ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

TEMA:

DESARROLLO E IMPLEMENTACION DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA, APLICANDO HERRAMIENTAS OPEN SOURCE Y LA METODOLOGIA EXTREME PROGRAMMING.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN COMPUTACIÓN

AUTORES:

QUISHPE CAHUEÑAS WILSON HOMERO

TIPAN SIMBAÑA JUAN PABLO

Latacunga, Marzo 2011

ESCUELA POLITÉCNICA DEL EJÉRCITO
EXTENSIÓN LATACUNGA

AUTORIZACIÓN

Nosotros, Quishpe Cahueñas Wilson H. y Tipan Simbaña Juan P.

Autorizamos a la Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la institución del trabajo "DESARROLLO E IMPLEMENTACION DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA, APLICANDO HERRAMIENTAS OPEN SOURCE Y LA METODOLOGIA EXTREME PROGRAMMING." Cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y auditoria.

Latacunga, Marzo del 2011

Wilson Homero Quishpe Cahueñas	Juan Pablo Tipán Simbaña
C.I. 1717638660	C.I. 1713996120

ESCUELA POLITÉCNICA DEL EJÉRCITO
EXTENSIÓN LATACUNGA

DECLARACIÓN DE RESPONSABILIDAD

Nosotros, Quishpe Cahueñas Wilson H. y Tipán Simbaña Juan P.

DECLARO QUE:

El proyecto de grado denominado "DESARROLLO E IMPLEMENTACION DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA, APLICANDO HERRAMIENTAS OPEN SOURCE Y LA METODOLOGIA EXTREME PROGRAMMING." Ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Marzo del 2011

Wilson Homero Quishpe Cahueñas	Juan Pablo Tipán Simbaña
C.I. 1717638660	C.I. 1713996120

ESCUELA POLITÉCNICA DEL EJÉRCITO
EXTENSIÓN LATACUNGA

CERTIFICADO

Ing. José Carrillo (Director)

Ing. Xavier Montaluisa (Codirector)

CERTIFICAN

Que el trabajo titulado "DESARROLLO E IMPLEMENTACION DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA, APLICANDO HERRAMIENTAS OPEN SOURCE Y LA METODOLOGIA EXTREME PROGRAMMING." Realizado por los señores Wilson Homero Quishpe Cahueñas y Juan Pablo Tipan Simbaña, ha sido guiado y revisado periódicamente y cumple las normas estatutarias establecidas por la ESPE, en el reglamento de estudiantes de la Escuela Politécnica del Ejército.

Debido a que constituye un trabajo de excelente contenido científico, coadyuvara a la aplicación de conocimientos y al desarrollo profesional, si recomiendan su publicación.

Latacunga, Marzo del 2011

Ing. José Carrillo	Ing. Xavier Montaluisa
DIRECTOR DEL PROYECTO	CODIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco primeramente a dios por darme la vida y en especial a mis padres y familiares por apoyarme en la culminación de mis estudios superiores, además a los señores maestros y compañeros alumnos que estuvieron enseñando y apoyándonos en todo este largo trayecto de mi carrera.

Juan P. Tipán S.

AGRADECIMIENTO

Le doy gracias a dios por permitirme culminar una etapa más de mi vida, a mi familia por su apoyo y comprensión durante todo el tiempo de estudio, a mis amigos y compañeros, que supieron demostrar su espíritu de estudiante. A todos nuestros profesores, por mostrarnos y enseñarnos sus sabios conocimientos.

Wilson H. Quishpe C.

DEDICATORIA

Dedico a toda mi familia y en especial a mi Esposa y a mis Hijos que me supieron apoyarme incondicionalmente en toda mi carrera profesional, triunfando exitosamente en mis estudios superiores obteniendo el conocimiento y un titulo que me ayudara en mi desempeño profesional en lo largo de mi vida.

Juan P. Tipán S.

DEDICATORIA

Este proyecto se la dedico a mi familia por su apoyo incondicional, que me permiten seguir cumpliendo mis metas, como es este título, que me ayudara en el desempeño profesional a lo largo de mi carrera profesional.

Wilson H. Quishpe C.

ÍNDICES DEL CONTENIDO

INDICES DE LAS FIGURAS

INDICES DE LAS TABLAS

PRESENTACIÓN

El presente proyecto de investigación está enfocado a investigar la Tecnología JavaBeans, Open Source y la metodología Extreme Programming y una base de datos mysql ya que dichas herramientas se utilizan en software libre proporcionándonos así facilidad de manejo y licencias.

Aplicando esta moderna tecnología se va a desarrollar una aplicación para la Gestión de Emisión y Recesión de documentos, empleando la tecnología Java Beans utilizando código abierto con el fin de facilitar el almacenamiento de datos.

RESUMEN

El presente proyecto de tesis se encuentra dividido en 4 capítulos:

Capitulo 1. Habla sobre el marco teórico que es el proceso de recolección de información documental, necesaria para la confección del desarrollo del sistema

Capitulo 2. Cita las características y las herramientas principales para el desarrollo del Sistema.

Capitulo 3. Cita las características y las herramientas principales para el diseño y el desarrollo del Sistema

Capitulo 4. Sobre las conclusiones y recomendaciones que se ha llegado al concluir en este proyecto de tesis

CAPITULO 1

1.1 INTRODUCCION

El presente trabajo tiene como objetivo ayudar a mejorar el sistema de control para la emisión y recepción de la documentación de la Escuela Politécnica del Ejercito Extensión Latacunga, para brindar un servicio más ágil y técnico a la colectividad. Ya que debido a la cantidad de información que en los diferentes departamentos se maneja y se da tramite a varios documentos, el volumen retrasa el movimiento de dicha información entre los diferentes departamentos que con estas se relacionan.

Los diferentes departamentos reciben, envían, generan y archivan documentación importante para la Escuela Politécnica del Ejercito Extensión Latacunga, por lo que nuestro trabajo se enmarca en dar facilidades en el tratamiento de esta documentación y en la labor de estos departamentos.

Con la automatización de ciertos procesos que ahora se realizan manualmente se vio la necesidad de implementar un sistema para agilizar las actividades que en estos departamentos se realizan.

Esto se lo verá reflejado al poner en funcionamiento el modulo que se realizo y al ver los resultados por rendimiento en tiempo costo y dinero de las actividades que se realizan en los diferentes departamentos.

Por ende como trabajo previo a la obtención del título de Tecnología en Computación, se presenta el siguiente trabajo como aporte a la Escuela Politécnica del Ejercito Extensión Latacunga.

1.2 OBJETIVO GENERAL

Desarrollar e Implementar un Sistema de Control para la Emisión y Recepción de la Documentación de la Escuela Politécnica del Ejército extensión Latacunga, aplicando herramientas Open Source y la Metodología Extreme Programming basándose en un punto de vista

estructural y funcional del sistema con herramientas de fácil identificación y de manejo fácil que ayuden al usuario a su satisfacción de la utilización del sistema e información personal o en cualquier ámbito que requiera el mismo.

1.3 OBJETIVOS ESPECÍFICOS

- Analizar la situación actual de los diferentes departamentos de la ESPE
- Determinar Marco teórico a aplicarse en todo el proceso de desarrollo
- Determinar necesidades y problemas de envió y recepción de documentos que se generan dentro de los departamentos
- Analizar la herramientas OPEN SOURCE a emplearse para el desarrollo del software
- Desarrollar e implementar el software de control de documentos

1.4 IMPORTANCIA DEL PROYECTO

- Para que ciertos procesos primordiales estén automatizados y se pueda realizar con mayor facilidad el trámite de la documentación
- Para generar respaldos de información en medios magnéticos y se pueda tener acceso a los archivos cuando se lo requiera y obtener esta información con mayor facilidad y rapidez.
- Para agilitar los diferentes trámites y evitar la congestión de documentación
- Para constatar la veracidad de envió y recepción de documentos.
- Mediante la realización de estos proyectos ayuden a estudiantes egresados de la escuela a desarrollar su tesis.

CAPITULO 2

2.1 MARCO TEORICO

En este capítulo vamos a ver todo lo referente a la tecnología que se va a utilizar, así destacamos los siguientes:

- La metodología Extreme Programming
- Open Source
- Java
- NetBeans
- Mysql
- IDE (Entorno de Desarrollo Integrado)
- UML (Lenguaje Unificado de Modelado)

Estos pueden ser fáciles de usarlos en una aplicación. La utilización de estos es que tienen un código fácil de mantener y mas legible para los diseñadores.

Esta moderna tecnología nos permitirá desarrollar una aplicación de un sistema de control de documentos para la ESPE-L empleando la tecnología Open Source, Java–NetBeans a fin de facilitar el almacenamiento de datos a través de una base de datos en Mysql para obtener los datos rápidos y oportunos el mismo que será desarrollado en Java–NetBeans por su facilidad de programación y seguridad.

2.2 METODOLOGIA EXTREME PROGRAMMING (XP)

2.2.1 INTRODUCCION

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de

los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

La filosofía de X.P es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo.

2.2.2 RESEÑA HISTORICA

En el año 2.001, miembros prominentes de la comunidad se reunieron en Snowbird, Utah, y adoptaron el nombre de "metodologías ágiles". Poco después, algunas de estas personas formaron la "alianza ágil", una organización sin fines de lucro que promueve el desarrollo ágil de aplicaciones. Muchos métodos similares al ágil fueron creados antes del 2000. Entre los más notables se encuentran: Scrum (1986), Crystal Clear (cristal transparente), programación extrema o XP (1996), desarrollo de software adaptativo.

Kent Beck creó el método de Programación Extrema (usualmente conocida como XP) en 1996 como una forma de rescatar el proyecto del Sistema exhaustivo de compensaciones.

2.2.3 QUÉ ES EXTREME PROGRAMMING (XP)

Es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

Se base en un grupo de gente que programa en estrecho contacto con sus clientes, trabajar por parejas etc.
Es una metodología ágil
Diseñada para entornos dinámicos
Pensada para equipos pequeños (hasta 10 programadores)
Orientada fuertemente hacia la codificación
Énfasis en la comunicación informal, verbal
Extreme Programming (XP) se funda en cuatro valores: comunicación, simplicidad, feedback y coraje.

2.2.4 DEFINICION

La Programación Extrema es una metodología de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado (reciclado de código), lo que buscan en definitiva es la reducción de costos.

2.2.4.1 LOS OBJETIVOS QUE PERSIGUE EXTREME PROGRAMING

2.2.4.1.1 EL OBJETIVO PRINCIPAL

Es la satisfacción del cliente. Esta metodología fue diseñada para proporcionar el software que el cliente necesita cuando lo necesite. Deberemos responder de forma rápida a los cambios en las necesidades del cliente, incluso cuando estos dicho software simplicidad y realimentación.

2.2.4.1.2 EL SEGUNDO OBJETIVO

Es potenciar al máximo el trabajo en equipo. Tanto los Directores de Proyecto, como los clientes y desarrolladores, son parte del equipo encargado de la implementación de software de calidad (ante todo la

comunicación). Esto implicará que los diseños deberán ser claros y sencillos. Y los clientes deberán disponer de versiones operativas cuanto antes para poder participar en el proceso creativo mediante sus sugerencias y aportaciones.

- El código será revisado continuamente, mediante la programación en parejas dos personas por máquina).
- Se harán pruebas todo el tiempo, no sólo de cada nueva clase (pruebas unitarias) sino que también los clientes comprobarán que el proyecto va satisfaciendo los requisitos.
- Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente (integración continua).
- Se rediseñará todo el tiempo, dejando el código siempre en el estado más simple posible.
- Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, de manera que nos podamos beneficiar de la retroalimentación tan a menudo como sea posible.

2.2.4.2 FUNDAMENTOS DE EXTREME PROGRAMING (X.P).

La programación extrema es una metodología reciente (alrededor de 5 años) utilizada en el desarrollo de software. La filosofía de X.P es satisfacer al completo las necesidades del cliente, por eso, lo integra como una parte más del equipo de desarrollo.

X.P fue inicialmente creada para el desarrollo de aplicaciones dónde el cliente no tiene una concepción clara de las funcionalidades que tendrá la aplicación que se desarrollará. Este desconocimiento podría provocar un cambio constante en los requisitos que debe cumplir la aplicación por lo que es necesaria una metodología ágil como X.P que se adapta a las

necesidades del cliente y dónde la aplicación se va revisando constantemente.

