La vista del modelo e implementación incluye componentes para probar que la arquitectura es ejecutable. Durante esta fase del desarrollo, se realizan los casos de uso más críticos que se identificaron en la fase de comienzo.

La fase de elaboración resuelve los siguientes criterios:

- Se desarrollan un modelo de casos de uso en el cual ya se han identificado los casos de uso. El modelo de casos de uso debe estar en un 80% completo.

- Una descripción de la arquitectura del software en un proceso del desarrollo del software.

- Caso del negocio y lista de riesgo que están revisadas.

- Un plan de desarrollo para el proyecto total.

Si el proyecto no puede pasar esta fase, debe ser reajustado, mediante las iteraciones que sean necesarias. Ya que al dejar esta fase, los cambios realizados posteriormente son más difíciles, perjudiciales y tienen un riesgo elevado.

c) FASE DE CONSTRUCCIÓN:

En esta fase el foco principal es el desarrollo de componentes y de otras características del sistema que esta siendo diseñado. En proyectos que son bastante complejos , varias iteraciones de la construcción se pueden desarrollar en un esfuerzo de dividir dos casos de uso en segmentos manejables que producen prototipos demostrables.
Se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. El resultado final es un sistema ejecutable.

- Planificar que subsistemas deben ser implementados y en que orden deben ser integrados, formando el Plan de Integración.

- Cada implementador decide en que orden implementa los elementos del subsistema

- Si encuentra errores de diseño, los notifica. Se integra el sistema siguiendo el plan.

El flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida.

- Encontrar y documentar defectos en la calidad del software
- Generalmente asesora sobre la calidad del software percibida.
- Provee la validación de los supuestos realizados en el diseño y especificación de requisitos por medio de demostraciones concretas.
- Verificar las funciones del producto de software según lo diseñado.
- Verificar que los requisitos tengan su apropiada implementación.

d) FASE DE TRANSICIÓN:

Esta actividad tiene como objetivo, producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- Probar el producto en su entorno de ejecución final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.
- Formar a los usuarios.
- Migrar el software existente o convertir bases de datos.

Durante todo el proyecto se vigila el cumplimiento de los objetivos, gestión de riesgos y restricciones para desarrollar un producto .Que sea acorde a los requisitos de los clientes y los usuarios.

- Proveer un marco de trabajo para la gestión de proyectos de software intensivos.
- Proveer guías practicas realizar planeación, contratar personal, ejecutar y monitorear el proyecto.
- Proveer un marco de trabajo para gestionar riesgos.

El control de cambios permite mantener la integridad de todos los artefactos que se crean en le proceso, así como de mantener información del proceso evolutivo que han seguido.

La finalidad de esta actividad es dar aporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento, asi como definir la instancia concreta del proceso que va a seguir. En concreto las responsabilidades de este flujo de trabajo incluyen:

- Selección y adquisición de herramientas.
- Establecer y configurar las herramientas para que se ejecuten a la organización.
- Configuración del proceso.
- Mejora del proceso.
- Servicios técnicos.

2.2.1.3 ESTRUCTURA DEL PROCESO UNIFICADO DE DESARROLLO.

El proceso unificado de desarrollo de software esta compuesto por 4 fases, cada una de estas fases es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala.
El eje horizontal representa la parte dinámica del proceso, el tiempo, iteraciones y las metas, el eje vertical representa la parte estática del proceso donde se describe los flujos de trabajo, requisitos, análisis, diseño, implementación y prueba. Las curvas son una aproximación hasta donde se llevan a cabo los flujos de trabajo en cada fase, como se puede ver en la figura 2.5[Jacobson, 2000].

FIGURA 2 (estructura del proceso de desarrollo de software)

FUENTE(Jacobson, Booch, R, 2000)

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías mas importantes para alcanzar un grado de certificación en el desarrollo del software.

2.3 UML(LENGUAJE UNIFICADO DE MODELADO)

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

[Larman, 1999]

Los sistemas, sobre todo aquellos que por sus características pueden llegar a denominarse complejos, son difíciles de comprender; más aun por aquellos que no son profesionales de las Tecnologías de la Información. Así encontramos que el sistema gira entorno a la visión que se tenga de él y de cómo la tecnología puede llegar a mejorar las cosas. Así el éxito de los proyectos de desarrollo de software giran entorno al enlace que existe entre quien tiene la idea de la nueva aplicación (el usuario) y quien puede crearla (el desarrollador), pero debe existir una manera de que ambos puedan expresar y registrar sus ideas, para así propagarlas

al equipo de trabajo, es aquí donde un lenguaje común se hace imprescindible y necesario, con esta idea se ha desarrollado el lenguaje UML (Unified Languaje Model) o Lenguaje Unificado de Modelado, esta herramienta cumple con esta función, permitiendo capturar la idea de un sistema para comunicarla con aquellos que hacen posible su materialización. Cada diagrama tiene fines distintos dentro del proceso de desarrollo.

El UML fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson a mediados de los años noventa, el lenguaje compuesto por diversos elementos gráficos se combinan para formar diagramas, debido a que es un lenguaje, cuenta con reglas para combinar tales elementos.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. El modelo UML de un sistema es similar a un modelo a escala de un edificio junto con la interpretación del artista del edificio. Es importante destacar que un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

2.3.1 DIAGRAMAS DE CASOS DE USO:

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario.

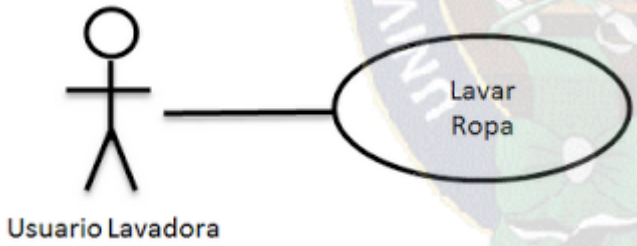


FIGURA 3 (Diagrama de Casos de Uso)
FUENTE("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 29)

2.3.2 DIAGRAMA DE SECUENCIA:

Muestra la forma como interactúan los objetos entre si. Por ejemplo, si se añade la ropa y el detergente a la lavadora y se activa su funcionamiento, la secuencia sería:

- El agua empezará a llenar el tambor mediante una manguera.
- El tambor permanecerá inactivo durante cinco minutos.

- La manguera dejará de abastecer agua.
- El tambor girará de un lado a otro durante quince minutos.
- El agua jabonosa saldrá por el drenaje.
- Comenzará nuevamente el abastecimiento de agua.
- El tambor continuará girando.
- El abastecimiento de agua se detendrá.
- El agua del enjuague saldrá por el drenaje.
- El tambor girará en una sola dirección y se incrementará su velocidad por cinco minutos.
- El tambor dejará de girar y el proceso de lavado habrá finalizado.

En un diagrama se secuencia se representaría:

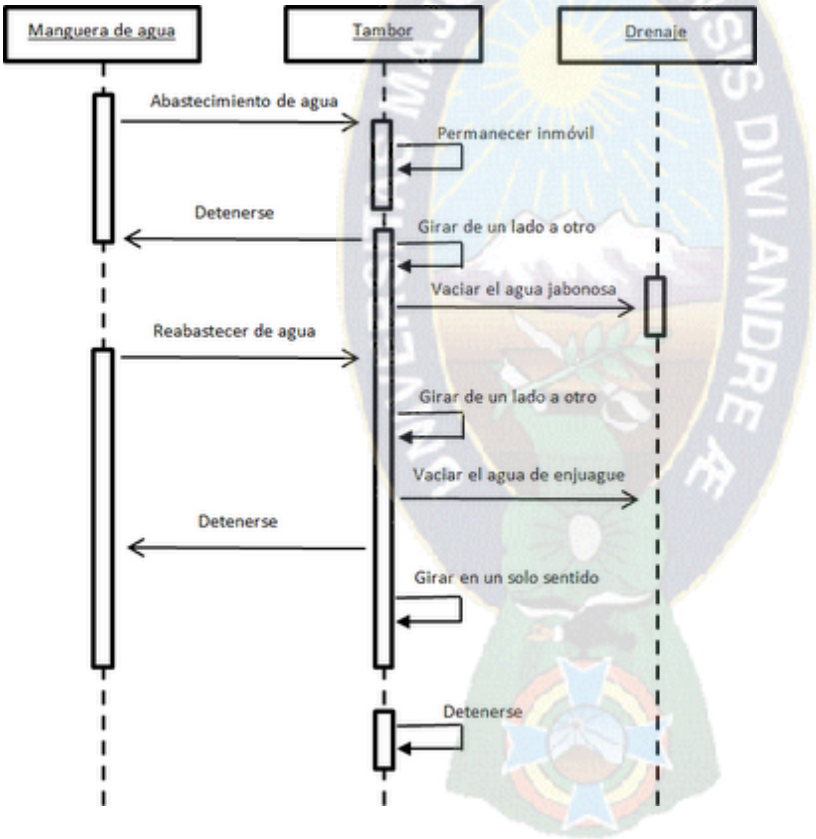


FIGURA 4 (Diagrama de Secuencia.)
FUENTE("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 31)

2.3.3 DIAGRAMA DE ESTADO:

En cualquier momento, un objeto se encuentra en un estado en particular, como por ejemplo un aeroplano, puede estar estacionado en un hangar, en maniobras de aterrizaje, despegue o sencillamente estar volando recto y nivelado.

FIGURA 5 (Diagrama de Estados)
FUENTE ("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 30)

2.3.4 DIAGRAMA DE ACTIVIDADES:

Las actividades que ocurren dentro de un caso de uso o dentro del comportamiento de un objeto se dan, normalmente, en secuencia, como en los once pasos de la sección anterior.

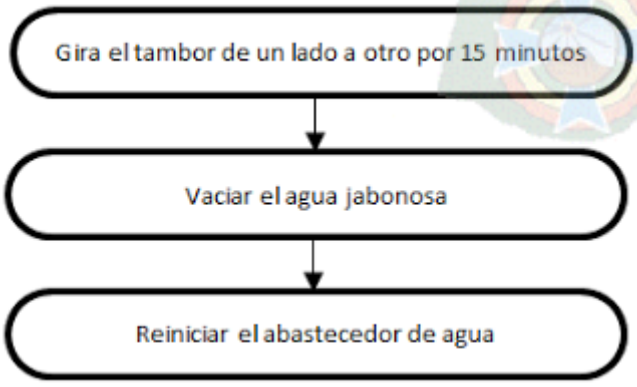


FIGURA 6 (Diagrama de Actividades)
FUENTE ("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 31)

2.3.5 DIAGRAMA DE COLABORACIÓN:

Muestra gráficamente la forma como los elementos de un sistema trabajan en conjunto.

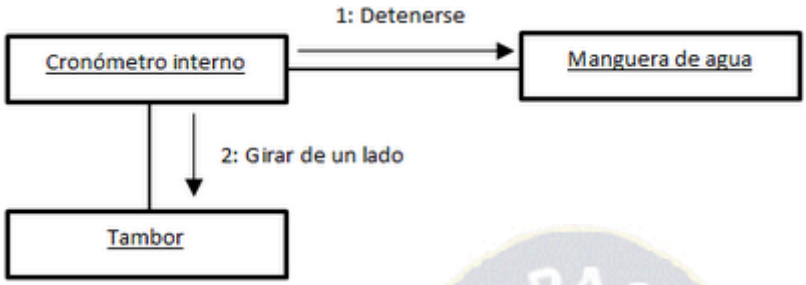


FIGURA 7 (Diagrama de Colaboración.)
FUENTE("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 32)

2.3.6 DIAGRAMA DE CLASES:

Una clase es una categoría o grupo de cosas que tienen atributos y acciones similares, como por ejemplo la clase lavadora, cuenta con los atributos marca, modelo, número de serie, capacidad. También posee ciertos comportamientos o métodos tal como: agregar ropa, agregar detergente y sacar ropa.

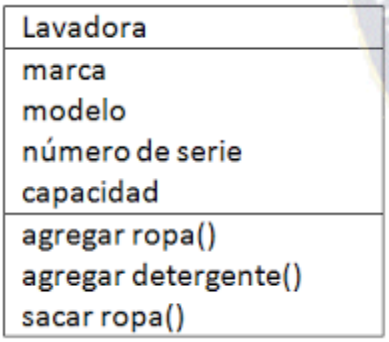


FIGURA 8 (Diagrama de Clases,)
FUENTE("Aprendiendo UML en 24 Horas". Schmuller, J. (2004). p. 28)

2.3.7 CONTROL DE CALIDAD Y PRUEBAS

Las razones por las que ocurren los errores en el software:

- Especificaciones: Al momento de redactar las especificaciones estas están incompletas, ambiguas, variables o simplemente no están realizadas.

- Diseño: El diseño de la solución esta incompleto o inadecuado o bien las especificaciones no se comprendieron correctamente.
- Codificación: el código es incorrecto porque se hizo rápidamente, el programador no conoce bien el lenguaje o no comprendió bien el diseño.

La calidad es un concepto difícil de definir ya que contiene múltiples facetas, en primer lugar según Pressman(2010), es una característica que se reconoce de inmediato aunque no pueda ser definida explícitamente, desde el punto de vista del usuario es un producto o servicio que satisface sus necesidades, para el fabricante es el cumplimiento de las especificaciones del producto fabricado o servicio prestado, la calidad también se encuentra relacionada con el precio que está dispuesto a pagar el comprador a cambio de un producto con tales características.

En el desarrollo de software, la calidad del diseño es el grado en el que éste cumple con las funciones y características especificadas en el modelo de requerimiento. La calidad de la conformidad se centra en el grado en el que la implementación se apega al diseño y en el que el sistema resultante cumple sus metas de requerimientos y desempeño.

La calidad del software, continua Pressman(2010), se presenta de la siguiente manera:

- Un proceso eficaz de software envuelve la calidad del proceso con el cual es llevado a cabo, envolviendo los aspectos de gestión y de buenas prácticas de ingeniería.
- En segundo lugar se relaciona con la producción de un producto útil que cumpla con las expectativas del usuario de forma confiable y libre de errores.
- Añade valor agregado al productor y el usuario del producto, beneficiando a la organización que lo utiliza, lo produce y a la comunidad de usuarios finales.
- Así el producto final redunda en una serie de beneficios, menor esfuerzo en mantenimiento, menor errores, menos costo de producción, mayor rentabilidad y disponibilidad de información.
- Señala Pressman(2010) los factores de calidad dictada por la norma ISO 9126, que sirven de guía de acción a la calidad:
- Funcionalidad: Es el grado en el que el software satisface las necesidades planteadas según su: adaptabilidad, exactitud, interoperatividad, cumplimiento y seguridad.
- Confiabilidad: Cantidad de tiempo que el software se encuentra disponible para su uso, según lo indican los atributos tales como: madurez, tolerancia a fallos y recuperación.
- Usabilidad: Grado en el que el software es fácil de usar, que sea entendible, aprendible y operable.
- Eficiencia: Grado en el que el software emplea óptimamente los recursos del sistema.
- Facilidad de recibir mantenimiento: Facilidad con la que pueden efectuarse reparaciones al software, que sea analizable, cambiable, estable y susceptible de someterse a pruebas.
- Portabilidad: Facilidad con la que el software puede llevarse de un ambiente a otro, cumpliendo con atributos de adaptabilidad, instalable, conformidad y sustituible.

2.4 TECNOLOGIAS DE SOFTWARE.

2.4.1 LENGUAJES DE PROGRAMACIÓN.

Un lenguaje de programación" es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

Por otro lado, el término "lenguaje natural" define un medio de comunicación compartido por un grupo de personas (por ejemplo: inglés o francés).

Los lenguajes que los equipos usan para comunicarse entre ellos no tienen nada que ver con los lenguajes de programación; se los conoce como protocolos de comunicación. Se trata de dos conceptos totalmente diferentes. Un lenguaje de programación es muy estricto:

A CADA instrucción le corresponde UNA acción de procesador.

El lenguaje utilizado por el procesador se denomina lenguaje máquina. Se trata de datos tal como llegan al procesador, que consisten en una serie de 0 y 1 (datos binarios).

El lenguaje máquina, por lo tanto, no es comprensible para los seres humanos, razón por la cual se han desarrollado lenguajes intermediarios comprensibles para el hombre. El código escrito en este tipo de lenguaje se transforma en código máquina para que el procesador pueda procesarlo.

El ensamblador fue el primer lenguaje de programación utilizado. Es muy similar al lenguaje máquina, pero los desarrolladores pueden comprenderlo. No obstante, este lenguaje se parece tanto al lenguaje máquina que depende estrictamente del tipo de procesador utilizado (cada tipo de procesador puede tener su propio lenguaje máquina). Así, un programa desarrollado para un equipo no puede ser portado a otro tipo de equipo. El término "portabilidad" describe la capacidad de usar un programa de software en diferentes tipos de equipos. Para poder utilizar un programa de software escrito en un código ensamblador en otro tipo de equipo, ¡a veces será necesario volver a escribir todo el programa!

Por lo tanto, un lenguaje de programación tiene varias ventajas:

- es mucho más fácil de comprender que un lenguaje máquina:
- permite mayor portabilidad, es decir que puede adaptarse fácilmente para ejecutarse

2.4.2 JAVA:

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis

deriva mucho de C y C++, pero tiene menos facilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

La compañía Sun desarrolló la implementación de referencia original para los compiladores de Java, máquinas virtuales, y librerías de clases en 1991 y las publicó por primera vez en 1995. A partir de mayo de 2007, en cumplimiento con las especificaciones del Proceso de la Comunidad Java, Sun volvió a licenciar la mayoría de sus tecnologías de Java bajo la Licencia Pública General de GNU. Otros también han desarrollado implementaciones alternas a estas tecnologías de Sun, tales como el Compilador de Java de GNU y el GNU Classpath.

El lenguaje para la programación en Java, es un lenguaje orientado a objeto, de una plataforma independiente.

El lenguaje para la programación en Java, fue desarrollado por la compañia Sun Microsystems, con la idea original de usarlo para la creacion de paginas WEB.

Esta programación Java tiene muchas similitudes con el lenguaje C y C++, asi que si se tiene conocimiento de este lenguaje, el aprendizaje de la programación Java sera de facil comprension por un programador que haya realizado programas en estos lenguajes.

Con la programación en Java, se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales, que se ejecutan dentro de un navegador al ser cargada una pagina HTML en un servidor WEB, Por lo general los applets son programas pequeños y de propositos especificos.

Otra de las utilidades de la programación en Java es el desarrollo de aplicaciones, que son programas que se ejecutan en forma independiente, es decir con la programación Java, se pueden realizar aplicaciones como un procesador de palabras, una hoja que sirva para calculos, una aplicacion grafica, etc. en resumen cualquier tipo de aplicacion se puede realizar con ella. Java permite la modularidad por lo que se pueden hacer rutinas individuales que sean usadas por más de una aplicacion, por ejemplo tenemos una rutina de impresion que puede servir para el procesador de palabras, como para la hoja de calculo.

La programación en Java, permite el desarrollo de aplicaciones bajo el esquema de Cliente Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultaneamente, y de esta forma logra distribuir el trabajo a realizar.

2.4.3 MICROSOFT .NET:

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Oracle Corporation y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

2.4.3.1 CARACTERÍSTICAS

Es el encargado de proveer lo que se llama código administrado, es decir, un entorno que provee servicios automáticos al código que se ejecuta. Los servicios son variados:

- Cargador de clases: permite cargar en memoria las clases.
- Compilador MSIL a nativo: transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- Administrador de código: coordina toda la operación de los distintos subsistemas del Common Language Runtime.
- Recolector de basura: elimina de memoria objetos no utilizados automáticamente.
- Motor de seguridad: administra la seguridad del código que se ejecuta.
- Motor de depuración: permite hacer un seguimiento de la ejecución del código aún cuando se utilicen lenguajes distintos.
- Verificador de tipos: controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- Administrador de excepciones: maneja los errores que se producen durante la ejecución del código.
- Soporte de multiproceso (hilos): permite desarrollar aplicaciones que ejecuten código en forma paralela.
- Empaquetador de COM: coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
- Biblioteca de Clases Base que incluye soporte para muchas funcionalidades comunes en las aplicaciones.

2.4.4 PHP:

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP.[2] Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término PHP

2.4.4.1 VISIÓN GENERAL

PHP puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores. El enorme número de sitios en PHP ha visto reducida su cantidad a favor de otros nuevos lenguajes no tan poderosos desde agosto de 2005. El sitio web de Wikipedia está desarrollado en PHP.[5] Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de sitios webs, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando alguna extensión como puede ser PHP-Qt, PHP-GTK,[6] WxPHP, WinBinder, Roadsend PHP, Phalanger, Phc o HiP Hop VM. También puede ser usado desde la línea de comandos, de la misma manera como Perl o Python pueden hacerlo; a esta versión de PHP se la llama PHP-CLI (Command Line Interface).[7]

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente.

Mediante extensiones es también posible la generación de archivos PDF,[8] Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tanto SQL como NoSQL tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird, SQLite o MongoDB.[9]

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (y de ese tipo, como Linux o Mac OS X) y Microsoft Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C# y Visual Basic .NET como lenguajes), a ColdFusion de la empresa Adobe, a JSP/Java, CGI/Perl y a Node.js/Javascript. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un entorno de desarrollo integrado comercial llamado Zend Studio. CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno de desarrollo integrado para PHP, denominado 'Delphi for PHP. También existen al menos un par de módulos para Eclipse, uno de los entornos más populares.[10]

2.5.1 SISTEMAS DE GESTIÓN DE BASES DE DATOS

2.5.1.1 INTRODUCCIÓN:

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

2.5.1.2 LAS BASES DE DATOS.

En el entorno informático, la gestión de bases de datos ha evolucionado desde ser una aplicación más disponible para los computadores, a ocupar un lugar fundamental en los sistemas de información. En la actualidad, un sistema de información será más valioso cuanto de mayor calidad sea la base de datos que lo soporta, la cual resulta a su vez un componente fundamental del mismo, de tal forma que puede llegarse a afirmar que es imposible la existencia de un sistema de información sin una base de datos, que cumple la función de "memoria", en todas sus acepciones posibles, del sistema.

Las bases de datos almacenan, como su nombre dice, datos. Estos datos son representaciones de sucesos y objetos, a diferente nivel, existentes en el mundo real: en su conjunto, representan algún tipo de entidad existente. En el mundo real se tiene percepción sobre las entidades u objetos y sobre los atributos de esos objetos; en el mundo de los datos, hay registros de eventos y datos de eventos. Además, en ambos escenarios se puede incluso

distinguir una tercera faceta: aquella que comprende las definiciones de las entidades externas, o bien las definiciones de los registros y de los datos.

La transferencia entre las entidades del mundo real, y sus características, y los registros contenidos en una base de datos, correspondientes a esas entidades, se alcanza tras un proceso lógico de abstracción, conjunto de tareas que suelen englobarse bajo el título de diseño de bases de datos. Sin embargo, es necesario definir, en primer lugar, qué es una base de datos, independientemente de su diseño y/o su orientación. Entre las numerosas definiciones que pueden encontrarse en la bibliografía, pueden escogerse, por su exhaustividad, las siguientes:

"Colección de datos correspondientes a las diferentes perspectivas de un sistema de información (de una empresa o institución), existentes en algún soporte de tipo físico (normalmente de acceso directo), agrupados en una organización integrada y centralizada en la que figuran no sólo los datos en sí, sino también las relaciones existentes entre ellos, y de forma que se minimiza la redundancia y se maximiza la independencia de los datos de las aplicaciones que los requieren." (GUILERA, 1993: 377)

"Una base de datos es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción han de ser únicas estando almacenadas junto a los mismos. Por último, los tratamientos que sufran estos datos tendrán que conservar la integridad y seguridad de éstos." (MOTA, CELMA y CASAMAYOR, 1994: 9)

La segunda definición añade los objetivos que debe cumplir un sistema de gestión de bases de datos, sobre los cuales se tratará más adelante. Por ahora, basta considerar que deben cumplir los objetivos de independencia de los datos (las aplicaciones no deben verse afectadas por cambios en la estructura de los datos), integridad de los datos (los datos deben cumplir ciertas restricciones que aseguren la correcta introducción, modificación y borrado de los mismos) y seguridad (establecer diferentes niveles de acceso a los datos a diferentes tipos de usuarios).

