UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

SISTEMA DE GESTIÓN Y CONTROL DE ACTIVOS FIJOS UNIVERSIDAD PRIVADA FRANZ TAMAYO

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERIA DE SISTEMAS INFORMÁTICOS

AUTOR: Luis Fernando Saisa López

TUTOR: Lic. Mario Loayza Molina M.Sc.

REVISOR: Lic. Luisa Velásquez López M.Sc.

LA PAZ – BOLIVIA

2007

DEDICATORIA

A Dios, por estar presente en todo momento de mi vida.

A mis queridos padres Martín y Elena por su apoyo incondicional y por estar siempre conmigo.

A mi hermano Victor, quien siempre estuvo conmigo, apoyándome en cada momento y alentándome para seguir adelante.

A mis hermanas Maria Elena y Magaly, quienes siempre estuvieron con migo brindándome todo su apoyo.

Luis Fernando

AGRADECIMIENTOS

Deseo expresar mi agradecimiento en primer lugar a mi tutor, Lic. Mario Loayza Molina M.Sc. por haberme asesorado y guiado en cada etapa del proyecto, brindándome apoyo y guía para la conclusión del mismo.

A mi docente revisor, Lic. Luisa Velásquez López M.Sc., por su apoyo incondicional, por guiarme y brindarme valiosas sugerencias durante el transcurso del desarrollo del proyecto.

Al Lic. Franc Barrientos Jaldin Jefe de la Dirección de Sistemas Informáticos y Redes-Unifranz, por su apoyo incondicional en el desarrollo del proyecto.

Al personal de la Dirección de Activos Fijos de la Universidad Privada Franz Tamayo quienes me brindaron su colaboración y orientación en el desarrollo del presente proyecto.

A los señores Fernando y Willy, quienes me apoyaron con el material bibliográfico y sobre todo por su amistad.

A mis amigos, Johnny, Vania Mercedes, Gladys, Maria Luisa, Miriam, Adolfo, Ana, Franolig, Juan Carlos y Daniel; con quienes compartí bellos momentos.

Luis Fernando

RESUMEN

La Universidad Privada Franz Tamayo, es una institución de formación superior, cuya misión es formar hombres y mujeres creativos, responsables y con principios éticos, capaces de aplicar conocimientos técnicos, científicos y humanos en beneficio del país.

El presente proyecto propone mejorar la administración de los Activos Fijos de la universidad, pues los mismos son importantes y primordiales en la formación de los estudiantes. El control que se le da en la Dirección de Activos Fijos no es el apropiado, ya que estos procesos (incorporación del Activo Fijo, asignación, devolución, depreciaciones y revalorización) se los realiza de forma semi-manual.

En el proyecto se realizo un análisis previo y con el mismo se realizó el diseño de cada uno de los procesos, utilizando como metodología de desarrollo de software el "Proceso Unificado de Desarrollo de Software (RUP)", que permite modelar el comportamiento que va ha tener el sistema utilizando el modelo UML (Lenguaje Unificado de Modelado), el cual modelará cada uno de los procesos. A partir de esta situación se procede a desarrollar las aplicaciones del producto final.

Durante la construcción del software se realizaron entrega de prototipos a los usuarios finales, producto de las iteraciones de la metodología utilizada, con ello se obtuvo la funcionalidad del sistema incrementalmente y la facilidad operacional de la misma, satisfaciendo los requerimientos y expectativas de los usuarios.

ÍNDICE

CAPÍTULO IV
ANÁLISIS Y DISEÑO DEL SISTEMA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

Hoy en día, el uso de la tecnología se ha convertido en un recurso primordial con mayores exigencias de calidad y servicio que constituyen en elemento esencial en las instituciones públicas y privadas, siendo estos sectores donde se genera mayor información que hace cada vez más prescindible el uso de la tecnología informática otorgando mayor beneficio y proyectándonos hacia el futuro.

En la actualidad, existen instituciones públicas y privadas que hacen el control de sus bienes de manera incorrecta tal es el caso de los Activos Fijos[1]. Los Activos Fijos son parte esencial de cualquier institución y más aun cuando estos tienden a crecer y su control es cada vez más complejo para quienes administran estos bienes.

Es importante contar con un sistema de administración de Activos Fijos, el cual permite el almacenamiento y recuperación de información. La documentación es un recurso organizacional, el cual provee información importante para la institución. La utilización de sistemas automatizados para el almacenamiento de información nos permite generar el documento requerido de manera eficaz, el cual mejora el rendimiento en el desempeño en las actividades de la institución.

La Universidad Privada Franz Tamayo, tiene como propósito, mejorar la administración de los Activos Fijos, se pretende contar con un sistema automatizado de manejo y disposición de los bienes, con el fin de mantener un control efectivo.

Es así, que el presente Proyecto es una herramienta que permite solucionar los problemas que se tiene en la Dirección de Activos Fijos de la universidad, con un enfoque que permita controlar y generar documentación para la administración de los Activos Fijos, obteniendo información oportuna para la toma de decisiones.

[1] Bienes tangibles que tengan una vida útil mayor o igual a un año, tales como terrenos, edificios, muebles y enseres, vehículos, etc

1.1. Antecedentes

A continuación se detalla algunos proyectos con relación al presente trabajo:

- "Sistema de información de Activos Fijos y Almacenes Cámara Nacional de Comercio", desarrollado por el postulante Martha Silva (2003), cuyo objetivo es; desarrollar un sistema de información de Activos Fijos y almacenes que proporcione información confiable, ágil y oportuna para optimizar la administración y adoptar políticas institucionales sobre Activos Fijos y almacenes. Podemos notar que este trabajo comprende los siguientes puntos: desarrollar subsistemas de Activos Fijos, que permita realizar el seguimiento de los bienes de uso de las instituciones, desarrollar un subsistema de almacenes que permita realizar un control de los artículos y suministros desde el momento de su compra, ingreso a almacenes, integrar los subsistemas de Activos Fijos y almacenes al sistema de control de personal y control de usuario.

- "Sistema de Información y Control de Activos Fijos Ministerio de Gobierno", desarrollado por el postulante James Williams Gutiérrez Flores (2005), cuyo objetivo es; Desarrollar e implementar un Sistema de Información automatizado para la administración, seguimiento y control de Activos Fijos del Ministerio de Gobierno, que permite tener información actualizada y centralizada. Podemos notar que este trabajo comprende los siguientes puntos: migrar la información de la base de datos del antiguo sistema, generar reportes e informes de los procesos realizados con el sistema, uniformar los formatos de actas e informes, diseñar y desarrollar las interfaces que faciliten la operabilidad del sistema, capacitar al personal sobre el manejo del nuevo sistema.

Los proyectos tratan de resolver problemas particulares en el manejo de Activos Fijos, que tienen las empresas, instituciones públicas, privadas y otros.

El sistema que se implementara a diferencia de los ya existentes, contemplará y realizará un control en el seguimiento de los Activos Fijos, el cual permitirá mejorar en

el control informativo de los mismos, mejorando en el desempeño de la Dirección de Activos Fijos.

1.2. Problema

En la Universidad Privada Franz Tamayo en la Dirección de Activos Fijos, se realizó un diagnostico previo y se pudo observar las siguientes dificultades:

Causa
Dificultad para obtener información de registro de Activos Fijos.
Efecto
Perdida de tiempo y trabajo para el personal.

Causa
Carencia de control en la universidad de seguimiento, supervisión y asignación de Activos Fijos.
Efecto
Perdida de tiempo y trabajo para el personal.

Causa
Inadecuado registro de los Activos Fijos.
Efecto
Redundancia en la actualización de Activos Fijos.

Causa
No existe un registro adecuado de las bajas que se realizan de los Activos Fijos.
Efecto
No se tiene actualizado la información de los Activos Fijos, lo cual afecta a la toma de decisiones.

1.3. Planteamiento del problema

Existe deficiencia en el manejo y control de los Activos Fijos, con lo que la información de los mismos no es integra, lo cual dificulta en la actualización de la información de estos activos con referencia a la asignación y devolución de los mismos en sus diferentes áreas de trabajo.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar e implementar un sistema automatizado, que permita administrar y controlar el movimiento de los Activos Fijos, reduciendo el tiempo en los procesos de manejo de información de los Activos Fijos en la Universidad Privada Franz Tamayo.

1.4.2. Objetivos Específicos

- Mejorar el control de los Activos Fijos, desde su ingreso, asignación, devolución y otros, disminuyendo los tiempos de respuesta en la obtención de informes ó reportes.
- Diseñar una Base de Datos para el registro y clasificación de los Activos Fijos, que permita generar reportes e informes de los procesos realizados con el sistema.
- Generación de códigos para los Activos Fijos, el mismo será generado de acuerdo a la ubicación, grupo y auxiliar contable al que pertenece, bajo determinados criterios.
- Mejorar el cálculo de las depreciaciones de los Activos Fijos aplicando el método de la Línea Recta.
- Obtener el valor neto de los Activos Fijos, basado en el método de las depreciaciones aplicados.
- Unificar los formatos de actas e informes.

1.5. Justificación

1.5.1. Justificación práctica

Con el presente Proyecto de Grado, se pretende contribuir de alguna manera en las actividades que se realiza en la Dirección de Activos Fijos de la Universidad Privada Franz Tamayo, con el fin de mejorar la administración que se realiza de los Activos Fijos.

1.5.2. Justificación técnica

La Dirección de Activos Fijos cuenta con un ambiente de trabajo y equipos necesarios para la implementación del sistema, en la Dirección de Sistemas Informáticos y Redes, se tiene equipos de computación para el desarrollo del presente Proyecto y apoyo necesario.

1.5.3. Justificación económica

La propuesta planteada contemplará el uso de equipos existentes, en la Dirección de Activos Fijos de la universidad. Con la automatización se minimizara el tiempo y el esfuerzo laboral, reduciendo gastos económicos.

1.5.4. Justificación social

El presente Proyecto beneficiara al personal de la Dirección de Activos Fijos y la universidad en su conjunto, al tener un mejor control de los Activos Fijos, brindando información oportuna que ayudara al personal.

1.6. Alcances

El "Sistema de Gestión y Control del Activos Fijos Universidad Privada Franz Tamayo",
esta orientado a:

- Registro de los Activos Fijos, el cual realizará el registro de los Activos Fijos de forma dinámica tomando en cuenta la información necesaria para su registro.
- Asignación de Activos Fijos, el cual realizará la asignación de Activos Fijos a un responsable.
- Devolución de Activos Fijos, el cual realizará la devolución de Activos Fijos de un responsable.
- Depreciación de Activos Fijos, el cual realizará la depreciación de los Activos Fijos, tomando en cuenta activos revalorizados, nuevos y existentes.
- Reporte de los Activos Fijos, proporcionará información necesaria de los Activos Fijos en formato para impresión, bajo determinación de criterios.

1.7. Aportes

- Sistema de autentificación de usuarios autorizados para el manejo del sistema en el aspecto administrativo.
- Algoritmo de búsqueda por clasificación del personal, para el manejo de su información.
- Algoritmo de verificación, control e información de monedas.

1.8. Límites

El presente sistema no tomará en cuenta otras áreas de la universidad, se abocara específicamente al control de los Activos Fijos en la Dirección de Activos Fijos.

1.9. Metodología

Para el desarrollo del Proyecto, se utiliza la Metodología RUP (Proceso Unificado de Desarrollo de Software), por ser un proceso de desarrollo de software que se adapta a proyectos de diferente complejidad, y por ajustarse a las necesidades del Proyecto.

Para el modelado de análisis y diseño se utiliza UML (Unified Modeling Language) que permite modelar, construir y documentar los elementos que forman parte del sistema.

CAPÍTULO II

MARCO INSTITUCIONAL

2.1. Universidad Privada Franz Tamayo

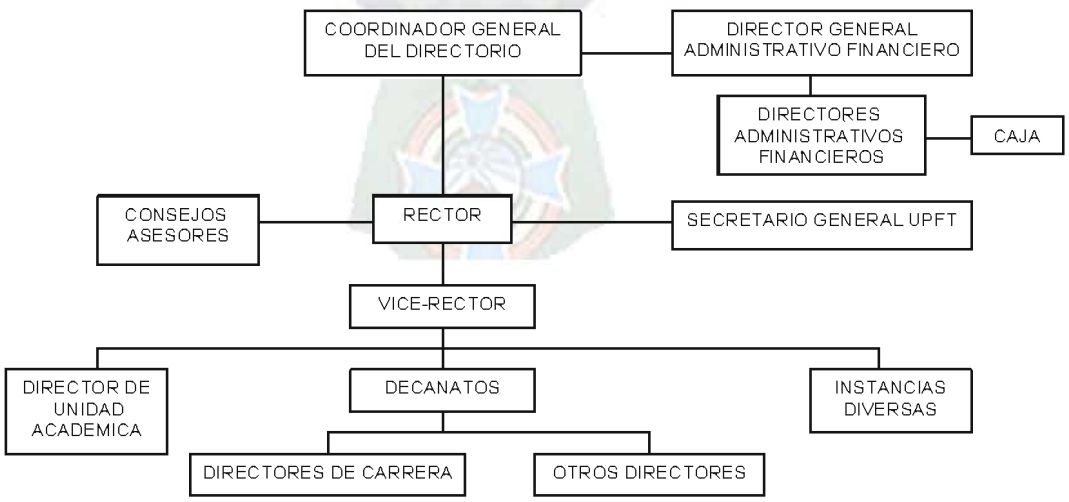
Misión

Formar recursos humanos altamente calificados en las diversas disciplinas de la educación superior y en todos sus niveles, desarrollando profesionales creativos, responsables y con principios éticos, capaces de aplicar los conocimientos técnicos, científicos y humanos, en beneficio del país y de la comunidad internacional.

Visión

Contar con profesionales creativos, formados a través de una educación para el desarrollo personal, con profunda orientación de servicio, alta formación técnica, científica y humanística; capaces de desenvolverse y desarrollarse en cualquier ámbito.

2.2. Organigrama General



Fuente: Estatuto Orgánico Universidad Privada Franz Tamayo

2.3. Organigrama Dirección Administrativa Financiera

Fuente: Estatuto Orgánico Universidad Privada Franz Tamayo

2.4. Dirección de Activos Fijos

La Dirección de Activos Fijos, tiene como misión administrar todo los activos existentes en la Universidad Privada Franz Tamayo, desde su incorporación, codificación, asignación, devolución, bajas y el cálculo de las depreciaciones concernientes a los Activos Fijos. La Dirección de Activos Fijos en forma permanente realiza inspecciones y seguimiento de todos los activos existentes en la universidad, este tipo de seguimiento tiene como objeto determinar el buen manejo de los Activos Fijos por parte de cada responsable.

CAPÍTULO III

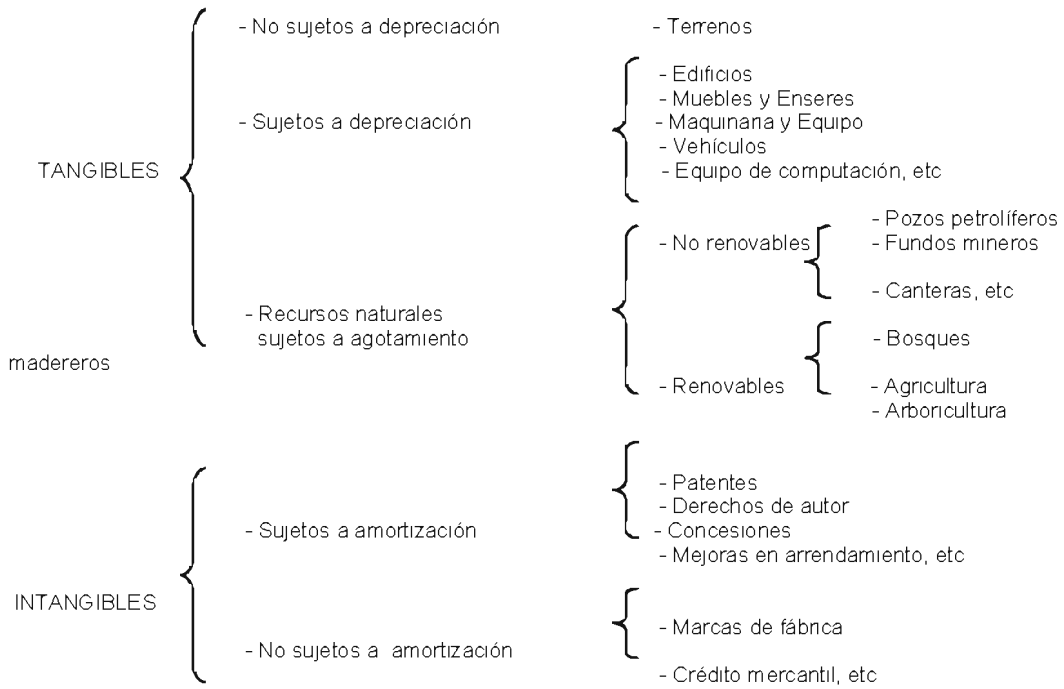
MARCO TEÓRICO

3.1. Activos Fijos

Los Activos Fijos, son todos aquellos bienes tangibles que se utilizan en la actividad de cualquier institución, que tenga una vida útil superior a un año y que no estén destinados a la venta.

3.1.1. Clasificación de los Activos Fijos

Los activos fijos se clasifican en dos grandes grupos:



Fuente: [FUNES03]

3.1.2. Depreciación de los Activos Fijos

Es la perdida de valor que sufre el bien de uso a través del tiempo por el servicio que presta, su manipulación y otros. La depreciación se estima por su vida útil y su valor residual, se entiende por vida útil el lapso de vida que tiene un activo en años.

3.1.2.1. Causas de la depreciación

Las causas de depreciación más importantes son: el deterioro físico y la obsolescencia.

- **Deterioro físico**
  El deterioro físico de un bien en uso, resulta de su desgaste por el uso normal, deterioro; daño ocasionado por factores climatológicos.
- **Obsolescencia**
  Es el proceso en el cual el Activo Fijo puede volverse desactualizado u obsoleto es decir, el Activo Fijo es remplazado por otro tomando en cuenta la oportunidad de uso económico y eficiencia, y no así de su condición física.

3.1.3. Métodos de Depreciación

Hay varios métodos alternativos para calcular la depreciación. Una Empresa no necesita utilizar el mismo método de depreciación para sus variados activos. Entre los métodos de depreciación se mencionan a continuación los más utilizados:

- Método de la Línea Recta.
- Método de unidades de producción.
- Método suma de los dígitos de los años.
- Método saldo decreciente.

3.1.3.1. Método de la Línea Recta

Este método permite hacer la distribución total de la depreciación por partes iguales a través de todos los periodos se su vida útil de un bien en uso.

El calculo del cargo por depreciación del periodo se hace restando el *valor de salvamento o residual* del costo del activo y dividiendo el remanente del *costo depreciable* por los años de vida útil estimada.

$$\frac{\text{Costo} - \text{Valor Residual}}{\text{Años de Vida Útil}}$$

3.2. El Proceso Unificado de Desarrollo de Software

El Proceso Unificado es un proceso de desarrollo de software. Un *proceso* de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software (ver Figura 3.1).