X.P está diseñada para el desarrollo de aplicaciones que requieren un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

2.2.4.3 CARACTERÍSTICAS ESENCIALES DE ESTA METODOLOGÍA SON LAS SIGUIENTES:

2.2.4.3.1 Comunicación.

Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

2.2.4.3.2 Simplicidad.

Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se requiere mantenimiento o ampliación resulta imposible hacerlo y se tienen que desechar y partir de cero.

2.2.4.3.3 Realimentación.

Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema adecuado a sus necesidades. Se le va mostrando el proyecto a tiempo para sugerir cambios y poder retroceder a una fase anterior para rediseñarlo a su gusto.

2.2.4.3.4 Aptitud.

Se debe tener coraje o valentía para cumplir los tres puntos anteriores; Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y por último hay que tener valor y confiar en que la realimentación sea efectiva.

2.2.4.4 ACTIVIDADES DE EXTREME PROGRAMING (X.P).

2.2.4..4.1 Codificar

Es necesario codificar y plasmar nuestras ideas a través del código. En programación, el código expresa la interpretación del problema, así podemos utilizar el código para comunicar, para hacer comunes las ideas, y por tanto para aprender y mejorar.

2.2.4..4.2 Hacer pruebas

Las características del software que no pueden ser demostradas mediante pruebas simplemente no existen. Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente. Las pruebas nos indican que nuestro trabajo funciona, cuando no podemos pensar en ninguna prueba que pudiese originar un fallo en nuestro sistema, entonces habremos acabado por completo.

2.2.4..4.3 Escuchar

Nos menciona en una frase, "Los programadores no lo conocemos todo, y sobre todo muchas cosas que las personas de negocios piensan que son interesantes. Si ellos pudieran programarse su propio software ¿para qué nos querrían?".

Si vamos a hacer pruebas tenemos que preguntar si lo obtenido es lo deseado, y tenemos que preguntar a quien necesita la información. Tenemos que escuchar a nuestros clientes cuáles son los problemas de su negocio, debemos de tener una escucha activa explicando lo que es fácil y difícil de obtener, y la realimentación entre ambos nos ayudan a todos a entender los problemas.

2.2.4..4.4 Diseñar

El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias. Si hay fallos en el diseño o malos diseños, estos deben de ser corregidos cuanto antes.

Resumiendo las actividades de X.P: Tenemos que codificar porque sin código no hay programas, tenemos que hacer pruebas por que sin pruebas no sabemos si hemos acabado de codificar, tenemos que escuchar, porque si no escuchamos no sabemos qué codificar ni probar, y tenemos que diseñar para poder codificar, probar y escuchar indefinidamente.

2.2.4.5 MODELO DE LA METODOLOGÍA EXTREME PROGRAMING (X.P).

La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que, de

estas cuatro variables, sólo tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo, en función de los valores de las otras tres. Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. Este modelo es analizado por Kent Beck, donde propone las ventajas de un contrato con alcances opcionales.

El ciclo de vida de un proyecto XP incluye, al igual que las otras metodologías, entender lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente. Sin embargo, XP propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto.
Por esto, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP. Típicamente un proyecto con XP lleva 10 a 15 ciclos o iteraciones. Si bien el ciclo de vida de un proyecto XP es muy dinámico, se puede separar en fases

2.2.4.6 MAPA DE LAS FASES DE EXTREME PROGRAMMING

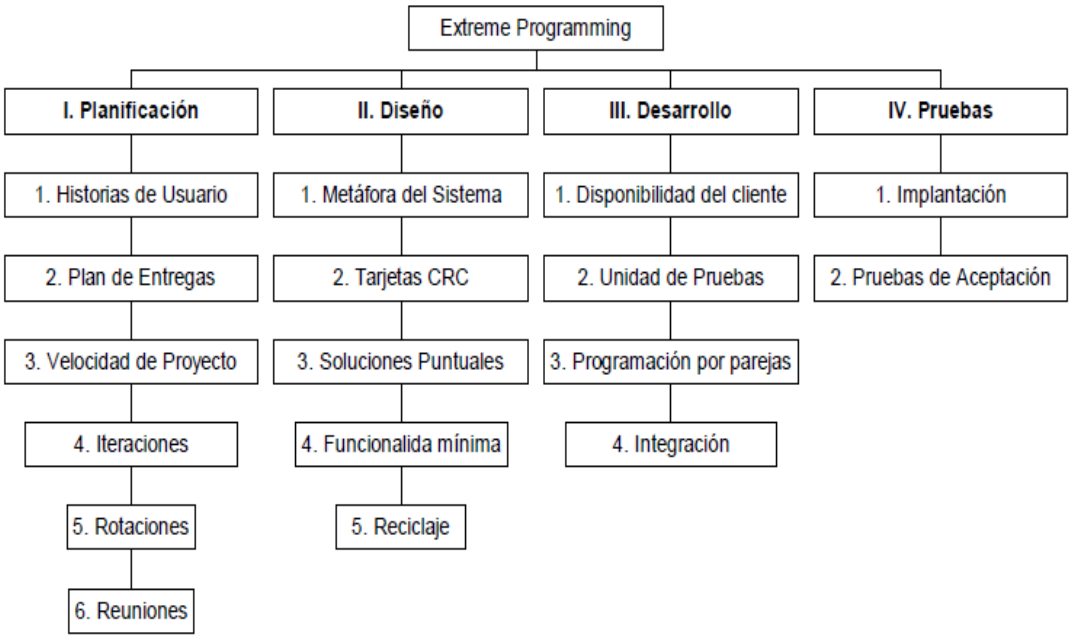


Figura 1.1: Del Mapa de las fases de Extreme Programming

2.2.4.1 FASES DE LA METODOLOGÍA EXTREME PROGRAMING (X.P)

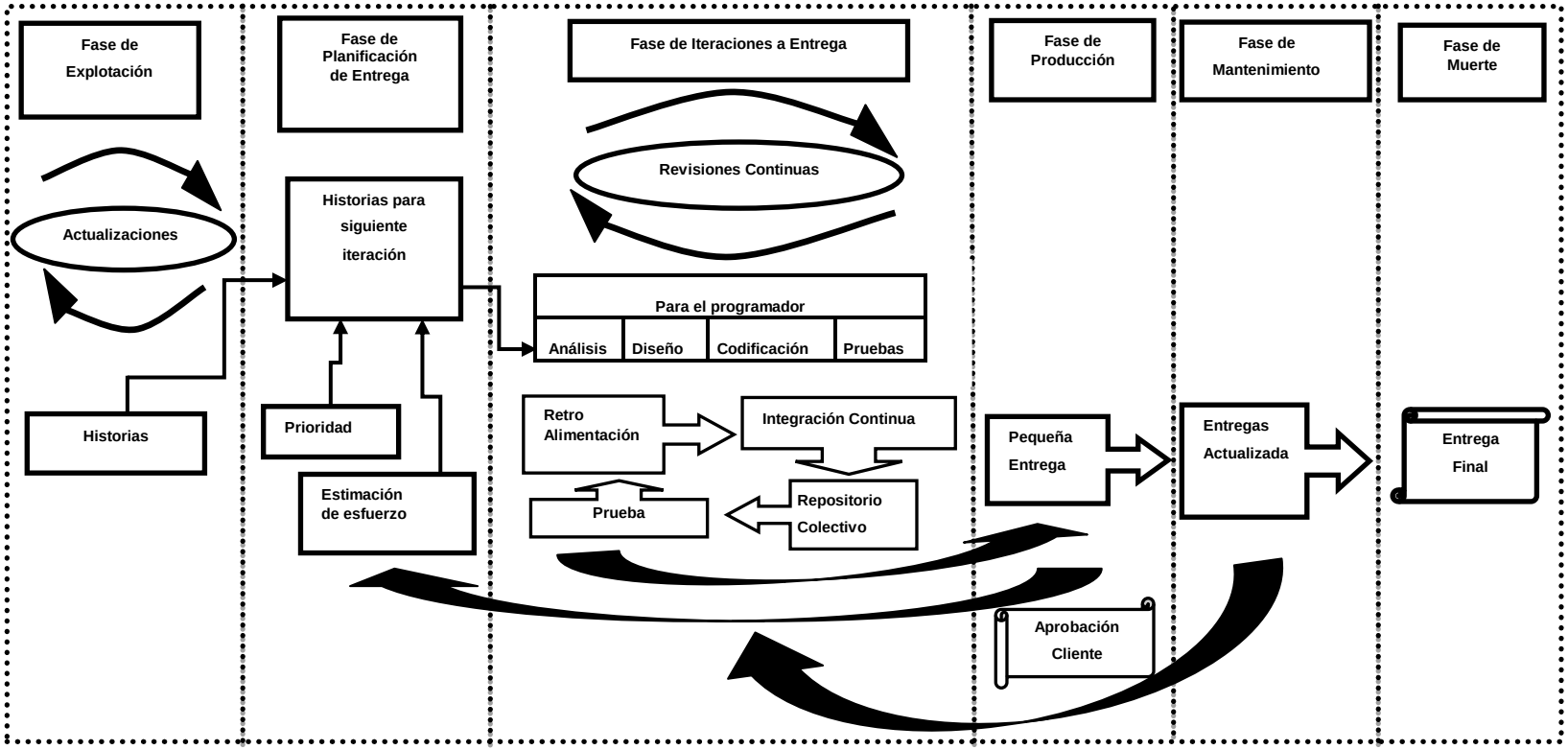


Figura 1.2: De las Fases de la m Mapa de las fases de Extreme Programming

2.2.4.7.1 FASE DE EXPLORACIÓN

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.2.4.7.1.1 REGLAS Y PRACTICAS

La metodología XP tiene un conjunto importante de reglas y prácticas. Se pueden agrupar en:

- Reglas y prácticas para la Planificación
- Reglas y prácticas para el Diseño
- Reglas y prácticas para el Desarrollo
- Reglas y prácticas para las Pruebas

2.2.4.7.2 FASE DE PLANIFICACIÓN DE LA ENTREGA

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma

de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.2.4.7.3 HISTORIAS DE USUARIOS

Las “Historias de usuarios” (“User stories”) sustituyen a los documentos de especificación funcional, y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios.

Las historias de usuarios deben poder ser programadas en un tiempo, entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia.

2.2.4.7.4 PLAN DE ENTREGAS (“RELEASE PLAN”)

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión “Juego de planeamiento” (“Planning game”), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, “Planning meeting” o “Planning workshop”).

Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario.

2.2.4.7.5 PLAN DE ITERACIONES (“ITERATION PLAN”)

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación.

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir.

2.2.4.7.6 REUNIONES DIARIAS DE SEGUIMIENTO (“STAND-UP MEETING”)

El objetivo de tener reuniones diarias es mantener la comunicación entre el equipo, y compartir problemas y soluciones. En la mayoría de estas reuniones, gran parte de los participantes simplemente escuchan, sin tener mucho que aportar. Para no quitar tiempo innecesario del equipo, se sugiere realizar estas reuniones en círculo y de pie.

2.2.4.7.7 PLANIFICACIÓN DEL PROYECTO

2.2.4.7.7.1 HISTORIAS DE USUARIO

El primer paso de cualquier proyecto que siga la metodología X.P es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados.

Son utilizadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si la aplicación cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia.

El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

2.2.4.7.7.2 LIBERACIÓN DE PLANIFICACIÓN

Tras definir las historias de usuario es necesario crear un plan de publicaciones, donde se indiquen las historias de usuario que se implementarán para cada versión de la aplicación y las fechas en las que

se publicarán dichas versiones. Una Liberación de Planificación es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias de usuario que serán implementadas en cada versión del programa.

Después de una Liberación de Planificación Tras definir las historias de usuario es necesario crear un plan de publicaciones, donde se indiquen las historias de usuario que se implementarán para cada versión de la aplicación y las fechas en las que se publicarán dichas versiones. Una Liberación de Planificación, es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias de usuario que serán implementadas en cada versión del programa, tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones de la aplicación, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado.

2.2.4.7.7.3 ITERACIONES.

Todo proyecto que siga la metodología X.P se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en la Liberación de Planificación que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior.

2.2.4.7.7.4 VELOCIDAD DEL PROYECTO

La velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el mismo; estimarla es muy sencillo: basta con contar el número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones.

Con la velocidad del proyecto controlaremos si todas las tareas se pueden desarrollar en el tiempo dispuesto para la iteración. Es conveniente reevaluar esta medida cada 3 ó 4 iteraciones y si se aprecia que no es adecuada hay que negociar con el cliente una nueva " Liberación de Planificación".