La entidad existente en el mundo real es objeto de un doble tratamiento, desde el momento en que convierte en objeto de la base de datos. El tratamiento de sus datos se va a realizar en un nivel lógico, por una parte, y en un nivel físico, por otra. En el primero de ellos, el lógico, se va a trabajar en los aspectos referidos a la identificación de las características de la entidad, su descripción y organización, mientras que en el segundo todo lo anterior se va a plasmar en la organización, acceso y almacenamiento de los datos en un soporte físico. Esta división entre un nivel lógico y otro físico se va reflejar en todos los métodos y conceptos subsiguientes.

2.5.1.3 EL MODELO DE ARQUITECTURA DE BASES DE DATOS.

Hasta fecha relativamente cercana, las bases de datos eran el resultado de una compleja programación y de complicados mecanismos de almacenamiento. Con la popularización de la microinformática, la aparición de aplicaciones específicas también trajo con ella la disponibilidad de herramientas de gestión de datos, que acabaron desembocando en los denominados sistemas de gestión de bases de datos, identificados por sus siglas SGBD (DBMS en inglés, siglas de DataBase Management Systems). De esta manera, la gestión de base de datos pudo liberarse de los grandes ordenadores centrales, pudiendo distribuirse según los intereses de los usuarios, y dotando de autonomía en la gestión de información a muchas entidades. Los SGBD permitieron a todo tipo de usuarios crear y mantener sus bases de datos, dotándolos de una herramienta que era capaz de transformar el nivel lógico que éstos diseñaban en un conjunto de datos, representaciones y relaciones, traduciéndolo al nivel físico correspondiente. Para que fuese posible, y para asegurar a los usuarios cierta seguridad en el intercambio de datos entre diferentes sistemas, y en el diseño de ficheros y bases de datos, fue necesario normalizar los esquemas que guiaban la creación de las bases de datos.

2.5.1.4 NIVELES DE LA ARQUITECTURA DE BASES DE DATOS

Las bases de datos respetan la arquitectura de tres niveles definida, para cualquier tipo de base de datos, por el grupo ANSI/SPARC. En esta arquitectura la base de datos se divide en los niveles externo, conceptual e interno (KORTH y SILBERSCHATZ, 1994:5; MIGUEL y PIATTINI, 1993: 83-107; MOTA, CELMA y CASAMAYOR, 1994: 11-12):

2.5.1.5 NIVEL INTERNO: es el nivel más bajo de abstracción, y define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso.

2.5.1.6 NIVEL CONCEPTUAL: es el nivel medio de abstracción. Se trata de la representación de los datos realizada por la organización, que recoge las vistas parciales de los requerimientos de los diferentes usuarios y las aplicaciones posibles. Se configura como visión organizativa total, e incluye la definición de datos y las relaciones entre ellos.

2.5.1.7 NIVEL EXTERNO: es el nivel de mayor abstracción. A este nivel corresponden las diferentes vistas parciales que tienen de la base de datos los diferentes usuarios. En cierto modo, es la parte del modelo conceptual a la que tienen acceso.

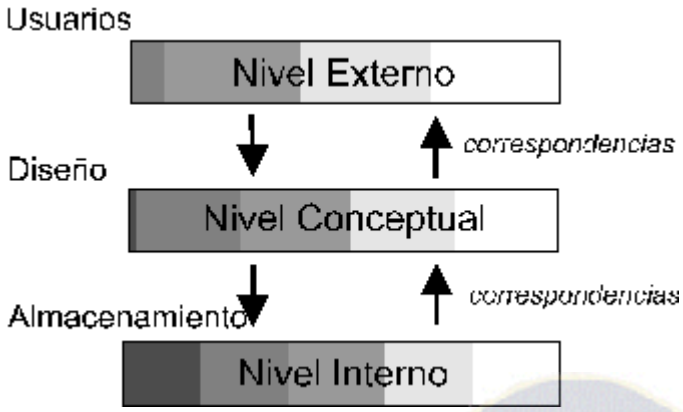


FIGURA 9 (niveles de la arquitectura de bases de datos.)

FUENTE (el modelo de arquitectura de bases de datos)

En ocasiones puede encontrarse el nivel conceptual divido en dos niveles, conceptual y lógico. El primero de ellos corresponde a la visión del sistema global desde un punto de vista organizativo independiente, no informático. El segundo correspondería a la visión de la base de datos expresada en términos del sistema que se va a implantar con medios informáticos.

El modelo de arquitectura propuesto permite establecer el principio de independencia de los datos. Esta independencia puede ser lógica y física. Por independencia lógica se entiende que los cambios en el esquema lógico no deben afectar a los esquemas externos que no utilicen los datos modificados. Por independencia física se entiende que el esquema lógico no se vea afectado por cambios realizados en el esquema interno, correspondientes a modos de acceso, etc.

http://www.monografias.com/trabajos56/sistemas-bases-de-datos/sistemas-bases-de-datos.shtml

2.6 CALIDAD DE SOFTWARE:

La calidad del software La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

Los requisitos del software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.

Los estándares o metodologías definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la ingeniería del software. Si no se sigue ninguna metodología siempre habrá falta de calidad.

Existen algunos requisitos implícitos o expectativas que a menudo no se mencionan, o se mencionan de forma incompleta (por ejemplo el deseo de un buen mantenimiento) que también pueden implicar una falta de calidad.

Las métricas se utilizan para supervisar y controlar un proyecto de software . El estándar ISO 9126, ha sido desarrollado en un intento de identificar los atributos de calidad para el software. El estándar identifica los siguiente atributos de calidad [PRESSMAN, 2002]

2.6.1 ISO/IEC 9126

'ISO 9126 es un estándar internacional para la evaluación de la calidad del software. Está reemplazado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos. Este estándar es el más usado... El estándar está dividido en cuatro partes las cuales dirigen, realidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido.El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas de la siguiente manera:

La subcaracterística Conformidad no está listada arriba ya que se aplica a todas las características. Ejemplos son conformidad a la legislación referente a usabilidad y fiabilidad.

Cada subcaracterística (como adaptabilidad) está dividida en atributos. Un atributo es una entidad la cual puede ser verificada o medida en el producto software. Los atributos no están definidos en el estándar, ya que varían entre diferentes productos software.

Un producto software está definido en un sentido amplio como: los ejecutables, código fuente, descripciones de arquitectura, y así. Como resultado, la noción de usuario se amplía tanto a operadores como a programadores, los cuales son usuarios de componentes como son bibliotecas software.

El estándar provee un entorno para que las organizaciones definan un modelo de calidad para el producto software. Haciendo esto así, sin embargo, se lleva a cada organización la tarea de especificar precisamente su propio modelo. Esto podría ser hecho, por ejemplo, especificando los objetivos para las métricas de calidad las cuales evalúan el grado de presencia de los atributos de calidad.

Métricas internas son aquellas que no dependen de la ejecución del software (medidas estáticas).

Métricas externas son aquellas aplicables al software en ejecución.

La calidad en las métricas de uso están sólo disponibles cuando el producto final es usado en condiciones reales.

Idealmente, la calidad interna no necesariamente implica calidad externa y esta a su vez la calidad en el uso.

Este estándar proviene desde el modelo establecido en 1977 por McCall y sus colegas, los cuales propusieron un modelo para especificar la calidad del software. El modelo de calidad McCall está organizado sobre tres tipos de Características de Calidad:

- Factores (especificar): Describen la visión externa del software, como es visto por los usuarios.
- Criterios (construir): Describen la visión interna del software, como es visto por el desarrollador.
- Métricas (controlar): Se definen y se usan para proveer una escala y método para la medida.

ISO 9126 distingue entre fallo y no conformidad. Un fallo es el incumplimiento de los requisitos previos, mientras que la no conformidad es el incumplimiento de los requisitos especificados. Una distinción similar es la que se establece entre validación y verificación.

2.6.2 CONFIABILIDAD

Es la certeza de que un componente, equipo o producto software realiza su función prevista sin incidentes por un periodo de tiempo. Para determinar la confiabilidad de cualquier sistema es necesario definir la función del sistema al igual que las situaciones o condiciones que hacen perder la funcionalidad sobre el sistema.

Se establece, hasta donde se puede esperar que un programa lleve a cabo su función con la exactitud requerida. En términos estadísticos como la probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y durante un tiempo específico. Este factor viene dado por cantidad de tiempo que el software esta disponible para su uso, relacionado por los siguiente atributos: madurez, tolerancia a fallos y facilidad de recuperación [PRESSMAN,2005].

Es posible expresar la confiabilidad de acuerdo a la siguiente ecuación.

$F = a*e^{-bx(t)}$

Donde:

t = el numero de fallas en el instante

a,b = constantes

Este modelo indica el número de horas restantes para garantizar la confiabilidad. El calculo de horas de prueba necesarias para cero fallas es.

$$\frac{\text{Ln}\ \frac{fallas}{0.5+fallas}\ * horas_{hasta_{ultima_{falla}}}}{Ln(\frac{0.5+fallas}{fallas_{probadas}}+fallas)}$$

Donde.

Fallas= Numero de fallas proyectando

Fallas_probadas=Numero de fallas observado

Horas_hasta_ultima_falla=Numero total de horas de ejecución de pruebas hasta la ultima falla.

2.6.3 USABILIDAD.

Es el esfuerzo necesario para aprender a operar con el sistema, prepara los datos de entrada e interpretar los de salida (resultados) de un programa, que es el grado que el software es de fácil de uso y viene reflejado por los siguientes atributos:

- Facilidad de comprensión, facilidad de aprendizaje y operatividad [PRESSMAN, 2005]. Este factor viene dado por la medida de las sub-caracteristicas de la capacidad de ser entendido y la operatividad.

2.6.4 EFICIENCIA

Es el rendimiento del funcionamiento de un programa, es decir, el grado que el software hace optimo el uso de los recursos del sistema. Esta determinado por los siguientes sub-atributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba[PRESSMAN,2005].

2.6.5 MANTENIBILIDAD

Es le esfuerzo necesario para localizar y arreglar un error e un programa la facilidad con que una modificación puede ser realizada. Esta determinada por los siguiente sub-atributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba [PRESSMAN,2002].

Para medir la matenibilidad del sistema se utilizan los índices de madurez del software (IMS) según el IEEE982, 1-1988, este nos proporciona una indicación de la estabilidad basado en los cambios presentados en cada versión durante el desarrollo del sistema.

IMS=(MT-(Fc+Fa+Fe))/MT

Dónde:

MT= Numero de módulos en la versión actual

Fc= Numero de módulos en la versión actual que se han cambiado

Fa=Numero de módulos en la versión actual que se han añadido

Fe=Numero de módulos en la versión actual que se han eliminado

A medida que el sistema se aproxima a 1 el producto se pone más estable según la siguiente relación:

75%<=IMS<=100% → Optima

50%<=IMS<=75% → Buena

25%<=IMS<=50% → Suficiente

0%<=IMS<=25% → Deficiente

2.6.6 PORTABILIDAD

Es el esfuerzo necesario para transferir el programa de un entorno hardware – software a otro entorno diferente, es decir, la facilidad con el que el software puede ser llevado de un entorno a otro, se mide probando el sistema en diferentes sistemas operativos. Esta determinado por los siguientes sub-atributos: facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio[PRESSMAN,2005].

La portabilidad viene dad por la medida de la subcaracteristica de facilidad de instalación, teniendo en cuenta la siguiente realcion:

$S=A / B$

Donde

A= numero de casos de éxito de la operación de instalación por parte del usuario

B=Numero total de operaciones de instalación que realizo el usuario.

Luego de obtener el resultado se hace una verificación con los siguientes valores:

75%<=IMS<=100% → Optima

50%<=IMS<=75% → Buena

25%<=IMS<=50% → Suficiente

0%<=IMS<=25% → Deficiente

2.7 ESTUDIO DE COSTOS Y BENEFICIOS:

2.7.1 COCOMO

El **Modelo Constructivo de Costos** (o **COCOMO**, por su acrónimo del inglés **CO**nstructive **CO**st **MO**del) es un modelo matemático de base empírica utilizado para estimación de costos[1] de software. Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

Este modelo fue desarrollado por Barry W. Boehm a finales de los años 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "Software Engineering Economics" (Prentice-Hall, 1981).

2.7.2 CARACTERÍSTICAS GENERALES

Pertenece a la categoría de modelos de subestimaciones basados en estimaciones matemáticas. Está orientado a la magnitud del producto final, midiendo el "tamaño" del proyecto, en líneas de código principalmente.

2.7.3 INCONVENIENTES

- Los resultados no son proporcionales a las tareas de gestión ya que no tiene en cuenta los recursos necesarios para realizarlas.
- Se puede desviar de la realidad si se indica mal el porcentaje de líneas de comentarios en el código fuente.
- Es un tanto subjetivo, puesto que está basado en estimaciones y parámetros que pueden ser "vistos" de distinta manera por distintos analistas que usen el método.
- Se miden los costes del producto, de acuerdo a su tamaño y otras características, pero no la productividad.
- La medición por líneas de código no es válida para orientación a objetos.

- Utilizar este modelo puede resultar un poco complicado, en comparación con otros métodos (que también sólo estiman).

2.7.4 MODELOS DE ESTIMACIÓN

La ecuaciones que se utilizan en los tres modelos son:[2]

- $E = a(Kl)^b * m(X)$, en persona-mes
- $Tdev = c(E)^d$, en meses
- $P = E/Tdev$, en personas

donde:

- E es el esfuerzo requerido por el proyecto, en persona-mes
- Tdev es el tiempo requerido por el proyecto, en meses
- P es el número de personas requerido por el proyecto
- a, b, c y d son constantes con valores definidos en una tabla, según cada submodelo
- Kl es la cantidad de líneas de código, en miles.
- m(X) Es un multiplicador que depende de 15 atributos.

A la vez, cada submodelo también se divide en **modos** que representan el tipo de proyecto, y puede ser:

- **modo orgánico**: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio).

- **modo semilibre** o **semiencajado**: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.

- **modo rígido** o **empotrado**: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

2.7.4.1 MODELO BÁSICO

Artículo principal: COCOMO Básico

Se utiliza para obtener una primera aproximación rápida del esfuerzo,[2] y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costes:

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Estos valores son para las fórmulas:

- Personas necesarias por mes para llevar adelante el proyecto (**MM**) = a*(Kl^b)
- Tiempo de desarrollo del proyecto (**TDEV**) = c*(MM^d)
- Personas necesarias para realizar el proyecto (**CosteH**) = MM/TDEV
- Costo total del proyecto (**CosteM**) = CosteH * Salario medio entre los programadores y analistas.

Se puede observar que a medida que aumenta la complejidad del proyecto (modo), las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que utilizar con mucho cuidado el modelo básico puesto que se obvian muchas características del entorno

2.7.4.2 MODELO INTERMEDIO

Este añade al modelo básico quince modificadores opcionales para tener en cuenta en el entorno de trabajo, incrementando así la precisión de la estimación.

Para este ajuste, al resultado de la fórmula general se lo multiplica por el coeficiente surgido de aplicar los atributos que se decidan utilizar.

Los valores de las constantes a reemplazar en la fórmula son:

MODO	a	b
Orgánico	3.20	1.05

Semilibre	3.00	1.12
Rígido	2.80	1.20

Se puede observar que los exponentes son los mismos que los del modelo básico, confirmando el papel que representa el tamaño; mientras que los coeficientes de los modos orgánico y rígido han cambiado, para mantener el equilibrio alrededor del semilibre con respecto al efecto multiplicador de los atributos de coste.

2.7.4.3 ATRIBUTOS

Cada atributo se cuantifica para un entorno de proyecto. La escala es **muy bajo** - **bajo** - **nominal** - **alto** - **muy alto** - **extremadamente alto**. Dependiendo de la calificación de cada atributo, se asigna un valor para usar de multiplicador en la fórmula (por ejemplo, si para un proyecto el atributo DATA es calificado como muy alto, el resultado de la fórmula debe ser multiplicado por 1000).

El significado de los atributos es el siguiente, según su tipo:

- De software
  - **RELY**: garantía de funcionamiento requerida al software. Indica las posibles consecuencias para el usuario en el caso que existan defectos en el producto. Va desde la sola inconveniencia de corregir un fallo (muy bajo) hasta la posible pérdida de vidas humanas (extremadamente alto, software de alta criticidad).
  - **DATA**: tamaño de la base de datos en relación con el tamaño del programa. El valor del modificador se define por la relación: D/K, donde D corresponde al tamaño de la base de datos en bytes y K es el tamaño del programa en cantidad de líneas de código.
  - **CPLX**: representa la complejidad del producto.

- De hardware
  - **TIME**: limitaciones en el porcentaje del uso de la CPU.
  - **STOR**: limitaciones en el porcentaje del uso de la memoria.
  - **VIRT**: volatilidad de la máquina virtual.
  - **TURN**: tiempo de respuesta requerido.

- De personal
  - **ACAP**: calificación de los analistas.
  - **AEXP**: experiencia del personal en aplicaciones similares.
  - **PCAP**: calificación de los programadores.
  - **VEXP**: experiencia del personal en la máquina virtual.

- **LEXP**: experiencia en el lenguaje de programación a usar.

- De proyecto
  - **MODP**: uso de prácticas modernas de programación.
  - **TOOL**: uso de herramientas de desarrollo de software.
  - **SCED**: limitaciones en el cumplimiento de la planificación.

El valor de cada atributo, de acuerdo a su calificación, se muestra en la siguiente tabla:

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de Base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		

Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	
Restricciones de tiempo de desarrollo	1,22	1,08	1,00	1,04	1,10	

2.7.4.4 MODELO DETALLADO

Presenta principalmente dos mejoras respecto al anterior:[2]

- Los factores correspondientes a los atributos son sensibles o dependientes de la fase sobre la que se realizan las estimaciones. Aspectos tales como la experiencia en la aplicación, utilización de herramientas de software, etc., tienen mayor influencia en unas fases que en otras, y además van variando de una etapa a otra.

- Establece una jerarquía de tres niveles de productos, de forma que los aspectos que representan gran variación a bajo nivel, se consideran a nivel módulo, los que representan pocas variaciones, a nivel de subsistema; y los restantes son considerados a nivel sistema.

2.7.5 VAN:

Valor actual neto

El **valor actual neto**, también conocido como valor actualizado neto o valor presente neto (en inglés net present value), cuyo acrónimo es VAN (en inglés, NPV), es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión. La metodología consiste en descontar al momento actual (es decir, actualizar mediante una tasa) todos los flujos de caja (en inglés cash-flow) futuros den determinar la equivalencia en el tiempo 0 de los flujos de efectivo futuros que genera un proyecto y comparar esta equivalencia con el desembolso inicial. Dicha tasa de actualización (k) o de descuento (d) es el resultado del producto entre el coste medio ponderado de capital (CMPC) y la tasa de inflación del periodo. Cuando dicha equivalencia es mayor que el desembolso inicial, entonces, es recomendable que el proyecto sea aceptado.

En las transacciones internacionales es necesario aplicar una tasa de inflación particular, tanto, para las entradas (cobros), como, para las de salidas de flujos (pagos). La condición que maximiza el margen de los flujos es que la economía exportadora posea un IPC inferior a la importadora, y viceversa.

La fórmula que nos permite calcular el Valor Actual Neto es:

$$\mathrm{VAN} = \sum_{t=1}^{n} \frac{V_t}{(1+k)^t} - I_0$$

V_t representa los flujos de caja en cada periodo t.

I_0 es el valor del desembolso inicial de la inversión.

n es el número de períodos considerado.

k, **d** o **TIR** es el tipo de interés.

Si el proyecto no tiene riesgo, se tomará como referencia el tipo de la renta fija, de tal manera que con el VAN se estimará si la inversión es mejor que invertir en algo seguro, sin riesgo específico. En otros casos, se utilizará el coste de oportunidad.

Cuando el VAN toma un valor igual a 0, k pasa a llamarse TIR (tasa interna de retorno). La TIR es la rentabilidad que nos está proporcionando el proyecto.

2.7.5.1 INTERPRETACIÓN

Valor	Significado	Decisión a tomar
VAN > 0	La inversión produciría ganancias por encima de la rentabilidad exigida (r)	El proyecto puede aceptarse
VAN < 0	La inversión produciría pérdidas por debajo de la rentabilidad exigida (r)	El proyecto debería rechazarse
VAN = 0	La inversión no produciría ni ganancias ni pérdidas	Dado que el proyecto no agrega valor monetario por encima de la rentabilidad exigida (r), la decisión debería basarse en otros criterios, como la obtención de un mejor posicionamiento en el mercado u otros factores.

El **valor actual neto** es muy importante para la valoración de inversiones en activos fijos, a pesar de sus limitaciones en considerar circunstancias imprevistas o excepcionales de mercado. Si su valor es mayor a cero, el proyecto es rentable, considerándose el valor mínimo de rendimiento para la inversión.

Una empresa suele comparar diferentes alternativas para comprobar si un proyecto le conviene o no. Normalmente la alternativa con el VAN más alto suele ser la mejor para la entidad; pero no siempre tiene que ser así. Hay ocasiones en las que una empresa elige un proyecto con un VAN más bajo debido a diversas razones como podrían ser la imagen que le aportará a la empresa, por motivos estratégicos u otros motivos que en ese momento interesen a dicha entidad.

Puede considerarse también la interpretación del VAN, en función de la creación de valor para la empresa:

- Si el VAN de un proyecto es positivo, el proyecto crea valor.

- Si el VAN de un proyecto es negativo, el proyecto destruye valor.

- Si el VAN de un proyecto es cero, el proyecto no crea ni destruye valor.

2.7.5.2 RENTAS FIJAS

Cuando los flujos de caja son de un monto fijo (rentas fijas), por ejemplo los bonos, se puede utilizar la siguiente fórmula:

$$\mathrm{VAN} = -I + \frac{R[1-(1+i)^{-n}]}{i}$$

Rrepresenta el flujo de caja constante.

$i$representa el coste de oportunidad o rentabilidad mínima que se está exigiendo al proyecto.

$n$es el número de periodos.

Ies la Inversión inicial necesaria para llevar a cabo el proyecto.

2.7.5.3 RENTAS CRECIENTES

En algunos casos, en lugar de ser fijas, las rentas pueden incrementarse con una tasa de crecimiento "g", siendo siempre g<i. La fórmula utilizada entonces para hallar el VAN es la siguiente:

$$\text{VAN} = -I + \frac{R[(1-(1+g)^{n} * (1+i)^{-n})]}{(i-g)}$$

Rrepresenta el flujo de caja del primer período.

$i$representa el coste de oportunidad o rentabilidad mínima que se está exigiendo al proyecto.

$g$representa el índice de incremento en el valor de la renta de cada período.

$n$es el número de periodos.

Ies la Inversión inicial necesaria para llevar a cabo el proyecto.

Si no se conociera el número de periodos a proyectarse (a perpetuidad), la fórmula variaría de esta manera:

$$\text{VAN} = -I + \frac{R}{(i-g)}$$«»

2.5.7.4 PROCEDIMIENTOS DEL VALOR ACTUAL NETO

Como menciona el autor Coss Bu, existen dos tipos de valor actual neto:

- Valor presente de inversión total. Puesto que el objetivo en la selección de estas alternativas es escoger aquella que maximice valor presente, las normas de utilización en este criterio son muy simples. Todo lo que se requiere hacer es determinar el valor presente de los flujos de efectivo que genera cada alternativa y entonces seleccionar aquella que tenga el valor presente máximo. El valor presente de la alternativa seleccionada deberá ser mayor que cero ya que de este manera el rendimiento que se obtiene es mayor que el interés mínimo atractivo. Sin embargo es posible que en ciertos casos cuando se analizan alternativas mutuamente exclusivas, todas tengan valores presentes negativos. En tales casos, la decisión a tomar es “no hacer nada”, es decir, se deberán rechazar a todas las alternativas disponibles. Por otra parte, si de las alternativas que se tienen solamente se conocen sus costos, entonces la regla de decisión será minimizar el valor presente de los costos.

- Valor presente del incremento en la inversión. Cuando se analizan alternativas mutuamente exclusivas, son las diferencias entre ellas lo que sería más relevante al

tomador de decisiones. El valor presente del incremento en la inversión precisamente determina si se justifican esos incrementos de inversión que demandan las alternativas de mayor inversión.

Cuando se comparan dos alternativas mutuamente exclusivas mediante este enfoque, se determinan los flujos de efectivo netos de la diferencia de los flujos de efectivo de las dos alternativas analizadas. Enseguida se determina si el incremento en la inversión se justifica. Dicho incremento se considera aceptable si su rendimiento excede la tasa de recuperación mínima.