El Proceso Unificado esta *basado en componentes*, utiliza el Lenguaje de Modelado (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software.

El Proceso Unificado se resumen en tres fases clave; dirigidos por casos de uso, centrado en la arquitectura, e iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

Figura 3.1. Un Proceso de Desarrollo de Software
Fuente: [JACOB00]

- **El Proceso Unificado está dirigido por casos de uso**

  Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representa los requisitos funcionales. Todo los casos de uso juntos constituyen el modelo de casos de uso el, cual describe la funcionalidad total del sistema.

- **El Proceso Unificado está centrado en la arquitectura**

  El proceso se centra en establecer al principio una arquitectura software que guían el desarrollo del sistema. Este diseño arquitectónico sirve como una sólida base sobre el cual se puede planificar y manejar el desarrollo del software basado en componentes.

  La arquitectura se desarrolla mediante iteraciones, principalmente durante la fase de elaboración, comenzando con los requisitos y siguiendo con el análisis, diseño, implementación y pruebas pero centrándose en los casos de uso relevantes desde el punto de vista de la arquitectura y en otros requisitos. El resultado al final de la fase de elaboración es una línea base de la arquitectura como muestra la Figura 3.2.

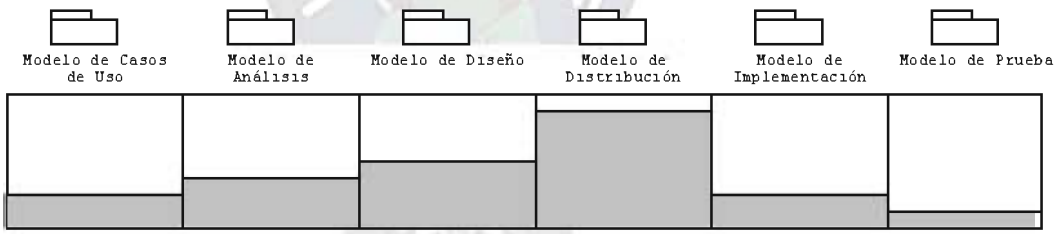


Figura 3.2. Línea base de la arquitectura
Fuente: [JACOB00]

- **El Proceso Unificado es iterativo e incremental**

  El desarrollo de un producto de software supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o subproyectos, cada subproyecto es una iteración que resulta de un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto.

En la Figura 3.3. se divide los elementos genéricos del flujo de trabajo de cada iteración.

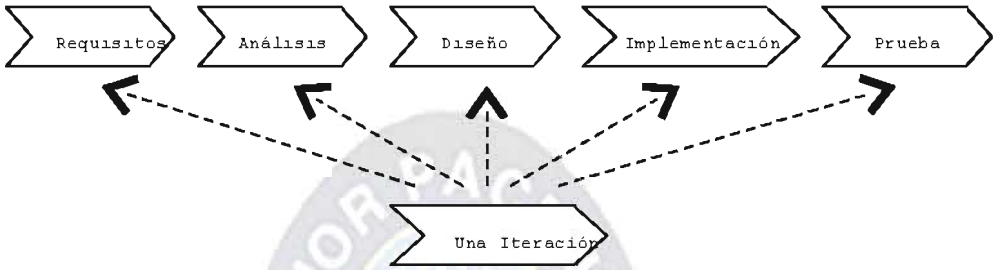


Figura 3.3. Cada iteración constituye cinco flujos de trabajo fundamentales
Fuente: [JACOB00]

Las iteraciones pueden solaparse en un sentido que una iteración está a punto de terminar cuando otra está comenzando, como se muestra en la Figura 3.4.

Figura 3.4. Las iteraciones a lo largo de los flujos de trabajo
Fuente: [JACOB00]

3.2.1. Vida del Proceso Unificado

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, cada ciclo concluye con una versión del producto para los clientes.

Cada ciclo consta de cuatro fases: Inicio, elaboración, construcción y transición. Cada fase se subdivide a su vez en iteraciones.

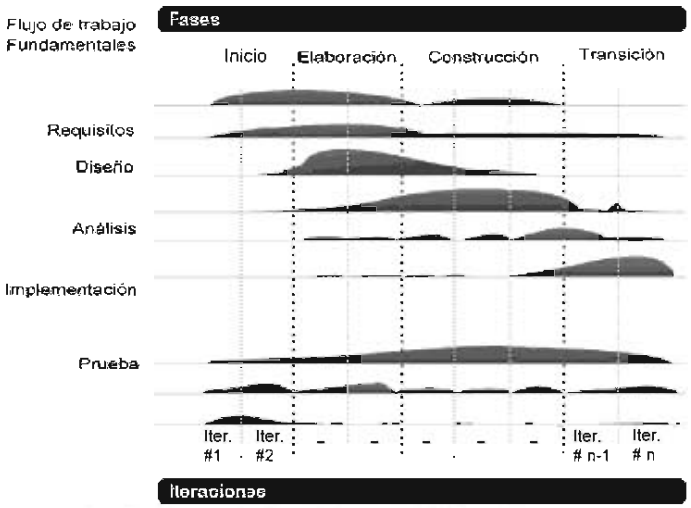


Figura 3.5. Fase de Desarrollo del Proceso Unificado
Fuente: [JACOB00]

3.2.2. Fases dentro de un ciclo

- **Fase de inicio**

  Durante está fase, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis del negocio para el producto.

- **Fase de elaboración**

  Durante está fase, se especifican en detalle la mayoría de los casos de uso del producto y se diseñan la arquitectura del sistema. Durante está fase del desarrollo, se realizan los casos de uso más críticos que se identificaron en la fase de inicio. El resultado de esta fase es una línea base de la arquitectura.

- **Fase de construcción**

  Durante está fase se crea el producto. En está fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de la fase de construcción se decide si el software, los lugares donde se instalará y los usuarios están todos preparados para empezar a funcionar.

- **Fase de transición**

  La fase de transición con lleva actividades como la fabricación, formación del usuario, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentran tras la entrega.

3.2.3. Proceso integrado

El Proceso Unificado esta basado en componentes, utiliza el *Lenguaje Unificado de Modelado* (UML), y se sostiene sobre tres ideas básicas que son: caso de uso, arquitectura y desarrollo iterativo e incremental.

3.2.4. Los flujos de trabajos fundamentales

El Proceso Unificado consiste en una serie de flujos de trabajo que van desde los requisitos hasta las pruebas. Los flujos de trabajo desarrollan modelos, desde el modelo de casos de uso hasta el modelo de prueba.

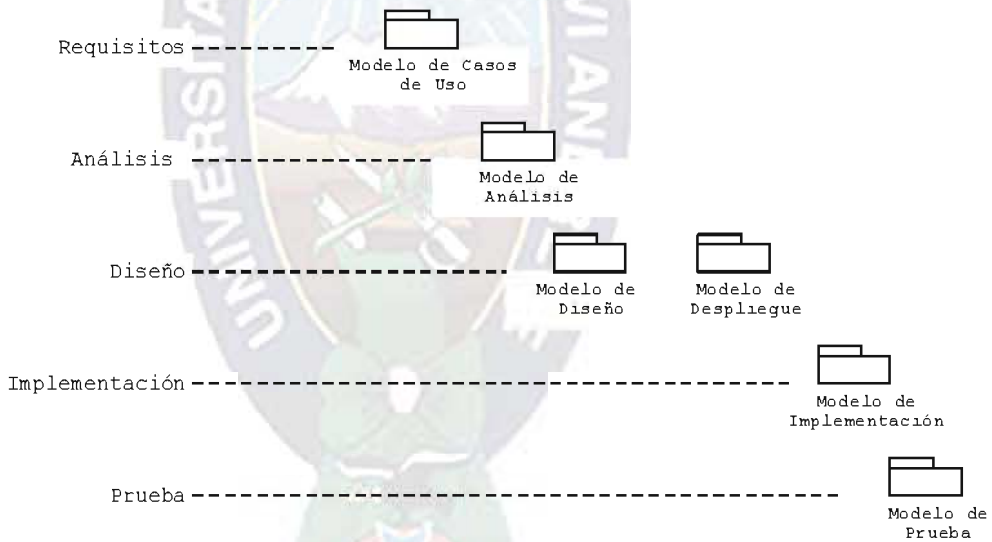


Figura 3.6. Flujos de trabajo fundamentales
Fuente: [JACOB00]

3.2.4.1. Requisitos

Llamamos *Captura de Requisitos* a este acto de descubrimiento. Cada proyecto de software es diferente. Esta singularidad proviene de las diferencias en el tipo de sistema, en el cliente, en la organización de desarrollo, en la tecnología, etc. De igual forma hay diferentes puntos de partida para la captura de requisitos. En algunas

ocasiones comenzamos haciendo un modelo del negocio, o comenzamos con un modelo del negocio que ya está desarrollado por parte de alguna otra empresa.

Este flujo de trabajo incluye los siguientes pasos:

Trabajo a realizar	Artefactos resultantes
Enumerar requisitos candidatos	Lista de características
Comprender el contexto del sistema	Modelo del dominio o del negocio
Captura de requisitos funcionales	Modelo de casos de uso
Captura de requisitos no funcionales	Requisitos adicionales o casos de uso concreto (para requisitos específicos de un casos de uso)

Figura 3.7. Artefactos necesarios para establecer el contexto del sistema
Fuente: [JACOB00]

3.2.4.1.1. Artefactos

- **Modelos del dominio**
  Este modelo captura los tipos más importantes de objetos en el contexto del sistema. El modelo del dominio se describe mediante diagrama de clases de UML.
- **Modelo del negocio**
  Describe los procesos del negocio de una empresa en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y lo clientes, respectivamente.

  Un modelo de negocio se desarrolla en dos pasos:
  1 Se debe desarrollar un modelo de casos de uso del negocio que identifique los actores del negocio y casos de uso del negocio que utilicen los actores.
  2 Desarrollar un modelo de objetos del negocio compuesto por *trabajadores*[1], entidades del negocio, y unidades de trabajo que juntos realizan los casos de uso del negocio.

[1] Puesto que puede ser asignado a una persona o equipo

3.2.4.2. Análisis

Durante el análisis se realiza los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos.

El lenguaje que se utiliza en el análisis se basa en un modelo conceptual que llamamos *modelo de análisis*. El modelo de análisis nos ayuda a refinar los requisitos según:

- Los casos de uso deben mantenerse tan independientes unos de otros como sea posible.
- Los casos de uso deben describirse utilizando el lenguaje del cliente. Esto se consigue al utilizar notaciones más formales, como diagramas de estado, actividad o interacción.
- Debe estructurarse cada caso de uso para que forme una especificación de funcionalidad completa e intuitiva.

3.2.4.2.1. Artefactos

- **Modelo de análisis**
  Dentro del modelo de análisis, los casos de uso se describen mediante clases de análisis y su objeto. Esto se representa mediante colaboraciones dentro del modelo de análisis que llámanos *realizaciones de caso de uso-análisis*.

  - **Realización de caso de uso-análisis**
    Una realización de caso de uso-análisis, es una colaboración dentro del modelo de análisis que describe como se lleva acabo y se ejecuta en un caso de uso determinando en términos de las clases del análisis y de sus objetos de análisis en interacción.
    Una realización de caso de uso posee una descripción textual del flujo de sucesos, diagrama de clases que muestran sus clases del análisis participantes y diagramas de interacción que muestran la realización de un

flujo o escenario particular del caso de uso en términos de interacciones de objetos de análisis.

- **Paquete de análisis**

  Los paquetes de análisis proporcionan un modelo para organizar los artefactos del modelo de análisis en piezas más pequeñas y manejables. Un paquete de análisis puede constar de realizaciones de casos de uso y otros paquetes de análisis (recursivamente).

Figura 3.8. Identificación de paquete de análisis a partir de casos de uso
Fuente: [BOOCH99]

3.2.4.3. Diseño

En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todo los requisitos, incluyendo todos los requisitos no funcionales y otras restricciones que se le suponen.

3.2.4.3.1. Artefactos

- **Modelo de diseño**

El modelo de diseño, es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales junto con otras restricciones relacionada con el entorno de implementación.

En el modelo de diseño, los casos de uso son realizados por las clases de diseño y sus objetos. Esto se representa por colaboraciones en el modelo de diseño y

denota realización de caso de uso-diseño.

- **Realización de caso de uso-diseño**
  Una realización de caso de uso-diseño es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso especifico y cómo se ejecuta en términos de clases de diseño y sus objetos.

3.2.4.4. Implementación

En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

3.2.4.4.1. Artefactos

El modelo de implementación describe cómo los elementos del modelo de diseño como las clases se implementan en términos de componentes como ficheros de código fuente, ejecutables, etc.

3.2.4.5. Prueba

En este flujo de trabajo verificamos el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema hacer entregados.

3.2.4.5.1. Artefactos

- **Caso de prueba**
  Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultados con la que se ha de probar y las condiciones bajo las que se ha de probarse.

3.3. Lenguaje Unificado de Modelado(UML), Unified Modeling Language

UML, es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

- **Diagrama en UML**

  Un diagrama es la representación gráfica de un conjunto de elementos, visualizado la mayoría de la veces como un grafo conexo de nodos (elementos) y arcos (relaciones). Los diagramas se dibujan para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema, un diagrama representa una vista resumida de los elementos que constituyen un sistema.

  UML incluye 5 vistas que son:

  - Vistas de casos de uso, comprende los casos de uso que describen el comportamiento del sistema. Con UML, los aspectos estáticos de esta vista se captura en los diagramas de casos de uso, los aspectos dinámicos de esta vista se captura en los diagramas de interacción y diagramas de estados.
  - Vista de diseño, comprende las clases, interfaces y colaboraciones que forman el vocabulario del problema y su solución. Con UML, los aspectos estáticos de esta vista se capturan en los diagramas de clases; los aspectos dinámicos se capturan en los diagramas de interacción y los diagramas de estados.
  - Vista de proceso, comprende los hilos y los procesos que forman los mecanismos de sincronización y concurrencia del sistema. Con UML, los aspectos estáticos y dinámicos de esta vista se capturan en el mismo tipo de diagrama que la vista de diseño.
  - Vista de implementación, comprende los componentes y archivos que se utilizan para ensamblar y hacer disponible el sistema físico. Con UML, el

aspecto estático de esta vista se captura en los diagramas de componentes; los aspectos dinámicos de esta vista se captura en los diagramas de interacción y diagramas de estados.

- Vista de despliegue, contienen los nodos que forman la topología hardware sobre la que se ejecuta el sistema. Con UML los aspectos estáticos de vista se capturan en los diagramas de despliegue; los aspectos dinámicos de esta vista se capturan en los diagramas de interacción y los diagramas de estados.

- **Diagrama de clases**

  Un *diagrama de clases* se utilizan para modelar la vista de diseño estática de un sistema. Este diagrama muestra un conjunto de interfaces, colaboraciones y sus relaciones.

- **Diagrama de casos de uso**

  Un *diagrama de casos de uso*, es un diagrama que muestra un conjunto de casos de uso, actores y sus relaciones.

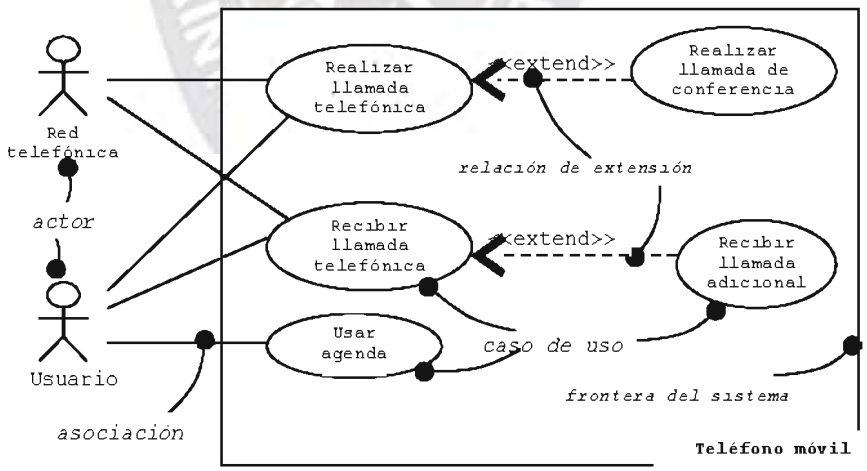


Figura 3.9. Diagrama de casos de uso
Fuente: [BOOCH99]

- **Diagrama de interacción**

  - **Diagrama de secuencia**

Un *diagrama de secuencia* destaca la ordenación temporal de los mensajes como se muestra en la Figura 3.10.

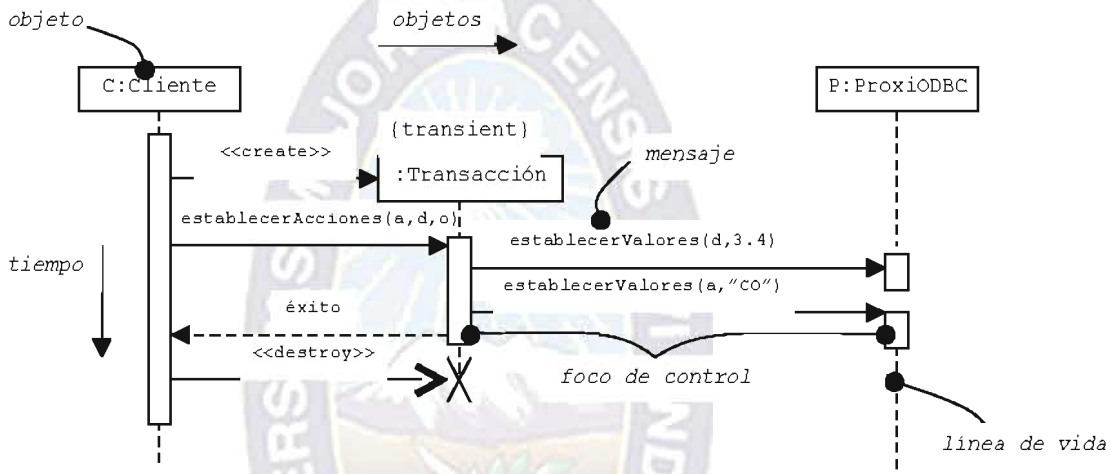


Figura 3.10. Diagrama de secuencia
Fuente: [BOOCH00]

  - **Diagrama de colaboración**

Un *diagrama de colaboración* destaca la organización de los objetos que participan en una interacción, como se muestra en la Figura 3.11.

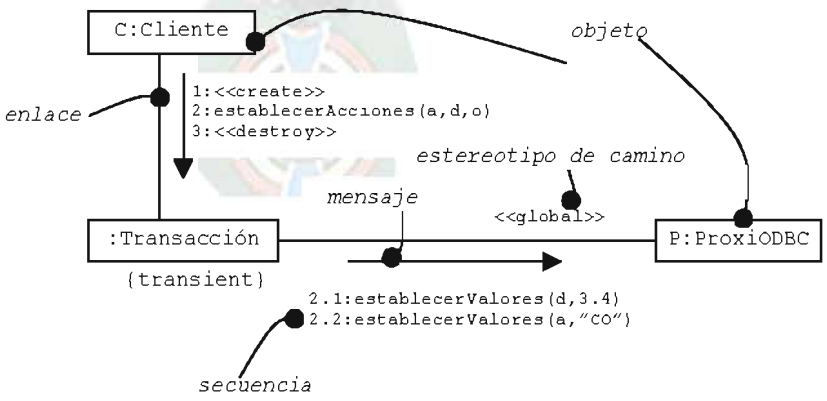


Figura 3.11. Diagrama de colaboración
Fuente: [BOOCH99]

- **Diagrama de despliegue**

  Un *diagrama de despliegue* es un diagrama que muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos. Gráficamente un diagrama de despliegue es una colección de nodos y arcos. Un diagrama de despliegue se utiliza para modelar la vista de despliegue estática de un sistema.