2.2.4.7.7.5 PROGRAMACIÓN POR PARES

La metodología X.P aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado. El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. De esta forma se consigue un código y diseño con gran calidad.

2.2.4.7.7.6 ROTACIONES

Las rotaciones evitarán que las personas se conviertan en sí mismas en un cuello de botella. Las rotaciones permitirán que todo el mundo conozca cómo funciona el sistema.

2.2.4.7.7.7 REUNIONES DIARIAS

Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo debe tener voz y voto.

2.2.4.7.8 FASE DE DISEÑO

La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes.

2.2.4.7.8.1 DISEÑOS SIMPLES

La metodología X.P sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácil de entender e implementar que a la larga costará menos tiempo y esfuerzo desarrollar.

2.2.4.7.8.2 GLOSARIOS DE TÉRMINOS

Usar una correcta especificación de los nombres de clases, métodos y propiedades ayudará a comprender el diseño y facilitará futuras ampliaciones y la reutilización del código.

2.2.4.7.8.3 RIESGOS

Si surgen problemas potenciales durante el diseño, X.P sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema dando.

2.2.4.7.8.4 FUNCIONALIDAD EXTRA

Nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10% de la misma es utilizada lo que demuestra que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

2.2.4.7.8.5 REFACTORIZAR

Consiste en mejorar el código modificando su estructura sin alterar su funcionalidad. Refactorizar supone revisar de nuevo la codificación para procurar optimizar su funcionamiento.

Es muy común reutilizar código ya creado que suele contener funcionalidades que no serán utilizadas y diseños obsoletos; esto es un grave error porque puede generar código inestable y mal diseñado; por este motivo, es necesario refactorizar cuando se va a reutilizar código.

2.2.4.7.8.6 TARJETAS C.R.C.

El uso de las tarjetas C.R.C (Clase, Responsabilidad y Colaboración) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación proceder al clásica.

Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

2.2.4.7.8.7 SOLUCIONES “SPIKE”(PRUEBAS)

Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados “spike”1), para explorar diferentes soluciones. Estos programas son únicamente para probar o evaluar una solución, y suelen ser desechados luego de su evaluación.

2.2.4.7.8.8 RECICLAJE

El reciclaje implicará mantener el código limpio y fácil de comprender, modificar y ampliar.

2.2.4.7.8.9 RE CODIFICACIÓN

La decodificación (“refactorización”) consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de

escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, más clara y eficientemente.

2.2.4.7.8.10 METÁFORAS

Es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema.

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema y explica de como guiar la estructura y arquitectura, que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema.

La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. Por ejemplo, puede ser una guía para la nomenclatura de los métodos y las clases utilizadas en el diseño del código. Tener nombres claros, que no requieran de mayores explicaciones, redunda en un ahorro de tiempo.

2.2.4.7.9 FASE CODIFICACIÓN

2.2.4.7.9.1 DISPONIBILIDAD DEL CLIENTE

Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP.

Al comienzo del proyecto, el cliente debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de “alto nivel”, no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo. No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, “cara a cara” a los desarrolladores.

2.2.4.7.9.2 PROGRAMACIÓN DIRIGIDA POR LAS PRUEBAS

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada sobre el final del proyecto, o sobre el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, en el que, lo primero que se escribe son los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas.

Las pruebas a los que se refieren esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona o “dirige” el desarrollo.

2.2.4.7.9.3 PROGRAMACIÓN POR PAREJAS

XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.

En un estudio realizado por Cockburn y Williams, se concluye que la programación en pares tiene un sobre costo aproximado de 15%, y no de

un 100% como se puede pensar. Este sobre costo es rápidamente pagado por la mejor calidad obtenida en el producto final.

2.2.4.7.9.4 INTEGRACIONES PERMANENTES

Todos los desarrolladores necesitan trabajar siempre con la "última versión", Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

2.2.4.7.10 FASE PRUEBA

2.2.4.7.10.1 PRUEBAS UNITARIAS

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o re codificar parte del mismo.

Uno de los pilares de la metodología X.P es el uso de test para comprobar el funcionamiento del código que estamos desarrollando.

Se deben crear los test que se aplicarán a una clase/método antes de implementarla; en el apartado anterior se explicó la importancia de crear antes los test que el código.

2.2.4.7.10.2 DETECCIÓN Y CORRECCIÓN DE ERRORES

Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

2.2.4.7.10.3 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

2.2.4.7.10.4 IMPLANTACIÓN

Este código debe ser implantado cuando supere correctamente todas unidades de test, revisada por todos los usuarios y clientes.

2.2.4.7.11 FASE PRODUCCIÓN

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación (por ejemplo, durante la fase de mantenimiento).

2.2.4.7.12 FASE MANTENIMIENTO

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

2.2.4.7.13 MUERTE DEL PROYECTO

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

2.2.5 IMPORTANCIA

- Es muy importante porque surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos
- Se plantea como una metodología a emplear en proyectos de riesgo
- Aumenta la productividad

- Una de las cosas más importantes de la programación extrema es que no se hace documentación en absoluto si no es necesaria para algo concreto. Un grupo de programadores pueden perfectamente pensar qué van a hacer usando una servilleta de papel que luego tiran.
- La programación extrema NO dice que NO se haga documentación. Sólo dice que se haga la estrictamente NECESARIA realmente para algo concreto.

2.2.6 VENTAJAS

- Es una programación Organiza para poder realiza cualquier proyecto sin ningún problema de desorganización
- Este código es el más adecuado para realizar este tipo de proyectos
- Múltiples desarrolladores contribuyen al diseño
- Propiedad colectiva del código
- Se puede tener una menor taza de errores
- Satisfacción del programador

2.2.7 DESVENTAJAS

- Es recomendable emplearlo solo en proyectos a corto plazo
- Reduce número de participantes en proyectos
- Conseguir su implantación en un equipo es algo que puede resultar dificultoso
- El equipo no está acostumbrado a este tipo de técnicas
- Altas comisiones en caso de fallar

2.3 TECNOLOGÍA OPEN SOURCE

2.3.1 INTRODUCCION.

El Software Libre aplica los principios de libre acceso a las fuentes de conocimiento que propugna la ciencia al ámbito del software.

Así pues, al igual que en ciencia toda persona que se lo proponga puede acceder al conocimiento patrimonio de todos, en el Software Libre no existen trabas para que cualquier individuo pueda disponer de toda la enciclopedia informática que constituye el corazón de los programas.

El Open Source o Código Abierto es una revolucionaria forma de desarrollar y distribuir el software. Ahora, moviéndose y creciendo paulatinamente por un movimiento revolucionario de personas alrededor del mundo que lo crean, utilizan y promueven.

El Código Abierto permite que varios programadores puedan leer, modificar y redistribuir el código fuente de un programa, por lo que ese programa ¡evoluciona! La gente lo mejora, lo adapta y corrige sus errores a una velocidad impresionantemente mayor a la aplicada en el desarrollo de software convencional o cerrado, dando como resultado la producción de un mejor software. La licencia de código abierto permite explícitamente:

Utilizar el programa para cualquier propósito y sin limitaciones, estudiar cómo funciona el programa.
Redistribuir copias del programa (no se paga por la licencia). Modificar el programa.
Con el código cerrado el usuario depende de que la empresa desarrolladora decida implementar la funcionalidad que necesita.
La diferencia entre el código Abierto y código Cerrado es que la licencia de código Cerrado permite explícitamente:
La oposición a las distintas licencias de software, - limitan el uso del programa, prohíben intentar conseguir el código.
Prohíben realizar (y distribuir) copias del programa, prohíben modificar el programa.

2.3.2 RESEÑA HISTORICA

Durante el año 1998, Eric S. Raymond, Bruce Pernees y otros hackers involucrados en el desarrollo de software libre lanzaron la Open Software Initiative y propusieron el uso de termino open source
(código abierto) en contraposición al termino free software (software libre) como termino más atractivo al entorno empresarial. El termino free software en el mundo anglófono creaba una situación incómoda debido a la doble acepción que en ingles tiene el termino free (que puede significar gratuito o libre). La gran mayoría de empresas en Estados Unidos usan principalmente el termino código abierto para evitar dar la percepción que el software libre es un recurso totalmente gratuito y para poner énfasis en valor diferencial que representa el hecho de que el código fuente está disponible.
Bruce Perens, de la Open Source Iniciative y antiguo coordinador de la distribución de Linux
Debían, creo una lista de condiciones que debe cumplir un programa para poder ser considerado Open Source. Estas condiciones son muy similares

y, de hecho están basadas, en las directrices de software libre de Debían. Estas condiciones también son aplicables a cualquier programa que sea software libre y pueden ayudarnos a matizar sus implicaciones:

El término para algunos no resultó apropiado como reemplazo para el ya tradicional free software, pues eliminaba la idea de libertad, confundida usualmente con la simple gratuidad. No obstante, el término código abierto continúa siendo ambivalente, puesto que se usa en la actualidad por parte de programadores que no ofrecen software libre pero, en cambio, sí ofrecen el código fuente de los programas para su revisión o modificación previamente autorizada por parte de sus pares académicos.

Dada la anterior ambivalencia, se prefiere el uso del término software libre para referirse a programas que se ofrecen con total libertad de modificación, uso y distribución bajo la regla implícita de no modificar dichas libertades hacia el futuro.

Desde el punto de vista de una "traducción estrictamente literal", el significado textual de "código abierto" es que "se puede examinar el código fuente", por lo que puede ser interpretado como un término más débil y flexible que el del software libre. Sin embargo, ambos movimientos reconocen el mismo conjunto de licencias y mantienen principios equivalentes.

2.3.3 QUE ES LA TECNOLOGIA OPEN SOURCE

El software Open Source se define por la licencia que lo acompaña, que garantiza a cualquier persona el derecho de usar, modificar y redistribuir el código libremente.

Código abierto (en ingleses open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

Al ambiguo nombre original en inglés del software libre (free software). Free en inglés significa dos cosas distintas dependiendo del contexto: gratuidad y libertad. Se prefiere el uso del término software libre para referirse a programas que se ofrecen con total libertad de modificación, uso y distribución bajo La idea bajo el concepto de código abierto es sencilla: cuando los programadores (en Internet) pueden leer, modificar y redistribuir el código

Fuente de un programa, éste evoluciona, se desarrolla y mejora. Los usuarios lo adaptan a sus necesidades, corrigen sus errores a una velocidad impresionante, mayor a la aplicada en el desarrollo de software convencional o cerrado, dando como resultado la producción de un mejor software.

La idea del código abierto se centra en la premisa de que al compartir el código, el programa resultante tiende a ser de calidad superior al software propietario, es una visión técnica.

2.3.4 DEFINICION

El código abierto promueve la fiabilidad y calidad del software al permitir las revisiones independientes de los colaboradores y una rápida evolución del código fuente. Para obtener la certificación OS (Open Source) el software debe distribuirse bajo una licencia que garantice el derecho a leerlo, redistribuirlo, modificarlo y usarlo libremente.

El software Open Source se define por la licencia que lo acompaña, que garantiza a cualquier persona el derecho de usar, modificar y redistribuir el código libremente.

La licencia no debe restringir otro software que se distribuya con el mismo. Por ejemplo, la licencia no debe indicar que todos los programas distribuidos conjuntamente con el deben ser libres.

Libre distribución. No haya restricciones para vender o distribuir el

Software.

Código fuente. El software debe incluir el código fuente y debe permitir crear distribuciones compiladas siempre y cuando la forma de obtener el código fuente esté expuesta claramente.

La plataforma que se utiliza para ejecutar estos programas es el sistema operativo en el que mayormente se usan estos programas es en Linux. Hay comunidades que realizan modificaciones, agregan programas, entornos, figuras, lo preparan en cd's, y se les llama distribución. Algunas de las distribuciones más famosas son Ubuntu, Federa, Suse, Mandrake, Red hat, Free BSD, etc.

2.3.4.1 CARACTERÍSTICAS DE OPEN SOURCE.

- Fiabilidad y seguridad: al contar con unos cuantos programadores mirando el mismo trabajo simultáneamente, los errores se detectan y corrigen con anterioridad, por lo que el producto resultante es más confiable y efectivo que el comercial.
- Rapidez de desarrollo: las actualizaciones y ajustes se llevan a cabo por medio de una comunicación constante vía internet. Debido a la gran cantidad de herramientas y librerías disponibles, se requieren menores tiempos de desarrollo.
- Relación con el usuario: el programador puede definir mejor las necesidades reales de su cliente.
- Libre: es de libre distribución, las personas pueden regalarlo, venderlo o prestarlo.
- El Software de Código Abierto, también llamado simplemente OS (por 'Open Source'), se diferencia del software de código cerrado en su licencia.