2.5.7.5 VENTAJAS

- Es muy **sencillo de aplicar**, ya que para calcularlo se realizan operaciones simples.
- Tiene en cuenta el **valor** de dinero en el **tiempo**.

2.5.7.6 INCONVENIENTES

- Dificultad para establecer el valor de K. A veces se usan los siguientes criterios
  - Coste del dinero a largo plazo
  - Tasa de rentabilidad a largo plazo de la empresa
  - Coste de capital de la empresa.
  - Como un valor subjetivo
  - Como un coste de oportunidad.
- El VAN supone que los flujos que salen del proyecto se reinvierten en el proyecto al mismo valor K que el exigido al proyecto, lo cual puede no ser cierto.
- El VAN es el valor presente de los flujos futuros de efectivo menos el valor presente del costo de la inversión.

2.7.6 TIR:

La **tasa interna de retorno** o **tasa interna de rentabilidad** (TIR) de una inversión es el promedio geométrico de los rendimientos futuros esperados de dicha inversión, y que implica por cierto el supuesto de una oportunidad para "reinvertir". En términos simples, diversos autores la conceptualizan como la tasa de descuento con la que el valor actual neto o valor presente neto (VAN o VPN) es igual a cero.[1] [2]

La TIR puede utilizarse como indicador de la rentabilidad de un proyecto: a mayor TIR, mayor rentabilidad;[3] [4] así, se utiliza como uno de los criterios para decidir sobre la aceptación o rechazo de un proyecto de inversión.[5] Para ello, la TIR se compara con una tasa mínima o tasa de corte, el coste de oportunidad de la inversión (si la inversión no tiene riesgo, el coste de oportunidad utilizado para comparar la TIR será la tasa de rentabilidad

libre de riesgo). Si la tasa de rendimiento del proyecto - expresada por la TIR- supera la tasa de corte, se acepta la inversión; en caso contrario, se rechaza.

Otras Definiciones

- Es la tasa que iguala la suma del valor actual de los gastos con la suma del valor actual de los ingresos previstos: $\sum_{i=1}^{N} VPIi = \sum_{i=1}^{N} VPCi$
- Es la tasa de interés para la cual los ingresos totales actualizados es igual a los costos totales actualizados: $ITAc = CTAc$
- Es la tasa de interés por medio de la cual se recupera la inversión.
- Es la tasa de interés máxima a la que se pueden endeudar para no perder dinero con la inversión.
- Es la tasa real que proporciona un proyecto de inversión y es aquella que al ser utilizada como tasa de descuento en el cálculo de un VAN dará como resultado 0.

Cálculo de la Tasa Interna de Retorno

La Tasa Interna de Retorno TIR es el tipo de descuento que hace igual a cero el VAN:

$$VAN = \sum_{t=1}^{n} \frac{F_t}{(1+TIR)^t} - I = 0$$

Donde:

F_t es el Flujo de Caja en el periodo t.
n es el número de periodos.
I es el valor de la inversión inicial.

La aproximación de Schneider usa el teorema del binomio para obtener una fórmula de primer orden:

$$(1+TIR)^{-n} \approx 1 - n * TIR$$

$$I = F_1 * (1 - TIR) + F_2 * (1 - 2 * TIR) + ... + F_n * (1 - n * TIR)$$

$$I - (F_1 + F_2 + ... + F_n) = -TIR * (F_1 + 2 * F_2 ... + n * F_n)$$

De donde: *

$$TIR = \frac{-I + \sum_{i=1}^{n} F_i}{\sum_{i=1}^{n} i * F_i}$$

Sin embargo, el cálculo obtenido puede estar bastante alejado de la TIR real.

Uso general de la TIR

Como ya se ha comentado anteriormente, la **TIR** o tasa de rendimiento interno, es una herramienta de toma de decisiones de inversión utilizada para conocer la factibilidad de diferentes opciones de inversión.

El criterio general para saber si es conveniente realizar un proyecto es el siguiente:

- Si TIR $\geq$r $\rightarrow$Se aceptará el proyecto. La razón es que el proyecto da una rentabilidad mayor que la rentabilidad mínima requerida (el coste de oportunidad).
- Si TIR $<$r $\rightarrow$Se rechazará el proyecto. La razón es que el proyecto da una rentabilidad menor que la rentabilidad mínima requerida.

r representa el costo de oportunidad.

2.7.6.1 DIFICULTADES EN EL USO DE LA TIR

- **Criterio de aceptación o rechazo**. El criterio general sólo es cierto si el proyecto es del tipo "prestar", es decir, si los primeros flujos de caja son negativos y los siguientes positivos. Si el proyecto es del tipo "pedir prestado" (con flujos de caja positivos al principio y negativos después), la decisión de aceptar o rechazar un proyecto se toma justo al revés:
    -
    - Si TIR >r $\rightarrow$Se rechazará el proyecto. La rentabilidad que nos está requiriendo este préstamo es mayor que nuestro costo de oportunidad.
    - Si TIR <= r $\rightarrow$Se aceptará el proyecto.

- **Comparación de proyectos excluyentes**. Dos proyectos son excluyentes si solamente se puede llevar a cabo uno de ellos. Generalmente, la opción de inversión con la TIR más alta es la preferida, siempre que los proyectos tengan el mismo riesgo, la misma duración y la misma inversión inicial. Si no, será necesario aplicar el criterio de la TIR de los flujos incrementales.

- **Proyectos especiales**, también llamado el problema de la inconsistencia de la TIR. Son proyectos especiales aquellos que en su serie de flujos de caja hay más de un cambio de signo. Estos pueden tener más de una TIR, tantas como cambios de signo. Esto complica el uso del criterio de la TIR para saber si aceptar o rechazar la inversión. Para solucionar este problema, se suele utilizar la TIR Corregida.

2.8 SEGURIDAD:

La seguridad informática o seguridad de tecnologías de la información es el área de la informática que se enfoca en la protección de la infraestructura computacional y todo lo relacionado con esta y, especialmente, la información contenida o circulante. Para ello existen una serie de estándares, protocolos, métodos, reglas, herramientas y leyes concebidas para minimizar los posibles riesgos a la infraestructura o a la información. La seguridad informática comprende software (bases de datos, metadatos, archivos), hardware y todo lo que la organización valore (activo) y signifique un riesgo si esta información confidencial llega a manos de otras personas, convirtiéndose, por ejemplo, en información privilegiada.

El concepto de seguridad de la información no debe ser confundido con el de «seguridad informática», ya que este último solo se encarga de la seguridad en el medio informático, pero la información puede encontrarse en diferentes medios o formas, y no solo en medios informáticos.

La seguridad informática es la disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinados a conseguir un sistema de información seguro y confiable

2.8.1 FISICO:

Control de la red

Los puntos de entrada en la red son generalmente el correo, las páginas web y la entrada de ficheros desde discos, o de ordenadores ajenos, como portátiles.

Mantener al máximo el número de recursos de red solo en modo lectura, impide que ordenadores infectados propaguen virus. En el mismo sentido se pueden reducir los permisos de los usuarios al mínimo.

Se pueden centralizar los datos de forma que detectores de virus en modo batch puedan trabajar durante el tiempo inactivo de las máquinas.

Controlar y monitorizar el acceso a Internet puede detectar, en fases de recuperación, cómo se ha introducido el virus.

Protección física de acceso a las redes

Independientemente de las medidas que se adopten para proteger a los equipos de una red de área local y el software que reside en ellos, se deben tomar medidas que impidan que usuarios no autorizados puedan acceder. Las medidas habituales dependen del medio físico a proteger.

A continuación se enumeran algunos de los métodos, sin entrar al tema de la protección de la red frente a ataques o intentos de intrusión desde redes externas, tales como Internet.

Redes cableadas

Las rosetas de conexión de los edificios deben estar protegidas y vigiladas. Una medida básica es evitar tener puntos de red conectados a los switches. Aún así siempre puede ser sustituido un equipo por otro no autorizado con lo que hacen falta medidas adicionales: norma de acceso 802.1x, listas de control de acceso por MAC addresses, servidores de DHCP por asignación reservada, etc.

Redes inalámbricas

En este caso el control físico se hace más difícil, si bien se pueden tomar medidas de contención de la emisión electromagnética para circunscribirla a aquellos lugares que consideremos apropiados y seguros. Además se consideran medidas de calidad el uso del cifrado (WPA, WPA v.2, uso de certificados digitales, etc.), contraseñas compartidas y, también en este caso, los filtros de direcciones MAC, son varias de las medidas habituales que cuando se aplican conjuntamente aumentan la seguridad de forma considerable frente al uso de un único método.

2.8.2LOGICO:

2.8.2.1 RESPALDO DE INFORMACIÓN

Artículo principal: Copia de seguridad

La información constituye el activo más importante de las empresas, pudiendo verse afectada por muchos factores tales como robos, incendios, fallas de disco, virus u otros. Desde el punto de vista de la empresa, uno de los problemas más importantes que debe resolver es la protección permanente de su información crítica.

La medida más eficiente para la protección de los datos es determinar una buena política de copias de seguridad o backups. Este debe incluir copias de seguridad completa (los datos son almacenados en su totalidad la primera vez) y copias de seguridad incrementales (solo se copian los ficheros creados o modificados desde el último backup). Es vital para las empresas elaborar un plan de backup en función del volumen de información generada y la cantidad de equipos críticos.

Un buen sistema de respaldo debe contar con ciertas características indispensables:

- **Continuo**

  El respaldo de datos debe ser completamente automático y continuo. Debe funcionar de forma transparente, sin intervenir en las tareas que se encuentra realizando el usuario.

- **Seguro**

  Muchos softwares de respaldo incluyen cifrado de datos, lo cual debe ser hecho localmente en el equipo antes del envío de la información.

- **Remoto**

  Los datos deben quedar alojados en dependencias alejadas de la empresa.

- **Mantenimiento de versiones anteriores de los datos**

  Se debe contar con un sistema que permita la recuperación de, por ejemplo, versiones diarias, semanales y mensuales de los datos.

Hoy en día los sistemas de respaldo de información online, servicio de backup remoto, están ganando terreno en las empresas y organismos gubernamentales. La mayoría de los sistemas modernos de respaldo de información online cuentan con las máximas medidas de seguridad y disponibilidad de datos. Estos sistemas permiten a las empresas crecer en volumen de información derivando la necesidad del crecimiento de la copia de respaldo a proveedor del servicio.

2.8.2.2 PROTECCIÓN CONTRA VIRUS

Los virus son uno de los medios más tradicionales de ataque a los sistemas y a la información que sostienen. Para poder evitar su contagio se deben vigilar los equipos y los medios de acceso a ellos, principalmente la red.

2.8.2.3 CONTROL DEL SOFTWARE INSTALADO

Tener instalado en la máquina únicamente el software necesario reduce riesgos. Así mismo tener controlado el software asegura la calidad de la procedencia del mismo (el software obtenido de forma ilegal o sin garantías aumenta los riesgos). En todo caso un inventario de software proporciona un método correcto de asegurar la reinstalación en caso de desastre. El software con métodos de instalación rápidos facilita también la reinstalación en caso de contingencia.

CAPITULO 3

MARCO APLICATIVO

En el presente capitulo es demostrar el análisis, diseño e implementación, con el objetivo de identificar y resolver los problemas en el control y seguimiento académico de la institución.

Se detallara las fases del Proceso Unificado de Desarrollo (RUP), el mismo que comprende 4 fases que en anterior capitulo fue mencionado y descrito.

3.1 FASE DE INICIO.

Se realiza un estudio del modelado de negocio y los requerimientos funcionales y no funciones, identificando los casos de uso.

a) MODELO DE NEGOCIOS

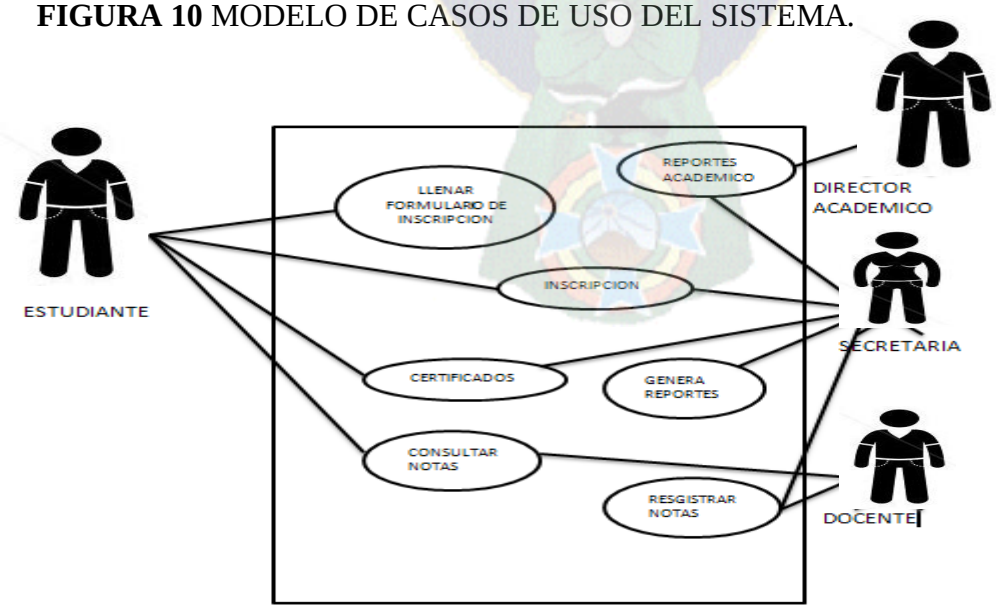
Nos permite capturar los requisitos requeridos para desarrollar el sistema, de acuerdo a las necesidades del usuario.

Se identifican los casos de uso de los actores

Se tiene el modelo de actividades del negocio

Se identifican las acciones de cada caso de uso.

FIGURA 10 MODELO DE CASOS DE USO DEL SISTEMA.



FUENTE (Elaboración Propia**)**

CASO DE USO EXISTENTES.

Las tablas que se describen a continuación expresan claramente los casos de uso de los procesos que se realizan los estudiantes para la solicitud de inscripción.

Identificación de actores y casos de uso.

Tabla 3.1 descripción de caso de uso: llenar formulario de inscripción.

Caso De Uso	Llenar formulario de inscripción
Actores	Estudiante, secretaria
Propósito	Saber si la persona que será inscrita es legitima
Resumen	La secretaria es la encargada de realizar la inscripción con la descripción del curso o carrera que desea.
Tipo	Primario
Curso normal de eventos	
Acción del actor	Respuesta del sistema
1.Solicita formulario de inscripción 3. presenta los documentos que pide como requisitos en el formulario.	2. entrega formulario de inscripción 4. se registran los datos y se guardan en un folder

Tabla 3.2 descripción de caso de uso: certificados.

Caso De Uso	certificados
Actores	Estudiante, secretaria
Propósito	Saber si la persona aprobó el curso
Resumen	Los estudiantes solicitan certificado (record académico, certificados de notas, etc.)a las instancias correspondientes
Tipo	Primario
Curso normal de eventos	
Acción del actor	Respuesta del sistema
1.el estudiante realiza un pago por el certificado.	2. registra el pago ,en una lista 3. el certificado es entregado con la debida nota.

Tabla 3.3 descripción de caso de uso: consulta de notas

Caso De Uso	Consulta de notas
Actores	Estudiante, docente, secretaria
Propósito	Saber si la persona aprobó o reprobó el curso
Resumen	El estudiante solicita su nota a las instancias correspondientes.
Tipo	Primario
Curso normal de eventos	
Acción del actor	Respuesta del sistema

1.Solicita su nota	2. solicita la nota la docente 3. docente da un informe de su nota.

Tabla 3.4 descripción de caso de uso :reportes académicos.

Caso De Uso	Reportes académicos
Actores	Director académico, secretaria.
Propósito	Registrar un informe detallado de la gestión.
Resumen	La secretaria da un informe detallado al director académico y se genera reportes académicos dela gestion.
Tipo	Primario
Curso normal de eventos	
Acción del actor	Respuesta del sistema
1.Solicita informe detallado	2. entrega informe de reportes académicos.

Tabla 3.5 Descripción de caso de uso: registrar notas.

Caso De Uso	Registra notas
Actores	Secretaria, docente
Propósito	Registra la nota del estudiante.
Resumen	La secretaria es la encargada de registrar las notas que proporciona el docente.
Tipo	Primario
Curso normal de eventos	
Acción del actor	Respuesta del sistema
1.Solicita nota	2. registra las notas del docente

3.1.2 REQUERIMIENTOS

3.1.2.1 REQUERIMIENTOS DE LOS USUARIOS.

Se identifican los requerimientos de los usuarios del sistema, donde se muestra en la tabla

Tabla 3.6 requerimientos de usuario

Nro.	Requerimientos del usuario	Categoría
R1	Tener un información centralizada, organizada y disponible en el sistema, para el apoyo de toma de decisiones	Evidente
R2	Obtener reportes mensuales, trimestrales y anuales de la capacitación.	Evidente
R3	Obtener reportes de faltas y retrasos de asistencia	Evidente

R4	Poder acceder a la información, según cargo o función que desempeñe.	Evidente

3.1.3 REQUERIMIENTOS DEL SISTEMA.

Se menciona los requerimientos del sistema de control y seguimiento académico, teniendo una muestra representativa del funcionamiento del sistema.

Tabla 3.7 requerimientos de sistema.

Nro.	Requerimientos Del Sistema	Categoría
R1	El sistema debe ser fácil, el uso por parte del usuario.	Evidente
R2	El sistema debe dar reportes actualizados programas y administración.	Evidente
R3	El sistema debe tener la base de datos actualizados para dar respuesta a las consultas ingresadas por el usuario.	Evidente
R4	El sistema debe asegurar el ingreso de usuarios autorizados, mediante pasword.	Evidente

3.1.3.1 REQUERIMIENTOS FUNCIONALES DEL SISTEMA.

Se identifican los requerimientos funcionales que hará como proceso, mostrando los eventos que se desarrolla dentro del sistema.

En las siguientes tablas.

Registro de datos del estudiante

Ref.	Función	Categoría
A1	el sistema proporciona formulario de inscripción	evidente
A2	El sistema verifica el llenado del formulario	evidente
A3	El sistema registra y almacena los datos inscritos	oculto
A4	Estudiante se inscribe	Evidente

Restringe a estudiantes no registrados

Ref.	Función	Categoría
A1	Restringe accesos u operaciones a usuarios	oculta
A2	Los usuarios deben identificarse para tener acceso dentro del sistema	evidente

Muestra registros, de búsqueda materias

Ref.	función	categoría
A1	Mostrar información sobre el seguimiento académico del estudiante	Evidente
A2	Mostrar materias a inscribirse	Evidente
A3	Emitir listas	Evidente
A4	Cumplir con las búsquedas necesarias	evidente

Pago de mensualidades

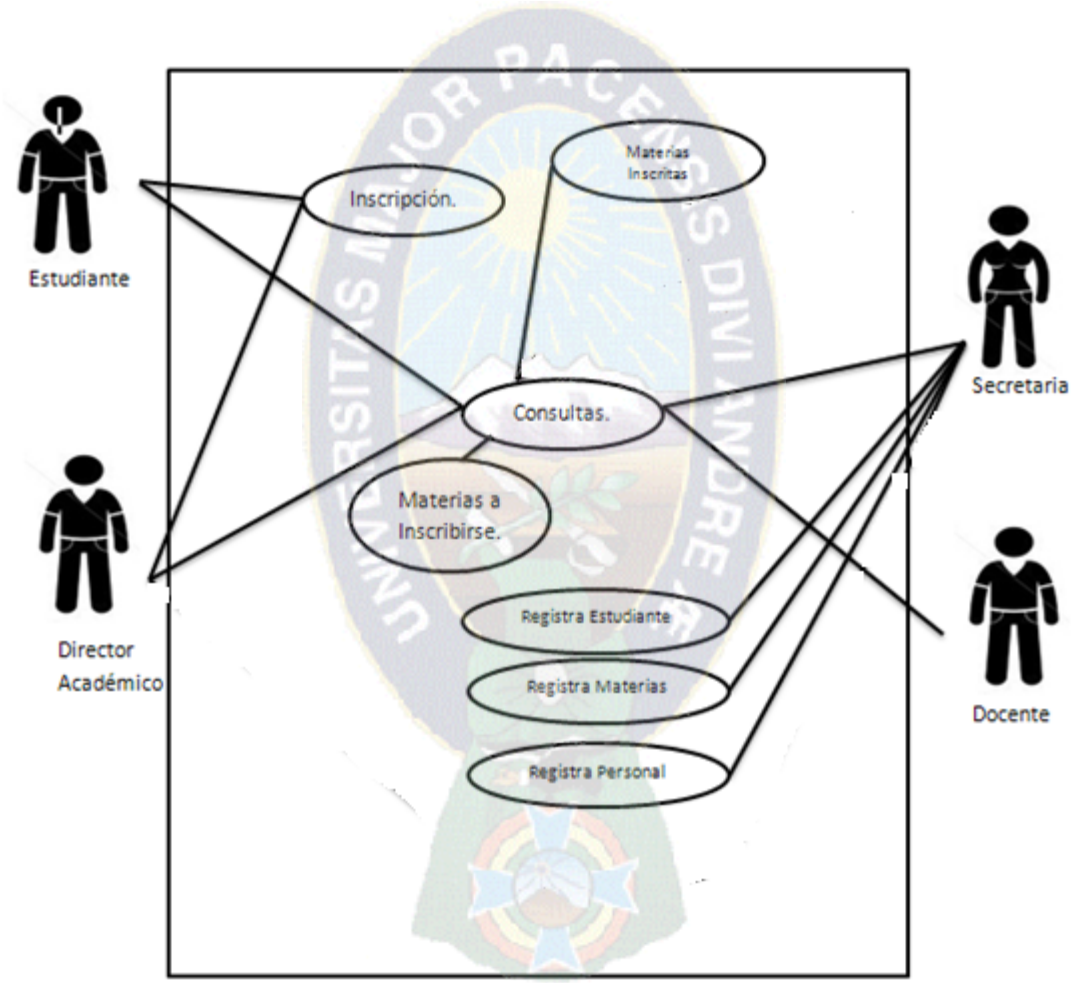
Ref.	función	categoría
A1	El sistema muestra el monto a pagar la mensualidad	Evidente
A2	El sistema genera pagos en cuotas	Evidente
A3	El sistema registra y actualiza el registro de pagos	oculto
A4	Se entrega un recibo y comprobante	evidente

3.1.4 MODELADO DE CASOS DE USO DEL SISTEMA.

Casos de USO DE ALTO NIVEL.

Esto se asemeja al concepto orientado a objetos de sub-clases, la figura 10 muestra el modelo de uso representado para la realización del sistema y los casos de uso que manejara el sistema

FIGURA 11 Modelo de casos de uso de alto nivel.



FUENTE(Elaboración Propia**)**

3.2 FASE DE ELABORACIÓN.

En esta fase todos los componentes se desarrollan e incorporan al sistema donde los requerimientos son estables y detallados.

3.2.1 IDENTIFICACION DE ACTORES.

Estudiante: es aquella persona que se inscribirá a algún curso determinado para su capacitación, además de entregar los datos actualizados.

Secretaria: es la encargada de registrar al estudiante, las notas respectivas, al personal , también realiza y elabora reportes y pago de mensualidades.

Director académico: es el encargado de hacer modificaciones al pensum, revisa el número de inscritos por curso y detalla a fondo los procesos del sistema.

Docente: es el encargado de hacer consultas y entrega las notas respectivas de los alumnos.