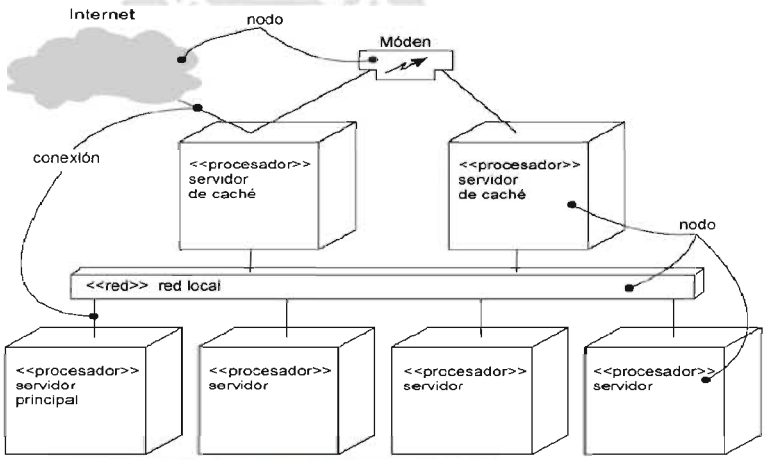


Figura 3.12. Diagrama de despliegue
Fuente: [BOOCH99]

- **Diagrama de estados**

  Un *diagrama de estados* muestra la secuencia de estados por los que pasa un caso de uso a lo largo de su existencia, teniendo como características los eventos que hacen la transición de un estado a otro.

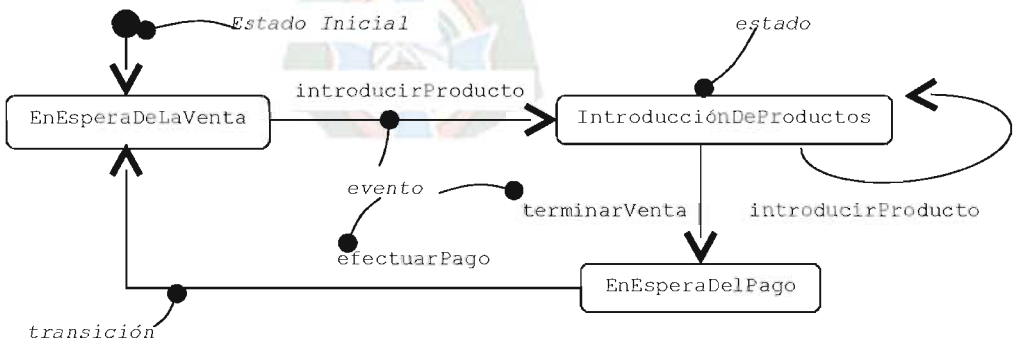


Figura 3.13. Diagrama de estados
Fuente: [LARMA99]

- **Diagrama de componentes**

  Un *diagrama de componentes* muestra un conjunto de componentes y sus relaciones. Se utilizan para visualizar aspectos estáticos de estos componentes físicos y sus relaciones, y para especificar sus detalles para la construcción, como se muestra en la Figura 3.14.

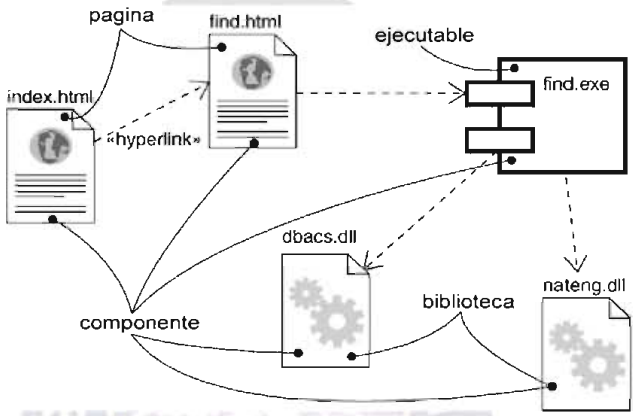


Figura 3.14. Diagrama de componentes
Fuente: [BOOCH99]

3.4. Herramientas de desarrollo

3.4.1. Sistema Operativo Linux

El proyecto GNU consiste en el desarrollo de un sistema operativo y juego de aplicaciones totalmente libre y compatible con UNIX. El proyecto incluye desarrollar una versión libre de cualquier aplicación que no se disponga libre. De esta forma, una computadora puede estar equipada con software libre y cumplir cualquier función; esto incluye el sistema operativo y todos los programas que uno necesite para cualquier función. Ya que sin un sistema operativo no puede usarse una computadora, se tomó esto como punto de partida para el proyecto GNU.

En 1990, se habían encontrado o escrito la mayoría de los componentes mayores del sistema operativo excepto uno: el kernel o núcleo. Para ese entonces, Linux comenzó como proyecto personal del entonces estudiante Linus Torvalds, que se basó en el

Minix de Andy Tanenbaum (profesor que creó su propio clon de UNIX para PC-XT para usarlo en su docencia). Combinando Linux con el resto del sistema GNU se llego a la meta inicial de un sistema operativo libre: El sistema GNU basado en Linux. Se estima que hoy hay millones de usuarios de GNU/Linux.

Actualmente Linus lo sigue desarrollando, pero a estas alturas el principal autor es la red Internet, desde donde un gigantesco grupo de programadores y usuarios aportan su tiempo y ayuda, tanto al núcleo Linux como al resto de las aplicaciones. La Free Software Foundation (Fundación software libre, FSF) continúa con el proyecto GNU desarrollando otras aplicaciones que todavía no tienen su versión libre.

3.4.2. Servidor Web Apache

En febrero de 1995, Brian Behlendorf y Chiff Skolnick ensamblaron una lista de envio, prepararon un computadora y consiguieron ancho de banda, donado por HotWired. Brian construyo un árbol CVS (Sistema de Versiones Simultanes), en virtud del cual todo el que quisiera podía contribuir a crear nuevas características y a reparar errores. De esta forma, un grupo de desarrolladores podian recoger las modificaciones a sus códigos y crear un producto combinado. Comenzó con HTTPd 1.3 NCSA, empezaron a aplicar estas soluciones. La primera versión de este producto, llamado Apache, fue la versión 0.6.2 lanzado en abril de 1995.

Los ocho socios fundadores del Grupo Apache eran Bchlendorf, Skolnick, Roy T. Fielding, Rob Hartill, David Robinson, Randy Terbush, Robert S. Thau y Andrew Wilson.

Poco después del primer lanzamiento Thau diseño una arquitectura completamente nueva. Comenzando con la versión 0.8.8 en agosto de 1995, Apache se incorporo a esta nueva base del código.

Netcraft muestra que Apache sobrepasa a NCSA como primer servidor HTTP a principios de 1996.

De acuerdo con la estadística de Netcraft, el servidor Web Apache se emplea mas que el resto del conjunto de servidores Web. De los cerca de 7 millones de sitios Web que tiene la World Wide Web, cerca de 4 millones (el 55%) ejecutan Apache. Si también se cuenta el software para servidor basado en código Apache, esta cifra se acerca al 60%.

NOTA

hecho encuestas en la Web desde julio de 1995, momento en el que registro 18.975 sitios en la Web. La empresa actualiza sus encuestas con carácter mensual, mostrando el crecimiento c declive de cada uno de los protagonistas, y ofrece comentarios sobre estas tendencias. Netcrafl es una empresa de investigación en Internet, que ofrece encuestas como ésta, así como consultaría de seguridad y varios servicios web e Internet.

La Figura 3.15. muestra a los servidores Web más populares y del número de sitios Web que utilizan estos servidores.

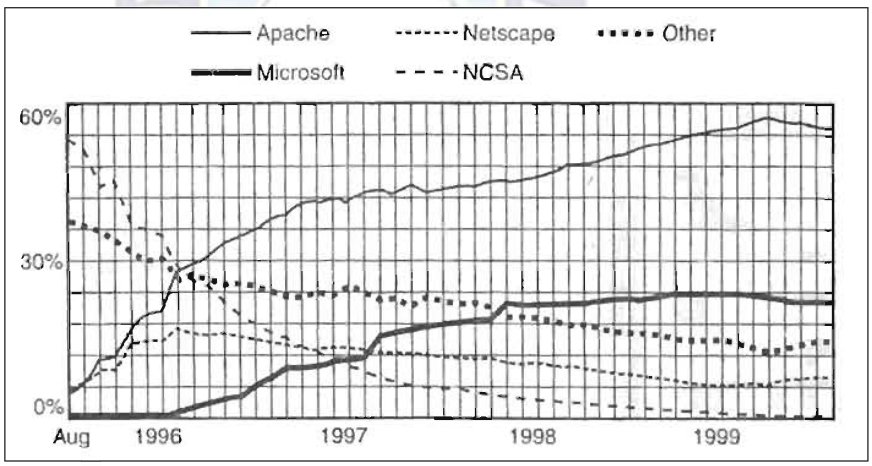


Figura 3.15. Servidores más populares
Fuente: [BOWEN00]

3.4.3. MySQL

MySQL es una idea que nace por la empresa Opensource MySQL AB establecida en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido ni flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

La procedencia del nombre de MySQL no es clara. Por más de 10 años, las herramientas han mantenido el prefijo My. También, se cree que tiene relación con el nombre de la hija del cofundador Monty Widenius quien se llama My.

Por otro lado, el nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso "Name the Dolphin".

MySQL probablemente sea el gestor Bases de Datos más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales características de este gestor de bases de datos MySQL son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
2. Soporta gran cantidad de tipos de datos para las columnas.
3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
4. Gran portabilidad entre sistemas.
5. Soporta hasta 32 índices por tabla.
6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

3.4.4. Lenguaje de Programación PHP

Rasmus Lerdof, fue quien desarrollo PHP en el año 1994. esta primera versión sin liberar, la incorporo a su página Web personal, cuya utilidad era controlar el número de usuarios que accedían a su currículum vitae online.

Nadie supo de la existencia de PHP hasta principios de 1995, cuando la primera versión al fin fue liberada. Por aquel entonces era conocido como Herramienta de Páginas Web Personales (Personal Home Page Tools).

A principios de 1995 PHP se puso a disposición de los desarrolladores como Personal Home Page Tools. Esta versión contenía un motor de análisis de sintáxis (perser) que utilizaba algunas macroinstrucciones (macros) y utilidades especiales que fueron incluidas a menudo en las páginas personales.

Esto hizo de PHP una opción bastante popular entre los desarrolladores que querían crear y realzar sus páginas. PHP les permitió que añadieran a sus páginas Web ciertas funciones, ahora habituales, como libros de invitados y contadores.

PHP tuvo varias versiones, y a mediados de 1995 Rasmus rescribió el analizador de sintaxis y lo tituló PHP/FI versión 2, hasta mediados de 1997, el analizador de sintaxis fue reescrito totalmente por Zeec Suraski y Andi Gulmnas, este formo el núcleo de la tercera versión de PHP, conocida como PHP 3. Hasta la fecha actualmente se cuenta con la versión de PHP 5.

3.5. Diseño de la Interfaz de Usuario

El diseño de la interfaz de usuario es la categoría de diseño que crea un medio de comunicación entre el hombre y la maquina. Con un conjunto de principios para el diseño de la interfaz, el diseño identifica los objetos y las acciones de la interfaz y crea entonces un formato de pantalla.

3.5.1. Las Reglas de Oro

Las "Reglas de Oro" para el diseño de la interfaz son:

- Dar el control al usuario
- Reducir la carga de memoria del usuario
- Construir una interfaz consecuente

3.5.2. Diseño de la Interfaz Web

Los sistemas de aplicaciones Web requieren otras consideraciones adicionales en el diseño de la interfaz que son:

- Los menús de navegación y las barras de cabecera se deberán diseñar consecuentemente y deberán estar disponibles en toda las páginas a las que el usuario tiene acceso. El diseño no deberá depender de las funciones del navegador para ayudar en la navegación.
- La estética nunca deberá sustituir la funcionalidad por ejemplo, un botón sencillo podría ser una opción de navegación mejor que una imagen o icono.
- Las opciones de navegación deberán ser obvias, incluso para el usuario casual. El usuario deberá buscar la pantalla para determinar cómo enlazar con otro contenido o servicio.

3.6. Prueba del Software

Las pruebas del software son un elemento critico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Cualquier producto de software puede probarse de una de estas dos formas:

1.- Conociendo la función especifica para la que fue diseñado el producto, se puede llevar a cabo pruebas que demuestren que cada función es completamente operativa y, al mismo tiempo buscando errores en cada función.

2.- Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que "todas las piezas encajan", o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

El primer enfoque de prueba se denomina prueba de caja negra y el segundo prueba de caja blanca.

3.7. Métricas de Calidad del Software

La Calidad del software se define como, concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.

Se define en tres puntos:

a) Los requisitos del software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.
b) Unos estándares especificados definen un conjunto de criterios de desarrollo que guían la manera en que se hace la ingeniería de software. Si no se siguen los criterios, habrá seguramente poca calidad.
c) Existe un conjunto de requisitos implícitos que a menudo no se nombran (tal es el caso la facilidad de mantenimiento). Si el software cumple con sus requisitos explícitos pero falta en los implícitos, la calidad del software no será fiable.

La calidad del software es una compleja mezcla de factores que varían a través de diferentes aplicaciones y según los clientes que la pidan.

3.7.1. Factores de Calidad ISO 9126

El estándar ISO 9126 ha sido desarrollado en un intento de identificar los atributos clave de la calidad para el software. El estándar identifica seis atributos clave de calidad:

Funcionalidad. El grado en que el software satisface las necesidades indicadas por los siguientes subatributos: idoneidad, corrección, interoperatividad, conformidad y seguridad.

Confiabilidad. Cantidad de tiempo que el software está disponible para su uso. Está referido por los siguientes subatributos: madurez, tolerancia a fallos y facilidad de recuperación.

Usabilidad. Grado en el que el software es fácil de usar. Viene reflejado por los siguientes subatributos: facilidad de comprensión, facilidad de aprendizaje y operatividad.

Eficiencia. Grado en el que el software hace óptimo el uso de recursos de sistema. Está indicado por los siguientes subatributos: tiempo de uso y recursos utilizados.

Facilidad de mantenimiento. La facilidad con que una modificación puede ser realizada. Está indicada por los siguientes subatributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba.

Portabilidad. La facilidad con que el software puede ser llevado de un entorno a otro. Está referido por los siguientes subatributos: facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio.

3.7.2. Métricas Orientadas a la Función

Las métricas orientadas a la función, utilizan una medida de la funcionalidad entregada por la aplicación como un valor de normalización. Ya que la funcionalidad no se puede medir directamente, se debe derivar indirectamente mediante otras medidas directas. Las métricas orientadas a la función fueron propuestas por primera vez por Albretch [ALB79], quien sugirió una medida llamada *punto de función*. Los puntos de función se derivan con una relación empírica según las medidas contables (directas)

del dominio de información del software y las evaluaciones de la complejidad del software.

Los puntos de función se evalúan según la siguiente tabla:

Parámetros de medición	Factor de ponderación					
	Cuenta		Simple	Medio	Complejo	
Número de estradas de usuario	☐	*	3	4	6	= ☐
Número de salidas de usuario	☐	*	4	5	7	= ☐
Número de peticiones de usuaric	☐	*	3	4	6	= ☐
Número de archivos	☐	*	7	10	15	= ☐
Número de interfaces externas	☐	*	5	7	10	= ☐

Se determinan cinco características de dominios de información y se proporcionan las cuentas en la posición apropiada de la tabla. Los valores de dominios de información se definen de la siguiente forma.

Número de estradas de usuario. Se cuenta cada entrada de usuario que proporcionan diferentes datos orientados a la planificación que proporciona al usuario información.

Número de salidas de usuario. Se cuenta cada salida que proporciona al usuario, información orientada a la aplicación.

Número de peticiones de usuario. Una petición se define como una entrada interactiva que produce la generación de alguna respuesta del software inmediata en forma de salida interactiva.

Número de archivos. Se cuenta cada archivo maestro lógico (base de datos o archivo de datos).

Número de interfaces externas. Se cuenta todas las interfaces legibles por la máquina (cinta de disco) que se utilizan para transmitir información a otro sistema.

Una vez que se han recopilado los datos anteriores, a la cuenta se asocia un valor de complejidad. Las organizaciones que utilizan métodos de puntos de función desarrollan criterios para determinar si una entrada en particular es simple, media o compleja. No obstante la determinación de la complejidad es algo subjetiva. Para calcular los puntos de función se utiliza la siguiente relación:

$$PF = cuenta\_total * [X + Min(Y) * \Sigma F_i]$$

En donde:

- **cuenta_total**: es la suma de todas las entradas PF obtenidas en la tabla.
- **X**: es la confiabilidad del sistema.
- **Min(Y)**: es el error mínimo aceptable a complejidad.
- $\Sigma \mathbf{F_i}$: son valores de ajuste de complejidad según las respuestas a las siguientes preguntas:

1.- ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2.- ¿Se requiere comunicación de datos?
3.- ¿Existen funciones de procesamiento distribuido?
4.- ¿Es crítico el rendimiento?
5.- ¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?
6.- ¿Requiere el sistema entrada de datos interactiva?
7.- ¿Requiere la entrada de datos interactiva, que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?
8.- ¿Se actualizan los archivos maestros de forma interactiva?
9.- ¿Son complejas las entradas, las salidas, los archivos o las peticiones?
10.- ¿Es complejo el procesamiento interno?
11.- ¿Se ha diseñado el código para ser reutilizable?
12.- ¿Están incluidas en el diseño la conversión y la instalación?
13.- ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14.- ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?

Cada una de las preguntas anteriores es respondida usando una escala de rangos desde 0 (no importante o aplicable) hasta 5 (absolutamente esencial). Los valores constantes de la ecuación y los factores de peso se aplican a las cuentas de los dominios de información se determinan empíricamente.

Entonces:

$$\%Funcionalidad = \% PF = (PF_{calculado} / PF_{maximo})*100\%$$

3.7.3. Confiabilidad del Software

La confiabilidad del sistema esta relacionado con los componentes del modelo de sistema. La prueba de cada uno de los modulo en un tiempo dado de su funcionamiento, determinará una métrica de calidad enunciada. Es así que la confiabilidad del sistema se define en términos estadísticos como: la probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado y durante un tiempo especifico.

La confiabilidad R(t) de un componente o un subsistema en determinado medio durante un periodo t se define como la probabilidad de que su tiempo para falla f excede a t; es decir:

$$R(t) = P(T>t) = 1 - F(t)$$

Donde:

R(t): Confiabilidad de un componente o subsistema en el tiempo t

F(t): Probabilidad de falla de un componente o subsistema en el tiempo t.