2.3.4.2 LA LICENCIA DE CÓDIGO ABIERTO PERMITE EXPLÍCITAMENTE.

- Utilizar el programa para cualquier propósito y sin limitaciones.
- Estudiar cómo funciona el programa.
- Redistribuir copias del programa (no se paga por la licencia).
- Modificar el programa.

2.3.4.3 EN OPOSICIÓN A ESTO, LAS DISTINTAS LICENCIAS DE SOFTWARE CERRADO EXPLÍCITAMENTE.

- Limitan el uso del programa.
- Prohíben intentar conseguir el código
- Prohíben realizar (y distribuir) copias del programa.
- Prohíben modificar el programa.

2.3.4.4 CUÁL ES EL MODELO DE DESARROLLO DEL CÓDIGO ABIERTO

El modelo de desarrollo del software de código abierto se basa en compartir el conocimiento.

Este es el modelo tradicional propio de los campos científicos y por ello fue el modelo inicial con que se desarrollo lo que hoy conocemos como internet, al igual que las herramientas que lo hicieron posible (tcp/ip, UNIX, C...)

2.3.4.5 DIFERENCIA ENTRE CODIGO CERRADO Y CODIGO ABIERTO.

En el modelo de desarrollo de código cerrado una empresa contrata a desarrolladores para realizar un proyecto y después vende en el mercado

licencias de uso con restricciones (no se puede copiar, no se puede estudiar, no se puede modificar...).

En el modelo de desarrollo de código abierto una persona u organización (puede ser una empresa) coordina una amplia comunidad de desarrolladores independientes distribuidos por todo el planeta y el software se puede ver, probar y modificar aún antes de que salga una primera versión completa.

Al proceder los usuarios y desarrolladores de distintos entornos económicos, sociales y legales, el resultado obtiene mayor flexibilidad, adaptabilidad y versatilidad.

2.3.4.6 CUÁL ES EL MODELO DE FINANCIACIÓN DEL CÓDIGO ABIERTO.

Las empresas que se dedican al OS no ingresan dinero por las licencias del software sino por el servicio que prestan a sus clientes.

Más concretamente, las empresas realizan sus ingresos por los servicios asociados al software, tales como formación, asesoría y consultoría, certificación, desarrollos a medida

De esta manera los clientes no pagan por poder utilizar un programa (lo use mucho o poco), los clientes invierten en tener un proveedor de servicios. Así optimizan su inversión, ya que solo pagan por los servicios recibidos y se mantienen independientes del proveedor.

Por otra parte uno de los servicios que puede proporcionar un proveedor es desarrollar una funcionalidad inexistente en una aplicación.

2.3.4.7 DIFERENCIA ENTRE CODIGO CERRADO Y CODIGO ABIERTO

De hecho con el código cerrado, mediante la venta de licencias, los clientes financian el desarrollo de un sistema y, generalmente, siguen

pagando por su uso cuando ya se ha financiado completamente el desarrollo.

En el código abierto un cliente únicamente financia, si lo desea, la funcionalidad que necesita y no existe.

De este modo contribuirá además al desarrollo del sistema, beneficiando a otras empresas en la misma medida que el desarrollo financiado por otras empresas le ha ayudado.

2.3.4.8 MANERAS DE OBTENER SOFTWARE LIBRE

- A través de copias en CD: los que a su vez se pueden conseguir en revistas especializadas, o comprándolos en una casa de computación, o pidiéndoselos a un amigo, pariente, etc.
- A través de Internet: a su vez, por medio de FTP, sitios Web, canales de chat, foros de noticias, programas de intercambio de archivos, etc.
- A través de una computadora: en este caso, comprando una que venga con Software Libre pre instalado, ya sea de fábrica o por su vendedor.

2.3.5 IMPORTANCIA.

2.3.5.1 ECONÓMICA

El costo de las licencias de Software Propietario es bastante importante, y por la situación económica actual, imposible de afrontar de la manera que los fabricantes de Software lo piden.

2.3.5.2 LEGAL

El Software Libre es siempre legal, salvo contadas excepciones (p/ej., que compilemos el código fuente y lo vendamos como propietario). Por lo

tanto, al utilizar este tipo de software estaremos siempre "por derecha", por lo que no seremos pasibles de multas y/o prisión.

2.3.5.3 TÉCNICA

Es sabido que Microsoft ha dejado de ofrecer soporte de desarrollo para Windows 95 y Windows 98, por lo que si hoy o mañana se descubre un error en ellos, Microsoft no está obligado a repararlo. Para solucionar esto, tendríamos dos caminos: a) Migrar a otras versiones de Sistema Operativos de Microsoft: esto lleva aparejado una serie de costos, principalmente en licencias, luego costos de implantación, soporte e interoperabilidad, y además implica volver a hacer lo mismo dentro de dos o tres años. b) Utilizar Software Libre.

2.3.5.4 LABORAL

La implementación de Software Libre plantea un futuro muy prometedor para aquellas personas que sepan programar, traducir, utilizar un programa, enseñar, etc. Si tenemos que elegir entre pagar una licencia de software a un coloso informático o darle trabajo directamente a una persona, es de esperar que nos volquemos a la segunda alternativa.

2.3.6 VENTAJAS

- El usuario no paga por la licencia de uso del programa.
- El proveedor cobra únicamente por los servicios que presta.
- Al disponer del código fuente, cualquier persona puede continuar ofreciendo soporte, desarrollo u otro tipo de servicios para el software.
- El software libre puede seguir siendo usado aun después de que haya desaparecido la persona que lo elaboro, dado que cualquier técnico informático puede continuar desarrollándolo, mejorándolo o adaptándolo.

- Ayudaría a las compañías a ahorrar costes y tiempos de diseño en sus trabajos.
- Ofrece menores tiempos de desarrollo debido a la amplia disponibilidad de herramientas y bibliotecas.
- El cliente es independiente del proveedor ya que, al disponer del código fuente, cualquier proveedor puede proseguir donde terminó el anterior.
- Puesto que el proveedor solo cobra por sus servicios y el cliente no tiene ninguna atadura, el proveedor concentra sus esfuerzos en dar un buen servicio al cliente –el único modo de mantenerlo.
- Los datos generados siempre serán accesibles, sin obligar para ello al cliente a invertir en licencias.
- Al conocerse el funcionamiento de los programas, podrán operar entre ellos sin restricciones.

2.3.7 DESVENTAJAS

- No existe ningún tipo de garantía
- La curva de aprendizaje en estos ambientes tiene una pendiente elevada.
- A veces existe poca documentación o documentación confusa.
- Una de las principales desventajas que se anuncia, es el costo de capacitación. Si bien, al analizarlo un poco, nos damos cuenta de que ese costo de capacitación es drásticamente menor al costo de adquirir un software comercial a escala.
- Su licencia no se puede vender en costos altos por lo que es gratis

2.4 HERRAMIENTAS DE DESARROLLO

2.4.1 CONOCIMIENTOS BÁSICOS DE JAVA.

2.4.1.1 INTRODUCCION.

Java es un lenguaje desarrollado por Sun con la intención de competir con Microsoft en el mercado de la red. Sin embargo, su historia se remonta a la creación de una filial de Sun (FirstPerson) enfocada al desarrollo de aplicaciones para electrodomésticos, microondas, lavaplatos, televisiones. Esta filial desapareció tras un par de éxitos de laboratorio y ningún desarrollo comercial.

Sin embargo, para el desarrollo en el laboratorio, uno de los trabajadores de FirstPerson, James Gosling, desarrolló un lenguaje derivado de C++ que intentaba eliminar las deficiencias del mismo. Llamó a ese lenguaje Oak. Cuando Sun abandonó el proyecto de FirstPerson, se encontró con este lenguaje y, tras varias modificaciones (entre ellas la del nombre), decidió lanzarlo al mercado en verano de 1995.

El éxito de Java reside en varias de sus características. Java es un lenguaje sencillo, o todo lo sencillo que puede ser un lenguaje orientado a objetos, eliminando la mayor parte de los problemas de C++, que aportó su granito (o tonelada) de arena a los problemas de C. Es un lenguaje independiente de plataforma, por lo que un programa hecho en Java se ejecutará igual en un PC con Windows que en una estación de trabajo basada en Unix.

Cabe mencionar también su capacidad multihilo, su robustez o lo integrado que tiene el protocolo TCP/IP, lo que lo hace un lenguaje ideal para Internet. Pero es su sencillez, portabilidad y seguridad lo que le han hecho un lenguaje de tanta importancia.

2.4.1.2 RESEÑA HISTORICA.

El lenguaje JAVA fue desarrollado en Sun Microsystems en 1991 como parte del proyecto Green.

Su objetivo fue desarrollar un lenguaje de programación que pudiera ser ejecutado en aparatos inteligentes del futuro.

Los investigadores, para arrancar la investigación desarrollaron un prototipo de dispositivo llamado Star7, semejante a un control remoto común que pudiera comunicar con otros de su propia clase, entonces James Gosling escribió un nuevo lenguaje para controlar al Star7. Oak

2.4.1.2.1 Versiones del lenguaje

- Java 1.0.2 (23 de enero de 1996) — Primer lanzamiento.
- Java 1.1.7 (19 de febrero de 1997)
- Java 2 (8 de diciembre de 1998) — Nombre clave Playground
- J2SE 1.3 (8 de mayo de 2000) — Nombre clave Kestrel.
- J2SE 1.4 (6 de febrero de 2002) — Nombre Clave Merlin
- J2SE 5.0 (30 de septiembre de 2004) — Nombre clave: Tiger
- Java SE 6 — Nombre clave Mustang. Estando en 2006 está en desarrollo bajo la JSR 270. Una versión Beta fue lanzada el 15 de febrero de 2006

2.4.1.3 QUE ES JAVA.

Java es un lenguaje de programación orientado a objetos, con características especiales que lo hacen ideal para programas.

La programación en Java es compilada en bytecode, el cuál es ejecutado por la máquina virtual Java. Usualmente se usa un compilador JIT.

El lenguaje es parecido a C y C++, aunque su modelo de objetos es más sencillo, y fue influenciado también por Smalltalk, y Eiffel.

Esto significa que posee ciertas características que hoy día se consideran estándares en los lenguajes OO:

- Objetos
- Clases
- Métodos
- Subclases
- Herencia simple
- Enlace dinámico
- Encapsulamiento

2.4.1.4 DEFINICION.

Java es un lenguaje de programación orientado a objetos que comparte gran parte de su sintaxis con C y C++. Java es uno de los lenguajes más poderosos que existen actualmente y desde la versión 6 es un proyecto open source, por lo que tiene el soporte de toda una comunidad de programadores, además de Sun Microsystems que fueron los creadores originales. En 2005 Java se utilizaba en uno de cada cuatro proyectos, casi diez por ciento por encima de su siguiente competidor (C++) y ha seguido creciendo. Se estima que un noventa por ciento de las computadoras cuentan con una máquina virtual de Java, además de que todos los celulares y una gran cantidad de dispositivos móviles también cuentan con Java.

Java es un lenguaje de programación por objetos que permite crear programas que funcionan en cualquier tipo de ordenador y sistema operativo.

Un programa java es una colección de clases. Algunas clases se escriben y algunas forman parte del lenguaje java. Un programa java debe contener un método estático denominado main ().

2.4.1.4.1 CARACTERÍSTICAS DEL LENGUAJE JAVA

2.4.1.4.1.1 LENGUAJE SIMPLE

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros. Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

2.4.1.4.1.2 ORIENTADO A OBJETOS

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

2.4.1.4.1.3 DISTRIBUIDO

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

2.4.1.4.1.4 INTERPRETADO Y COMPILADO A LA VEZ

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

2.4.1.4.1.5 ROBUSTO

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

2.4.1.4.1.6 SEGURO

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

2.4.1.4.1.7 INDIFERENTE A LA ARQUITECTURA

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

2.4.1.4.1.8 PORTABLE

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

2.4.1.4.1.9 INTERPRETADO.

Cuando se compila un programa de Java traduce el código fuente a código intermedio, para su ejecución el código intermedio es interpretado por la máquina virtual de Java (JVM, por sus siglas en Inglés) convirtiéndolo en código ejecutable según el sistema operativo donde se ejecute el programa.