LISTA DE ACTORES Y CASOS DE USO

Tabla 3.8 LISTA DE ACTORES Y CASOS DE USO

Actor	Casos de uso
Estudiante	Inscripción consulta
Secretaria	Registra estudiante Registra notas Registra personal Realiza búsqueda
Director académico	Consultas Modificaciones de pensum Inscripción
docente	consultas

3.2.2 CASO DE USO EXPANDIDO

3.2.2.1 SOLICITA INSCRIPCIÓN.

FIGURA 11(Caso de uso expandido: solicita inscripción**)**

FUENTE(Elaboración Propia**)**

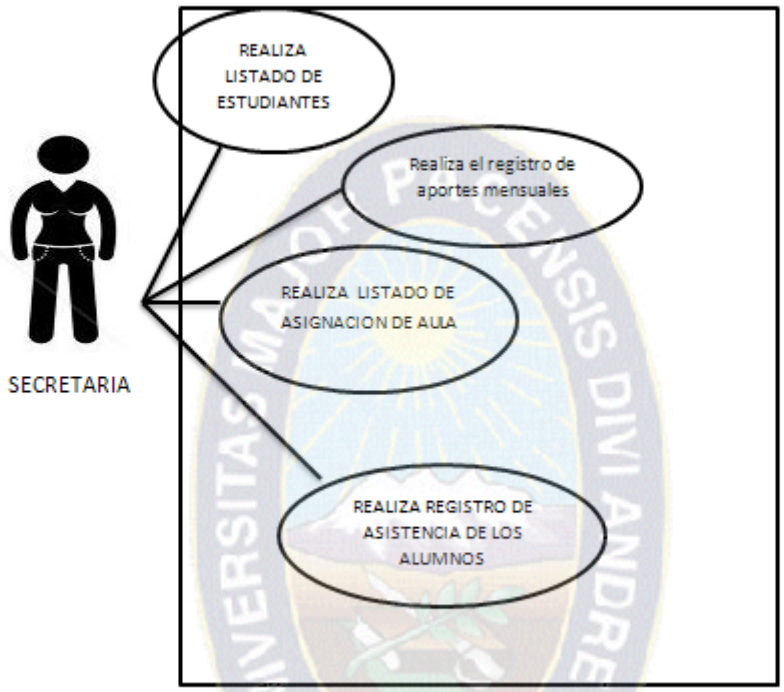
Tabla 3.9 LISTA DE ACTORES Y CASOS DE USO

Caso de Uso:	**Solicita inscripción**
Actores:	Estudiante, secretaria
Propósito:	Iniciar proceso de inscripción
resumen	El estudiante se dirige a la secretaria de la institución, entrega los requisitos de inscripción para estudiantes nuevos, entonces la secretaria ingresa al sistema para procesar la debida inscripción , además solicita que se genere un nombre de usuario y una contraseña para el estudiante.
Tipo	Primario , esencial
Referencias cruzadas	A1,A2,A4
Precondición.	Se necesita la estudiante para su inscripción
postcondicion	El estudiante que se inscribió puede obtener su usuario y contraseña.

curso normal de eventos	
Accion del Actor	Respuesta del sistema
1. El estudiante se dirige a al institución 3 . Usuario consulta e inicializa proceso de inscripción. 5. el estudiante procede a llenar el formulario. 8.obtiene su usuario y contraseña 8. se entrega la factura al debido estudiante 9. el estudiante realiza su inscripción satisfactoriamente.	2. Despliega una pagina con todos los menús 4 . el sistema proporciona el registro de inscripción. 6 el sistema registra y verifica el llenado del formulario. 7 el sistema almacena los datos en una base de datos.

3.2.2.2 REGISTRO DE REPORTES

FIGURA 12 (Caso de uso expandido: Registro de reportes**)**



FUENTE (Elaboración Propia**)**

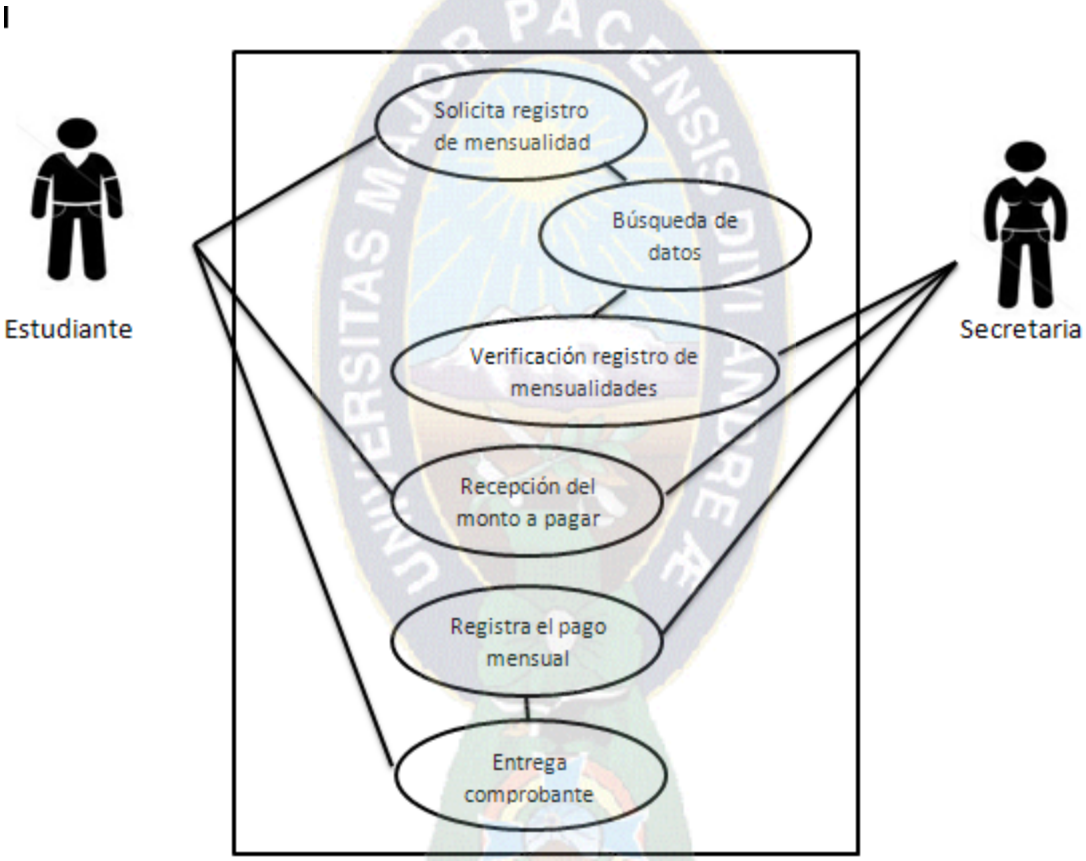
Tabla 4.0 LISTA DE ACTORES Y CASOS DE USO

Caso de Uso:	ELABORACION DE REPORTES
Actores:	secretaria
Propósito:	Consultar información (según nombre de usuario y contraseña), tener el control y seguimiento de los diferentes registros de área.
resumen	Se realiza reportes, asistencias, y asignación de aulas.
Tipo	Primario y esencial
Referencias cruzadas	A1,A2,A3,A4
Precondición	El usuario debe estar registrado
Postcondicion	Solo el usuario autorizado puede realizar reportes
curso normal de eventos	

Acción del Actor	Respuesta del sistema
1. La secretaria inicializa el sistema con la elaboración de reportes 3 . la secretaria saca un informe de formulario actualizado.	2. El sistema genera formulario de reportes de historial 4 . el sistema genera registro de aportes mensuales y observaciones.

3.2.2.3 REGISTRO DE MENSUALIDADES

FIGURA 13 (Caso de uso expandido: Registro de mensualidades**)**



FUENTE (Elaboración Propia**)**

Tabla 4.1 LISTA DE ACTORES Y CASOS DE USO

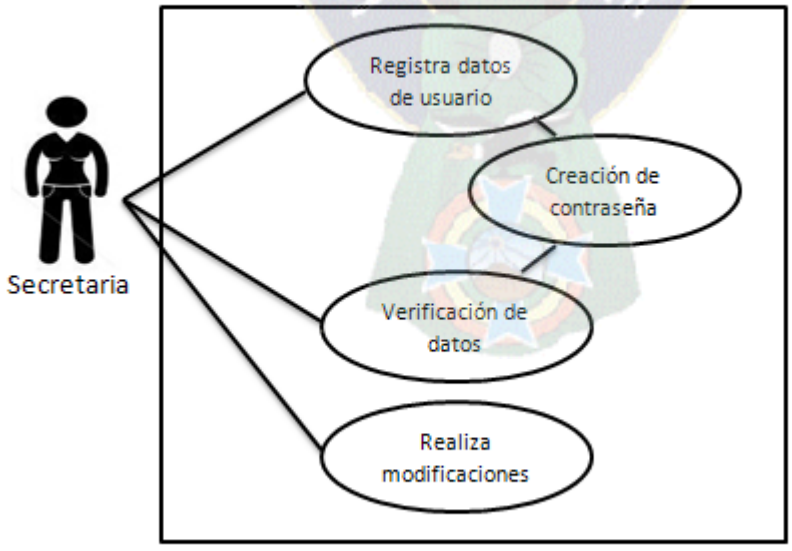
Caso de Uso:	Registro de mensualidades
Actores:	Secretaria,estudiante
Propósito:	Tener registrado todos los pagos de mensualidades
resumen	Muestra el registro de pago de mensualidades , donde se hace el cobro del monto a pagar se registra y se imprime un registro de

	comprobante
Tipo	Primario y esencial
Referencias cruzadas	A1,A2,A3,A4
Precondición	Se necesita al estudiante para los pagos
Postcondicion	Solo el usuario autorizado puede registrar las mensualidades

curso normal de eventos	
Acción del Actor	Respuesta del sistema
1. El estudiante realiza el pago de mensualidad 2 . la secretaria habilita el sistema para realizar la búsqueda de pago. 4 . el estudiante paga la mensualidad 5 el responsable entrega un recibo de comprobante.	3 El sistema registra y actualiza el registro de pago. 4 . el sistema registra y actualiza el registro de pago

3.2.2.4 REGISTRO Y MODIFICACION DE USUARIOS

FIGURA 14(Caso de uso expandido: Registro y modificación de usuarios)



FUENTE (Elaboración Propia**)**

Tabla 4.2 LISTA DE ACTORES Y CASOS DE USO

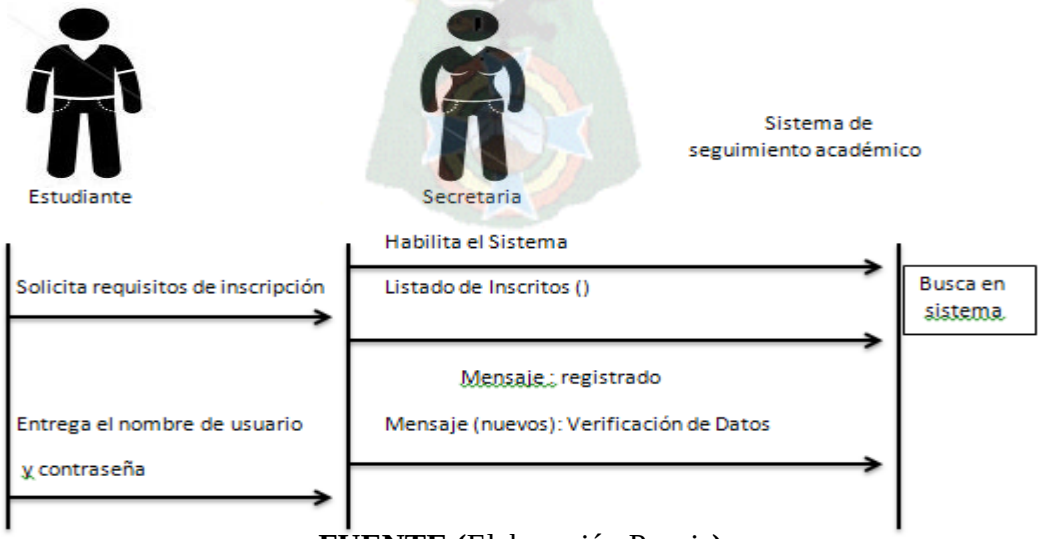
Caso de Uso:	Registro y modificación de usuario
Actores:	secretaria
Propósito:	Tener registrado a los usuarios autorizados, teniendo seguridad en la creación de contraseña.
resumen	Se registra a los usuarios, crean su contraseña los autorizados en utilizar el sistema
Tipo	Primario y esencial
Referencias cruzadas	A1,A2,A3,A4
Precondición	Se necesita usuarios que manejaran el sistema
Postcondicion	Solo los usuarios autorizados pueden ingresar
curso normal de eventos	

Acción del Actor	Respuesta del sistema
1 Secretaria habilita al sistema. 2 Secretaria registra datos de usuarios	2. El sistema almacena los datos 4 . el sistema registra contraseña 3. El sistema visualiza la verificación.

3.2.3 DIAGRAMA DE SECUENCIA DE SISTEMA

3.2.3.1 SOLICITUD DE INSCRIPCION

FIGURA 15 (Diagrama de secuencia: Solicitud de inscripción**)**



FUENTE (Elaboración Propia**)**

Contrato de operaciones de solicitud de inscripcion

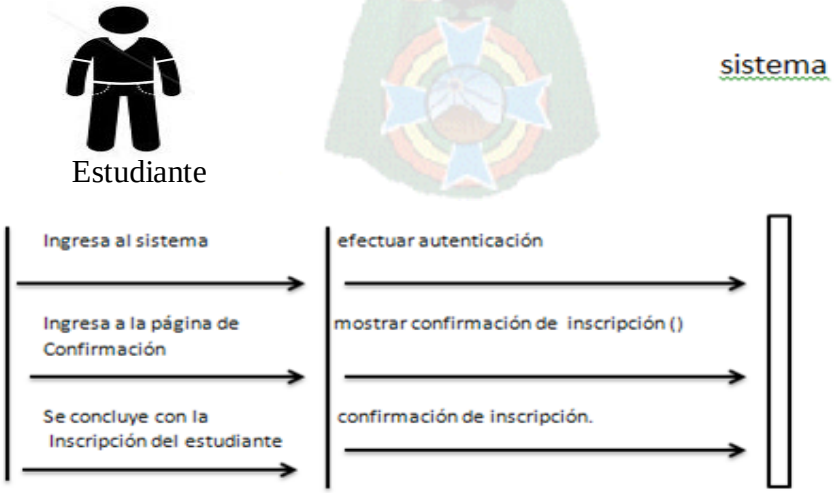
Nombre	Solicita inscripcion()
Responsable	El sistema comunica la Verificaion del Llenado de datos
Tipo	Sistema
Referencias	Inscripcion,A2
Precondicion	La secretaria tiene la informacion a generar
PostCondicion	Este caso de uso servira para registro de datos,registro reportes y mensualidades

Nombre	Listado de inscritos()
Responsable	El sistema realiza la busqueda para visualizar el formulario
Tipo	Sistema
Referencias	Inscripcion,A1
Precondicion	La secretaria realiza busqueda
PostCondicion	Este caso servira para realizar la busqueda , visualizar el formulario de inscripcion

Nombre	Verificacion de datos()
Responsable	La secretaria verifica el sistema
Tipo	Sistema
Referencias	Inscripcion,A1
Precondicion	La secretaria ingresa los datos al sistema
PostCondicion	Este caso de uso servira para registrar datos

3.2.3.2 CONFIRMAR INSCRIPCION

FIGURA 16 (Diagrama de secuencia: Confirmar inscripción**)**

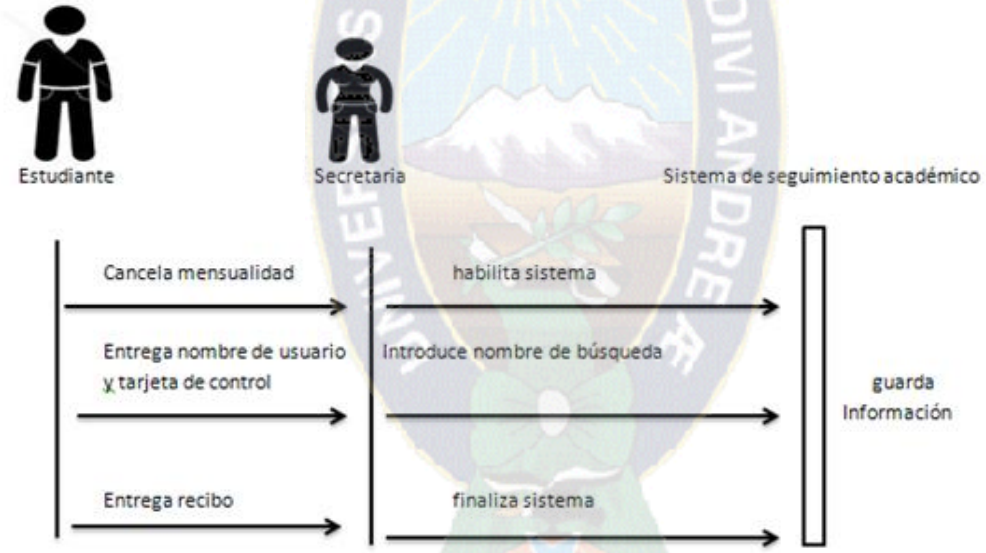


FUENTE (Elaboración Propia**)**

Nombre	mostrarConfirmacionlnscripcion()
Responsable	Se muestra la confirmación de inscripción en la página principal del sistema de aquellos estudiantes antiguos en el establecimiento educativo.
Tipo	Sistema
Referencias	Inscripcion,A2
Precondicion	Especificación de los datos personales del estudiante inscrito en el instituto en una gestión anterior a la presente.
PostCondicion	Se asocia el Formulario de Confirmación de inscripción correctamente desplegado por el sistema.

3.2.3.3 REGISTRO DE MENSUALIDADES

FIGURA 17(Diagrama de secuencia: Registro de mensualidades**)**



FUENTE (Elaboración Propia**)**

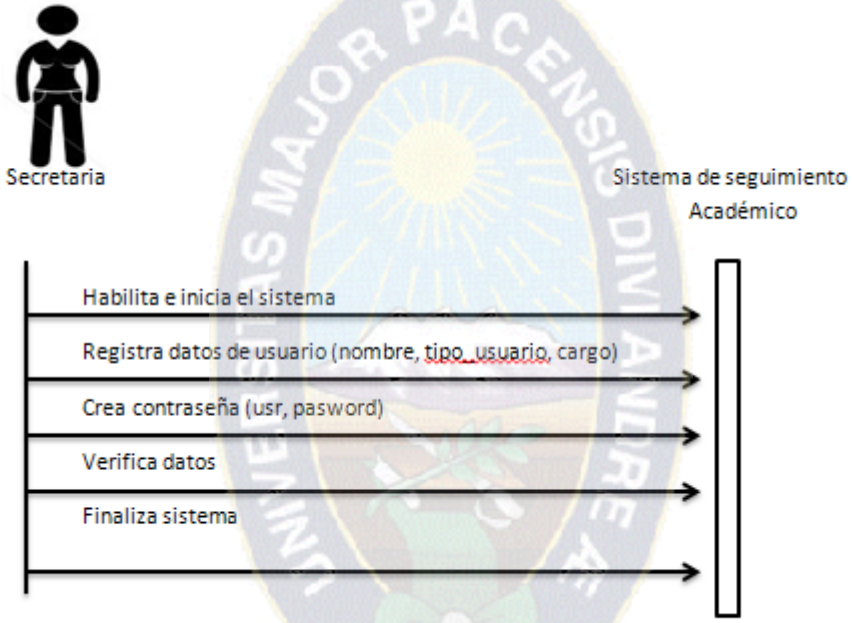
Nombre	Registra mensualidad()
Responsable	La secretaria registra el pago del curso
Tipo	Sistema
Referencias	Inscripcion,A1
Precondicion	La secretaria tiene el monto a registrar
PostCondicion	Este caso de uso servira para registrar el apgo, tener actualizado el registro

Nombre	Verificacion de datos()
Responsable	El visualiza la cancelacion para ser verificado

Tipo	Sistema
Referencias	Inscripcion,A2
Precondicion	La secretaria tiene que verificar el pago
PostCondicion	Este caso de uso servira para registrar el apgo , tener actualizado el registro

3.2.3.4 REGISTRO O MODIFICACIÓN DE USUARIO

FIGURA 18 (Diagrama de secuencia: Registro o modificación de usuario**)**



FUENTE (Elaboración Propia**)**

Nombre	Registro de datos()
Responsable	El administrador registra los datos de cada usuario
Tipo	Sistema
Referencias	Usuario,A1
Precondicion	El usuario, debe tener el registro de usuarios en documento manual
PostCondicion	Se visualiza la información

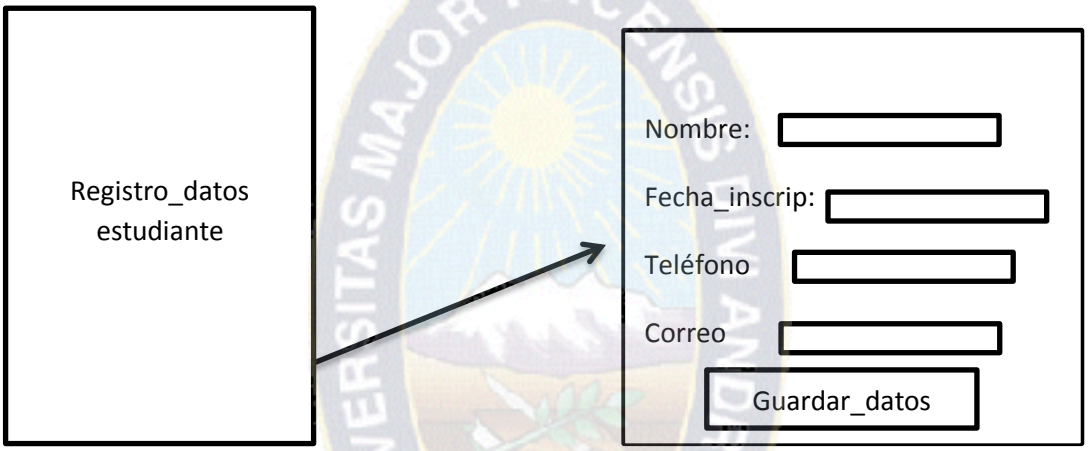
Nombre	Registro_de _password
Responsable	Los usuarios deben crear su contraseña para ingresar al sistema
Tipo	Sistema
Referencias	Usuario,A2

Precondicion	El usuario, debe ser registrado anteriormente
PostCondicion	Debe recordar su contraseña

3.2.4 MODELO DE CASOS DE USO PARA EL DISEÑO REAL

SE representa mediante las pantallas que están gráficamente diseñadas para posteriormente se desarrolle como sistema.