T: Tiempo para fallar o la duración del tiempo de trabajo sin fallar.

Considerando en tiempo T para fallar es una variable exponencial, se tiene:

$$R(t)= 1 – [1 – e^{-\lambda t}]$$

donde se obtiene: $R(t)= e^{-\lambda t}$

Donde:

λ: Tasa constante de pruebas de fallo (λ = Nº de fallas de acceso / Nº total de accesos al sistema).

T: Periodo de operaciones en tiempo.

Para hallar la confiabilidad del sistema, se considera dos situaciones:

a) El sistema falla si cualquiera de sus componentes falla. En esa situación el modelo se presenta mediante una combinación en serie de todos sus componente y su confiabilidad esta dada por la relación:

$$R_T(t) = R_1(t) * R_2(t) * \ldots\ldots\ldots\ldots * R_{n-1}(t) * R_n(t)$$

b) El sistema falla si y solo si todos sus componentes fallan. Este caso el modelo se presenta mediante la combinación en paralelo de todos su componentes y su confiabilidad esta dada por la relación:

$$R_T(t) = 1 - \{ [\ 1 - R_1(t)] * [\ 1 - R_2(t)] * \ldots\ldots\ldots * [\ 1 - R_{n-1}(t)] * [\ 1 - R_n(t)] \}$$

3.7.4. Usabilidad

La facilidad de uso es un intento de cuantificar "Lo amigable que puede ser un sistema con el usuario" y se puede medir en función a cuatro características:

1) Habilidad intelectual y/o física requerida para aprender el sistema.
2) El tiempo requerido para llegar hacer moderadamente eficiente en el uso del sistema.
3) Aumento neto en productividad, medida cuando alguien usa el sistema moderadamente y eficientemente.
4) Valoración subjetiva (a veces obtenida mediante un cuestionario) de la disposición de los usuarios hacia el sistema.

3.7.5. Facilidad de mantenimiento

Se han propuesto métricas diseñadas explícitamente para actividades de mantenimiento. El estándar IEEE 982.1 – 1988 [IEE94] sugiere un índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto software (basada en los cambios que ocurren con cada versión del producto). Se determina la siguiente información:

M_T = Número de módulos en la versión actual.
F_c = Número de módulos en la versión actual que se han cambiado.
F_a = Número de módulos en la versión actual que se han diseñado.
F_d = Número de módulos de la versión anterior que se han borrado en la versión actual.

El índice de madurez del software se calcula de la siguiente manera.

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

A medida que el IMS se aproxima a 1.0, el producto se empieza a estabilizar. El IMS puede emplearse también como métrica para la planificación de las actividades del mantenimiento del software. El tiempo medio para producir una versión de un producto software puede correlacionarse con el IMS desarrollándose modelos empíricos para el mantenimiento.

3.7.6. Prueba del camino Básico

La prueba del camino básico es una técnica de prueba de caja blanca. El método permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

- **Complejidad Ciclomática**

  La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un limite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

  La complejidad ciclomática está basada en la teoría de grafos y nos da una métrica del software extremadamente útil, y se puede calcular de la siguiente forma:

  La complejidad ciclomática, *V(G)*, de un grafo de flujo *G* se define como:

  $$V(G) = A - N + 2$$

  donde:

  A: Es el número de aristas del grafo de flujo .
  N: Es el número de nodos del mismo.

CAPÍTULO IV

ANÁLISIS Y DISEÑO DEL SISTEMA

Aplicando la metodología RUP, a continuación se modela el: "*Sistema de Gestión y Control de Activos Fijos Universidad Privad Franz Tamayo*" (GCAF).

4.1. Requisitos

4.1.1. Modelo del negocio

Basado en un estudio preliminar de la institución, se pasa a identificar y describir cada uno de los procesos del negocio. Lo cual permite comprender toda la actividad de la institución relacionada con el sistema a implementar.

4.1.1.1. Modelos de casos de uso del negocio-Dirección de Activos Fijos

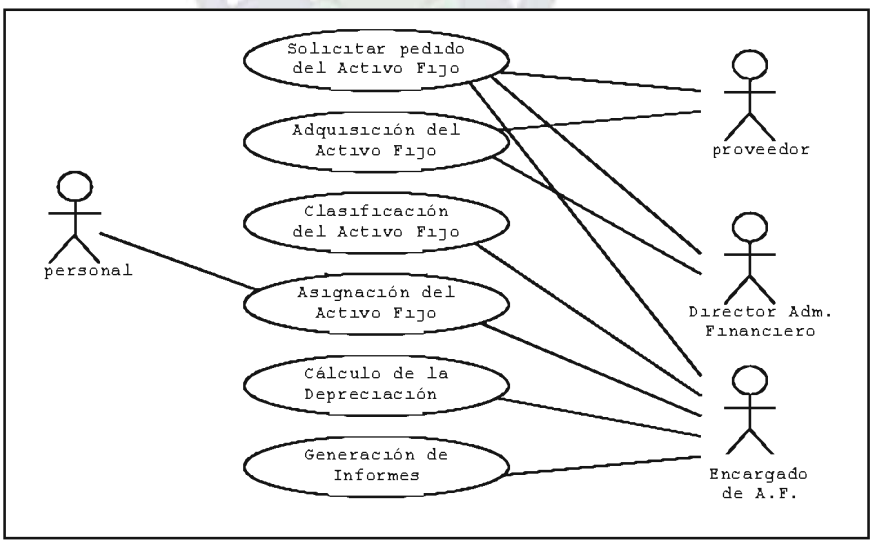


Figura 4.1. Modelo de casos de uso del negocio
Fuente: Elaboración Propia

4.1.1.2. Modelo de objetos del negocio

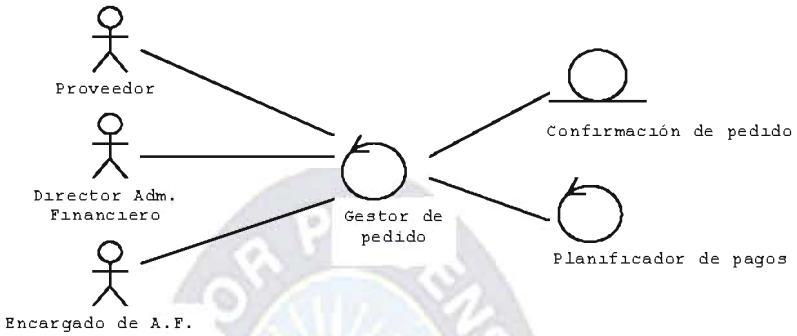


Figura 4.2. Caso de uso: solicitar pedido del Activo Fijo
Fuente: Elaboración Propia

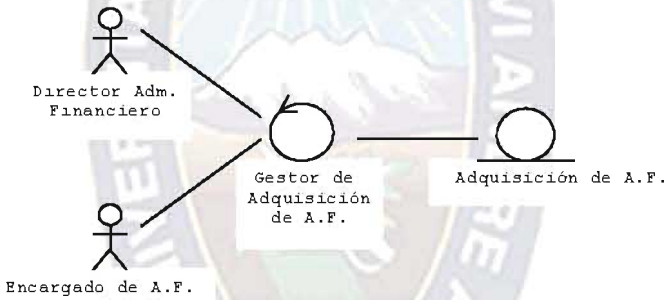


Figura 4.3. Caso de uso: adquisición del Activo Fijo
Fuente: Elaboración Propia

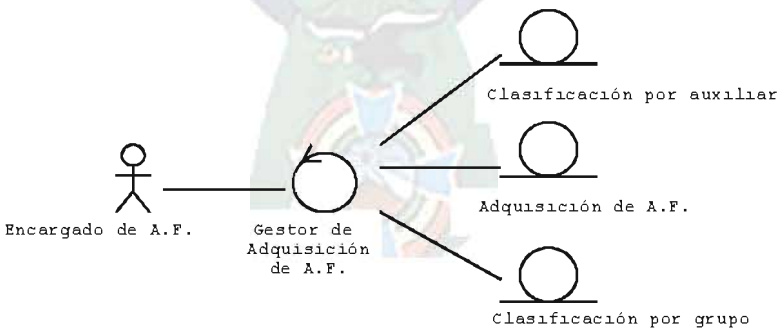


Figura 4.4. Caso de uso: clasificación del Activo Fijo
Fuente: Elaboración Propia

Para más detalles ver Anexo A.

4.1.1.3. Requisitos del Sistema

En la Dirección de Activos Fijos de la Universidad Privada Franz Tamayo (U.P.F.T.), el personal y los encargados de los Activos Fijos; para realizar la gestión de los Activos Fijos, requieren tener información actualizada y disponible.

A continuación se mencionan los requisitos y necesidades de los usuarios identificados:

- Manejo eficaz sobre la ubicación, registro, asignación, depreciación, revalúo y bajas de los Activos Fijos.
- Generación de informes sobre la depreciación, el estado, revalúo y otros de los Activos Fijos.
- Seguimiento y control de la información respecto a cambios de las características y detalles de los Activos Fijos.
- Generación de informes para los cierres de gestión.
- Emitir informes de manera eficaz y concisa.
- Procedimientos para la obtención de información sobre el registro de los Activos Fijos, asignación, devolución, baja y sus responsables respectivamente.
- Manejo de información de manera eficaz y concisa.

Una vez descrito los requerimientos del usuario, se procede a la descripción de las funciones del sistema y cada uno de los módulos que este comprenderá para satisfacer las necesidades y requerimientos del usuario.

4.1.1.3.1. Funciones del Sistema

- Implementar una Base de Datos para los Activos Fijos.
- Establecer una interfaz de comunicación con los módulos del sistema.
- Control de acceso de usuarios.
- Realizar el control y seguimiento de cada uno de los activos con respecto a su ubicación y responsables.

Todas las funciones descritas que son realizadas por el sistema se implementan en los siguientes módulos:

- Modulo de registro del Activo Fijo.
- Modulo de asignación y clasificación de Activos Fijos.
- Modulo de devolución de Activos Fijos.
- Modulo de baja de Activos Fijos.
- Modulo de revalúo de Activos Fijos.
- Modulo de depreciación de Activos Fijos.
- Modulo de registro de tipo de cambio de moneda.
- Modulo de emisión de reportes.

4.1.1.4. Captura de requisitos como caso de uso

En esta etapa se establecerá la captura de requisitos con el modelo de casos de uso que incluye los casos de uso y los actores del sistema.

4.1.1.4.1. Actores del Sistema

- *Encargado de Activos Fijos*: Tiene la misión de controlar todo el movimiento de los activos existentes en la U.P.F.T., desde su registro, asignación, baja, etc. A su vez también está a su cargo el cálculo de las depreciaciones y emitir los reportes.
- *Director Administrativo y Financiero*: Es aquél a quién se emite los reportes sobre el movimiento y estado de los activos.
- *Personal:* Es el personal que pertenece a distintos ambientes de la U.P.F.T., es aquel a quien se le asigna el activo y dispone del mismos para la realización de sus funciones y responsable directo del activo.

- *Administrador del sistema:* Es el personal que se encuentra en la Dirección de Sistemas Informáticos y Redes de la U.P.F.T., que se encarga de administrar la Base de Datos y el control de accesos al sistema.

4.1.1.4.2. Casos de uso del Sistema

- *Registro del tipo de cambio de la moneda*: En este caso de uso se realiza el registro del tipo de cambio de la moneda actual.
- *Incorporación del Activo Fijo*: En este caso de uso se hace el registro del Activo Fijo y a su vez se clasifica según al grupo contable al que pertenece.
- *Clasificación del Activo Fijo*: En este caso de uso se realiza la clasificación del activo según al grupo contable al que pertenece.
- *Asignación del Activo Fijo*: En este caso de uso se realiza la asignación del Activo Fijo a un nuevo responsable.
- *Devolución del Activo Fijo*: En este caso de uso es cuando el responsable del activo devuelve el mismo a la Dirección de Activo Fijos.
- *Depreciación del Activo Fijo*: En este caso de uso es cuando el encargado de Activos Fijos realiza el cálculo de la depreciación de los activos.
- *Baja del Activo Fijo*: En este caso de uso se realiza la baja del activo, de acuerdo a una evaluación técnica.
- *Generación de reportes*: En este caso de uso se muestra los reportes de los procesos que se tiene con los activos.
- *Administrador de la Base de Datos*: En este caso de uso se realiza la administración de la Base de Datos, obteniendo backups de respaldo por parte de la Dirección de Sistemas Informáticos y Redes

4.1.1.4.3. Modelo de casos de uso del Sistema

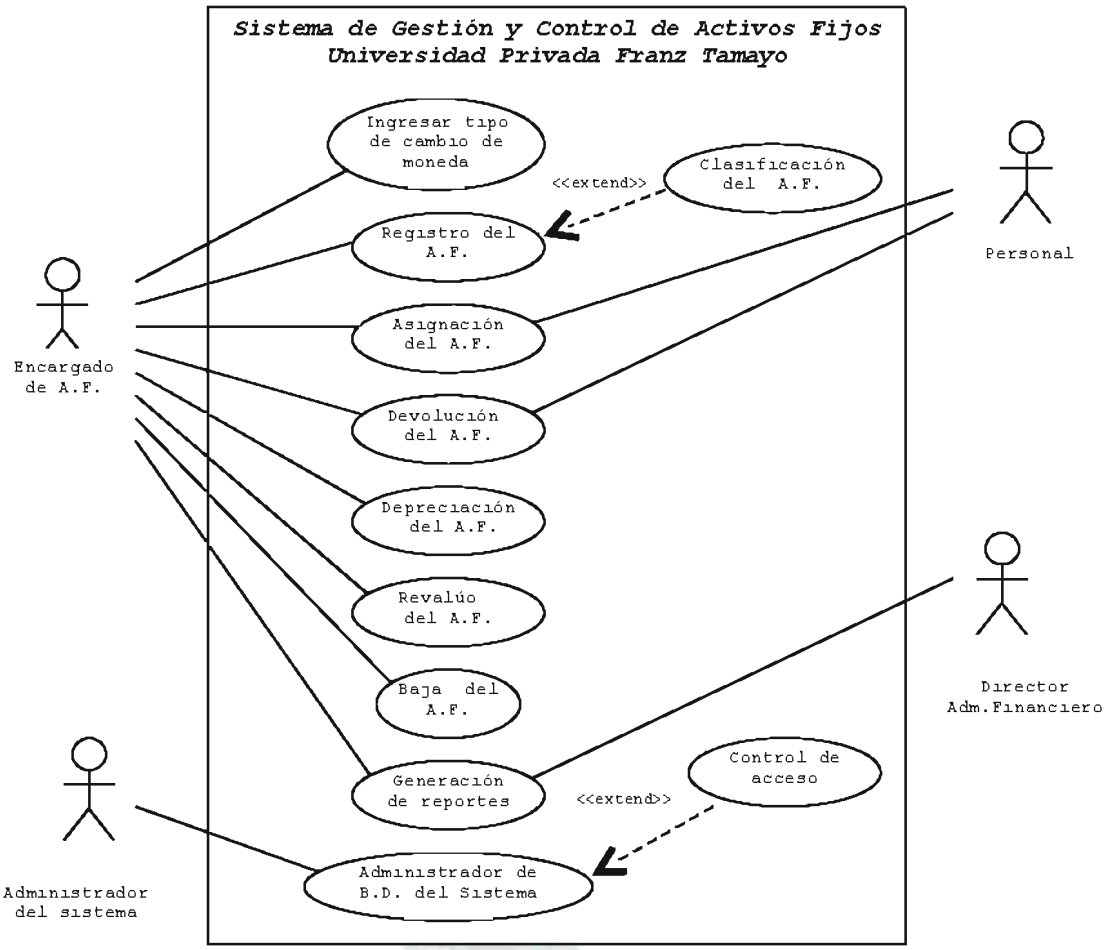


Figura 4.5. Modelo de casos de uso del sistema
Fuente: Elaboración Propia

4.1.1.4.4. Casos de Uso del Sistema

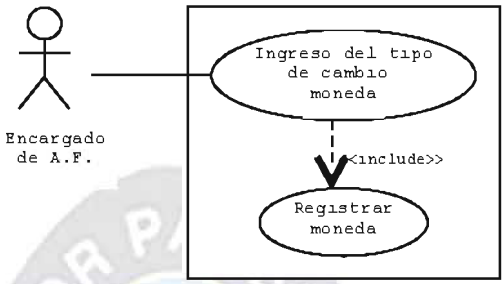


Figura 4.6. Caso de uso: ingreso de tipo de cambio moneda
Fuente: Elaboración Propia

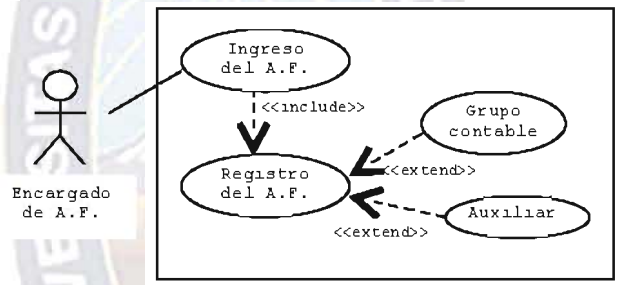


Figura 4.7. Caso de uso: registro del Activo Fijo
Fuente: Elaboración Propia

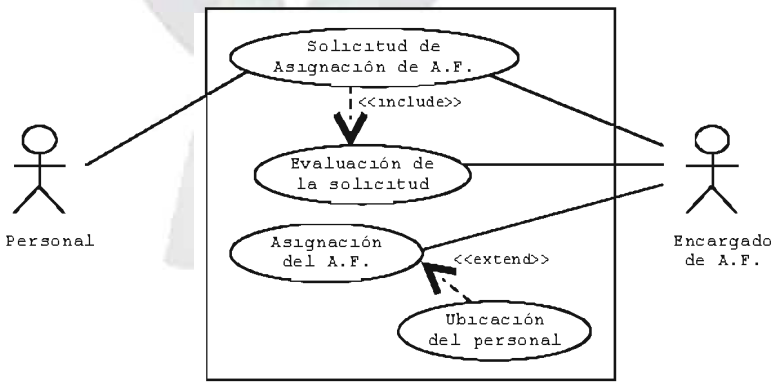


Figura 4.8. Caso de uso: asignación del Activo Fijo
Fuente: Elaboración Propia

Para más detalles ver Anexo B.

4.2. Inicio

4.2.1. Análisis

4.2.1.1. Análisis de la arquitectura

4.2.1.1.1. Identificación de paquetes de análisis a partir de los casos de uso

La identificación de paquetes de análisis a partir de los casos de uso proporciona un medio para organizar el modelo de análisis en piezas más pequeñas y manejables.