2.4.1.4.1.10 ALTO RENDIMIENTO.

2.4.1.4.1.11 MULTIHEBRA

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

2.4.1.4.1.12 DINÁMICO

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

2.4.1.4.1.13 PRODUCE APPLETS

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets.

Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java.

Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los figuras o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

2.4.1.5 IMPORTANCIA.

Sirve para programar "hacer software" y es muy importante porque es multiplataforma y es unos de los lenguajes de programación más potentes.

2.4.1.6 VENTAJAS

- Capacidad del Programa para ser ejecutado en sistemas y plataformas diferentes.
- Máquina Virtual.
- Java es un lenguaje de programación orientado a objetos, y tiene todos los beneficios que ofrece esta metodología de programación.
- Java es un lenguaje y por lo tanto puede hacer todas las cosas que puede hacer un lenguaje de programación: Cálculos matemáticos, procesadores de palabras, bases de datos, aplicaciones gráficas, animaciones, sonido, hojas de cálculo, etc.

2.4.1.7 DESVENTAJAS

- Los programas hechos en Java no tienden a ser muy rápidos
- Este lenguaje es un poco complicado aprenderlo no es cosa fácil. Especialmente para los no programadores.
- Este lenguaje es nuevo todavía no se conocen bien todas sus capacidades.
- Al ejecutar los programas es un poco lento
- Al tener que ser ejecutado mediante la JVM hace que no sea tan rápido como con otras tecnologías, por ejemplo C++.

2.4.2 CONOCIMIENTOS BÁSICOS NETBEANS.

2.4.2.1 INTRODUCCION.

NetBeans es un entorno de desarrollo integrado (IDE por sus siglas en inglés). Esto quiere decir que integra todas las herramientas que necesitamos para poder desarrollar. Originalmente la programación en Java era algo complicada porque Java cuenta con una enorme cantidad de librearías y funciones que era preciso aprenderse de memoria, viendo esto muchas compañías construyeron diferentes entornos de programación para facilitar la tarea del programador. Entre los más populares surgió Eclipse que reinó como el único y más importante IDE de Java durante varios años. Sun Microsystems desarrollo su propio IDE, que tenía la ventaja de que fue creado por las mismas personas que crearon Java años antes, este IDE fue NetBeans y después de varios años de desarrollo ha llegado a ser tan útil y poderoso como Eclipse o quizás un poco más.

2.4.2.2 RESEÑA HISTORICA.

NetBeans comenzó como un proyecto estudiantil en República Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad Carolina en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la

de Del phi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-reléase en 1997-. Xelfi fue un proyecto divertido para trabajar, ya que las IDE escritas en Java eran un territorio desconocido en esa época. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. Casi todos ellos siguen trabajando en NetBeans.

Tiempo después, fueron contactados por Roman Stanek, un empresario que ya había estado relacionado con varias iniciativas. Estaba buscando una buena idea en la que invertir, y encontró en Xelfi una buena oportunidad. Así, tras una reunión, el negocio surgió.

El plan original era desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, propuso la idea de llamarlo NetBeans, a fin de describir este propósito. Cuando las especificaciones de los Enterprise JavaBeans salieron, decidieron trabajar con este estándar, ya que no tenía sentido competir contra él, sin embargo permaneció el nombre de NetBeans.

En la primavera de 1999, Netbeans DeveloperX2 fue lanzado, soportando Swing. Las mejoras de rendimiento que llegaron con el JDK 1.3, lanzado en otoño de 1999, hicieron de NetBeans una alternativa realmente viable para el desarrollo de herramientas. En el verano de 1999, el equipo trabajó duro para rediseñar DeveloperX2 en un NetBeans más modular, lo que lo convirtió en la base de NetBeans hoy en día.

Algo más ocurrió en el verano de 1999. Sun Microsystems quería una herramienta mejor de desarrollo en Java, y comenzó a estar interesado en NetBeans. En otoño de 1999, con la nueva generación de NetBeans en Beta, se llegaría a un acuerdo.

Seis meses después, se tomó la decisión de hacer a NetBeans Open Source. Mientras que Sun había contribuido considerablemente con

líneas de código en varios proyectos de código abierto a través de los años, NetBeans se convirtió en el primer proyecto de código abierto patrocinado por ellos. En Junio del 2000 NetBeans.org fue lanzado.

Un proyecto de código abierto no es nada más ni nada menos que un proceso. Toma tiempo encontrar el equilibrio. El primer año, fue crucial como inicio. Los dos años siguientes, se orientó hacia código abierto. Como muestra de lo abierto que era, en los primeros dos años había más debate que implementación.

Con NetBeans 3.5 se mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se re implementó el sistema de ventanas y la hoja de propiedades, y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente.

Versiones NetBeans

Versión	Fecha de Lanzamiento
NetBeans 6.9.1	4 de agosto de 2010
NetBeans 6.9	15 de Junio de 2010
NetBeans 6.8	10 de Diciembre de 2009
NetBeans 6.7.1	27 de Julio de 2009
NetBeans 6.7	29 de Junio de 2009
NetBeans 6.5	Noviembre 25 de 2008
NetBeans 6.1	Abril 28 de 2008
NetBeans 6.0	Diciembre 3 de 2007
NetBeans 5.5.1	Mayo 24 de 2007
NetBeans 5.5	Octubre 30 de 2006
NetBeans 5.0	Enero de 2006
NetBeans 4.1	Mayo de 2005
NetBeans 4.0	Diciembre de 2004
NetBeans 3.6	Abril de 2004
NetBeans 3.5	Junio de 2003

Figura 1.2: Versiones de NetBeans

2.4.2.3 QUE ES NETBEANS

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y subiendo!) en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

2.4.2.4 DEFINICION.

NetBeans IDE es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación.
La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

2.4.2.4.1 LA PLATAFORMA NETBEANS

La plataforma netbeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para

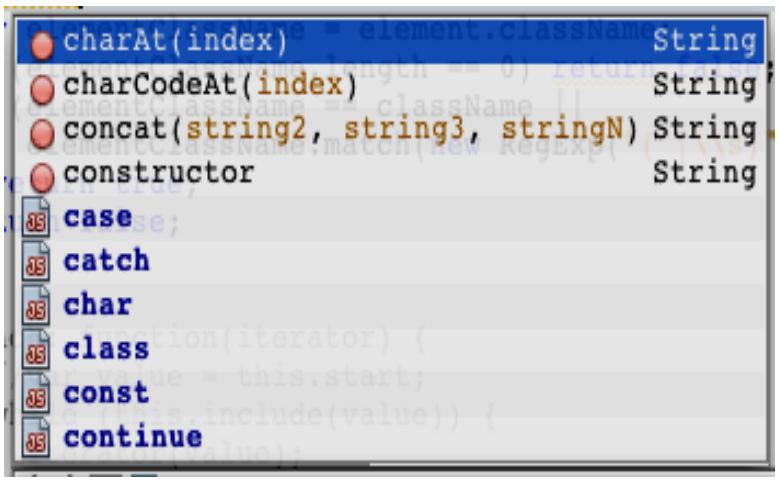
desarrollar sus propias herramientas y soluciones.la plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas)
- Administración de las configuraciones del usuario
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato)
- Administración de ventanas
- Framework basado en asistentes (diálogos paso a paso)

2.4.2.4.2 CARACTERÍSTICAS DESTACADAS

2.4.2.4.2.1 SOPORTE JAVASCRIPT

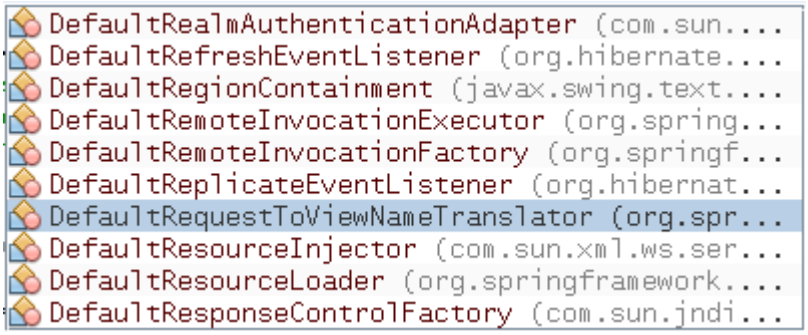
- Sintaxis Resaltada
- Completar el Código y Análisis de Tipeo
- Soluciones Rápidas (Quick Fixes) y Verificación de Sintaxis
- Refactorización



2.4.2.4.2.2 MEJORAS EN EL DESEMPEÑO

- Inicio hasta 40% más rápido
- Promociones más inteligentes, así que la completación de código es más rápida
- Menor consumo de memoria

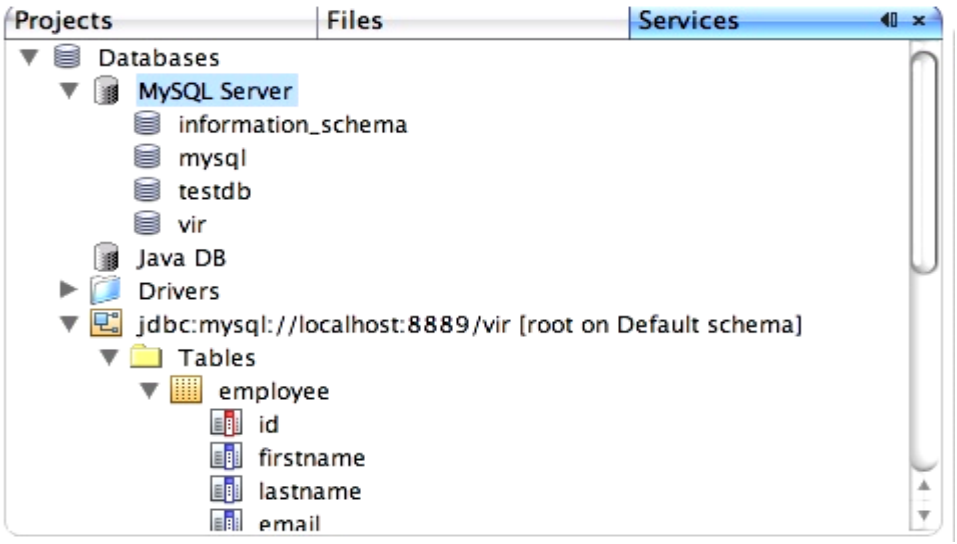
- Agregado de la librería Spring Framework 2.5
- Asistentes para configuración de archivos XML y controladores Spring Web MVC
- Completar el Código de nombres bean y clases y propiedades Java
- Soporte de entorno Spring Web MVC en proyectos web



2.4.2.4.2.3 NUEVO SOPORTE MYSQL EN EXPLORACIÓN DE BASES DE DATOS

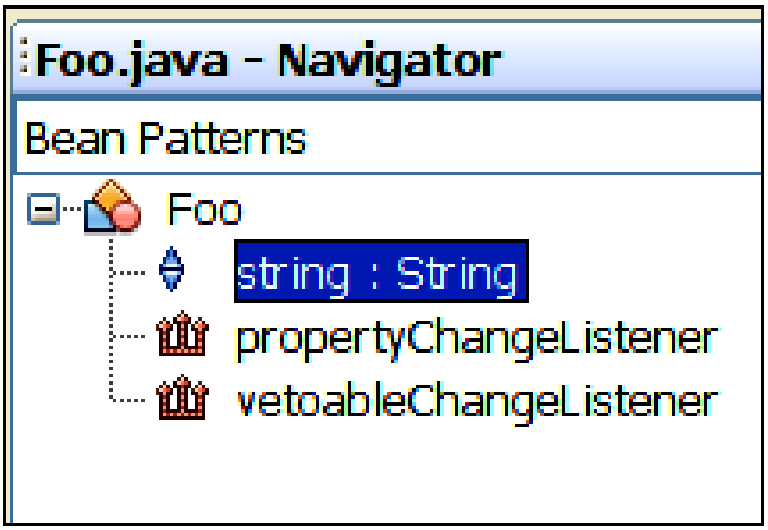
- Registro de servidores MySQL

- Ver, crear y borrar bases de datos
- Fácil lanzamiento de la herramienta de administración para MySQL