FIGURA 19 (Diagrama de caso de uso diseño real Pantalla Inscripción **)**

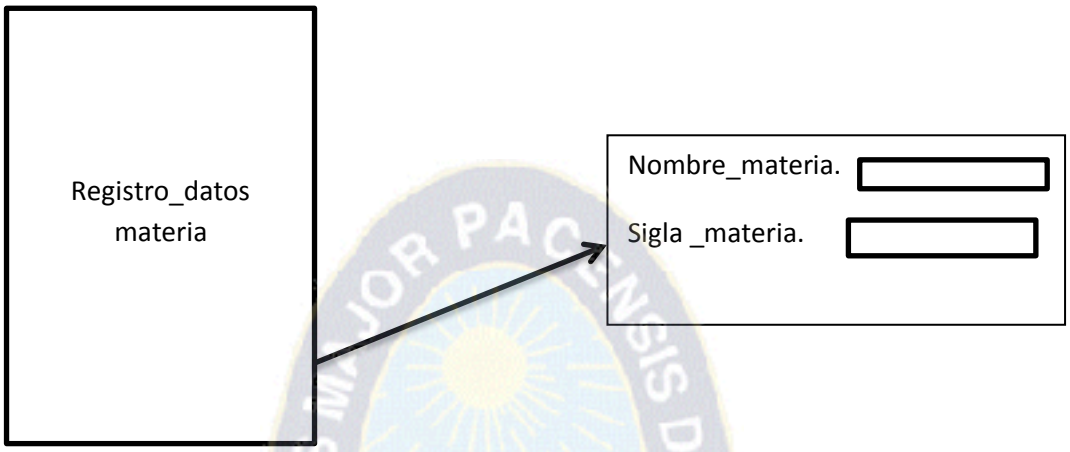


Caso de Uso:	Pantalla de inscripción
Actores:	secretaria
Propósito:	Tener registrado a los estudiantes
resumen	Se registra a los usuarios, crean su contraseña los autorizados en utilizar el sistema
Tipo	Primario y esencial

curso normal de eventos

Acción del Actor	Respuesta del sistema
1 Secretaria habilita al sistema. 2 Secretaria registra datos de usuarios 4 elabora registro de datos	3 El sistema genera un formulario de inscripción. 5 el sistema solicita guardar los datos

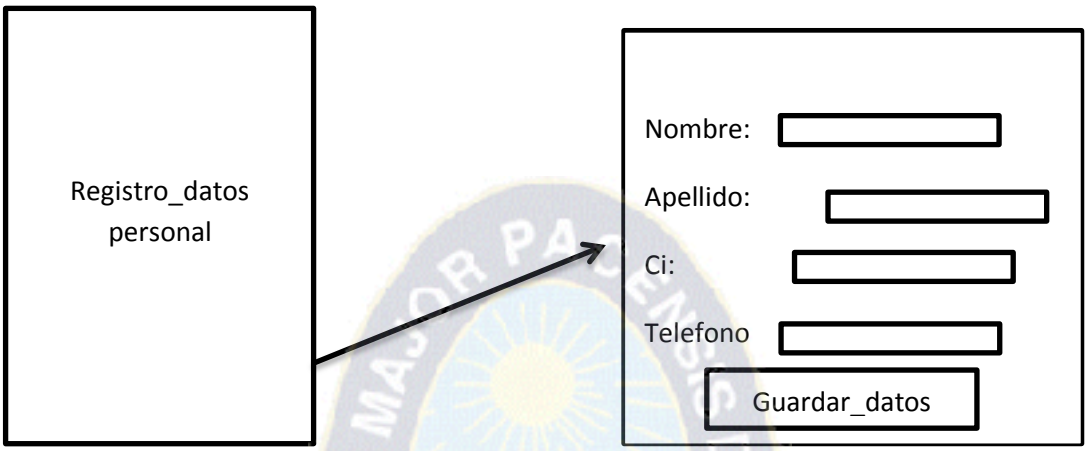
FIGURA 20 (Diagrama de caso de uso para el diseño real asignar una nueva materia**)**



Caso de Uso:	Pantalla de registro de datos de materia
Actores:	Secretaria, docente
Propósito:	Tener registrado las materias necesarias
resumen	Se registra materias nuevas donde se asignan con el docente de la area
Tipo	Primario y esencial

curso normal de eventos	
Acción del Actor	Respuesta del sistema
1 Secretaria habilita al sistema. 2 Secretaria registra datos de materia 4 elabora registro 5 guarda la nueva materia	3 El sistema genera un formulario para asignar una nueva materia 5 el sistema solicita guarda los datos de la nueva materia

FIGURA 21 (Diagrama de caso de uso para el diseño real asignar un nuevo docente**)**



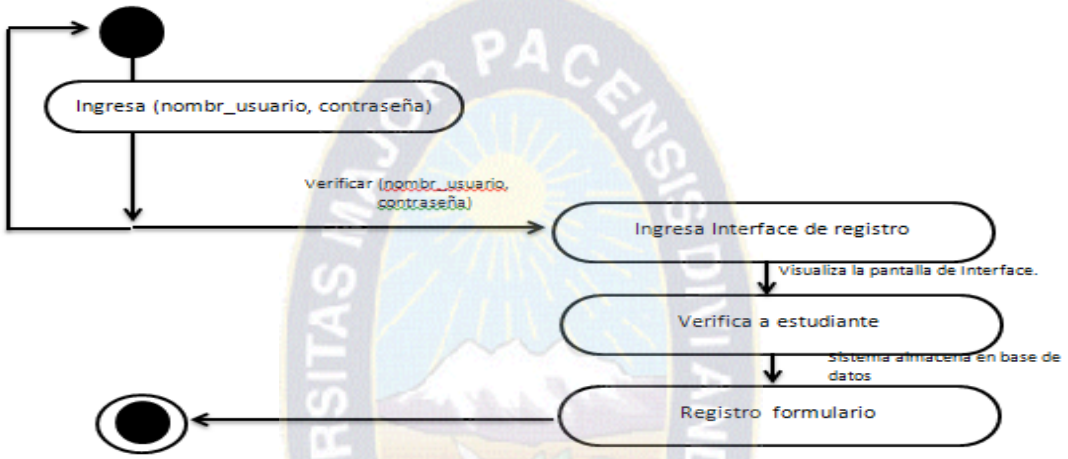
Caso de Uso:	Pantalla de registro de datos de nuevo personal
Actores:	Secretaria
Propósito:	Tener registrado a los nuevos docentes
resumen	Se registra a un nuevo personal
Tipo	Primario y esencial
curso normal de eventos	
Acción del Actor	Respuesta del sistema
1 Secretaria habilita al sistema. 2 Secretaria registra datos de nuevo de nuevo personal 4 elabora registro 5 guarda la nueva materia	3 El sistema genera un formulario para asignar un nuevo personal 5 el sistema solicita guarda los datos de la nueva materia

3.2.5 DIAGRAMAS DE ESTADO.

El diagrama de estado representa la secuencia de un objeto que pasa durante su tiempo de vida, teniendo como característica los eventos que hacen posible la transición de un estado a otro

3.2.5.1 SOLICITUD DE INSCRIPCIÓN

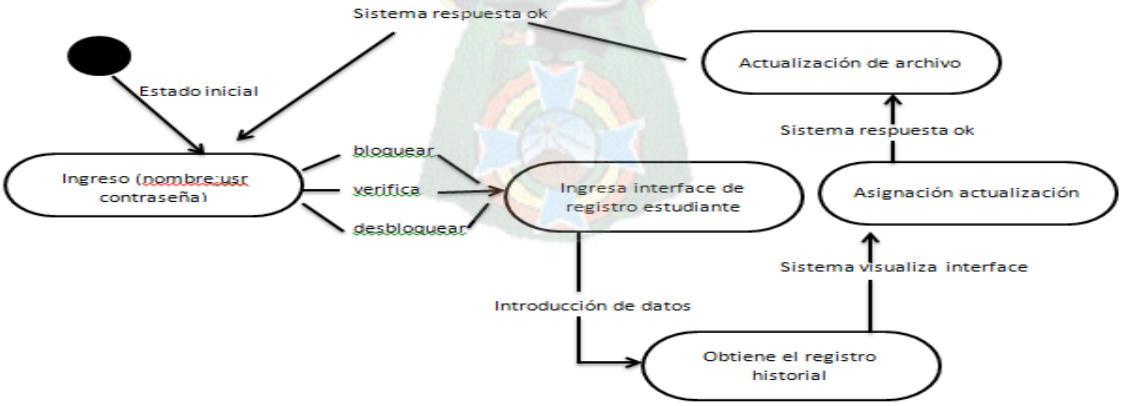
FIGURA 22 (Diagrama de estado: Solicitud de inscripción**)**



FUENTE (Elaboración Propia**)**

3.2.5.2 REGISTRO DE ESTUDIANTE

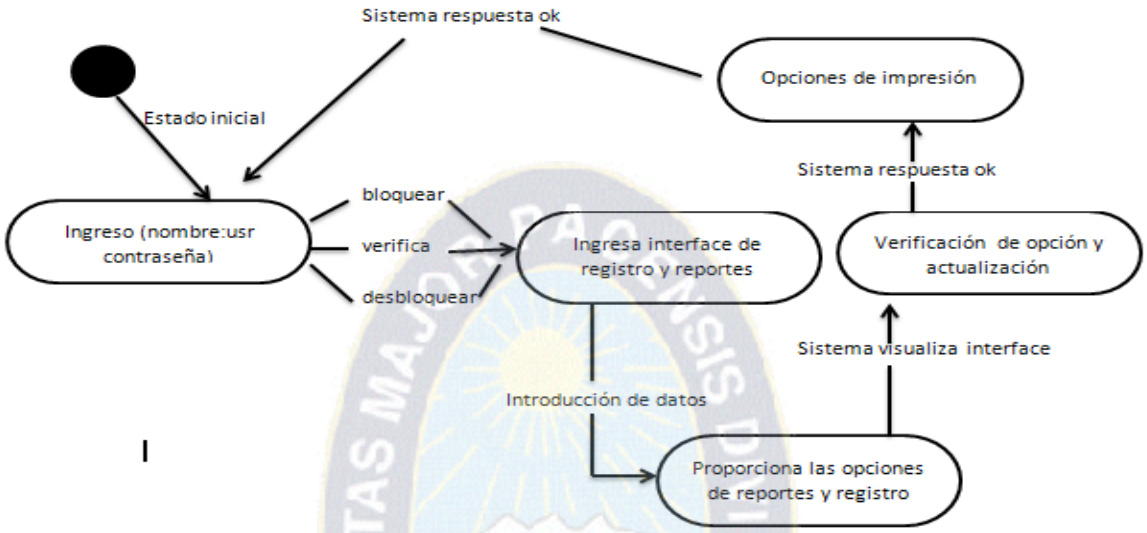
FIGURA 23 (Diagrama de estado: Registro de estudiante**)**



FUENTE (Elaboración Propia**)**

3.2.5.3 REGISTRO DE MENSUALIDADES

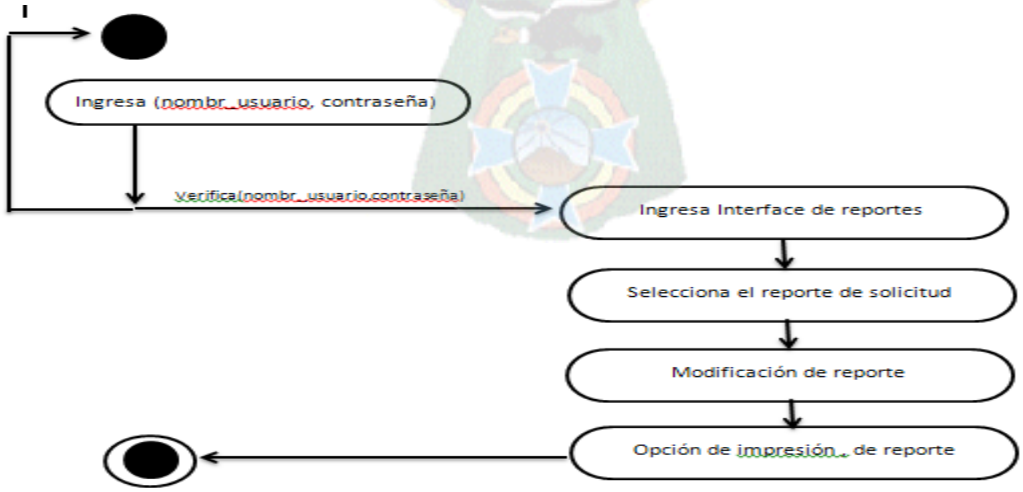
FIGURA 21 (Diagrama de estado: Registro de mensualidades**)**



FUENTE (Elaboración Propia**)**

3.2.5.4 ELABORACIÓN DE REPORTE

FIGURA 21 (Diagrama de estado: elaboración de reporte**)**



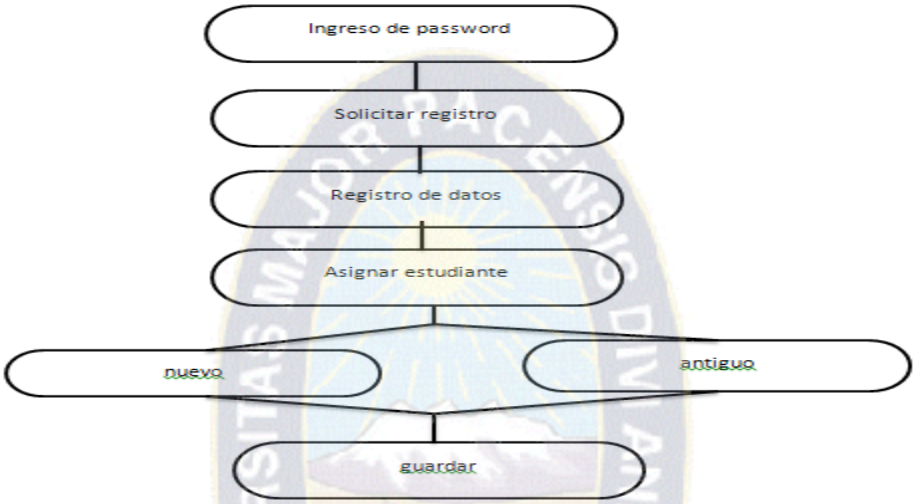
FUENTE (Elaboración Propia**)**

3.2.6 DIAGRAMA DE ACTIVIDADES:

En los diagramas de actividades se explican los procesos actuales

3.2.6.1 REGISTRO DE DATOS

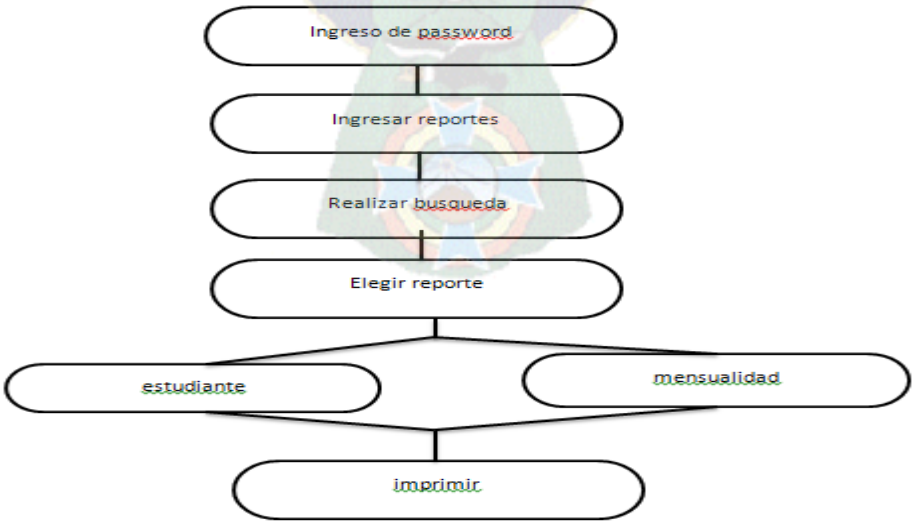
FIGURA 23(Diagrama de actividades: Registro de datos**)**



FUENTE (Elaboración Propia**)**

3.2.6.2 ELABORACIÓN DE REPORTES

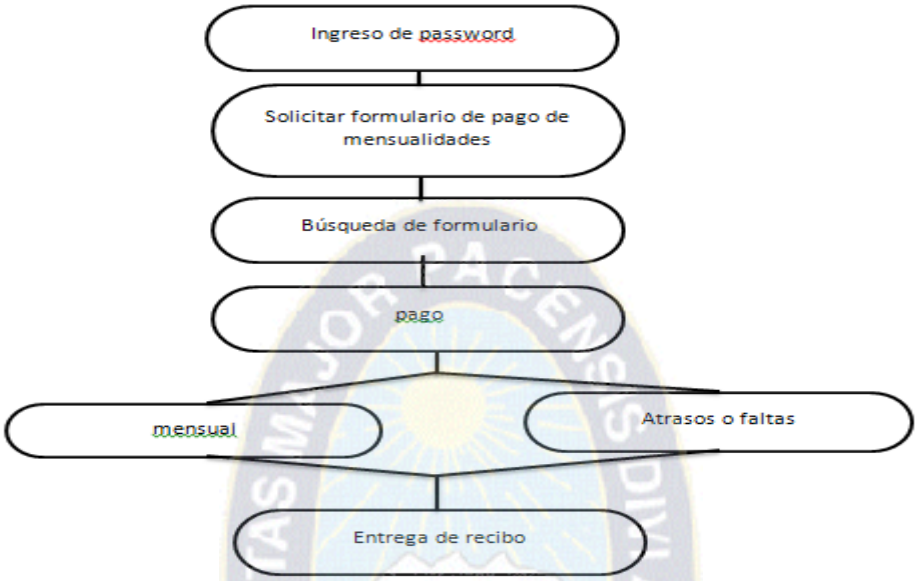
FIGURA 24 (Diagrama de actividades: Elaboración de reportes**)**



FUENTE (Elaboración Propia**)**

3.2.6.3 REGISTRO DE MENSUALIDADES

FIGURA 25 (Diagrama de actividades: registro de mensualidades**)**

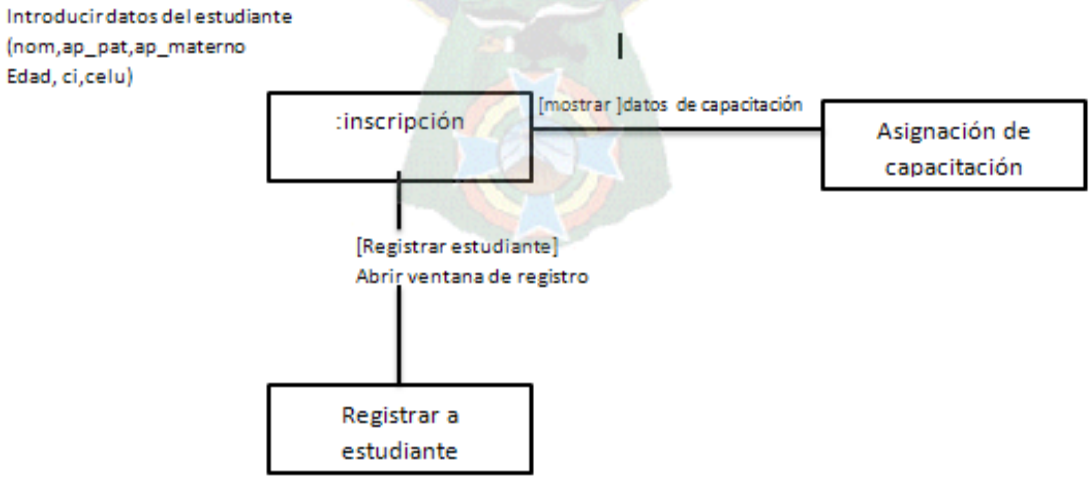


FUENTE (Elaboración Propia**)**

3.2.7 DIAGRAMA DE COLABORACION

3.2.7.1 SOLICITUD DE INSCRIPCION

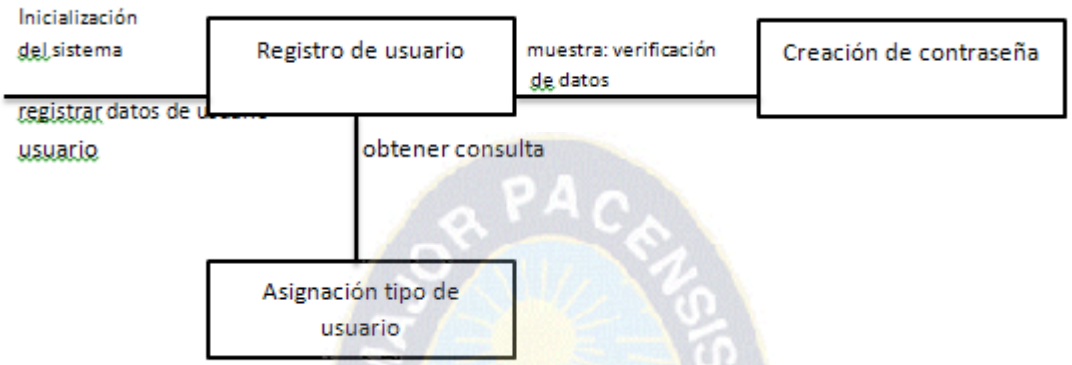
FIGURA 26 (Diagrama de Colaboración: Solicitud de inscripción**)**



FUENTE (Elaboración Propia**)**

3.2.7.2 REGISTRO Y MODIFICACION DE USUARIO

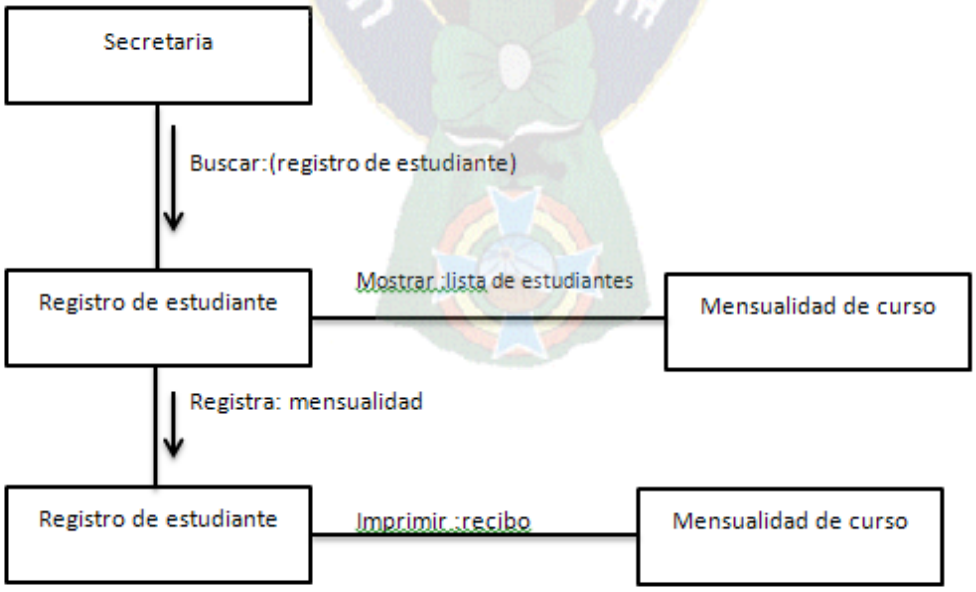
FIGURA 27 (Diagrama de Colaboración: Registro y modificación de usuario**)**



FUENTE (Elaboración Propia**)**

3.2.7.3 REGISTRO DE MENSUALIDADES

FIGURA 28 (Diagrama de Colaboración: Registro de mensualidades**)**



FUENTE (Elaboración Propia**)**

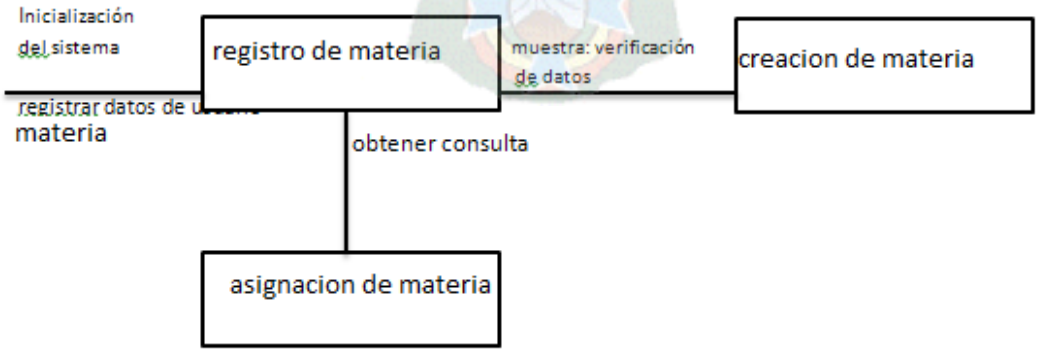
3.2.7.4 ELABORACION DE REPORTES

FIGURA 29 (Diagrama de Colaboración: Elaboración de reportes**)**

FUENTE (Elaboración Propia**)**

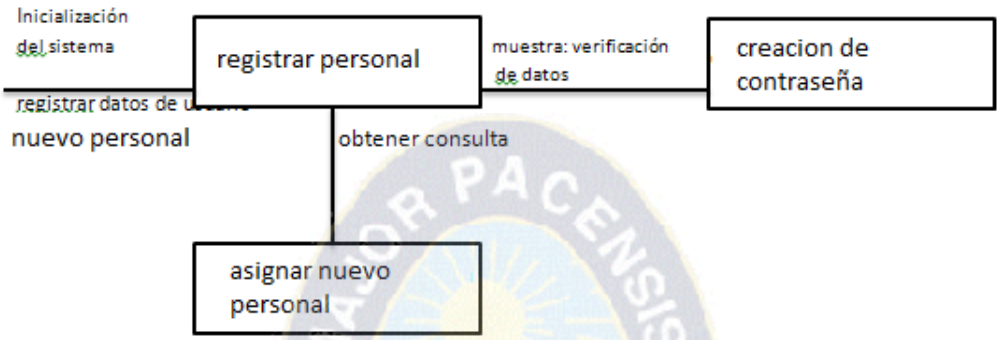
3.2.7.5 ASIGNACION DE MATERIAS

FIGURA 30 (Diagrama de Colaboración: asignar una nueva materia)