Figura 4.9. Registro tipo de cambio de moneda

Fuente: Elaboración Propia

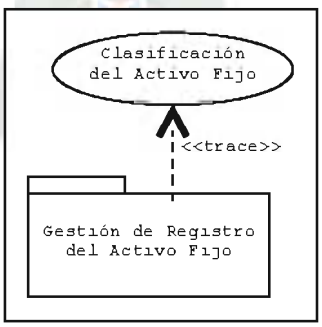


Figura 4.10. Registro del Activo Fijo

Fuente: Elaboración Propia

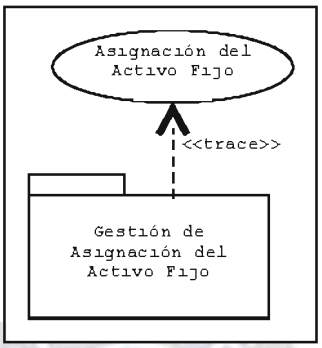


Figura 4.11. Asignación del Activo Fijo
Fuente: Elaboración Propia

Para más detalles ver Anexo C.

4.2.1.2. Análisis de los casos de uso

Aquí se realiza el *refinamiento de casos de uso*. Refinamos cada caso de uso como colaboración de clases del análisis.

4.2.1.2.1. Diagrama de colaboración

Analizamos cada caso de uso para describir como interactúan sus correspondientes objetos de análisis.

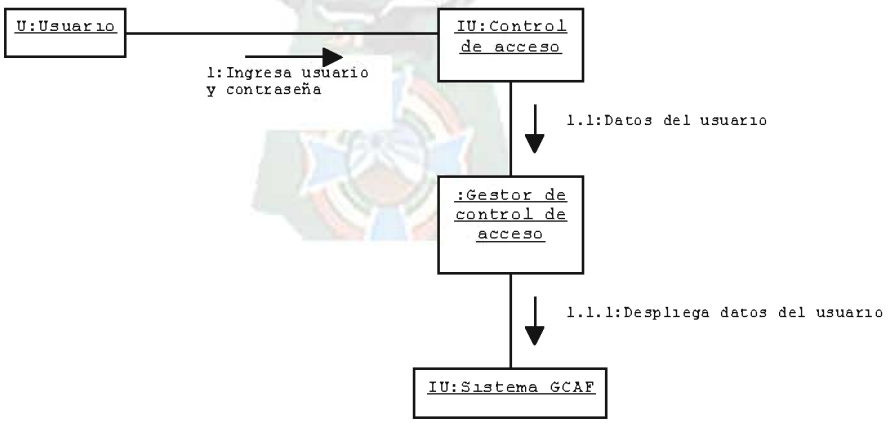


Figura 4.12. Diagrama de colaboración: acceso al Sistema GCAF
Fuente: Elaboración Propia

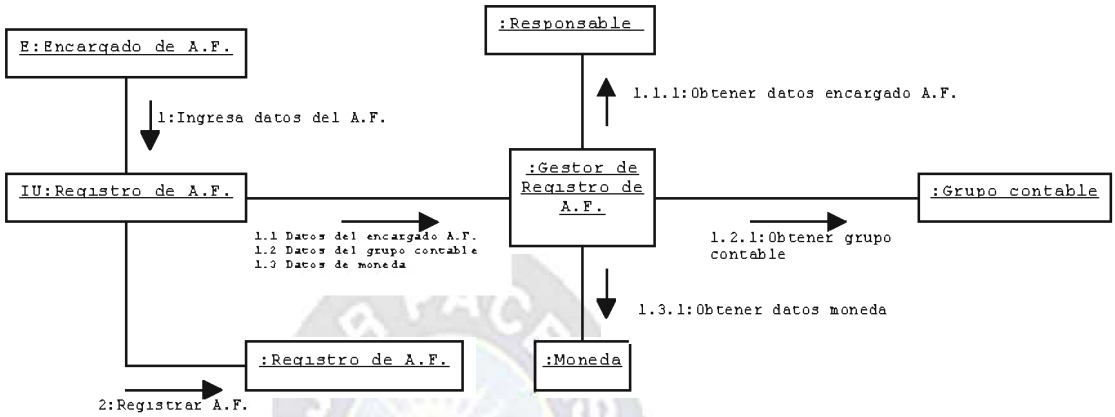


Figura 4.13. Diagrama de colaboración: registro del Activo Fijo
Fuente: Elaboración Propia

Figura 4.14. Diagrama de colaboración: asignación del Activo Fijo
Fuente: Elaboración Propia

Para más detalles ver Anexo D.

4.3. Elaboración

4.3.1. Diseño

El objetivo del diseño es esbozar los modelos de diseño y despliegue y su arquitectura. Modelamos el sistema y encontramos su forma para que soporte todo los requisitos.

4.3.1.1. Diagrama de despliegue

El diagrama de despliegue describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo que se identifico durante la implementación.

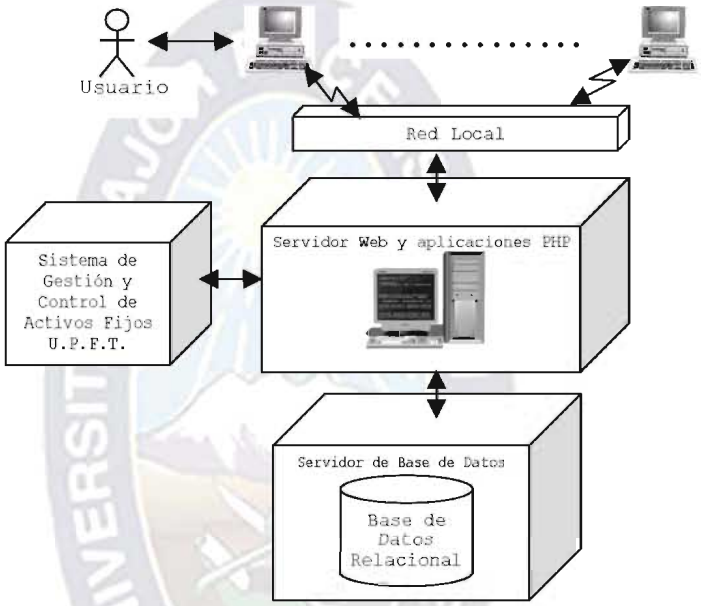


Figura 4.15. Diagrama de despliegue: Sistema GCAF
Fuente: Elaboración Propia

4.3.1.2. Diagrama de secuencia

Para describir como interactúan los objetos del diseño, se lo realiza mediante diagramas de secuencia que contienen las instancias de los actores, los objetos de diseño, y la transmisiones de mensajes entre éstos, que participan en el caso de uso.

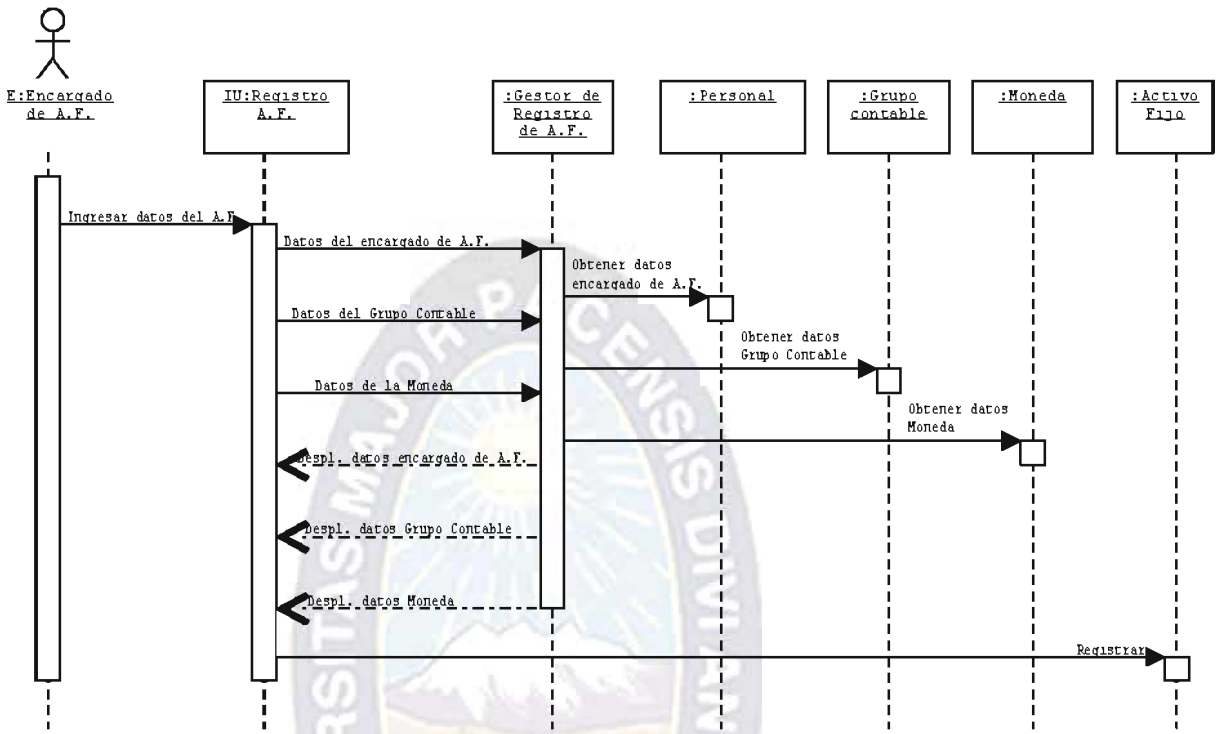


Figura 4.16. Diagrama de secuencia: registro del Activo Fijo
Fuente: Elaboración propia

Para más detalles ver Anexo E.

4.3.1.3. Diseño de clases

El propósito de diseñar una clase del sistema es crear una clase de diseño que cumpla su papel en las realizaciones de los casos de uso.

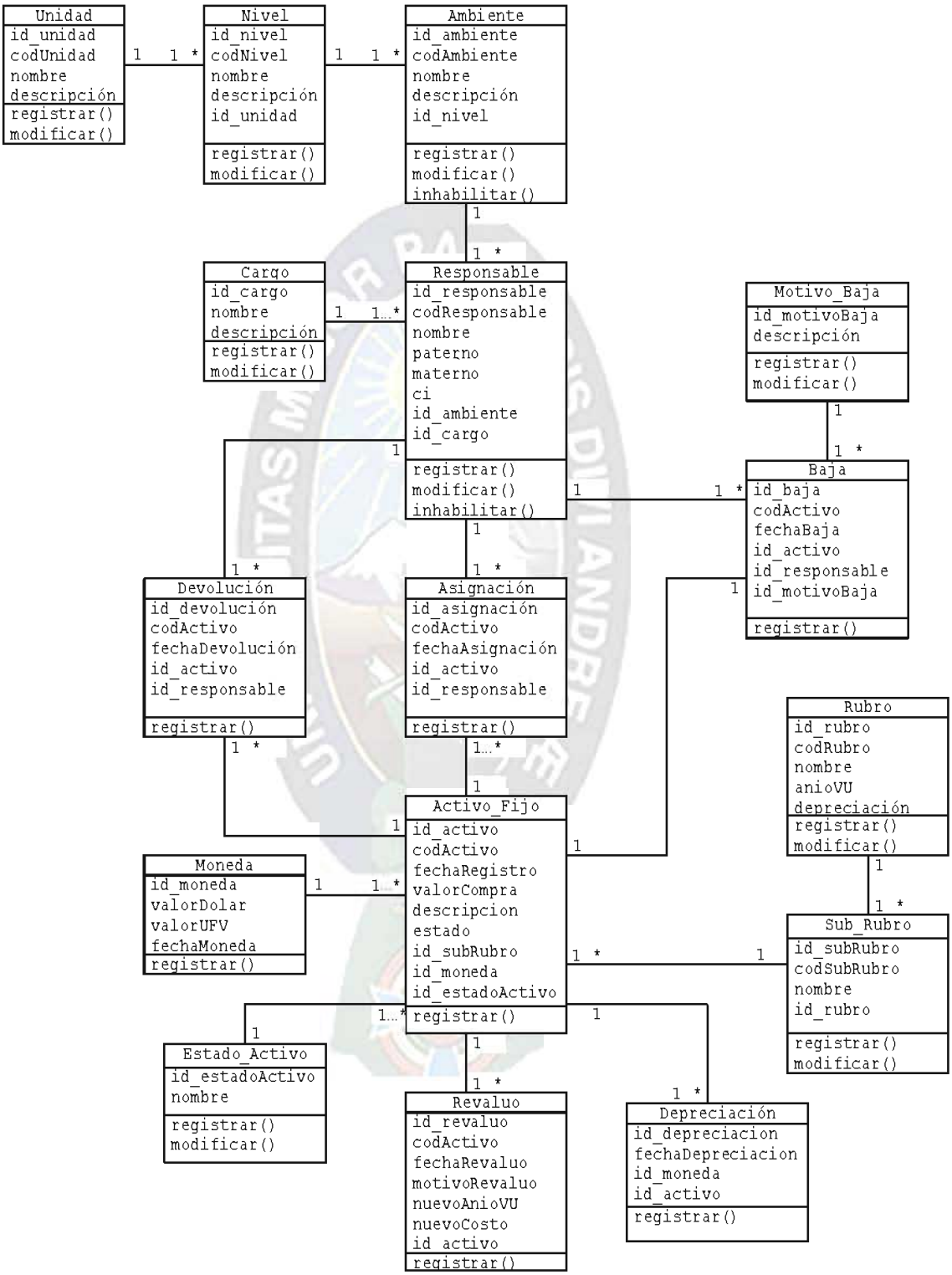


Figura 4.17. Diseño de clases: Sistema GCAF
Fuente: Elaboración propia

4.3.1.4. Diagrama de estados

La utilización de diagrama de estados significa describir las diferentes transacciones de estado de un objeto de diseño del objeto.

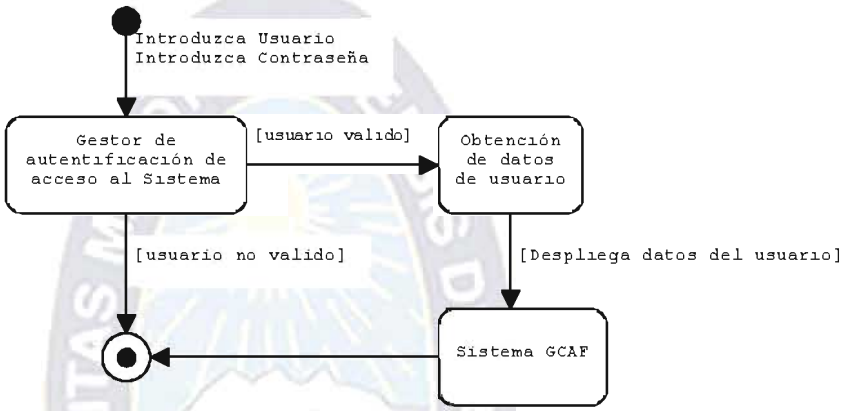


Figura 4.18. Diagrama de estados: acceso al Sistema GCAF
Fuente: Elaboración propia

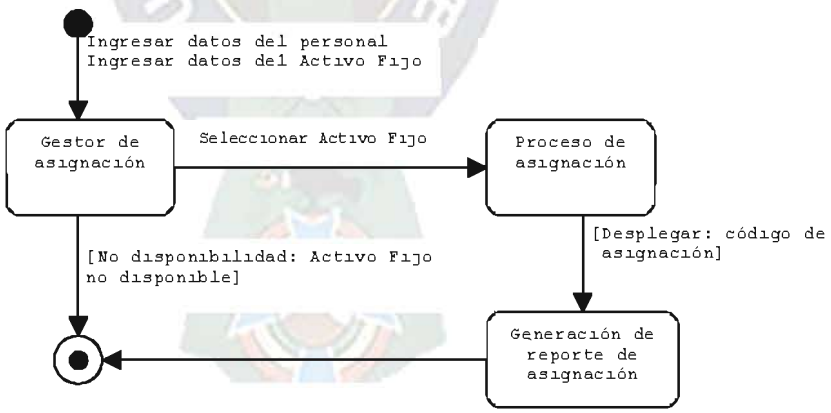


Figura 4.19. Diagrama de estados: asignación del Activo Fijo
Fuente: Elaboración propia

Para más detalles ver Anexo F.

4.4. Construcción

4.4.1. Implementación

La implementación describe cómo los elementos del modelo de diseño, se implementarán en términos de componentes, como ficheros de código fuente, ejecutables, etc.

4.4.1.1. Diagrama de componentes

El diagrama de componentes define el código fuente de uno o varios componentes, garantizando que cada componente implementa la funcionalidad correcta.

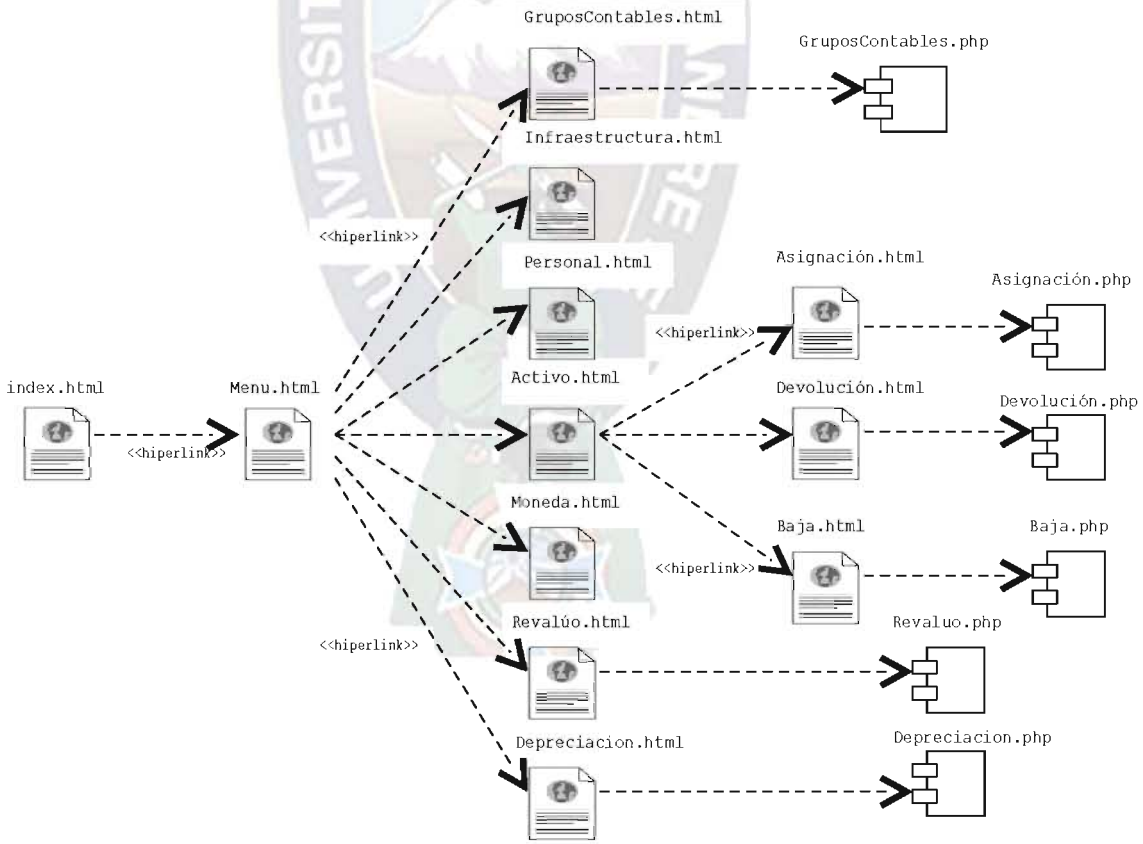


Figura 4.20. Diagrama de componentes: Sistema GCAF
Fuente: Elaboración propia

4.4.1.2. Diseño de la interfaz de usuario

La interfaz de usuario fue diseñada para darle al usuario el control directo sobre el sistema. El objetivo es satisfacer las necesidades de todos los usuarios potenciales, adaptando la tecnología Web a fin de cubrir sus expectativas.