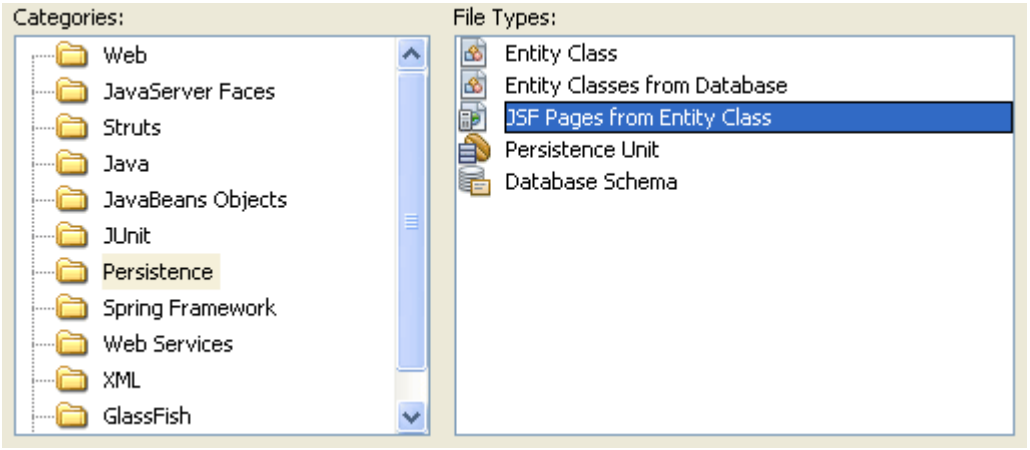
2.4.2.4.2.4 SOPORTE JAVA BEANS

- Modelos Bean en el Navegador
- Generador de Propiedades Bean
- Editor BeanInfo



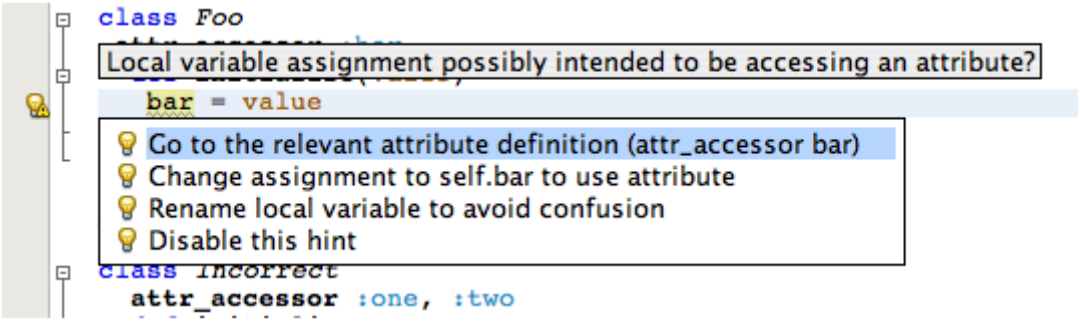
2.4.2.4.2.5 GENERADOR JSF CRUD

- Generador de aplicaciones JavaServer Faces CRUD a partir de clases de entidades.
- Soporta todo tipo de relaciones de entidades (uno-a-uno, uno-a-varios, varios-a-uno y varios-a-varios).
- Soporta todo tipo de claves principales (columna simple, compuesta y generada).



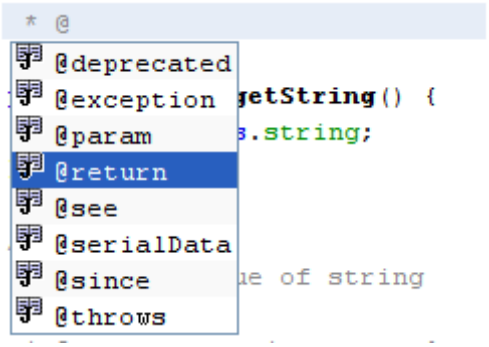
2.4.2.4.2.6 SOPORTE RUBY/JRUBY

- Mejoras en el editor, incluyendo nuevas sugerencias y soluciones
- Soporte de depuración rápida en JRuby
- Administrador de Plataforma
- Mejoras en la integración de servidores y bases de datos en proyectos Rails



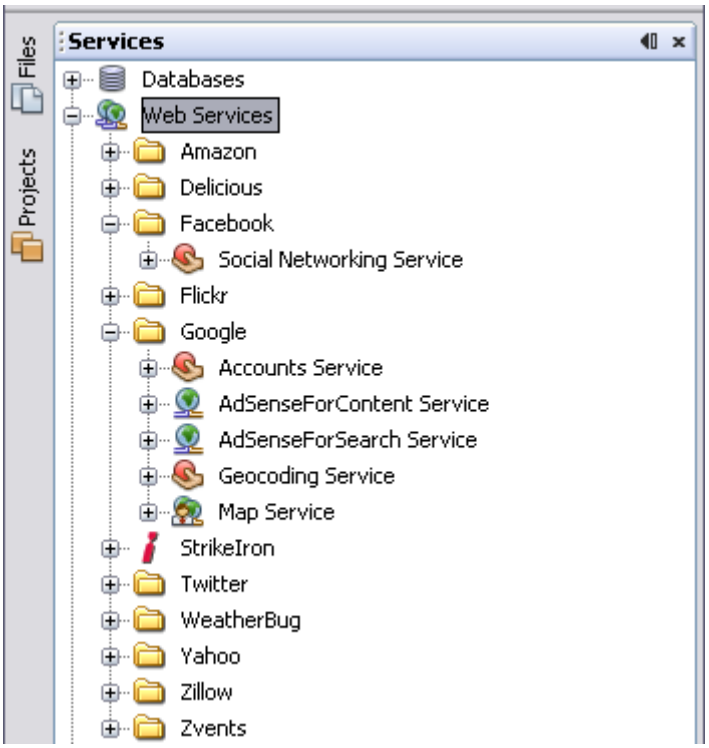
2.4.2.4.2.7 COMPLETACIÓN DE CÓDIGO JAVADOC

- Soporte de etiquetas (tags) estándares: @param, etc.
- Completar el Código para parámetros, excepciones, etc.



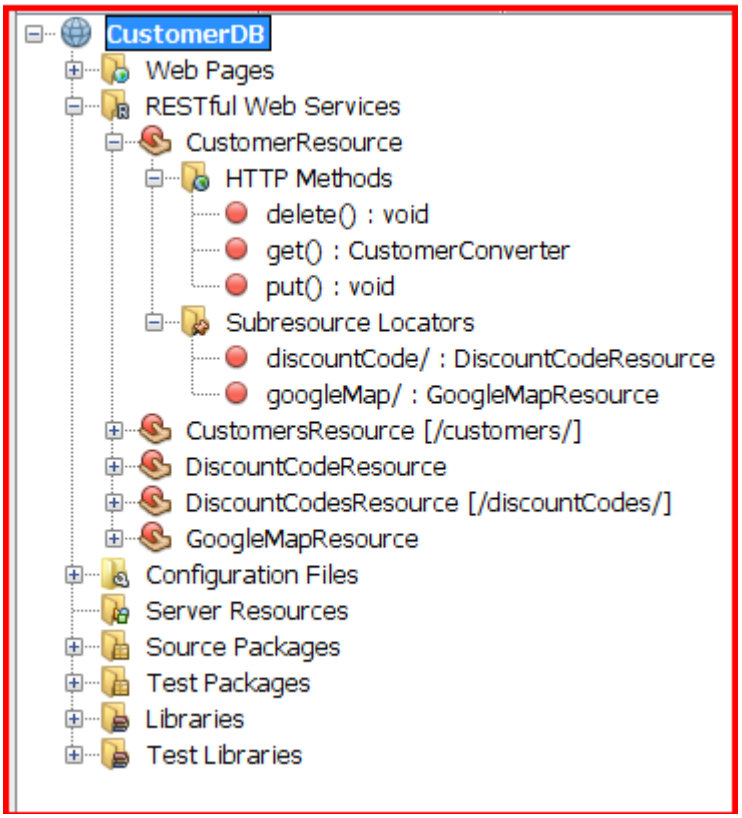
2.4.2.4.2.8 SOPORTE PARA LOS WEB APIS MÁS USADOS

- Fácil creación de aplicaciones re mezcladas (mashup)
- Operaciones de Arrastrar y soltar dentro del entorno POJO, Servlet, JSP y servicios web RESTful para que NetBeans IDE genere todo el código para acceder a los servicios
- Soporte de web APIs tales como Google, Facebook, Yahoo y YouTube



2.4.2.4.2.9 SOPORTE RESTFUL WEB SERVICE

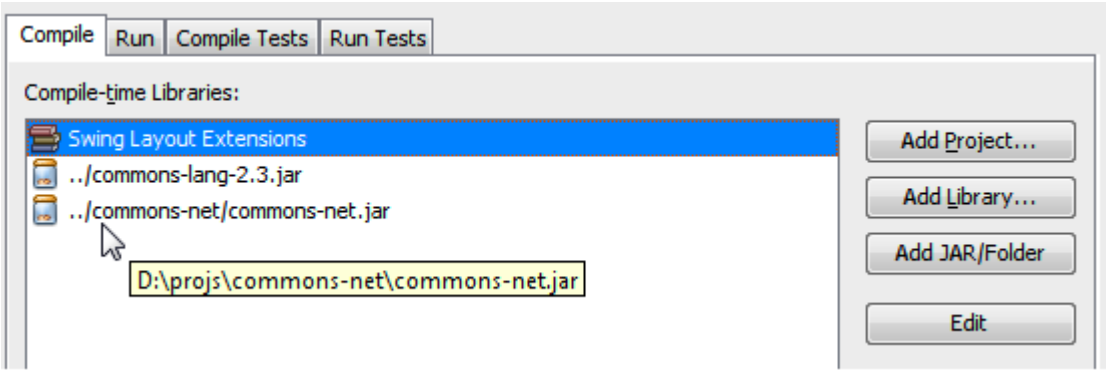
- Compilación de servicios JSR311-compliant RESTful Web utilizando Jersey
- Pruebe sus aplicaciones con el nuevo REST Test Client
- Use asistentes para crear servicios REST a partir de clases de entidades JPA y para generar aplicaciones de resumen (stubs) clientes JavaScript a partir de WADL



2.4.2.4.2.10 COMPARTIR PROYECTOS (LIBRERÍAS COMPARTIDAS AKA)

- Especificación de dependencias de librerías usando direcciones de librerías relativas (por defecto Java, Web y todos los proyectos tipo Java EE)

- Compartir proyectos más fácilmente con otros miembros de su equipo, cuando por ejemplo use un sistema de control de versión
- Habilite compilaciones no finalizadas en sus proyectos



2.4.2.4.2.11 NUEVAS EXTENSIONES (PLUGINS)

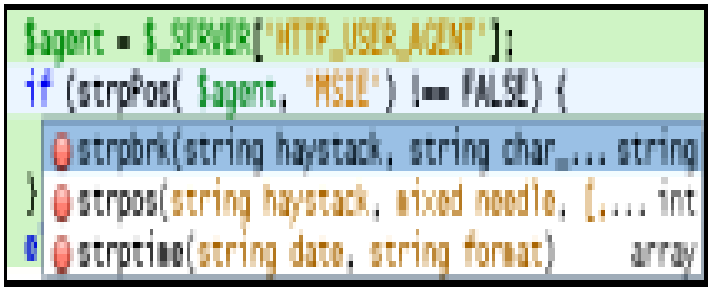
- Control de versión ClearCase
- Soporte AXIS
- Soporte SOAP UI
- Soporte Hibernate Framework (Ver mas...)

2.4.2.4.2.12 JAVA MOBILITY (APLICACIONES PARA MÓBILES)

- Emulador Mpowerplayer MIDP para aplicaciones MIDP en MacOS X (disponible en el centro de extensiones)
- Estructurador SVG (SVG Composer) para Componentes SVG de Uso Frecuente (SVG Custom Components)
- Documentación y estabilidad mejoradas

2.4.2.4.2.13 VISTA PREVIA DE LAS CARACTERÍSTICAS POST-6.1

- Soporte de Edición para PHP: Completar el código, sintaxis resaltada, navegación, depuración y mucho más
- Depurador JavaScript
- Extensión JavaFX



2.4.2.5 IMPORTANCIA.

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE.

NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.

Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo.

2.4.2.6 VENTAJAS

- Netbeans tienen la ventaja de ser libres, por lo que no necesitas pagar por desarrollar aplicaciones con el uso de este.
- No se necesita comprar ninguna licencia
- El uso de Netbeans tiene como resultado un código más simple fácil de mantener y mas legible (para los diseñadores de páginas).