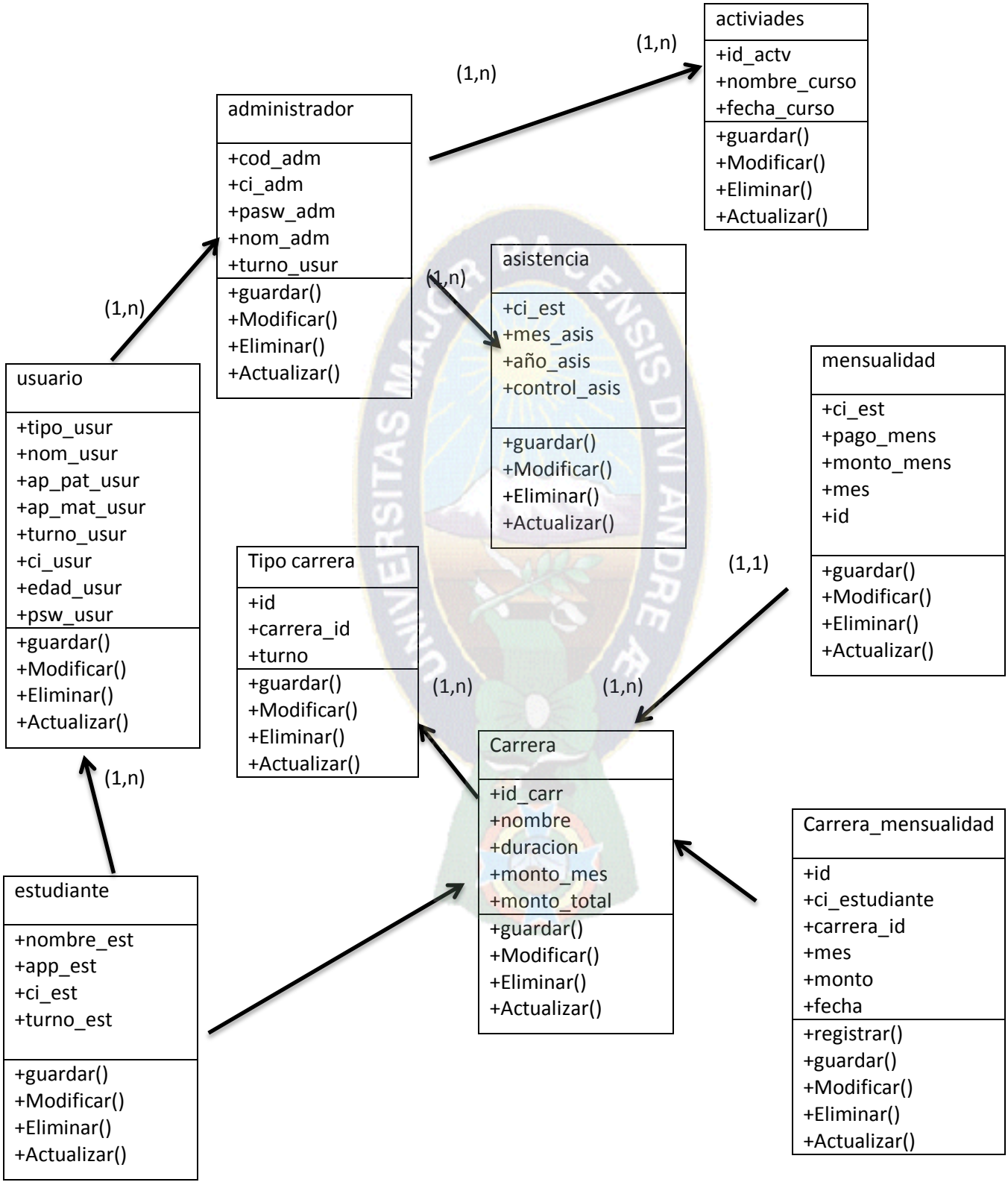


3.2.7.6 ASIGNACION DE PERSONAL

FIGURA 31(Diagrama de Colaboración : Asignar nuevo personal**)**

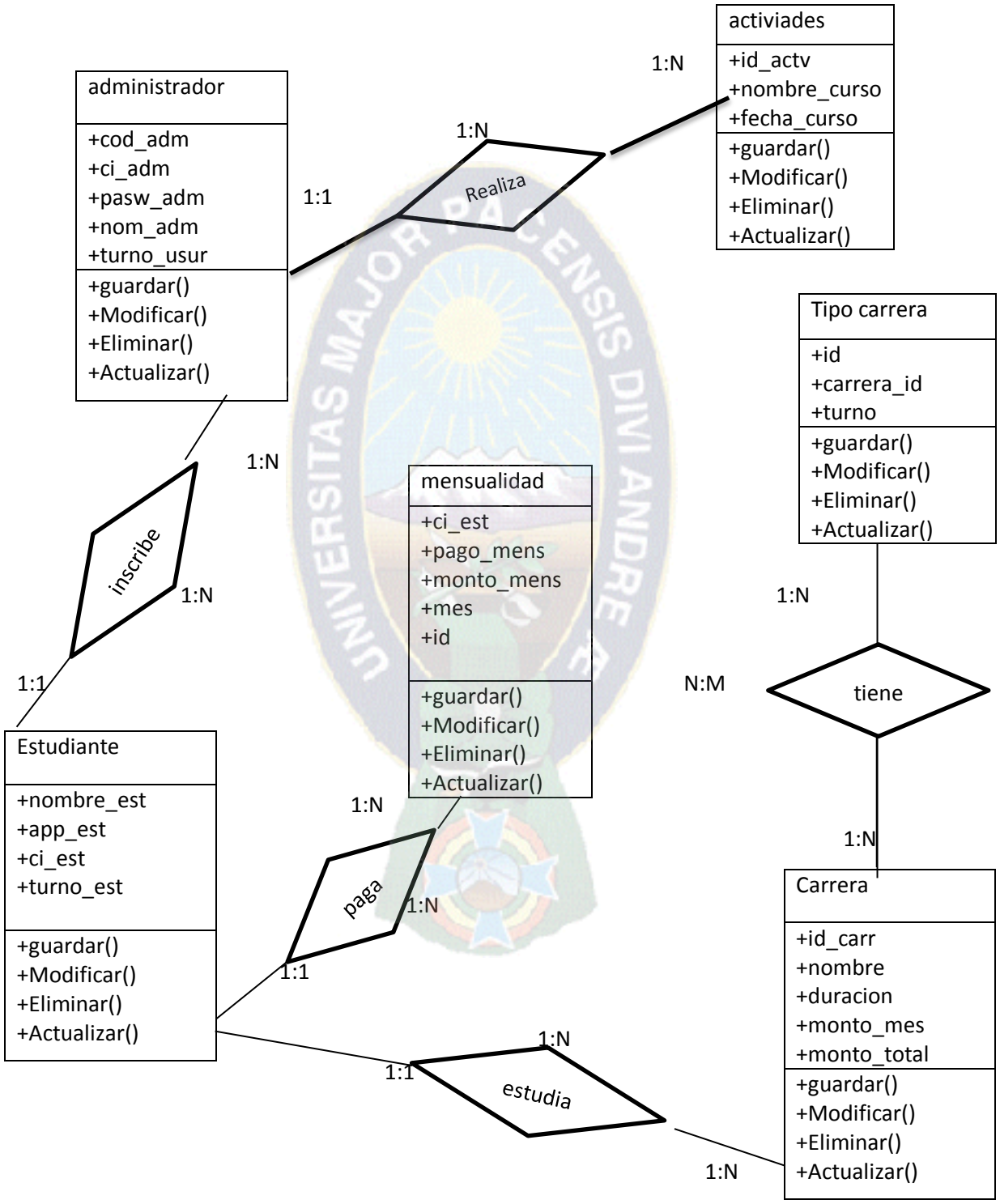


3.2.8 DIAGRAMA DE CLASES



3.2.9 DIAGRAMA ENTIDAD RELACION

Utilizando el diagrama de clases se diseña el diagrama Entidad Relación.

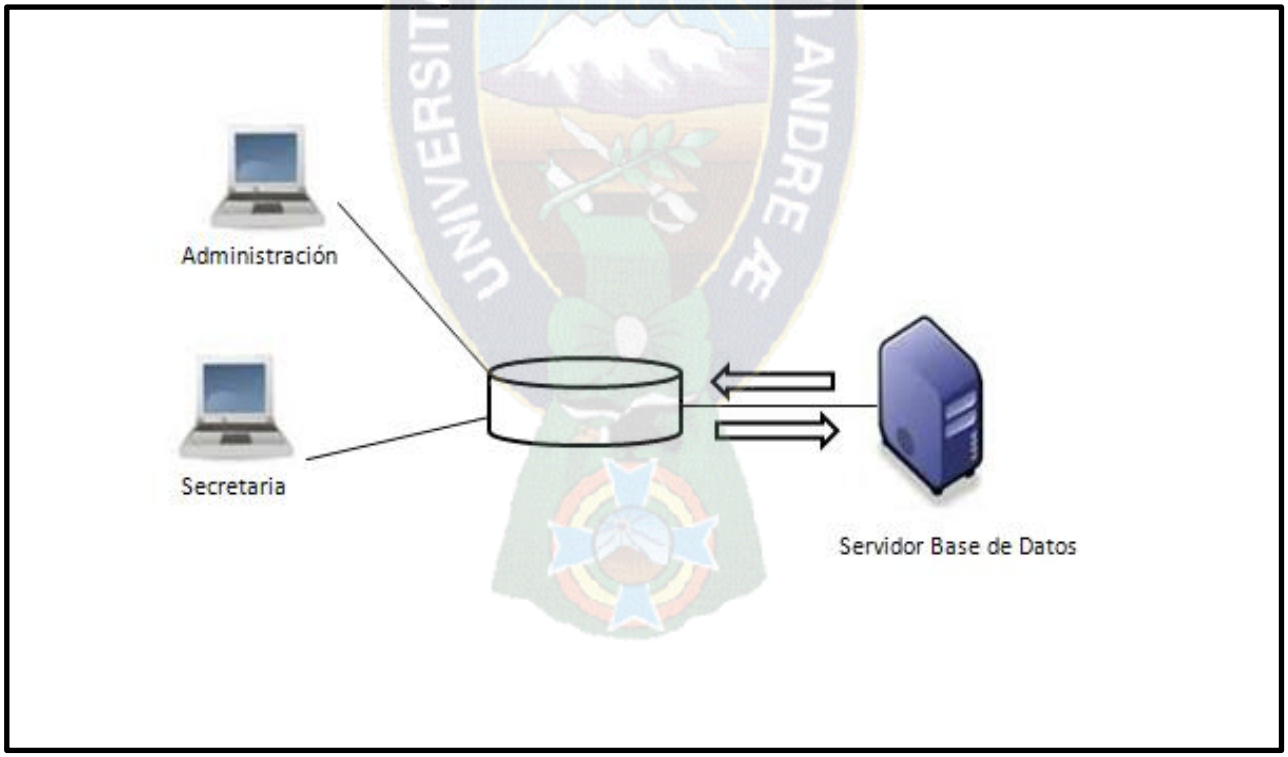


3.3 FASE DE CONSTRUCCIÓN.

En esta fase el foco principal es el desarrollo de componentes y de otras características del sistema que está siendo diseñado. En proyectos que son bastante complejos, varias iteraciones de la construcción se pueden desarrollar en un esfuerzo de dividir dos casos de uso en segmentos manejables que producen prototipos demostrables.

3.3.1 DISEÑO DE ARCUITECTURA DEL SISTEMA

La arquitectura física describe detalladamente al sistema en términos de software y hardware para el verificar los procesos y los usuarios que intervendrán de acuerdo a la configuración de red. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.



3.3.2 DISEÑO FISICO

Se identifican a una tabla los campos, describiendo a cada una. En las siguientes tablas fue extraído desde la base de datos para mostrar el diseño físico del sistema.

Usuarios

Campo	Tipo	Nulo	comentarios
Ci	Varchar(11)	no	Clave primaria
Nombre	Varchar(25)	No	Nombre del estudiante
Ap_paterno	Varchar(25)	No	Apellido paterno del estudiante
Ap_materno	Varchar(25)	No	Apellido materno del estudiante
Foto	Varchar(100)	No	Identificación del estudiante
Dirección	mediumtext	No	Dirección del estudiante
Teléfono	Int(11)	No	Teléfono del estudiante
Correo	Varchar(50)	No	Correo del estudiante
Celular	Int(11)	No	Celular del estudiante
Usuario	Varchar(25)	No	Usuario del estudiante
Clave	Varchar(25)	No	Contraseña del estudiante
tipo	Varchar(20)	no	Tipo de estudiante

Docente

Campo	Tipo	Nulo	comentarios
Id_doc_materia	Varchar(11)	no	Clave primaria
Id_materias	Varchar(25)	No	Nombre del estudiante
Ci_profesor	Int(11)	No	Apellido paterno del estudiante
Ci_alumnos	Int(11)	No	Apellido materno del estudiante
Id_horarios	Varchar(100)	No	Identificación del estudiante
Nota	Int(11)	No	Dirección del estudiante
Fecha_inscripcion	Varchar(25)	No	Teléfono del estudiante

Administrador

Campo	Tipo	Nulo	comentarios
Id_usuarios	Int(11)	no	Clave primaria
Usuarios	Varchar(25)	No	Nombre del usuario
pass	Varchar(25)	No	Password del usuario
Nombre_adm	Varchar(25)	No	Nombre de entrada
Fecha_activad	date	No	Fecha de inicio
Estado	Int(5)	No	Estado activo o inactivo
tipo	Varchar(25)	No	Tipo administrador

Horarios

Campo	Tipo	Nulo	comentarios
Id_horarios	Int(11)	no	Clave primaria
Días	Varchar(25)	No	Días
horas	Varchar(11)	No	Horas

Materias

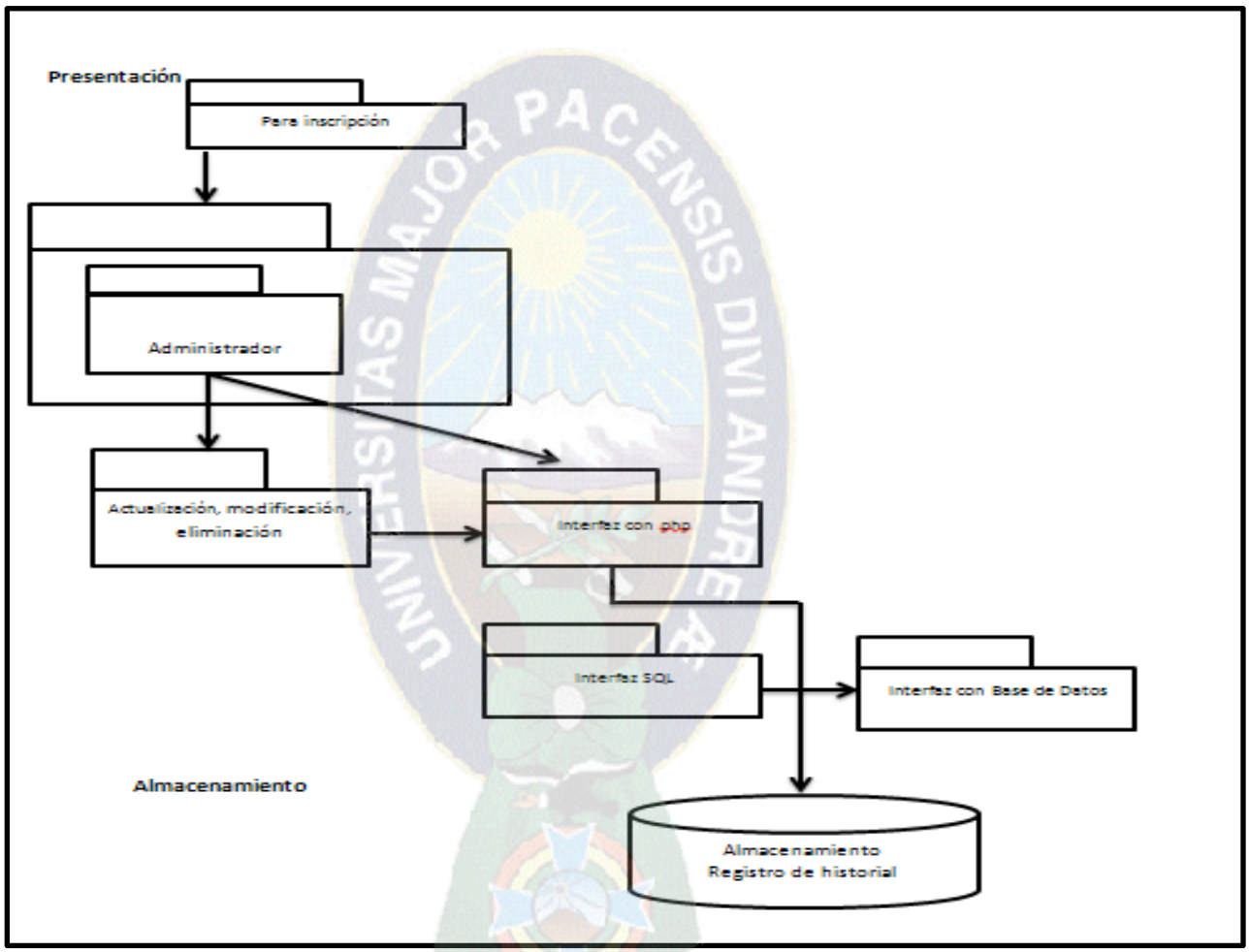
Campo	Tipo	Nulo	comentarios
Id_materia	Varchar(50)	no	Clave primaria
materias	Varchar(25)	No	materias

3.3.3 DIAGRAMA DE PAQUETES.

Permite identificar el proceso de desarrollo del sistema anidado.

3.3.3.1 SOLICITUD DE INSCRIPCION.

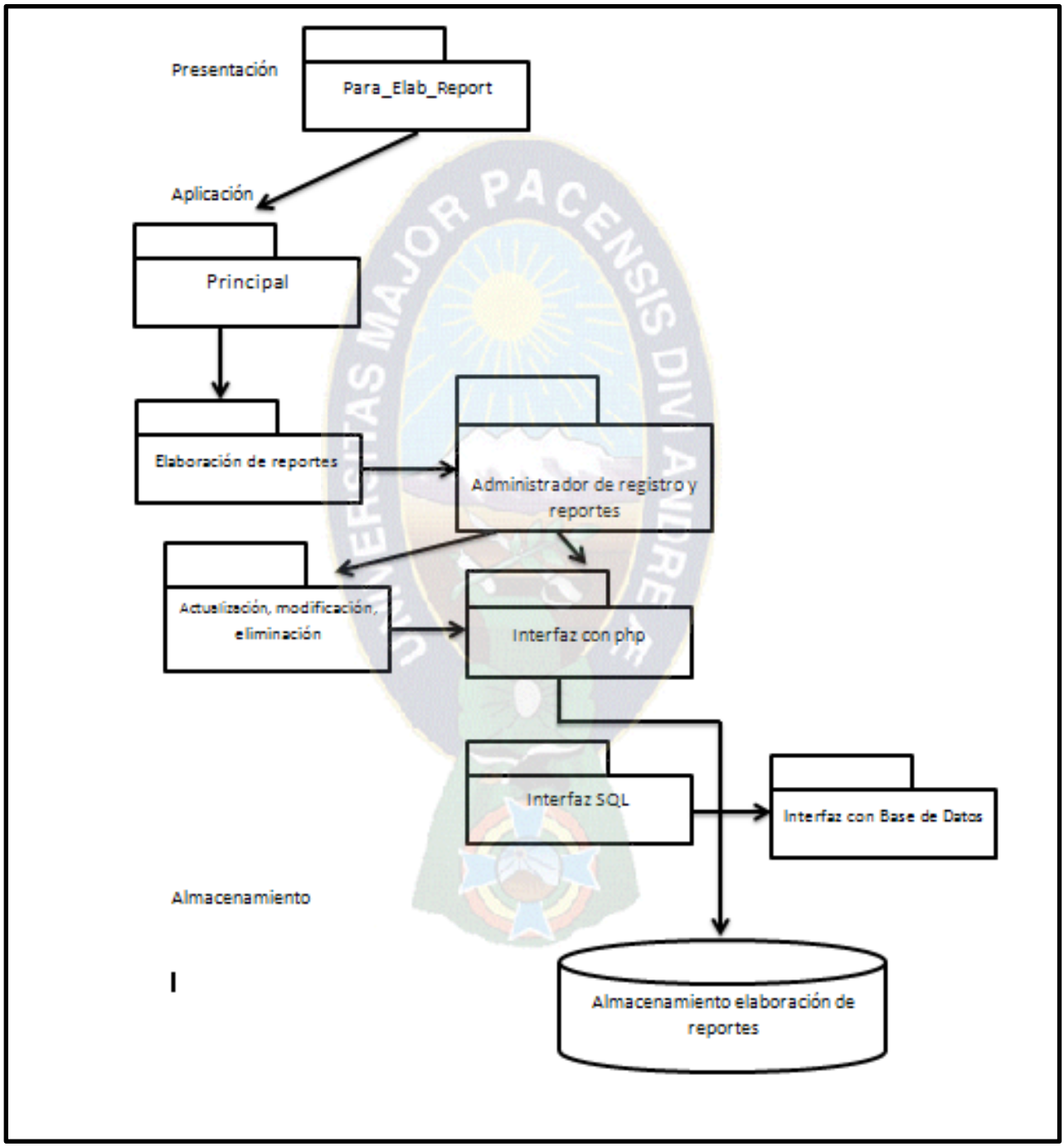
FIGURA 32 (Diagrama de paquetes: Solicitud de inscripción)



FUENTE (Elaboración Propia**)**

3.3.3.2 ELABORACIÓN DE REPORTES

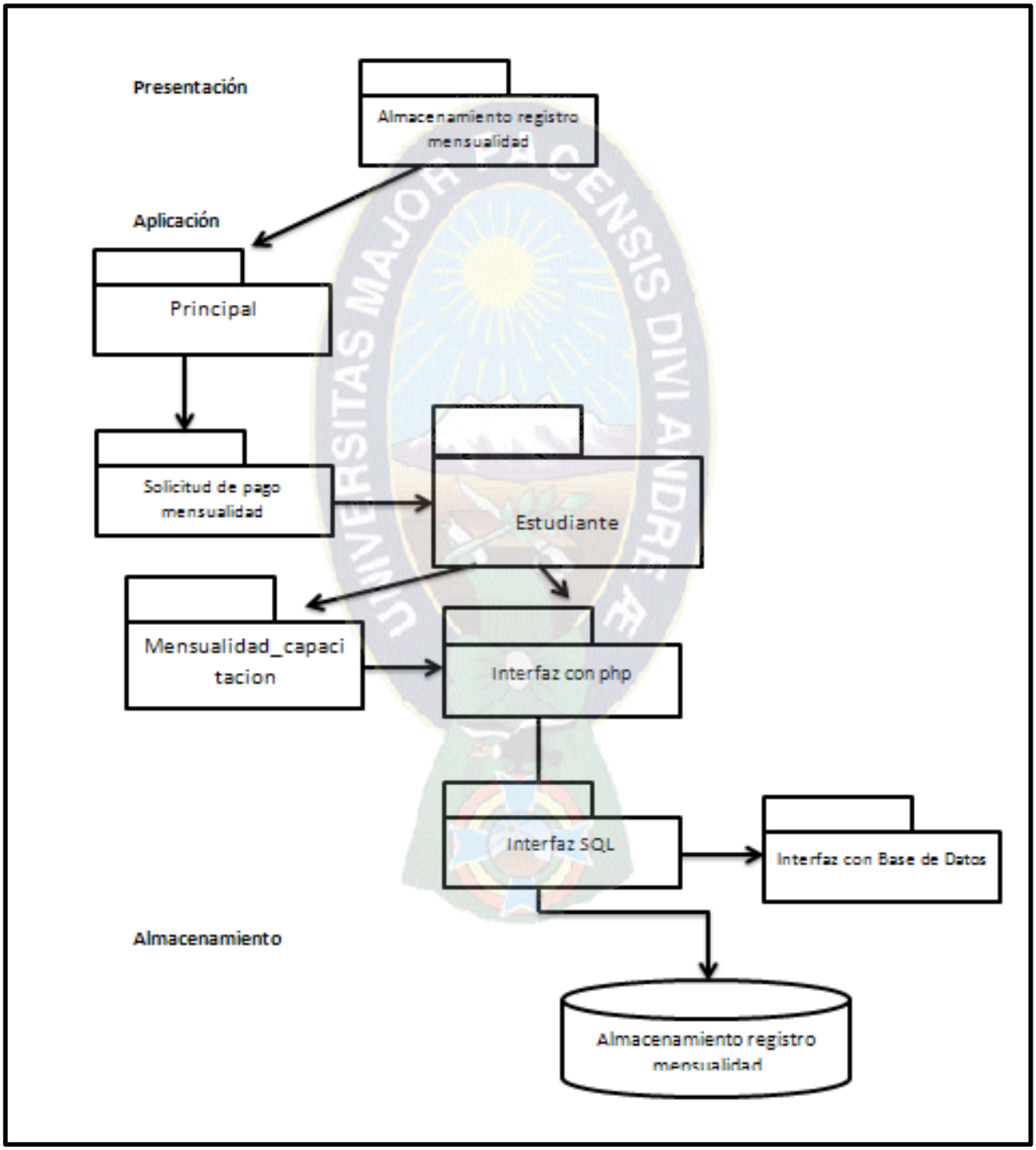
FIGURA 33 (Diagrama de paquetes: Elaboración de reportes)