Para el diseño de la interfaz se tomo en cuenta los siguientes componentes de una interfaz Web:

- **El dintel**, el dintel es un elemento común en las páginas Web que bien utilizado se convierte en un elemento identificador del sitio, aportando al mismo un estilo propio que lo distingue e identifica. Se entiende por dintel una zona de la interfaz Web situada en la parte superior de la misma (de ahí su nombre), de anchura generalmente igual a la de la página y altura variable, en la que se ubica generalmente el logotipo del sitio Web o de la empresa propietaria, acompañado generalmente de un texto identificador de la misma.

- **Los elementos de navegación**, los elementos de navegación son, los elementos de una interfaz que permiten la navegación por las diferentes secciones y páginas que componen el sitio Web. Generalmente se presentan como menús formados por diferentes opciones, con las que el usuario puede interactuar.

Su ubicación en la interfaz debe permitir un cómodo acceso a las opciones que lo forman, pero sin llegar a estorbar al resto de elementos de la misma. Los menús tipo lista y los de árbol se sitúan generalmente en la zona lateral izquierda de la página, mientras que los de cortinilla suelen ocupar una franja horizontal bajo el dintel (si lo hay). La zona lateral izquierda como lugar para situar el menú de navegación se ha convertido en un estándar en el diseño web.

- **El cuerpo de una página**, El cuerpo de la página es la parte de la interfaz web que presenta a los usuarios información específica sobre un tema concreto. Al ser la parte más importante de la interfaz, el espacio destinado a ella debe ser el mayor de todos, ocupando generalmente entre el 50% y el 85% del total. Su ubicación es siempre central, bajo el dintel (si lo hay) y al lado del menú lateral de navegación (si lo hay).

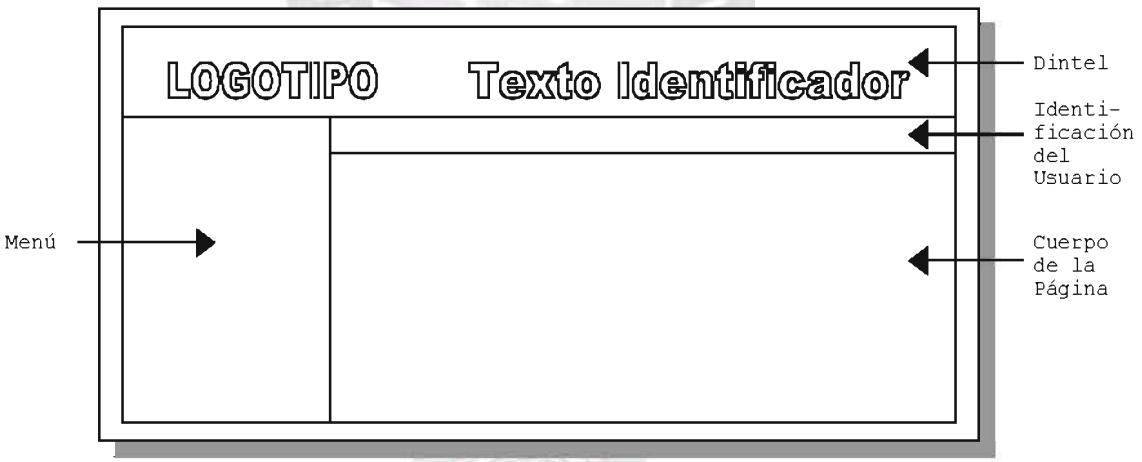


Figura 4.21. Diseño de la interfaz de usuario
Fuente: Elaboración propia

4.4.1.3. Interfaz de usuario

El diseño de la interfaz de usuario fue desarrollado de manera transparente al usuario, de tal manera que puede ser utilizado incluso por personas que no tengan conocimiento del área.

Presentación e Ingreso al Sistema GCAF, nos muestra la presentación inicial del sistema donde el usuario debe introducir el *usuario* y su *contraseña*.

Figura 4.22. Interfaz de usuario: pantalla inicial
Fuente: Elaboración propia

Menú de Opciones Sistema GCAF, nos muestra las opciones que puede elegir el administrador de Activos Fijos.

Figura 4.23. Interfaz de usuario: menú de opciones del Sistema GCAF
Fuente: Elaboración propia

Interfaz de Registro del tipo de cambio de la Moneda, nos muestra el formulario donde el administrador de Activos Fijos introduce el tipo de cambio del dólar y la UFV.

Figura 4.24. Interfaz de usuario: registro del tipo de cambio de la moneda

Fuente: Elaboración propia

Cuadro general de depreciaciones de los Activos Fijos, nos muestra el cuadro general de depreciaciones de los Activos Fijos de acuerdo al grupo contable.

Figura 4.25. Interfaz de usuario: cuadro general de depreciaciones

Fuente: Elaboración propia

Interfaz de Baja del Activo Fijo, nos muestra el formulario donde el administrador de Activos Fijos realiza la baja del activo de cuerdo a determinados criterios.

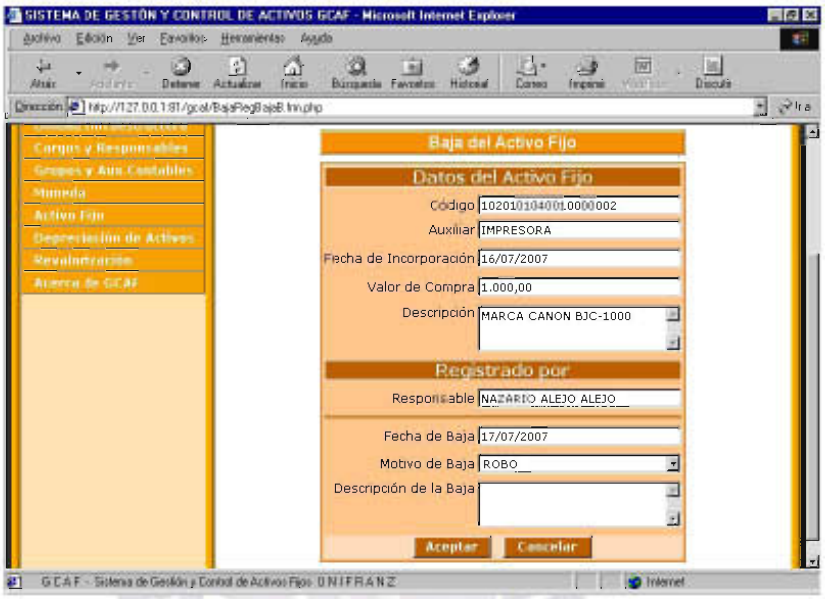


Figura 4.26. Interfaz de usuario: baja del Activo Fijo

Fuente: Elaboración propia

4.4.1.4. Pruebas de Caja Blanca

Es un método de diseño de casos de prueba. Para esta prueba utilizaremos el método de complejidad ciclomática que permite determinar el número de casos de prueba que debe ejecutarse y así garantizar las sentencias de cada módulo.

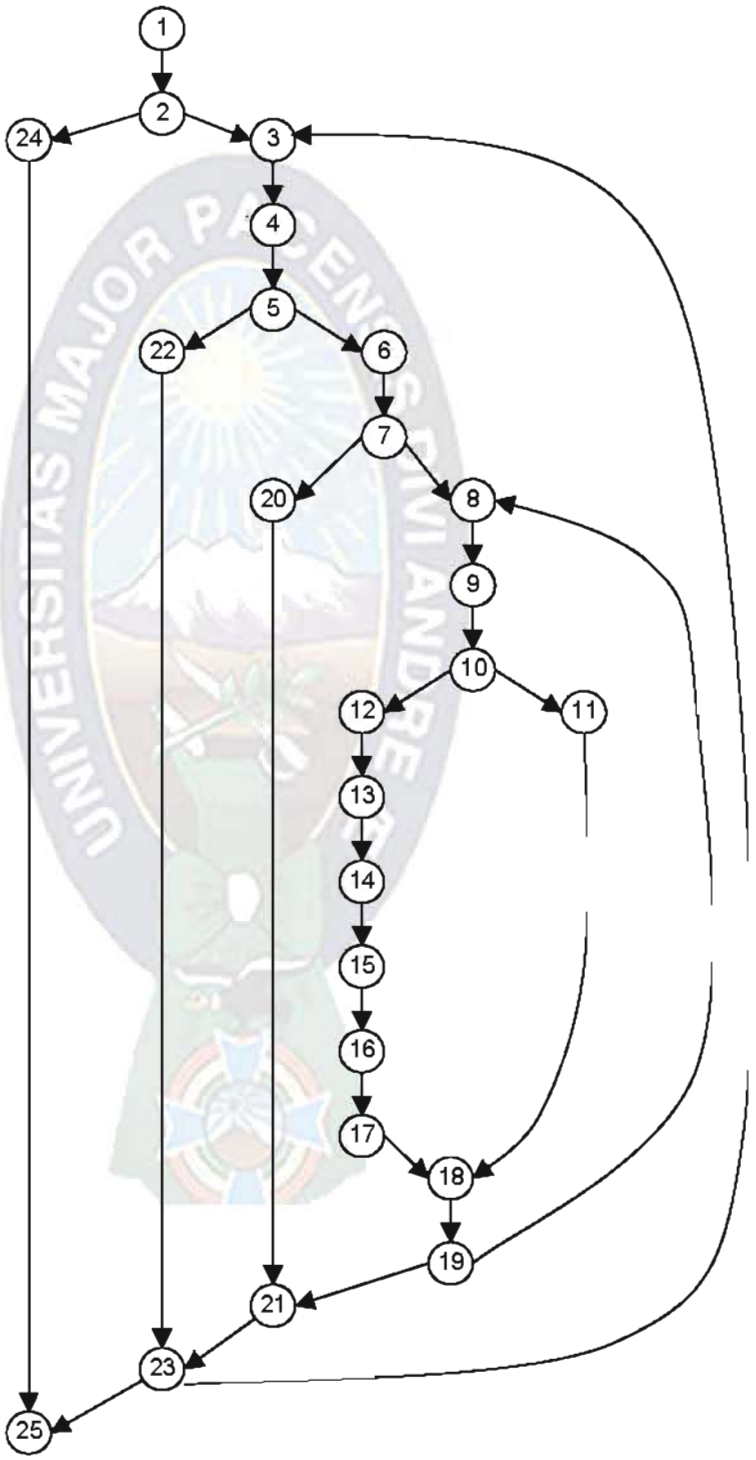
A continuación se muestra el código del módulo de Depreciación de Activos Fijos.

```
PROCEDURE_DepActivosFijos

sqlM="select id_moneda,fecha                                    1
        from  MONEDA
        where fecha=".date()." ' "
resultado=query(sqlM)                                           2
if(num_rows(resultado)!=0)
    while(datoM=get(resultado))                                 3
```

```
id_nomeda=datoM(0)                                                    4
fecha=datoM(1)
sqlD="select id_depreciación
        from DEPRECIACIÓN
        where id_moneda=".id_moneda
resultado=query(sqlD)
if(num_rows(resultado)==0)                                            5
    sqlA="select A.id_activo,RanioVU,M.fecha                          6
          from ACTIVO A,MONEDA M, SUBRUBRO SR, RUBRO R "
    resultado=query(sqlA)
    if(num_rows(sqlA)!=0)                                             7
      while(datoA=get(resultado))                                     8
        id_activo=datoA(0)                                            9
        anioVU=datoA(1)
        fecha1=datoA(2)
        fechaVal=fechas(fecha1,fecha)
        n=count(fechaVal)
        if((anioVU*365)-n>=0)                                         10
          sqlinsert="insert DEPRECIACION (".fecha.",".id_moneda.",".id_activo.")"   11
          resultado=query(sqlinsert)
        else
          fechaF=fechaVal((anioVY*365)-1)                             12
          sqlM="select id_moneda
                from MONEDA
                where fechaMoneda=".fechaF
          resultado=query(slqM)
          if(num_rows(resultado)!=0)                                  13
          datoM=get(resultado)                                        14
          id_moneda=datoM(0)
          endif
          sqlD="select D.id_activo                                    15
                from DEPRECIACION
                where D.id_activo=id_activo and D.id_moneda=id_moneda "
          resultado=query(sqlD)
          if(num_rows(resultado)!=0)                                  16
            sqlinsert="insert DEPRECIACION (".fecha.",".id_moneda.")"  17
            resultado=query(sqlinsert)
          endif
        endif                                                         18
      endwhile                                                        19
      print("Activos depreciados correctamente")
    else
      print("No existen activos registrados")                         20
    endif                                                             21
  else
    print("Error ya se realizo el calculo de la depreciación")        22
  endif                                                               23
 endwhile
else
 print("No existe tipo de cambio con la fecha de hoy")                24
endif                                                                 25
END_ DepActivosFijos
```

Usando el código del módulo anterior, dibujamos el correspondiente grafo de flujo.



Determinamos la complejidad ciclomática del grafo de flujo resultante:

$$V(G) = A - N + 2$$
$$V(G) = 30 \text{ aristas} - 25 \text{ nodos} + 2$$
$$V(G) = 7$$

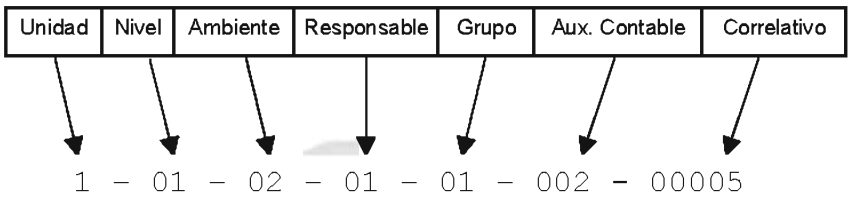
Por tanto se tiene 7 caminos linealmente independientes, el cual garantiza la ejecución de al menos una vez cada sentencia del programa y que cada condición se habrá ejecutado en sus vértices verdadera y falsa.

4.5. Descripción de Objetivos

- Para cumplir con uno de los objetivos específicos, en la generación de los códigos que tienen los Activos Fijos, se utilizo el criterio que se maneja en la Dirección de Activos Fijos que identifica la ubicación física donde estará el activo y el responsable, está codificación se realiza de acuerdo a un código único que tiene cada uno de los involucrados de acuerdo a lo siguiente:

  Código Unidad: identifica la sede, caso La Paz.
  Código Nivel: identifica el nivel de la sede, caso 1er Piso.
  Código Ambiente: identifica el ambiente de la sede, caso Rectorado.
  Código Responsable: identifica al responsable que pertenece a un ambiente, caso jefe de área y/o auxiliar de oficina.
  Código Grupo : identifica el grupo contable al que pertenece el activo,caso Muebles y Enseres.
  Código Auxiliar Contable: identifica el auxiliar contable al que pertenece el activo, caso silla, escritorio, etc.
  Correlativo: identifica el correlativo que tiene el activo fijo.

A continuación se muestra el código generado:



Código del Activo Fijo

- Para mejorar el cálculo de las depreciaciones, se aplico el método de la *Línea Recta* que se describió en el Capítulo 3, permite hacer una mejor distribución en cuanto a las depreciaciones en diferentes periodos de acuerdo a su vida útil.

A continuación se muestra la Figura que involucra el calculo, utilizando el método de la Línea Recta.

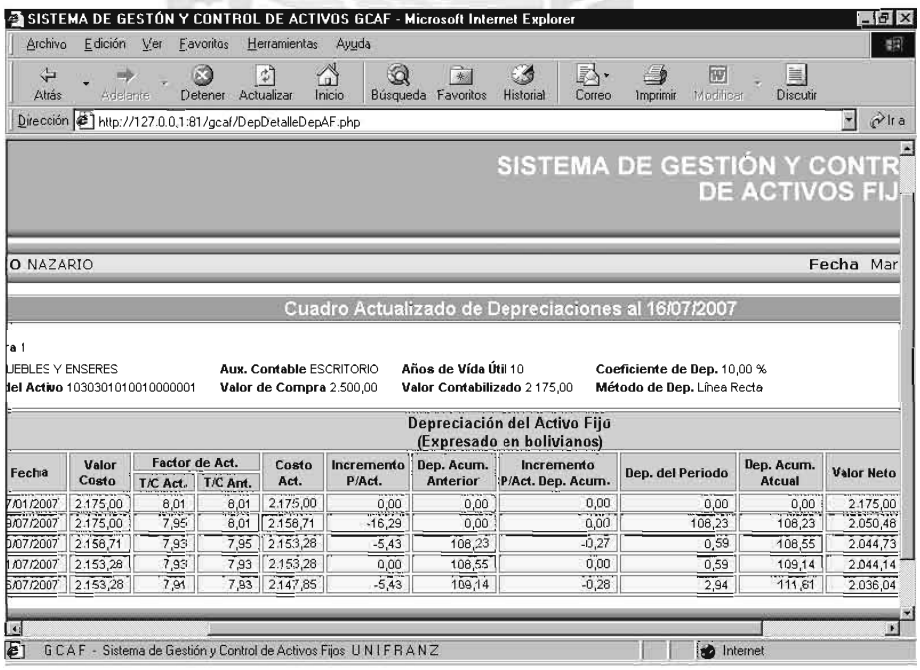


Figura 4.27. Cuadro de Depreciaciones

Fuente: Elaboración propia

CAPÍTULO V

CALIDAD DEL SOFTWARE

A continuación se describe los factores de calidad con el objeto de evaluar la calidad del software.

5.1. Confiabilidad

La confiabilidad de un sistema es un elemento importante en su calidad general. Para determinar la confiabilidad se toma en cuenta las fallas que se produce en el sistema en un tiempo determinado. Primeramente se considera la confiabilidad de cada módulo independientemente.

Para ello se requiere hallar el "modelo del sistema", mediante funciones de transferencia el cual nos permite delimitar el comportamiento del mismo, en cada uno de sus módulos.