2.4.2.7 DESVENTAJAS

- Netbeans consume muchos recursos de un computador.
- Para la construcción de un sistema se debe tener un conocimiento amplio de la programación en java o base solidas de este lenguaje

2.4.3 CONOCIMIENTOS BÁSICOS DE MYSQL.

2.4.3.1 INTRODUCCION.

MySQL es un sistema gestor de bases de datos relacionales en SQL, esto significa que permite la gestión de los datos de una BBDD relacional usando un lenguaje de consulta estructurado. Y, por tanto, que a partir de una oración, MySQL llevará a cabo una determinada acción sobre nuestra base de datos.

MySQL es un excelente gestor de bases de datos que la sitúan después de Oracle cómo la mejor solución a nivel técnico por las características que detallaremos en este artículo.

2.4.3.2 RESEÑA HISTORICA.

SQL (Lenguaje de Consulta Estructurado) fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL: 92, SQL: 99, SQL: 2003. MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael "Monty" 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 y flexible para sus necesidades. Esto lo llevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

El nombre de MySQL procede de la combinación de My, hija del cofundador Michael "Monty" Widenius, con el acrónimo SQL (según la documentación de la última versión en inglés). Por otra parte, el directorio base y muchas de las bibliotecas usadas por los desarrolladores tenían el prefijo My. El nombre del delfín de MySQL es Sakila y fue seleccionado por los fundadores de MySQL AB en el concurso “Name the Dolphin”. Este nombre fue enviado por Ambrose Twebaze, un desarrollador de software de código abierto africano, derivado del idioma SiSwate, el idioma local de Swazilandia y corresponde al nombre de una ciudad en Arusha, Tanzania, cerca de Uganda la ciudad origen de Ambrose.

2.4.3.3 QUE ES MYSQL.

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB — desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

2.4.3.4 DEFINICION.

El software MySQL proporciona un servidor de base de datos SQL (Structured Query Language) veloz, multihilo, multiusuario y robusto. El servidor está proyectado tanto para sistemas críticos en producción soportando intensas cargas de trabajo como para empotrarse en sistemas de desarrollo masivo de software. El software MySQL tiene licencia dual, pudiéndose usar de forma gratuita bajo licencia GNU o bien adquiriendo licencias comerciales de MySQL AB en el caso de no desear estar sujeto

a los términos de la licencia GPL. MySQL es una marca registrada de MySQL AB.

Es un sistema de gestión de base de datos (SGBD) multiusuario, multiplataforma y de código abierto. Es muy popular en sus aplicaciones de las plataformas LAMP, MAMP, WAMP; entre otras.

2.4.3.4.1 LAS CARACTERÍSTICAS PRINCIPALES DE MYSQL SON.

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y password, manteniendo un muy buen nivel de seguridad en los datos.
- Soporte a multiplataforma.
- Soporta gran cantidad de datos. Mysql Server tiene bases de datos de hasta 50 millones de registros
- Mysql contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de Mysql.
- Transacciones y claves foráneas.
- Conectividad segura

2.4.3.5 IMPORTANCIA.

Porque es un producto reconocido, de prestigio, fiabilidad, velocidad, rendimiento, facilidad de administración y conexión con otros productos,

bien documentados, con una buena evolución y soporte. Es un producto del que es fácil obtener información, con buenas herramientas, y para los que incluso podamos recibir cursos de formación si fuese necesario. Es un producto que está siendo utilizado en muchos entornos productivos y que nos dan la suficiente confianza.

2.4.3.6 VENTAJAS

- Fiabilidad, estabilidad, fácil uso
- Velocidad y precio
- Muy sencilla de manejar
- Más asequible tanto económica como en lo relativo a la dificultad
- Gratis, rapidez, robustez, interacción con web

2.4.3.7 DESVENTAJAS

- Muy limitada
- No es tan robusto como un ORACLE.
- No soportar muchos usuarios
- No tiene integridad referencial, lento con grandes bases de datos
- No tiene tantas capacidades como otros gestores profesionales

CAPÍTULO 3

3. DESARROLLO DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA.

3.1 MODELO ORIENTADO A OBJETOS (UML)

3.1.1 DEFINICIÓN

El éxito de un proyecto depende en gran medida de un buen plan y de una buena organización. En vista de ello, se hace necesario contar con herramientas
Eficientes para desarrollar sistemas.

La utilización del UML como herramienta para el Análisis y Diseño del sistema, es actualmente un estándar popular por la aceptación que ha tenido y la efectividad que ha representado para muchos analistas y diseñadores de sistemas.

3.1.1.1 OBJETIVO GENERAL

Ejemplificar claramente cada uno de los diagramas UML para el Análisis y Diseño eficiente del Control para la Emisión y Recepción de la Documentación de la ESPE-L

3.1.1.2 OBJETIVOS ESPECÍFICOS

- Mostrar los beneficios que ofrece cada uno de los diagramas UML a la hora de diseñar sistemas.
- Ofrecer un panorama claro de los usos prácticos del UML.
- Explicar los elementos principales de una interfaz gráfica de usuario, sus beneficios y usos en los sistemas informáticos actuales.

- Dar a conocer cómo los diagramas UML pueden ser fundamentales a la hora de realizar proyectos de ingeniería inversa.

3.1.1.3 JUSTIFICACIÓN

Debido a que muchos analistas de sistemas no utilizan un método ordenado y sistemático para el análisis y diseño de sus proyectos, la elaboración de este documento se justifica ante la necesidad de conocer esta popular y globalmente aceptada herramienta conocida como UML, que permite desarrollar sistemas metódicamente y paso a paso, documentando gráficamente mediante diagramas lo que se podría conocer como "el plano del sistema", si se hiciera la analogía con los planos a la hora de construir un edificio. Por esa razón, conocer sobre UML y sus usos se hace necesario y fundamental para todo analista de sistemas.

3.1.2 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE (ERS)

3.1.2.2. INTRODUCCION

Este documento detallará la Especificación de Requisitos de Software (ERS) para el Sistema de control para la emisión y recepción de la documentación de la Escuela Politécnica del Ejército extensión Latacunga, esta documentación se realiza con la colaboración de todo el personal que trabaja en los diferentes departamentos de la ESPE-L.

Este Sistema de Control Para la emisión y recepción de documentación ayudara a cubrir todas las necesidades que hoy en día existen en los diferentes departamentos de la ESPE-L, en donde se podrá realizar el almacenamiento de todos los documentos que entran y salen de la ESPE-L.

3.1.2.3. REQUERIMIENTO

El sistema estará apoyado bajo la descripción de la especificación de requisitos de software estándar IEEE-830.

3.1.2.4. PROPÓSITO

El objetivo principal es definir en una forma precisa, clara y ordenada las funcionalidades y restricciones del sistema a desarrollar.

El presente documento va dirigido al personal de desarrollo, la persona que lo pondrá a prueba y a los usuarios finales.

Esta especificación está sujeta a revisiones y modificaciones por parte de las partes implicadas, de acuerdo a sus necesidades hasta lograr la aprobación de todas las partes.

3.1.2.5. ALCANCE

El sistema se denominara con el nombre de SISCERDESPE-L (Sistema de control para la emisión y recepción de la documentación de la Escuela Politécnica del Ejercito extensión Latacunga).

Tiene como objetivo principal manejar la información de emisión y recepción de documentos que se manejan diariamente en los diferentes departamentos de la ESPE-L, debidamente elaborados y supervisados, y será aplicado en la ESPE-L.

La necesidad que impulsa a la realización del sistema es la decisión de todos los Departamentos de la ESPE-L, con el objetivo de realizar y registrar la documentación para poder disponer en cualquier momento la información en una forma muy rápida, oportuna, segura, actualizada y eficiente que ayuden a dar una buena atención a todos los usuarios

.

La situación de diseñar e implantar el sistema, es por el motivo de que no existe un sistema automatizado en la ESPE-L.

Los departamentos no proporcionan una información inmediata de los documentos enviados y recibidos de los años anteriores, por el motivo de encontrarse esta documentación en archiveros muy grandes lo cual su búsqueda causa congestión y pérdida de tiempo.

En vista que los diferentes departamentos no posee un Control Automatizado para la información de los documentos enviados y recibidos, lo cual se ha visto la necesidad de un sistema que funcione simultáneamente en cada una de los Departamentos, de manera automática, pero con capacidad de integración.

3.1.3 DEFINICIONES DE ACRÓNIMOS Y ABREBIATURAS

3.1.3.1 Definiciones

Administrador	Es la persona autorizada para **Altas, Bajas y Cambios** de la información de los Documentos Enviados y Recibidos en el sistema.
Usuario	Es la persona autorizada para Realizar y Registrar Documentos, consultas, modificaciones de la Información de los Documentos

3.1.3.2 Acrónimos

ERS	Especificación de requisitos de software
ARS	Análisis de Requerimiento del Sistema

3.1.3.3 Abreviaturas

SISCERDESPE-L	Sistema de control para la emisión y recepción de la documentación de la Escuela Politécnica del Ejercito extensión Latacunga.

3.1.3.4 Referencias

Referencia	Titulo	Ruta	Fecha	Autor
Referencia	Recomendaciones Prácticas para el Software Requisitos	ANSI/IEEE std 810	1998	IEEE

3.1.4 DESCRIPCIÓN GENERAL DEL DOCUMENTO

El documento presentará una introducción general de la especificación de requisitos de software, acompañado de una descripción general del sistema, con el objetivo de conocer las funciones y restricciones del sistema.

Una breve descripción del sistema de una manera global, las funciones que deben realizar, la información utilizada, las restricciones y otras funciones que afectan a la construcción y desarrollo del sistema.

3.1.4.1 PERSPECTIVA DEL PRODUCTO

El sistema, en primera versión no interactuará con ningún otro sistema informático.

3.1.4.2 DESCRIPCIÓN DE SISTEMA

El sistema está conformado por menús o ventanas, una base de datos, en la que se encuentran distribuidas de acorde a las gestiones requeridas por el usuario.

3.1.4.3 FUNCIONES DEL SISTEMA

En términos generales, el sistema deberá proporcionar soporte las siguientes tareas de gestión del SISCERDESPE-L:

- Gestión de Usuario
- Gestión de Departamentos
- Gestión de Personal del Departamento
- Gestión de Tipo de Documento
- Gestión de Documentos Enviados
- Gestión de Documentos Recibidos
- Gestión de Reportes

A continuación se describirá todas las tareas como estará soportado el sistema.

3.1.4.3.1 Gestión de Administrador

El Administrador será la persona encargada de administrar y manipular el Sistema de Control para la Emisión y Recepción de la Documentación de la Escuela Politécnica del Ejército Extensión Latacunga.

Es el encargado de realizar las funciones de altas, bajas, cambios, y consultas

Para dar de alta a un Administrador se ingresará los siguientes atributos como: Usuario, Nueva Contraseña, Repetir contraseña y Tipo de Usuario.

Para realizar las modificaciones el sistema permitirá realizara por medio del nombre

Para dar de baja a un Administrador se realizara una búsqueda por su nombre, el sistema permitirá la visualización de todos los atributos del administrador para ser dado de baja

La consulta General permitirá conocer todos los Administradores que se encuentran registrados en el sistema

3.1.4.3.2 Gestión de Usuario

El Usuario será la persona encargada de Administrar, Realizar, Registrar, digitar, Archivar y manipular toda la Documentación de los Diferentes Departamentos de la Escuela Politécnica del Ejército Extensión Latacunga.

El sistema permitirá realizar las funciones de altas, modificación y consultas de usuario

Para dar de alta a un usuario se ingresaran los atributos: Usuario, Nueva Contraseña, Repetir contraseña y Tipo de Usuario.

3.1.4.3.3 Gestión de Departamentos

En la gestión de Departamentos se encuentran los nombres de los Departamentos existentes en la ESPE Extensión Latacunga.

Para dar de alta a un departamento, se seleccionara el Id, nombre del departamento en caso de no existir el sistema permitirá el ingreso de los siguientes atributos como Id y Nombre del departamento

En caso de existir el sistema permitirá visualizar todos los departamentos para su registro se elegirá el nombre del departamento.

En caso de no existir el sistema permitirá ingresar un nuevo tipo de departamento.

3.1.4.3.4 Gestión de Tipo de Documentos

En la gestión de Tipo de Documento se encuentran los nombres de los Documentos que se pueden realizar y enviar.