FUENTE (Elaboración Propia**)**

3.3.3.3 PAGO DE MENSUALIDAD

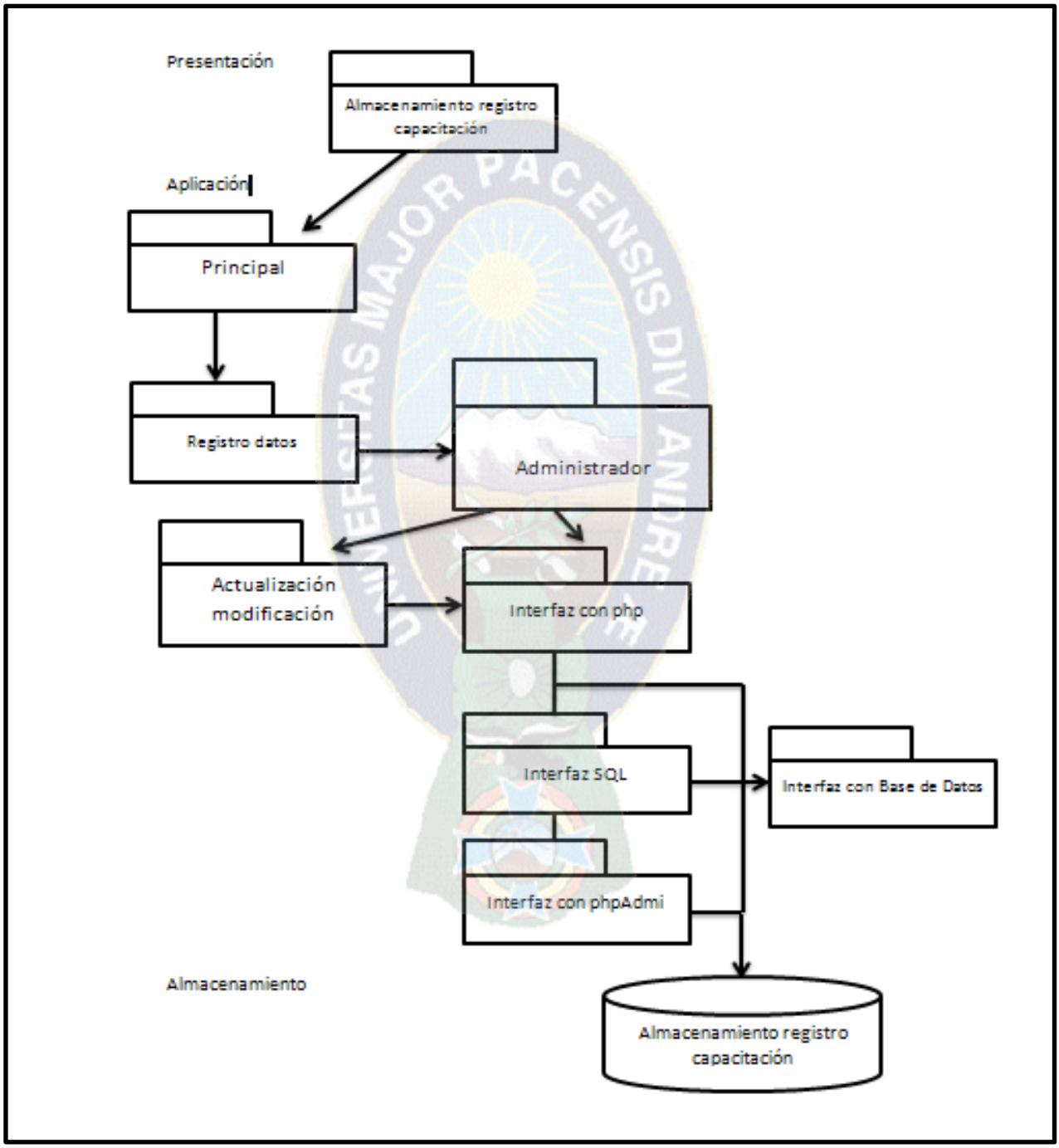
FIGURA 34 (Diagrama de paquetes: Elaboración de reportes)



FUENTE (Elaboración Propia**)**

3.3.3.4 REGISTRO DE DATOS

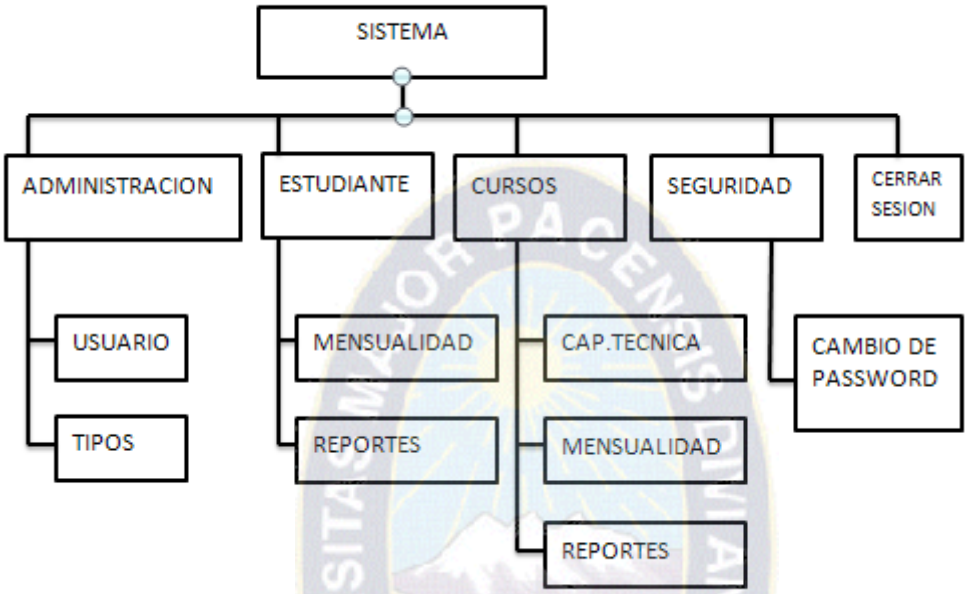
FIGURA 35 (Diagrama de paquetes: Registro de datos)



FUENTE (Elaboración Propia**)**

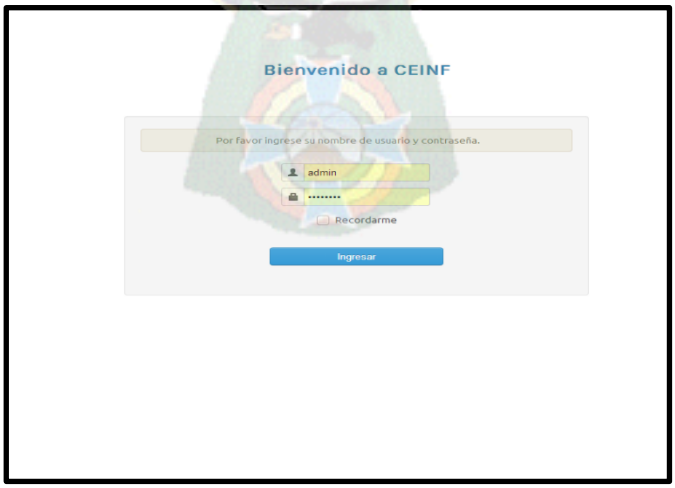
3.3.4 DISEÑO DE INTERFAZ

Se identifican los usuarios que interactúan en el sistema.



El usuario ingresa su identificador y su contraseña. Estos datos serán proporcionados por el administrador del sistema.

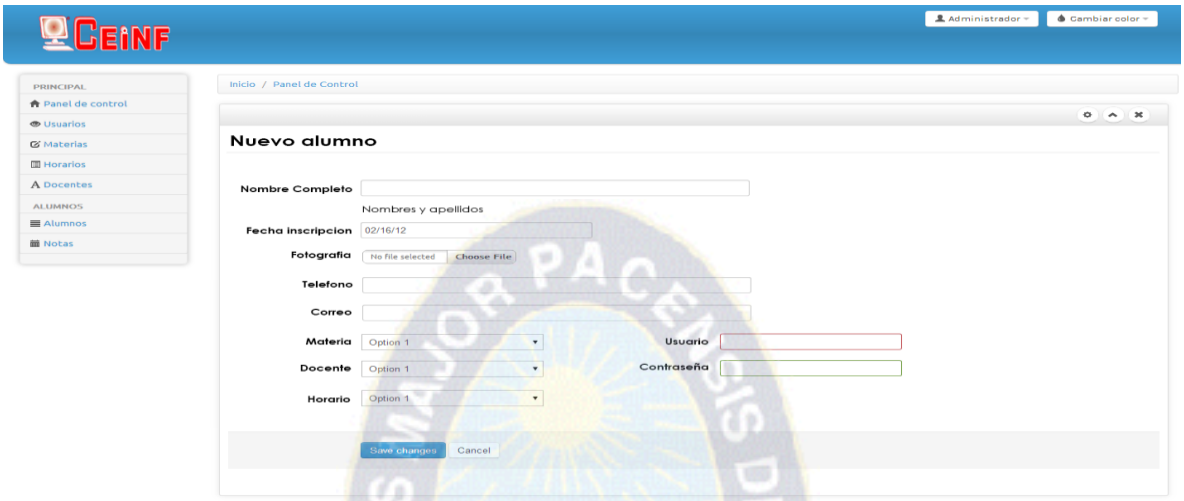
FIGURA 36 (El usuario ingresa su identificador y contraseña**)**



FUENTE (elaboración propia)

Ventana de inscripción esta ventana recibe los datos necesarios para registrar a un alumno.

FIGURA 37 (ingresa los datos necesarios del alumno)



FUENTE (elaboración propia)

Ventana de inscripción el alumno escoge la materia a tomar y el horario.

FIGURA 38 (ingresa los datos necesarios)

FUENTE (elaboración propia)

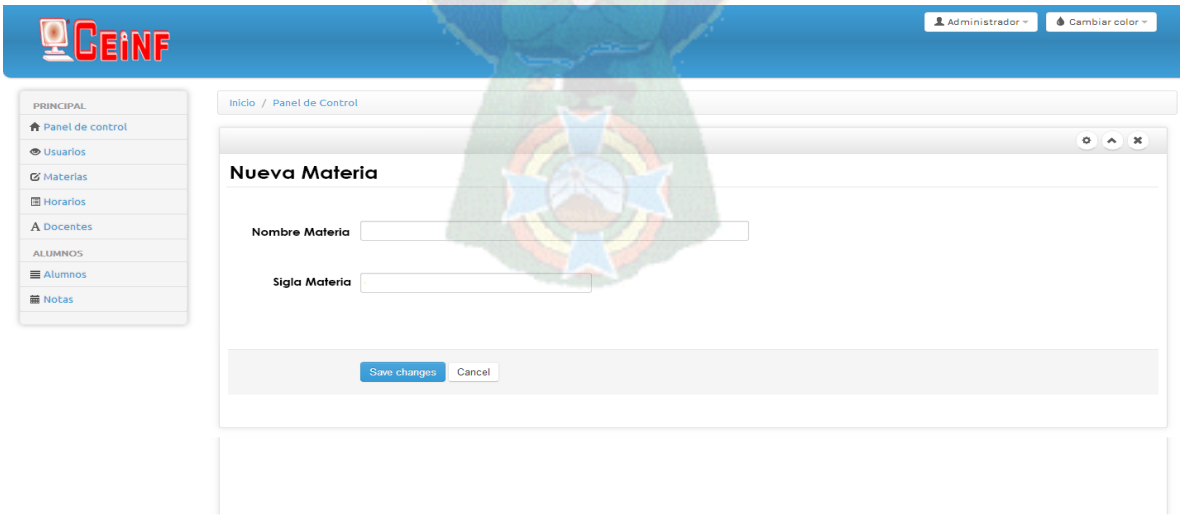
Ventana para crear un nuevo administrador.

Figura 39(Crear un nuevo administrador)

FUENTE (elaboración propia)

Ventana Asignar una nueva materia.

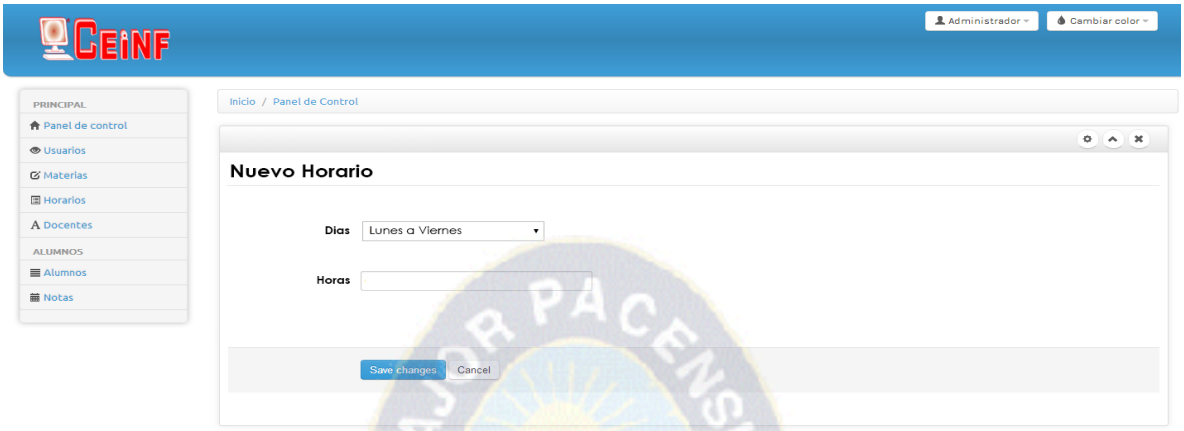
FIGURA 40(asignar una nueva materia)



FUENTE (elaboración propia)

Ventana para asignar horario.

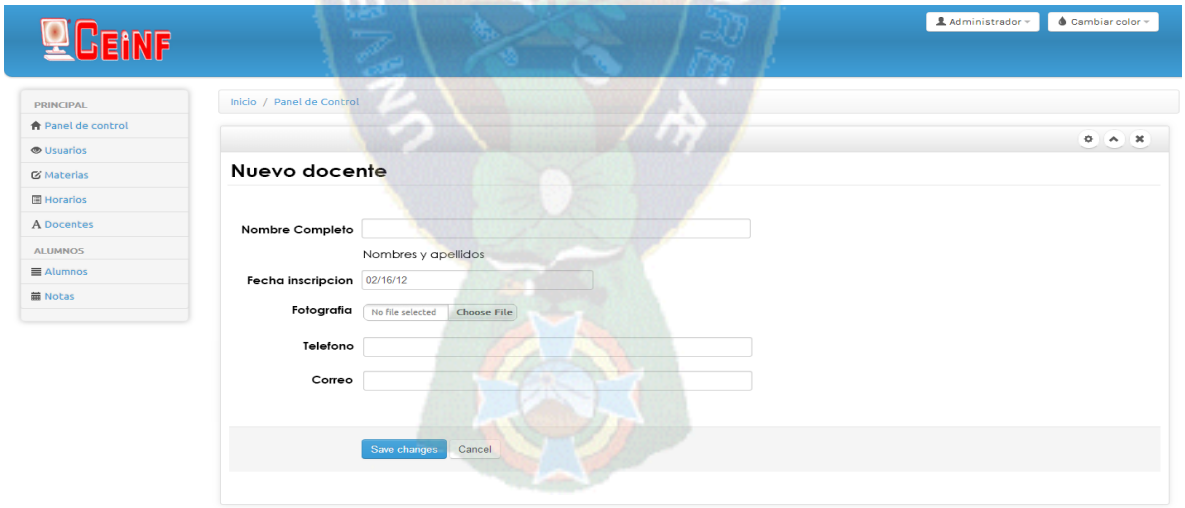
FIGURA 41(asignar un nuevo horario)



FUENTE (elaboración propia)

Ventana para asignar un docente.

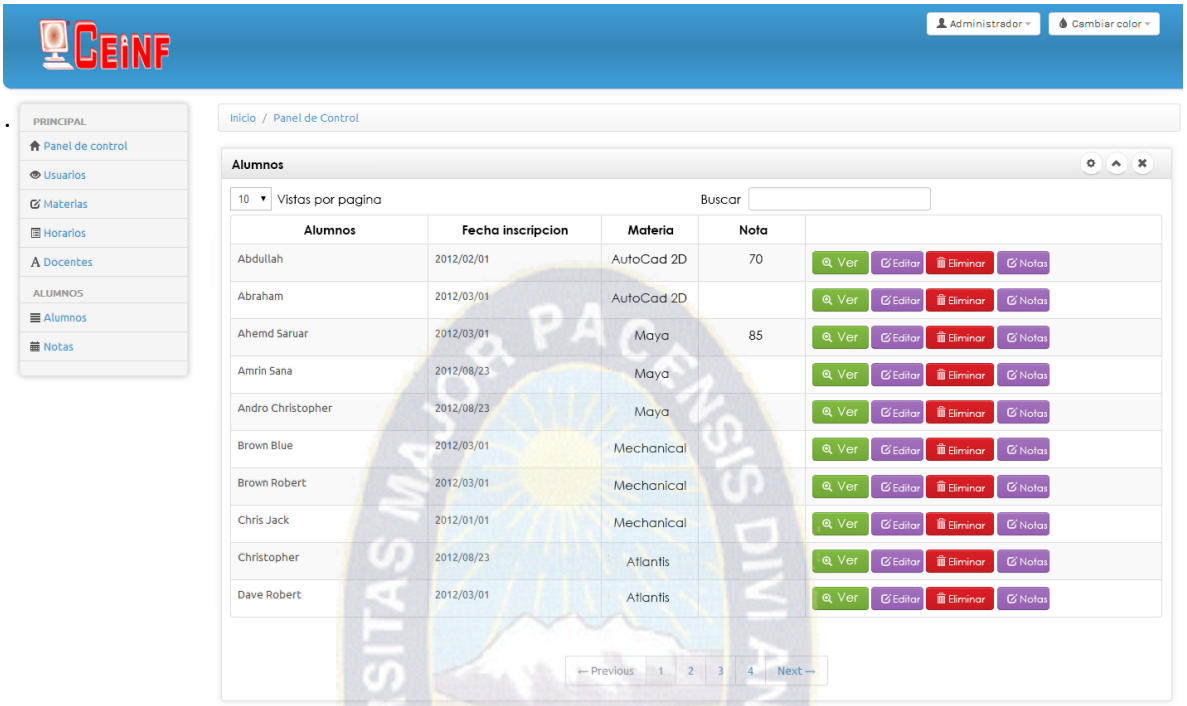
Figura 42(asignar un nuevo docente)



FUENTE (elaboración propia)

Ventana de reportes de alumnos inscritos

FIGURA 43(reportes del alumno)



FUENTE (elaboración propia)

Ventana de notas del estudiante. Esta ventana recibe modificaciones.

FIGURA 44(notas de alumno)

FUENTE (elaboración propia)

3.4 FASE DE TRANSICIÓN.

Esta es la última fase. el objetivo de esta fase es transportar el software desarrollado a los usuarios.

Actividades para la implementación del sistema.

- Instalar y configurar el sistema
- Instalar y configurar el servidor que requiere nuestro software (Apache).
- Instalar y configurar el gestor de base de datos (MySQL)
- Se configuran las máquinas de los usuarios cliente dentro de la institución.
- Se capacita a los administrativos de la institución.

CAPÍTULO 4

4.1 MÉTRICAS DE CALIDAD

La calidad del software La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo

La **ISO 9126** plantea un modelo normalizado que permite evaluar y comparar productos sobre la misma base ISO 9126

El modelo de calidad establecido en la primera parte del estándar, ISO 9126, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas de la siguiente manera:

4.2 FIABILIDAD.

Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período establecido.

Ahora se plantea la siguiente fórmula matemática.

Fiabilidad=1-(número de errores/número de líneas de código)

Fallas identificadas durante el funcionamiento del sistema.

Tiempo de servicio	Peticiones realizadas	Fallas encontradas	Probabilidad de fallo bajo demanda
10 hrs	70	4	0.0286
16hrs	54	2	0.045

Fiabilidad=1-(6/450)

Fiabilidad = 0.9288*100

Así vemos que la fiabilidad es de un 92%

4.3 USABILIDAD.

Un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso.

Se realiza una evaluación del sistema en base a encuestas de un grupo de usuarios.

Pregunta	Ponderación positiva
¿El interfaz de usuario es amigable?	**90%**
¿El manejo del sistema es comprensible y sencillo?	**92%**
¿El sistema satisface todos los requerimientos?	**90%**
¿Los datos de salida son confiables?	**91%**

∑xi/n = suma de ponderaciones = 459

N = cantidad de preguntas 4

Entonces:

U=∑xi/n=365/4=91.25

Así vemos que la usabilidad es de un 91%.

4.4 MANTENIBILIDAD

Es un conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

Así se tiene la siguiente formula:

IMS= $[M_t-(F_c+F_a+F_d)]/M_t$

Donde:

M_t= numero de módulos en la versión actual.

F_c=Numero de módulos en la versión actual que han cambiado

F_a=numero de módulos en la versión actual añadido

F_d=numero de módulos en la versión anterior que se ha borrado.

Entonces

M_t=6 F_c=1 F_a=0 F_d=0

IMS=(6-(1+0+0))/6

IMS=0.83

Por tanto el sistema tiene una mantenibilidad de un 83%

4.5 EFICIENCIA

Son atributos relacionados con la relación entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

EFICIENCIA DEL SOFTWARE=EFICIENCIA/LCC

Dónde:

Eficiencia=disponibilidad*confiabilidad*mantenibilidad*capacidad

LCC=costo de ciclo de vida

La disponibilidad, es una medida frecuente que el sistema está bien y listo para operar, para ello tenemos que la disponibilidad es de un 95%.

La capacidad es de un 90%, está relacionada con la entrega productiva, mide la capacidad del sistema para desempeñar su función.

Así

LCC=costo del ciclo de vida, se considera en 85%.

Reemplazando en la formula tenemos.

Eficiencia=(0.95*0.92*0.83*0.90)/0.85 = 0.77

Así decimos que la eficiencia es de un 77%

4.6 PORTABILIDAD.

Son atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

1-0.1(número de días para portar sistema/número de días para implementar sistema)

Entonces:

Portabilidad=1-(0.5 días/2.5 días)

Portabilidad = 0.80

Así vemos que la portabilidad es de un 80%, puede ser transferido de un entorno a otro.

CAPÍTULO 5

5.1 EVALUACIÓN DE COSTOS Y BENEFICIOS.

En este capítulo se evaluara los costos y beneficios para el sistema.

5.2 MÉTODO COCOMO.

El Modelo Constructivo de Costos es un modelo matemático de base empírica utilizado para estimación de costos de software.

La ecuaciones que se utilizan en los tres modelos son:[2]

- $E = a(Kl)^b * m(X)$, en persona-mes
- $Tdev = c(E)^d$, en meses
- $P = E/Tdev$, en personas

E = es el esfuerzo requerido por el proyecto, en persona-mes

Tdev = es el tiempo requerido por el proyecto, en meses

P = es el número de personas requerido por el proyecto

a, b, c y d = son constantes con valores definidos en una tabla, según cada submodelo

Kl = es la cantidad de líneas de código, en miles.

m(X) = Es un multiplicador que depende de 15 atributos.