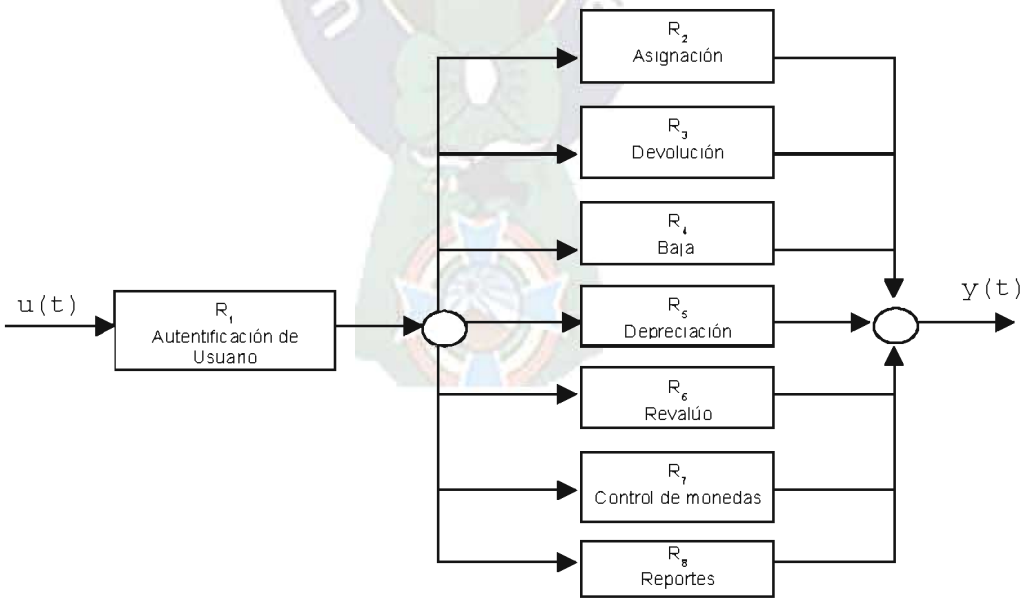


Figura 5.1. Modelo del Sistema GCAF

Fuente: Elaboración propia

Tomando en cuenta la siguiente relación: $R(t) = e^{-\lambda t}$

Se realiza el cálculo de la confiabilidad de cada módulo del sistema.

	Módulo	λ	T	R(t)
1	Autentificación del Usuario	0,03	2 horas	0,94
2	Asignación	0,03	5 horas	0,86
3	Devolución	0,03	5 horas	0,86
4	Baja	0,03	1 hora	0,97
5	Depreciación	0,07	3 horas	0,81
6	Revalúo	0,07	2 horas	0,86
7	Control de Monedas	0,03	1 hora	0,97
8	Reportes	0,02	3 horas	0,94

El modelo de sistema nos muestra una conexión compuesta donde se tiene inicialmente una conexión en serie y posteriormente un conexión en paralelo.

Por lo tanto realizando los cálculos correspondientes tenemos:

$$R = R_I * R_S$$

en donde: $R_I = R_1 = 0.94$

$$R_S = 1 - \{ [1-R_2(t)] * \ldots\ldots\ldots\ldots * 1-R_8(t)]\}$$

$$R_S = 1 - (0.14)*(0.14)*(0.03)*(0.19)*(0.14)*(0.03)*(0.06)$$

$$R_S = 0.999999971$$

$$R = [(0.97)*(0.999999971)] = 0.94$$

$$\%R = \mathbf{94}\%$$

De acuerdo con el resultado obtenido podemos decir que el "*Sistema de Gestión y control de Activos Fijos Universidad Privada Franz Tamayo*", presenta una confiabilidad del 94%, entonces podemos afirmar que es un sistema confiable.

5.2. Funcionalidad

La funcionalidad se valora evaluando el conjunto de características y capacidades del sistema. Para determinar la funcionalidad del sistema mediante la métrica *punto de función*, se debe determinar cinco característica de dominios de información y se

proporciona las cuentas en la posición apropiada a la tabla. Los valores de los dominios de información se definen de la forma siguiente:

- Número de entradas de usuario.
- Número de salidas de usuario.
- Número de peticiones de usuario.
- Número de archivos.
- Número de interfaces externas.

A continuación se realiza el conteo de los parámetro mencionados.

Tabla 5.1. Conteo de parámetros de Punto de Función

Entradas de usuario	3
Salidas de usuario	8
Peticiones de usuario	7
Archivos	1
Interfaces externas	2

Una vez recopilado los datos se le asocia un valor de complejidad que se muestra en la siguiente tabla.

Tabla 5.2. Cálculos de Punto de Función

Parámetros de Medición	Factor de Ponderación				
	Cuenta	Simple	Medio	Complejo	
Número de entradas de usuario	3	3	4	6	12
Número de salidas de usuario	8	4	5	7	40
Número de peticiones de usuario	7	3	4	6	28
Número de archivos	1	7	10	15	10
Número de interfaces externas	2	5	7	10	14
Cuenta Total					104

A continuación se obtiene los "valores de ajuste de la complejidad" según las respuestas a las siguientes preguntas.

Tabla 5.3. Tabla de valores de ajuste de complejidad

		No importante	Incidental	Moderado	Medio	Significativo	Esencial
	Factor	0	1	2	3	4	5
1	¿Requiere el sistema copias de seguridad y de recuperación fiables?						X
2	¿Se requiere comunicación de datos?					X	
3	¿Existen funciones de procesamiento distribuido?	X					
4	¿Es crítico el rendimiento?					X	
5	¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?					X	
6	¿Requiere el sistema entrada de datos interactiva?					X	
7	¿Requiere la entrada de datos interactiva, que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?					X	
8	¿Se actualizan los archivos maestros de forma interactiva?					X	
9	¿Son complejas las entradas, las salidas, los archivos o las peticiones?			X			
10	¿Es complejo el procesamiento interno?			X			
11	¿Se ha diseñado el código para ser reutilizable?					X	
12	¿Están incluidas en el diseño la conversión y la instalación?	X					
13	¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?		X				
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?						X

Entonces:

$$\sum F_i = F_1 + F_2 + \ldots\ldots\ldots\ldots\ldots\ldots + F_{14}$$

$$\sum F_i = 5 + 4 + 0 + 4 + 4 + 4 + 4 + 4 + 2 + 2 + 4 + 0 + 1 + 5$$

$$\sum F_i = 43$$

Reemplazando en la relación siguiente:

$$PF = Cuenta\_Total * [X + Min(Y) * \sum F_i]$$

X: confiabilidad del sistema, es igual a la confiabilidad halla anteriormente
X = 0,94

Min(Y): error mínimo aceptable
Min(Y) = 0,01

Cuenta_total = 104

$\sum F_i$ = 43

Reemplazándolo:

PF = 104 * [0,94 + 0.01 * 43]

PF = 143

Este PF supone 143 líneas de código.

Si se considera la $\sum F_i$ = 70, considerando el 100%, reemplazando en la relación anterior se tiene:

$$PF_{max} = 143 * [0,94 + 0,01 * 70]$$

$$PF_{max} = 235$$

Por lo tanto la funcionalidad esta dado por:

Funcionalidad = PF / PF_{max} = 143/ 235

Funcionalidad = 0.61

%Funcionalidad = 0.61 * 100 = **61**%

Esto quiere decir que la funcionalidad del sistema es del 61%.

5.3. Mantenibilidad

Para determinar el grado mantenibilidad del software. El estándar IEEE 982.1-1988 [IEE94], sugiere un índice de madurez del software(IMS).

El índice de madurez del software se calcula de la siguiente manera:

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

$$IMS=[\ 8 - (0+1+0)] / 8$$

$$IMS= 0.87$$

$$\%IMS= \mathbf{87}\%$$

Por lo tanto el software es maduro en un 87% y es estable. Lo que nos indica la facilidad de mantenimiento con la que se puede corregir el software si es que se puede encontrar un error o se puede adaptar si su entorno cambia o mejorar de acuerdo a los requerimientos del usuario.

5.4. Portabilidad

La portabilidad del software se enfoca en tres aspectos: Hardware del Servidor, Sistema Operativo y Software Servidor.

Hardware del servidor	Pentium IV o superior Velocidad 1 GHz o más Memoria RAM 256 MB o más Disco Duro 40 GB o más Tarjeta de Red
Sistema Operativo	Microsoft Windows: Win 9X, Win 2000, Win 2003, Win XP. GNU Linux: Redhat, Fedora Core, Susse.
Software Servidor	Apache, MySQL4 y PHP4

Por lo mencionado anteriormente el software del "*Sistema de Gestión y Control de Activos Fijos Universidad Privada Franz Tamayo*", es portable en sus diferentes entornos tanto en hardware y software por lo que se puede considerar una portabilidad del 100%.

5.5. Análisis costo-beneficio

Las formulas que permite el cálculo del costo del esfuerzo requerido del software son:

- Número de Personas requeridas par el Proyecto(NPM).

  $NPM = 1.1 * (PF)^{0.3}$

- Tiempo de Programación (TP)

  $TP = 1.25 * (NPM)^{0.5}$

Donde PF: es el valor obtenido de la medida del Punto de Función.

Calculando los valores de tiene:

PF = 143

$NPM = 1.1 * (143)^{0.3} = 4.87$ pm

$TP = 1.25 * (4.87)^{0.5} = 2.7$ meses

Por lo tanto:

El número de personas requeridas es:

NPM / TP = 4.87 / 2.7 = 2

Número de personas requeridas es igual a 2 personas.

Costo. El costo aproximado referente al tiempo en este caso hombre / maquina, considerando un promedio de 20$us por 20 día hábil de trabajo, dado un lapso de 2 meses se tiene un costo de 1200$us. Considerando que se contratan 2 personas, el costo del software desarrollado es de 2400$us.

Beneficio. El beneficio aproximado se puede calcular de la siguiente manera:

Beneficio del capital de la institución / costo total de software = 1500 / 2400 = 0.6

La relación de costo VS beneficio es del 60%, lo que indica que los beneficios son del **40**%.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- El "*Sistema de Gestión y Control de Activos Fijos Universidad Privada Franz Tamayo*", llego a su conclusión de forma satisfactoria, cumpliendo con todos los requisitos especificados en la etapa de análisis, dando lugar así al cumplimiento de su objetivo principal.
- Al obtener el producto final, el "*Sistema de Gestión y Control de Activos Fijos Universidad Privada Franz Tamayo*" se ha logrado construir un Base de Datos relacional para el sistema, el cual logra cumplir con uno de los objetivos específicos.
- Se puede afirmar que la implementación del presente sistema, ha mejorado el desempeño en la Dirección de Activos Fijos de la Universidad Privada Franz Tamayo.
- Se ha disminuido la carga de trabajo a los administradores de la Dirección de Activos Fijos.
- Por otra parte el "*Sistema de Gestión y Control de Activos Fijos Universidad Privada Franz Tamayo*" presenta ventajas en cuanto a:

  - Tiempo de respuesta óptimo en cada proceso.
  - Bajos costos y tiempo reducido en la obtención de reportes.

- En cuanto a la seguridad del sistema en un determinado nivel, se ha implementado un módulo de autentificación y control para el ingreso de usuarios, en donde se tiene restricciones a módulos, para determinados usuarios del sistema.

- De esta manera se concluye que todo los objetivos mencionados al inicio del proyecto han sido cumplidos en su totalidad.

6.2. Recomendaciones

- Se recomienda implementar el sistema en el resto de las unidades que forman parte de la Universidad Privada Franz Tamayo, para que en el futuro se tenga una Base de Datos centralizada.
- Concienciar a los administradores de Activos Fijos a emplear el uso del sistema sin prejuicios o temores.

BIBLIOGRAFÍA

[JACOB00] Jacobson, I., Booch, G. y Rumbaugh, J., "El Proceso Unificado de Desarrollo de Software", Pearson Educación, Madrid, 2000.

[BOOCH99] Booch, G., Rumbaugh, J. y Jacobson, I., "El Lenguaje Unificado de Modelado", Addison Wesley Iberoamericana, Madrid, 1999.

[RUMBA00] Rumbaugh, J., Jacobson, I. y Booch, G., "El Lenguaje Unificado de Modelo. Manual de Referencia", Pearson Educación, Madrid, 2000.

[LARMA99] Larman, G., "UML y Patrones. Introducción al análisis y diseño orientado a objetos", Prentice Hall, México, 1999.

[SILBE98] Silberschatz, A., Korth S. y Sudarshan S., "Fundamentos de Bases de Datos", 3ª ed., McGraw-Hill, Madrid, 1998.

[BOWEN00] Bowen, R. y Coar, K., "Servidor Apache al Descubierto", Pearson Educación, Madrid, 2000.

[FUNES03] Funes, J., "Contabilidad Intermedia", Sabiduría, Bolivia, 2003.

[PRESS03] Pressman, R. S., "Ingeniería del Software. Un enfoque práctico", 5ª ed., McGraw-Hill, 2003.

[EUPFT00] Estatuto Orgánico Universidad Privada Franz Tamayo, 2000.

REFERENCIAS WEB

"Diseño de Interfases"
http://www.imageandart.com/tutoriales/web_design/diseno_interfaces/index.html

"Componentes de una Interfaz Web"
http://www.desarrolloweb.com/manuales/64/

Anexos

ANEXO A

Modelo de objetos del negocio

Figura. Caso de uso: asignación del Activo Fijo
Fuente: Elaboración Propia

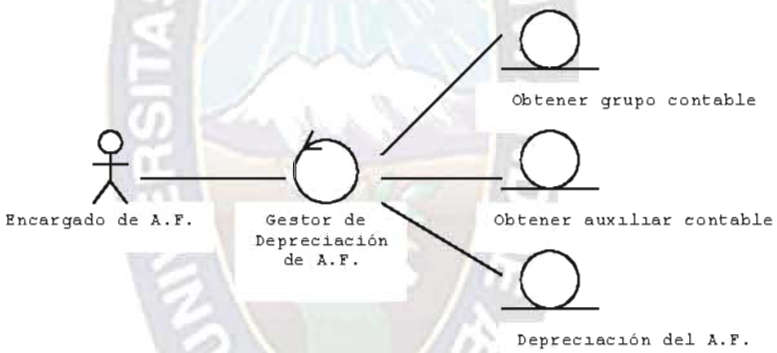


Figura. Caso de uso: cálculo de depreciación del Activo Fijo
Fuente: Elaboración Propia

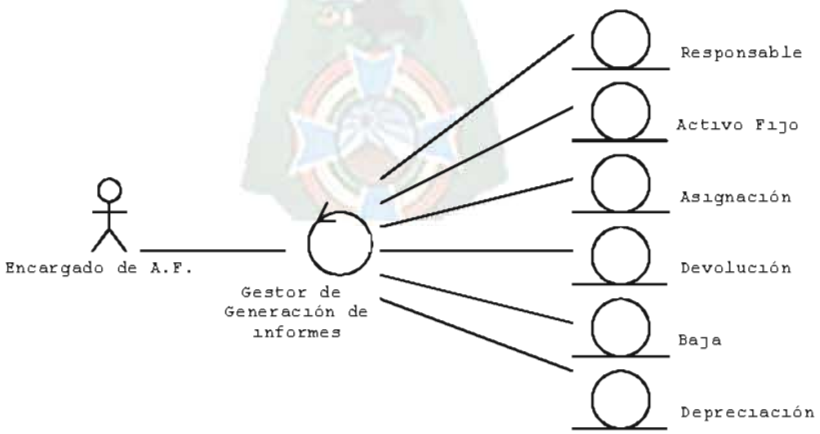


Figura. Caso de uso: generación de informes
Fuente: Elaboración Propia

ANEXO B

Casos de Uso del Sistema

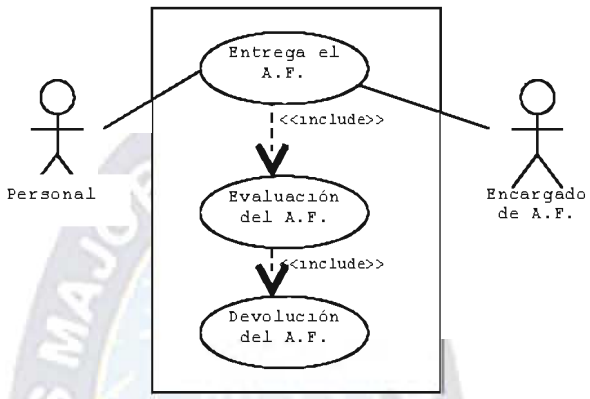


Figura. Caso de uso: devolución del Activo Fijo
Fuente: Elaboración Propia

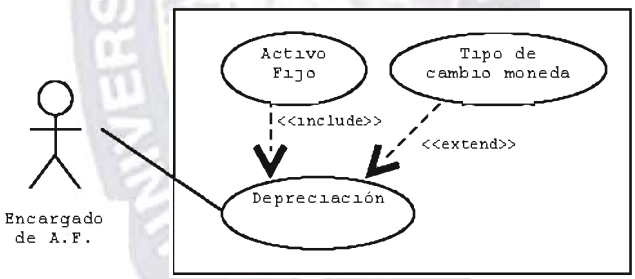


Figura. Caso de uso: depreciación
Fuente: Elaboración Propia

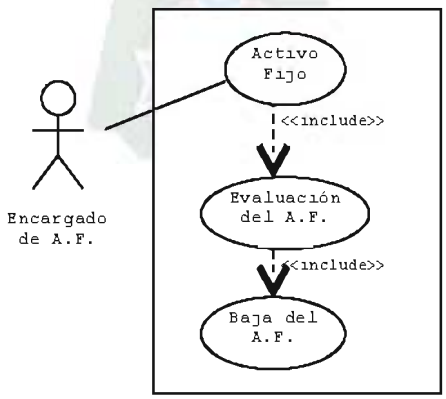


Figura. Caso de uso: baja del Activo Fijo
Fuente: Elaboración Propia

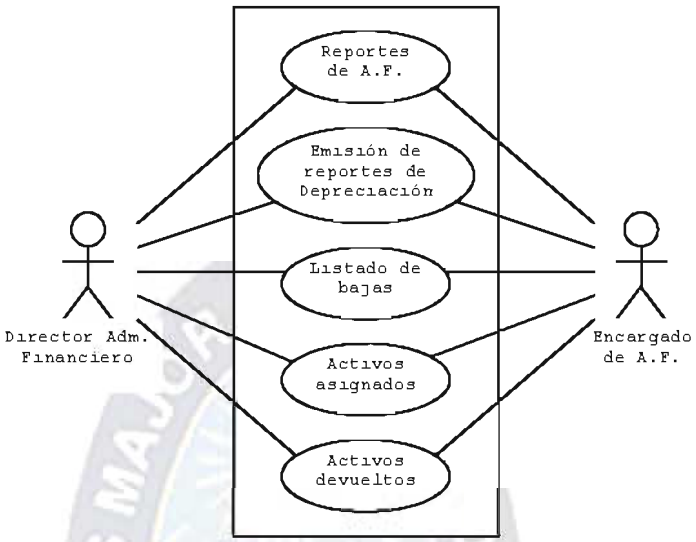


Figura. Caso de uso: generación de reportes
Fuente: Elaboración Propia

ANEXO C

Identificación de paquetes de análisis a partir de casos de uso

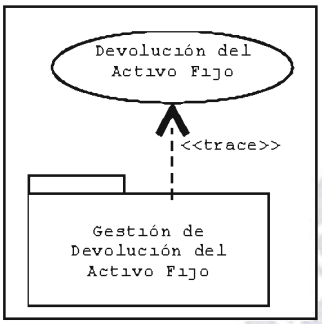


Figura. Devolución del Activo Fijo
Fuente: Elaboración Propia

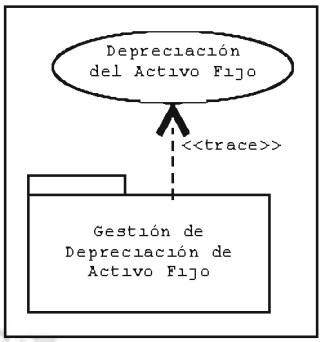


Figura. Depreciación de Activo Fijo
Fuente: Elaboración Propia

Figura. Revalúo de Activo Fijo
Fuente: Elaboración Propia

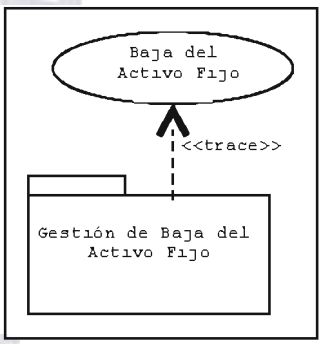


Figura. Baja de Activo Fijo
Fuente: Elaboración Propia

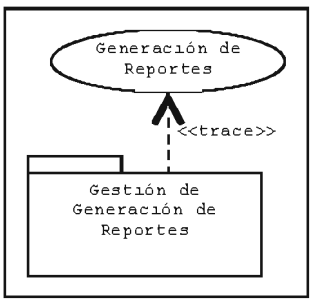


Figura. Generación de Reportes
Fuente: Elaboración Propia

ANEXO D

Diagrama de colaboración

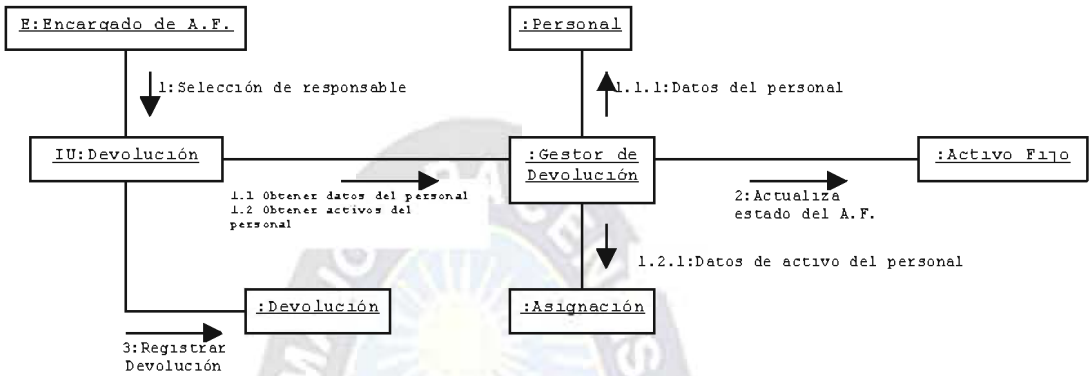


Figura. Diagrama de colaboración: devolución del Activo Fijo
Fuente: Elaboración Propia