Se tiene un total de líneas de código Klcd= 8363

Asi

Kl=Klcd/1000

Kl=8363/1000 = 8.363 Kl

En este caso es el tipo orgánico será más apropiado ya que el número de líneas de código no supera los 50 Kl

Se utiliza la siguiente tabla para obtener una primera aproximación rápida del esfuerzo, y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costes

MODO	**A**	**b**	**c**	**d**
Orgánico	3.2	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Y también debemos de hallar la variable FAE es el factor de ajustes de esfuerzo la cual se obtiene mediante la multiplicación de los valores evaluados.

	VALOR					
	Muy bajo	bajo	nominal	alto	Muy alto	Extra alto
fiabilidad	0.75	0.88	1.00	1.15	1.40	-
Tamaño de base de datos	-	0.94	1.00	1.08	1.16	-
Complejidad	0.70	0.85	1.00	1.15	1.30	1.65
Restricciones de tiempo de ejecución	-	-	1.00	1.11	1.30	1.66
Restricciones de memoria virtual	-	-	1.00	1.06	1.21	1.56
Volatilidad de la maquina virtual	-	0.87	1.00	1.15	1.30	-
Tiempo de respuesta	-	0.87	1.00	1.07	1.30	-
Capacidad de análisis	1.46	1.19	1.00	0.86	0.71	-
Experiencia en la aplicación	1.29	1.29	1.00	0.91	0.82	-
Calidad de los programadores	1.42	1.17	1.00	0.86	0.70	-
Experiencia en la maquina virtual	1.21	1.10	1.00	0.90	-	-
Experiencia en el lenguaje	1.14	1.07	1.00	0.95	-	-
Técnicas actualizadas de programación	1.24	1.10	1.00	0.91	0.82	-
Utilización de herramientas software	1.24	1.10	1.00	0.91	0.83	-
Restricciones de tiempo de desarrollo	1.23	1.23	1.00	1.04	1.10	-

m(X)=1.15*1.00*0.85*1.00*1.00*1.00*1.07*0.86*0.82*0.70*1.00*0.95**1.00*0.91*1.23

=0.54901094

Justificación de valores.

- Fiabilidad.-si se produce un fallo por alguna mensualidad o registro , puede ocasionar perdida de información (valoración alta)
- Tamaño de base de datos.-la base de datos es de tipo estándar
- Complejidad.-la aplicación no realizara cálculos complejos
- Restricciones de tiempo de ejecución.- en los requerimientos se exige un alto rendimiento
- Restricciones de memoria virtual.- no hay restricciones
- Volatilidad de la máquina virtual.- se usaran sistemas de la familia a Windows, como XP
- Tiempo de respuesta.-deberá ser interactivo con el usuario
- Capacidad del análisis.-capacidad alta , debido a la experiencia en análisis en proyecto similar
- Experiencia en la aplicación.-se tiene cierta experiencia en aplicaciones de esta envergadura
- Calidad de los programadores.-teóricamente deberá tenerse una capacidad muy alta por la capacidad de un programador
- Experiencia en la máquina virtual.-con Windows XP la experiencia es a nivel usuario
- Experiencia en el lenguaje.-es alta, dado que se controlan las nociones básicas y propias del proyecto.
- Técnicas actualizadas de programación.-se usaran prácticas de programación mayormente convencional
- Utilización de herramientas software.- se usaran herramientas estándar que no exigirán formación.
- Restricciones de tiempo de desarrollo.-existen pocos límites de planificación .

Calculo del esfuerzo del desarrollo

$E = a(Kl)^b * m(X)$, en persona-mes

E= 3.2*(8.363)^1.05*0.549

E=16.33

Calculo tiempo de desarrollo

$Tdev = c(E)^d$, en meses

T=2.5*12.25 ^0.38

T=7.22 meses

Productividad

$P = E/Tdev$, en personas

P=16.33/7.22

P=2.261 personas

Estimación de costos.

Al tener los datos definidos, se realiza el supuesto estimación de costo.

La hora por trabajar =3$

día =3*8= 24$

Semana = 24*5= 120$

Al mes 120*4 semanas = 480 * 2 personas = 960 $/mes.

Calculamos 960$ * 7 meses = 6720 $, el costo del producto elaborado en 7 meses.

CAPITULO 6

SEGURIDAD DEL SISTEMA

La seguridad informática o seguridad de tecnologías de la información es el área de la informática que se enfoca en la protección de la infraestructura computacional y todo lo relacionado con esta y, especialmente, la información contenida o circulante.

6.1 SEGURIDAD FISICA.

Comprende la parte tangible, de los recursos que permite controlar el acceso a los equipos de computación.

El instituto CEINF:

Tiene una oficina de trabajo seguro, con 5 personas que trabajan en la semana, una persona con su respectiva computadora en red, una impresora que comparten.

6.2 PROTECCION FISICA DE ACCESO A LAS REDES.

Es necesario tener seguridad en este punto, y para proteger a los equipos de una red de área local y el software que reside en ellos, se deben tomar medidas que impidan que usuarios no autorizados puedan acceder

6.3 SEGURIDAD LOGICA.

La seguridad del software y los sistemas, consiste en restringir el acceso a programas y archivos, así como la del acceso ordenado y autorizado de los usuarios a la información.

La información y los recursos deben estar protegidos. Ya que el internet es un factor primordial en la comunicación y también un evidente riesgo a los servicios de información disponibles.

6.3.1 SEGURIDAD DE APLICACIÓN.

Las aplicaciones web están en parte definidas por su uso del protocolo HTTP como medio de comunicación entre cliente y servidor. Este protocolo:

Es simple y basado en ASCII - no se requiere gran esfuerzo para generar pedidos y descifrar el contenido de las respuestas.

Utiliza un puerto TCP bien conocido – de poco sirve un firewall para proteger una aplicación si tiene que admitir el tráfico a través del puerto 80.

6.3.2 SEGURIDAD EN LA BASE DE DATOS.

Es necesario tener seguridad en este punto ya que se cuenta con personas que ingresan a información muy importante que pueden realizar modificaciones, adiciones hasta eliminar algunos datos registrados en la base de datos por tanto se usa una criptografía para tener mayor seguridad.

6.3.3 SEGURIDAD DE SISTEMA OPERATIVO

La seguridad de los Sistemas Operativos es solo una pequeña parte del problema total de la seguridad en los sistemas de computación, pero éste viene incrementándose en gran medida.

Generalmente se dividen responsabilidades, de esta manera un operario no debe conocer la totalidad del sistema para cumplir con esas responsabilidades.

CAPITULO 7

CONCLUSIONES Y RECOMENDACIONES.

7.1 CONCLUSIÓN.

Para desarrollar el sistema es necesario conocer las actividades que realiza la empresa de forma clara y precisa, es por esto que las entrevistas y colaboraciones con las personas encargadas se hacen indispensables.

Al concluir el presente proyecto, se cumplió los objetivos específicos proporcionando información óptima y confiable, centralizada y automatizada los procesos académicos.

Es así que el sistema realizara de manera eficiente los procesos de seguimiento académico e inscripciones de los estudiantes registrando sus datos personales de forma automática y segura en la base de datos el cual permitirá a los administrativos realizar los procesos con mayor eficiencia y facilidad.

Entre los puntos principales se llega a las siguientes especificaciones.

- Se automatizo el registro de estudiantes y del personal
- La base de datos fue construida y estructurada.
- Se elaboró el interfaz de usuario, que es fácil acceso a la información y hace posible las consultas ofreciendo resguardo de seguridad de la base de datos.
- Establece controles apropiados de seguridad y protección de información de datos, previa verificación de usuarios.
- Se cuenta con una base de datos que contiene información trascendental para el desempeño de sus actividades en el área académica.

Es así que al haber realizado los puntos principales se pudo organizar la información y optimizar el tiempo, en los procesos de administración de la institución Ceinf.

Es así que el software de seguimiento académico ayudara a solucionar las tareas , en un tiempo más óptimo.

7.2 RECOMENDACIONES.

El proyecto también sirve como base para la implementación de otra aplicación distinta para el control y seguimiento académico, también podemos recomendar los siguientes puntos.

- Realizar el mantenimiento del sistema en periodos de acuerdo a las políticas y normas de la institución.
- La información nueva introducida al sistema al sistema deberá ser previamente revisada, para que los reportes emitidos por el sistema sean fiables, para los usuarios que lo requieran.
- Es recomendable que se realice copias de seguridad cada año para evitar la perdida de información.

Se debe coadyuvar los procesos administrativos en términos de sistematización, para toma de decisiones eficientes y óptimas.

BIBLIOGRAFIA.

- [Larman 1999] Larman 1999 ”UML y Patrones ” 1ra edición Editorial Pretince Hall, mexico
- [Pressman 2003] Pressman R.S. 2003 ”Ingenieria de Software” 5ta Edición Editorial Mc Graw Hill, Mexico
- [fcoGil 2001] Fco. Javier gil rub 2001 “Sitios Web” 2da Edición Editorial Mc Graw Hill, Madrid.
- [Larman 1999]Larman 1999 “UML y Patrones ” 1ra Edición, Editorial Pretince Hall, Mexico.
- [Jacobson 2000] Jacobson,G. Booch, J Rumbaugh, El proceso unificado de desarrollo de software, edición, Madrid - España ,2000
- [Kendall , 1997]Kendall, Keneth E: Kendall, Julie E.”analisis y diseño de sistemas” tercera edición Prentice Hall Hispanoamericana SA 1997

Internet

- http\\:www.wikipedia.com\seguridad_informatica.htm
- http\\:www.softwareseguridad.com
- http\\:www.monografias.com RUP
- http\\:www.reifer.com\cocomo_edd.htm
- http\\:www.monografias.com\trabajos\calidad-software

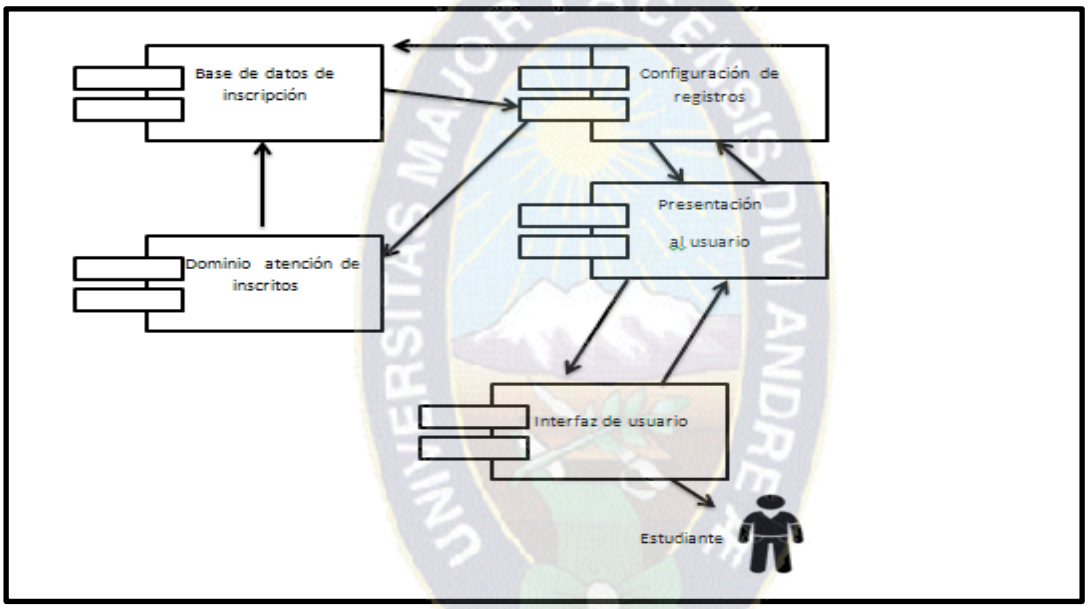
- **ANEXOS.**

DIAGRAMA DE DESPLIEGUE

Es el proceso de casos que se desarrolla dentro de la institución.

SOLICITUD DE INSCRIPCION

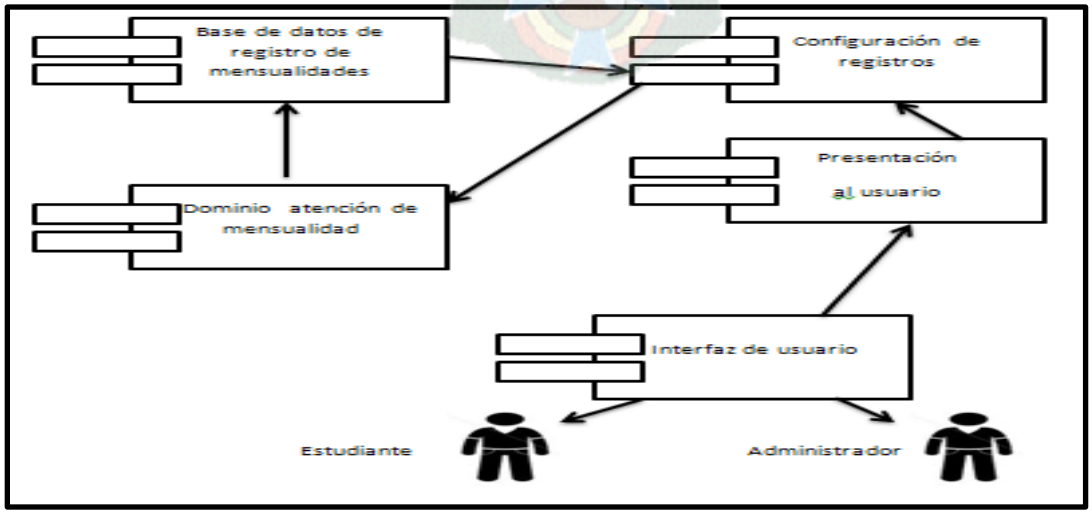
(Diagrama de despliegue: Solicitud de inscripción**)**



FUENTE (Elaboración Propia**)**

RECEPCION Y REGISTRO DE MENSUALIDADES

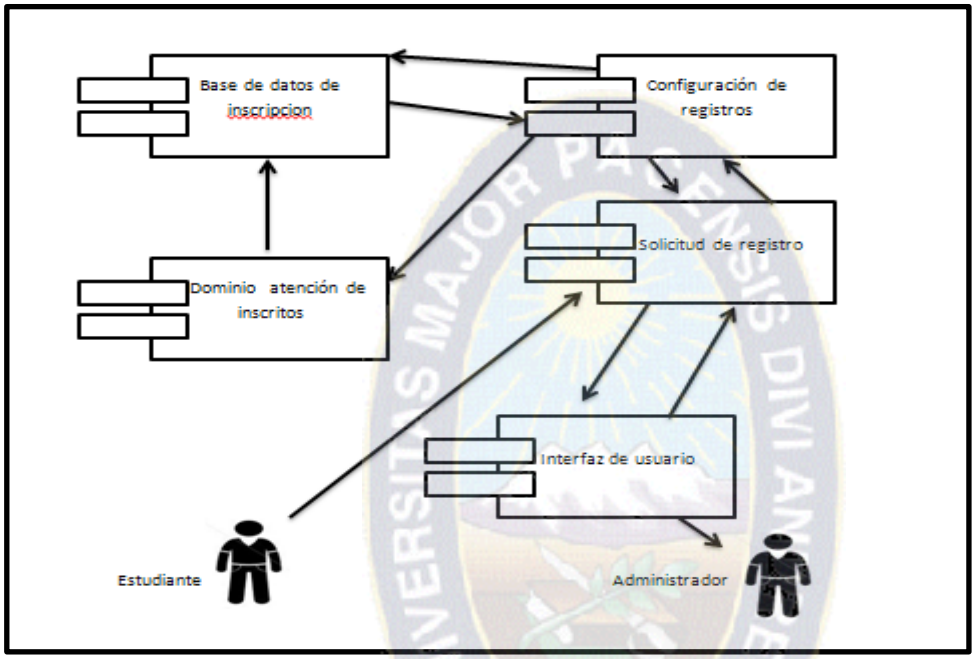
(Diagrama de despliegue: recepción y registro de mensualidades**)**



FUENTE (Elaboración Propia**)**

REGISTRO DE INSCRIPCION

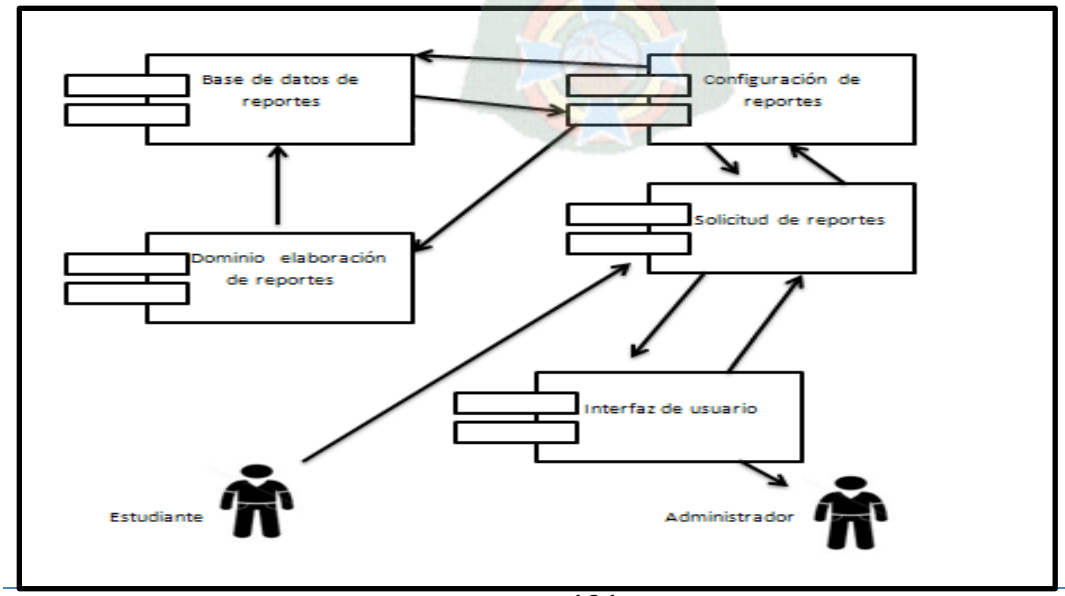
(Diagrama de despliegue: registro de inscripción **)**



FUENTE (Elaboración Propia**)**

ELABORACION DE REPORTES

(Diagrama de despliegue: elaboración de reportes)



FUENTE (Elaboración Propia**)**

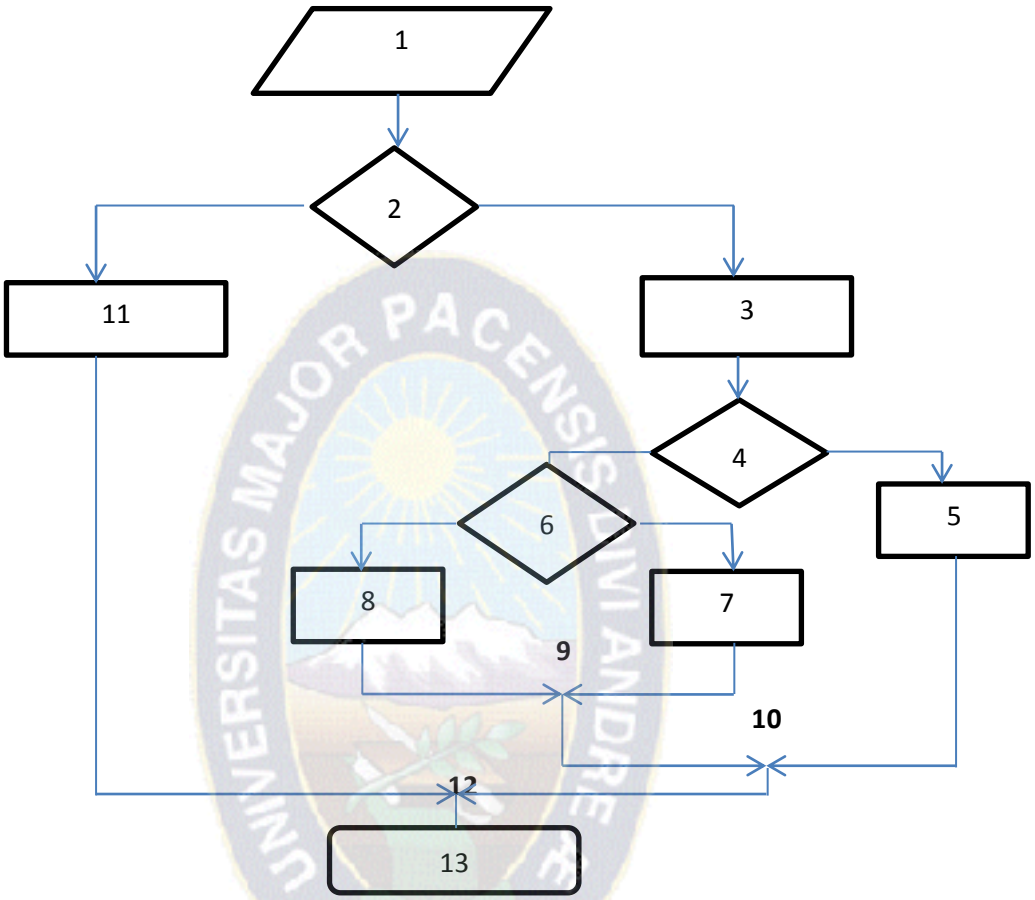
PRUEBA DE CAJA BLANCA.

Esta prueba se la ha realizado utilizando los pasos de prueba de caja blanca y complejidad ciclomática.

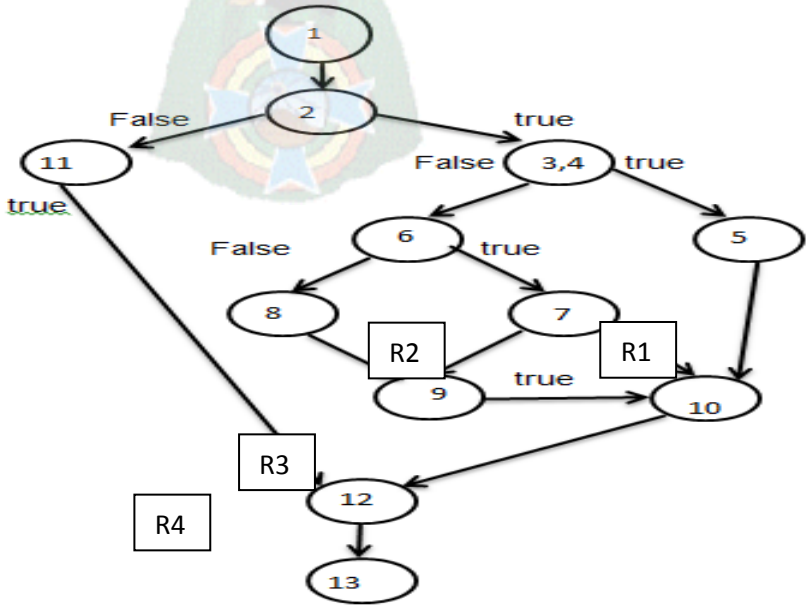
Se realizó la obtención del código del primer formulario de ingreso al sistema desarrollado en PHP

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$hostname_ceinf = "localhost";
$database_ceinf = "ceinf";
$username_ceinf = "root";
$password_ceinf = "";
$ceinf = mysql_pconnect($hostname_tienda, $username_tienda, $password_tienda) or
trigger_error(mysql_error(),E_USER_ERROR);
$_session["act"]=$usuario
$_session[tipo]=$dato[tipo_adm]
if(&dato[tipo_adm]==admi)
{
 header("location:index.php")
}
else
{ if(($dato[tipo_adm]== persona) )
  header("location:index.php")
else
{ if($dato[tipo_adm]== estudiante)
 header("location:index.php") }
 }
 }
 else
 { echo"Error: el usuario no existe"
   header("Location:index.php") }
?>
```

(Diagrama De Flujo De Código Fuente)



(Diagrama Prueba De Ruta Grafica)



Hallar la complejidad Ciclomática.

V(G) es igual al número de regiones del grafo de flujo.

V(G)={R1,R2,R3,R4}=5

V(G) de un grafo de flujo G, se define como.

V(G)=Aristas-Nodos+2

V(G)=15-12+2

A partir de un valor de la complejidad ciclomática obtenemos el número de caminos independientes, que nos dan un valor límite para el número de pruebas que se tiene que diseñar.

R1(G)={1,2,3,4,5,10,12,13}

R2(G)={1,2,3,4,6,7,10,12,13}

R3(G)={1,2,3,4,6,8,9,10,12,13}

R4(G)={1,2,11,12,13}

Documentos