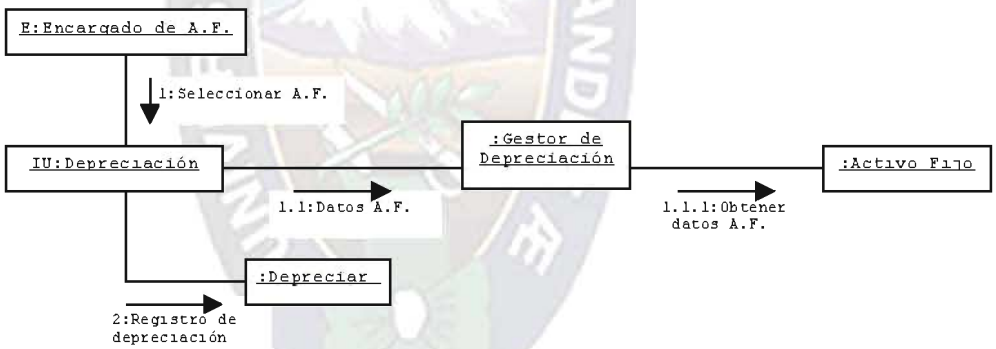


Figura. Diagrama de colaboración: depreciación de los Activos Fijos
Fuente: Elaboración Propia

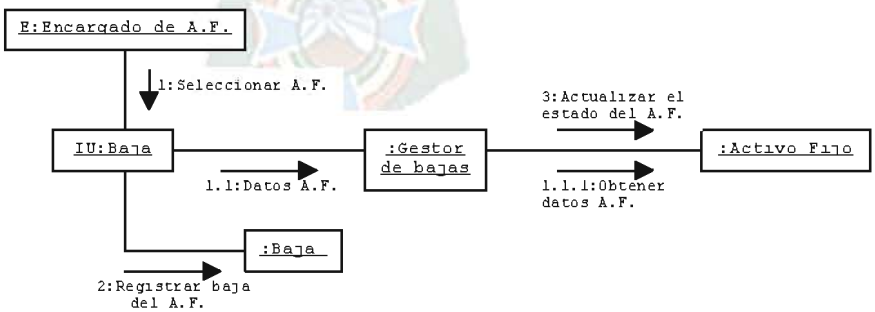


Figura. Diagrama de colaboración: baja del Activo Fijo
Fuente: Elaboración Propia

ANEXO E

Diagrama de secuencia

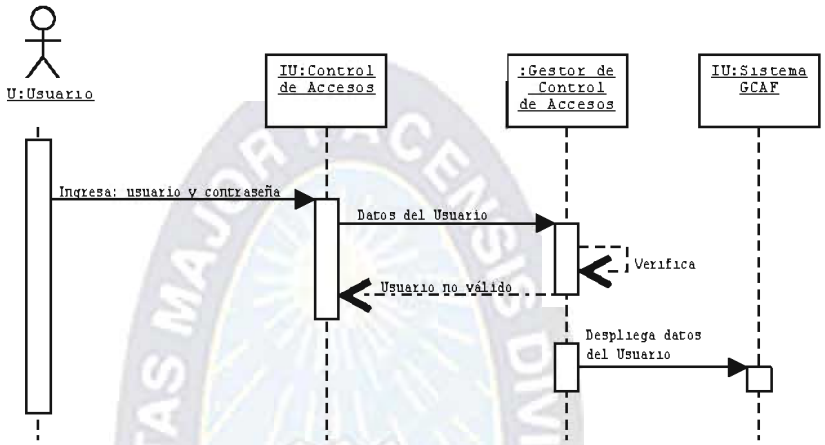


Figura. Diagrama de secuencia: control de accesos al Sistema GCAF
Fuente: Elaboración propia

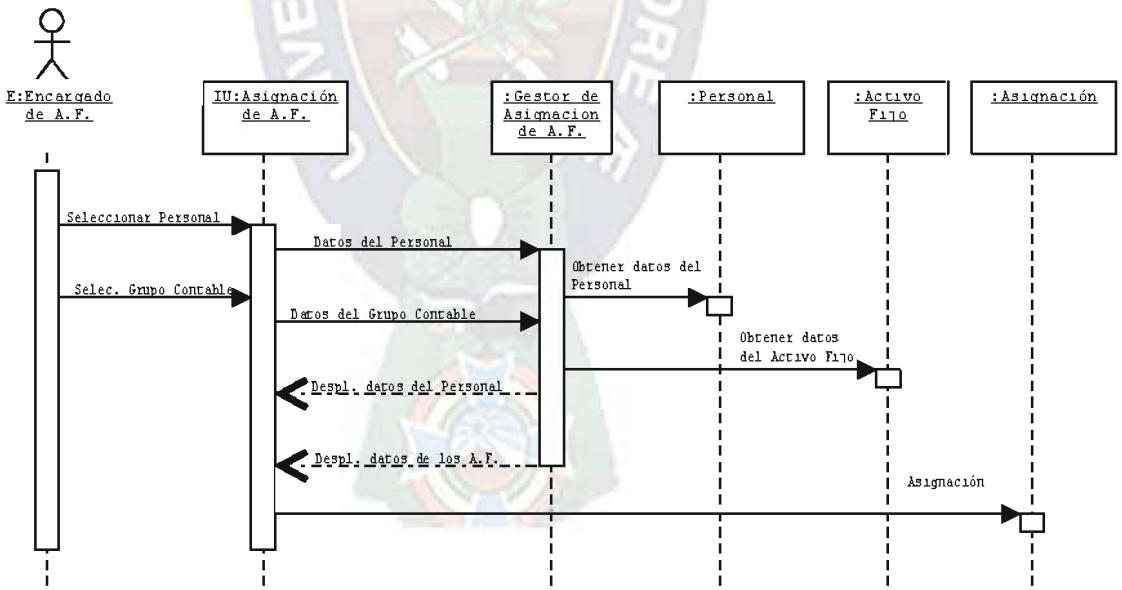


Figura. Diagrama de secuencia: asignación de Activo Fijo
Fuente: Elaboración propia

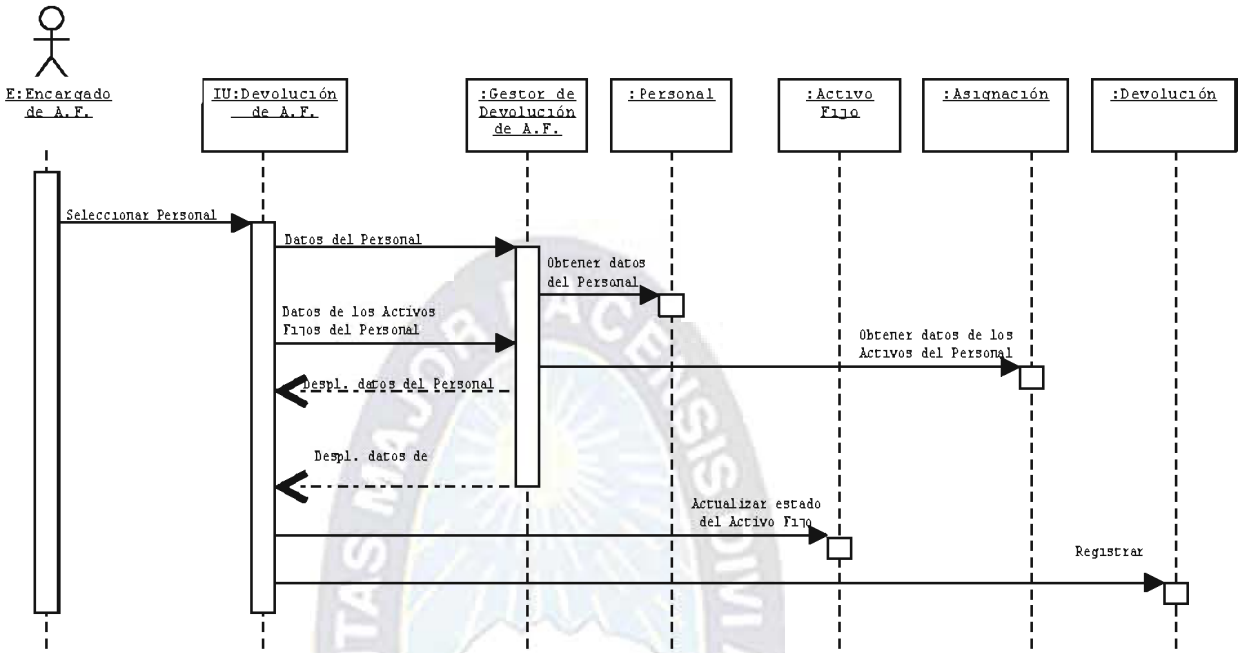


Figura. Diagrama de secuencia: devolución del Activo Fijo
Fuente: Elaboración propia

Figura. Diagrama de secuencia: depreciación del Activo Fijo
Fuente: Elaboración propia

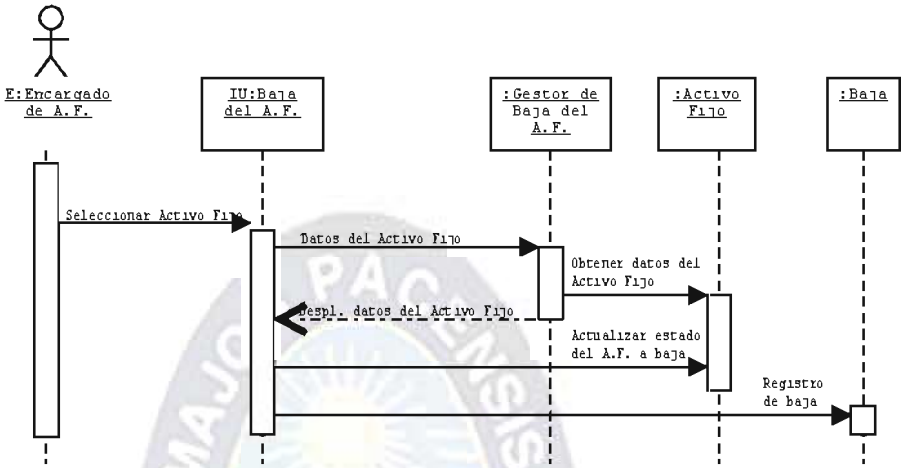


Figura. Diagrama de secuencia: baja del Activo Fijo
Fuente: Elaboración propia

ANEXO F

Diagrama de estados

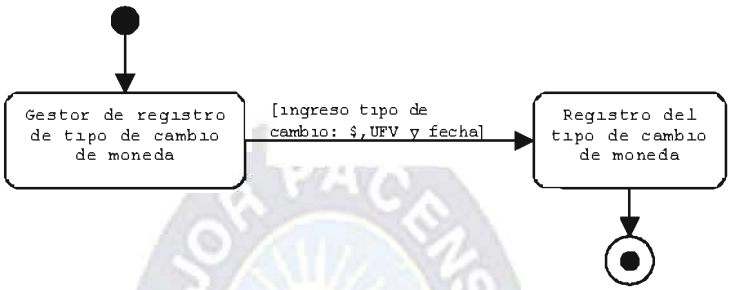


Figura. Diagrama de estados: registro de tipo de cambio de moneda
Fuente: Elaboración propia

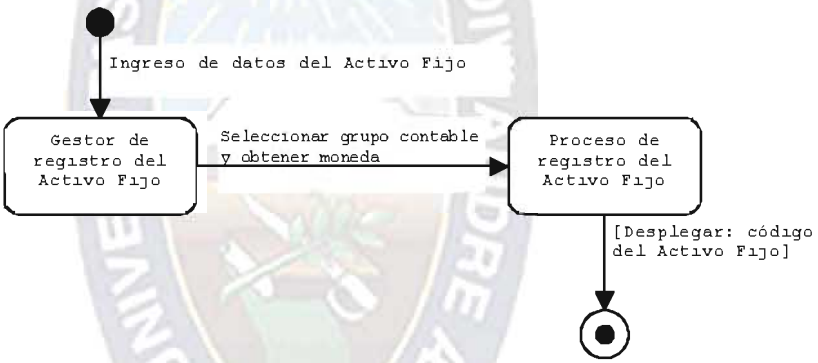


Figura. Diagrama de estados: registro del Activo Fijo
Fuente: Elaboración propia

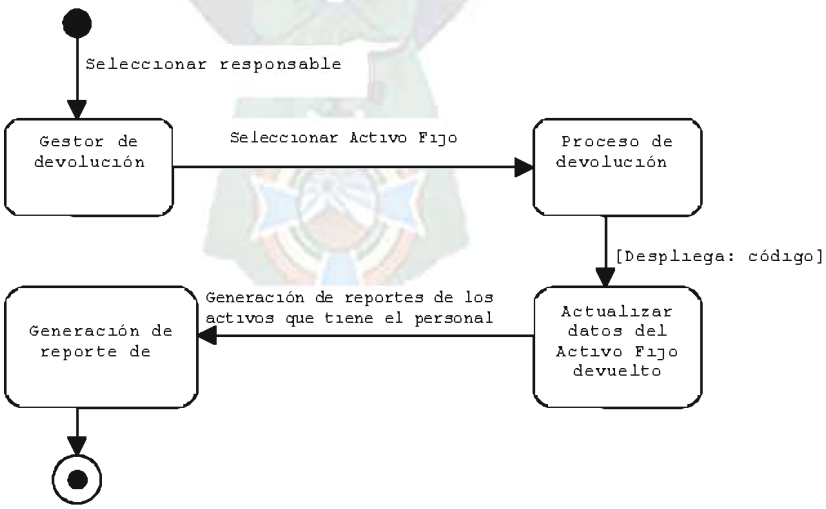


Figura. Diagrama de estados: devolución del Activo Fijo
Fuente: Elaboración propia

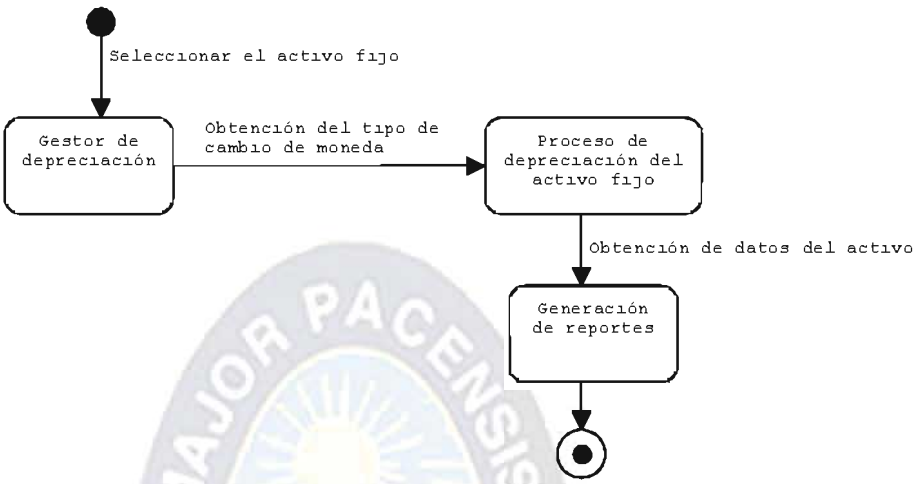


Figura. Diagrama de estados: depreciación del Activo Fijo
Fuente: Elaboración propia

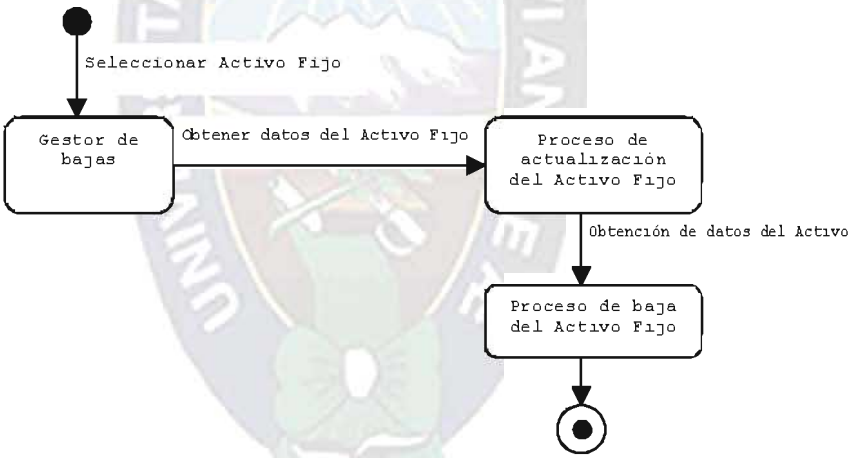


Figura. Diagrama de estados: baja del Activo Fijo
Fuente: Elaboración propia