Para dar de alta a un Tipo de documento, se seleccionara el nombre del Documento: Oficios Internos, Oficios Externos, Memorándum, Telegramas, Actas de entrega y recepción, informes, Oficios externos y Certificados.

En caso de existir el sistema permitirá visualizar todos los departamentos para su registro se elegirá el nombre del Documento

En caso de no existir un tipo de documento el programa no permitirá ingresar un nuevo documento.

3.1.4.3.5 Gestión del Personal del Departamento

En la gestión del Personal del Departamento se encuentran los nombres de las Personas Que Trabajan en los diferentes Departamentos de la ESPE Extensión Latacunga.

El sistema permitirá realizar las funciones de altas, bajas.

Para dar de alta a los atributos de una Persona que trabaja en un Departamento, ingreso los siguientes atributos Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo,

En caso de existir el sistema permitirá visualizar todas la Personas que se encuentran registradas en el sistema y se elegirá el nombre de la persona Para dar de baja a una persona el sistema permitirá la visualización de todos sus atributos de la persona ser dado de baja.

3.1.4.3.6 Gestión de Documentos Enviados.

Los documentos serán realizados y enviados de acuerdo a las necesidades de cada departamento

El sistema permitirá realizar las funciones de altas, bajas, modificación y consultas

Para dar de alta a un documento se deberá ingresar los atributos Fecha actual, No del documento, De, Para, Asunto, Tipo de Documento, Ano, Cargo De, Cargo Para, Contenido, Adjunto, elaborado por y revisado, firma.

Para realizar modificaciones el sistema permitirá realizar la selección del documento realizado y obtendrá todos los atributos del documento realizado

Para dar de baja a un documento realizado se realiza una búsqueda por su número, el sistema permitirá la visualización de todos los atributos del documento realizado para ser dado de baja

La consulta general permitirá conocer todos los documentos realizados que se encuentran registrados en el sistema

La consulta individua permitirá conocer cada uno de los documentos realizados que están almacenados en el sistema

3.1.4.3.7 Gestión de Documentos Recibidos.

Los documentos se registraran de acuerdo a su hora y fecha de llegada a cada departamento de la ESPE extensión Latacunga.

El sistema permitirá realizar las funciones de altas, bajas, y consultas

Para dar de alta a un documento se deberá ingresar los atributos Fecha actual, No del Tramite, No de Documento, Dependencia, Tipo de Documento, hora, Ano, Fecha del Documento, Recepción, asunto, para y de.

Para dar de baja a un documento registrado se realiza una búsqueda por su número, el sistema permitirá la visualización de todos los atributos del documento realizado para ser dado de baja

La consulta general permitirá conocer todos los documentos registrados que se encuentran almacenados en el sistema

La consulta individua permitirá conocer cada uno de los documentos registrados que están almacenados en el sistema

3.1.4.3.8 Gestión de Reportes

En esta gestión de Reportes se visualizara e imprimirá todos los datos requeridos por el usuario ya se ha de forma individual, general, por Tipo de departamento, tipo de documento, por rango de fechas.

En caso de no existir el dato deseado por cualquier usuario no se visualizara

3.1.5 REQUISITOS ESPECÍFICOS

Se describe los requisitos funcionales que serán satisfechos por el prototipo, todos los requisitos aquí expuestos son esenciales, es decir, no sería aceptable un sistema que no satisfaga. Estos requisitos se han especificado teniendo en cuenta el criterio de teste habilidad: dado un requisito, debería ser fácilmente demostrable si es satisfecho o no por el sistema.

3.1.6 REQUISITOS FUNCIONALES

Es el comportamiento interno del sistema, cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

Un requisito funcional típico contiene un nombre y un número de serie único y un resumen. Esta información se utiliza para ayudar al lector a entender por qué el requisito es necesario, y para seguir al mismo durante el desarrollo del producto.

3.1.7 MODELO DE CASO DE USO DEL ACTOR ADMINSTRADOR O USUARIO

3.1.7.1 Gestión de Administrador

Esta gestión deberá permitir al administrador realizar las siguientes funciones:

- Ingresar un nuevo usuario
- Modificar datos del Usuario
- Eliminar los datos de un Usuario
- Consultas

3.1.7.2 Gestión de Usuarios

El administrador deberá permitir al usuario realizar las siguientes funciones:

- Ingresar un nuevo usuario
- Modificar datos del Usuario
- Eliminar usuario
- Consulta individual y general del usuario.

3.1.7.3 Gestión de Departamentos

En esta ventana nos permitirá realizar las siguientes funciones.

- Ingresar un nuevo departamento
- Modificar Un registro
- Eliminar Departamento
- Consultar

3.1.7.4 Gestión de Personal del Departamento

En esta ventana nos permitirá realizar las siguientes funciones.

- Ingresar un nuevo registro
- Modificar un registro
- Eliminar un registrar
- Consultar

3.1.7.5 Gestión de Tipo de Documento

En esta ventana nos permitirá realizar las siguientes funciones.

- Ingresar un nuevo registro
- Modificar un registro
- Eliminar un registrar
- Consultar

3.1.7.6 Gestión de Documentos Enviados

En esta ventana nos permitirá realizar las siguientes opciones.

- Ingresar un nuevo documento
- Modificar datos del documento
- Eliminar un Registro
- Buscar un documento guardado
- Reportes

3.1.7.7 Gestión de Documentos Recibidos

En esta ventana nos permitirá realizar las siguientes opciones.

- Ingresar un nuevo documento
- Modificar datos del documento
- Eliminar un Registro
- Buscar un documento guardado
- Reportes

3.1.7.8 Gestión de Reportes

En este menú podremos realizar las siguientes funciones de reportes de los Documentos Enviados y Recibidos.

- Verificación de datos Individuales
- Verificación de datos Generales
- Verificar por fechas
- Verificar por numero
- Imprimir, visualizar.

3.1.8 REQUISITOS DE INTERFACES EXTERNAS

3.1.8.1 Interfaces del Usuario

La interfaz debe ser orientado a través de ventanas o menús, el manejo del programa se realizara a través del teclado y mouse.

3.1.8.2 Interfaces Hardware

No se definido

3.1.8.3 Interfaces Software

SISCERDESPE-L tiene una interfaz cliente

3.1.8.4 Interfaces Comunicación

SISCERDESPE-L está orientado para funcionar mediante una base de datos y una impresora.

3.1.9 REQUISITOS DE DESARROLLO

El ciclo de vida elegido para desarrollar el prototipo en el interactivo incremental, básico por su flexibilidad para incorporar cambios.

3.1.10 REQUISITOS TECNOLÓGICOS DE HADWARE Y SOFTWARE

	Interface	Servidor
Sistema Operativo Ubuntu 10.10	Multiplataforma	Mysql

3.1.10.1 Seguridad

El sistema permitirá el manejo de la información únicamente de las Personal del Departamento autorizadas, es decir el administrador y usuarios los mismos que poseerán un Tipo de usuario, un Login y una contraseña, la información será verificada por el sistema para comprobar que se trata de un usuario registrado en el prototipo. Si el usuario introducido corresponde a un usuario no identificado, este no podrá ingresar al sistema.

3.1.10.2 Administrador del Sistema

Se encargara de definir los perfiles de los diferentes usuarios, que podrán acceder al sistema.

3.1.11 ANÁLISIS Y PLANIFICACIÓN

3.1.11.1 Primera Interacción

Con los requerimientos recolectados en la etapa de exploración, se establece el desarrollo con la estimación de la duración del proyecto y la fabricación de un plan de entregas.

En base al plan de entregas inicial, se determina en cada iteración una planificación concreta para las historias de usuario fijadas, la estimación del tiempo de implementación y prioridad pueden restablecerse en caso de requerirse.

3.1.11.2 Priorización y Estimación

En la Tabla 3.2.1.1 se presenta la estimación del tiempo requerido para la implementación y la prioridad de las historias de usuario recolectadas, tomando en cuenta que un día laborable para el equipo de desarrollo es de 8 horas.

Tabla 3.1. Puntos de Función

Historias de Usuarios	Tiempo de Estimación		
	Prioridad	Días Estimados	Horas/Hombre Estimación
01-Datos del Usuario	Alta	5	40
02-Datos del Departamento	Alta	5	40
03-Datos de la Persona	Alta	5	40
04-Datos del Tipo documento	Alta	5	40
02-Datos de los Documentos Enviados	Alta	10	80

03-Datos de los Documentos Recibidos	Alta	10	80
10-Reportes	Alta	10	80
Total		**50**	**400**

3.1.11.3 Distribución Funcional

Se identificó los módulos de la aplicación, con el propósito de tomar en cuenta la distribución funcional de las historias de usuario recolectadas en la elaboración del plan de entrega, a más de la prioridad y el tiempo de implementación estimado.

3.1.11.4 Estimación Durante el Proyecto

Para establecer una estimación de la duración del proyecto se tiene en cuenta el tiempo total de horas que tomará la implementación de las historias de usuario, además el equipo de desarrollo trabajará 5 días a la semana con una duración 8 horas diarias.

Tabla 3.2. Estimación Duración del Proyecto.

Actividades	**Tiempo en Horas**
Requisitos definidos en la etapa de explotación	200
Requerimientos Futuros (a)	100
Imprevistos (b)	20
Pruebas de aceptación y reuniones (c)	250
Total	**570**

El 30% del total de horas estimado para la implementación de los requerimientos recolectados en la etapa de exploración, para requerimientos no descubiertos inicialmente.

- 5% del tiempo de horas estimado para requerimientos futuros.
- 45% del total de horas para las pruebas.
- 20% del total de horas estimado para la implementación.

Después de realizar los cálculos respectivos se estima que la duración del proyecto será de 5 a 6 meses.
Una vez determinada la estimación de la duración del proyecto, se establece el plan de entrega haciendo uso de las estimaciones de las historia de usuario.

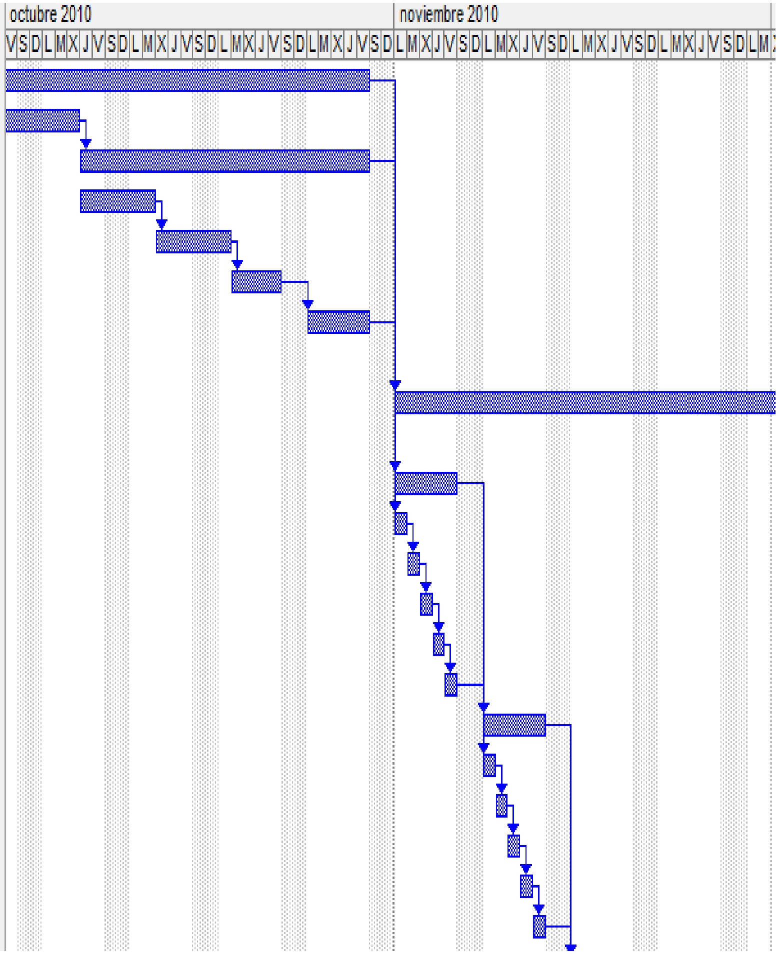
3.1.11.5 Plan de Entregas

Tabla 3.3. Plan de entregas de acuerdo a la prioridad de las historias de usuario

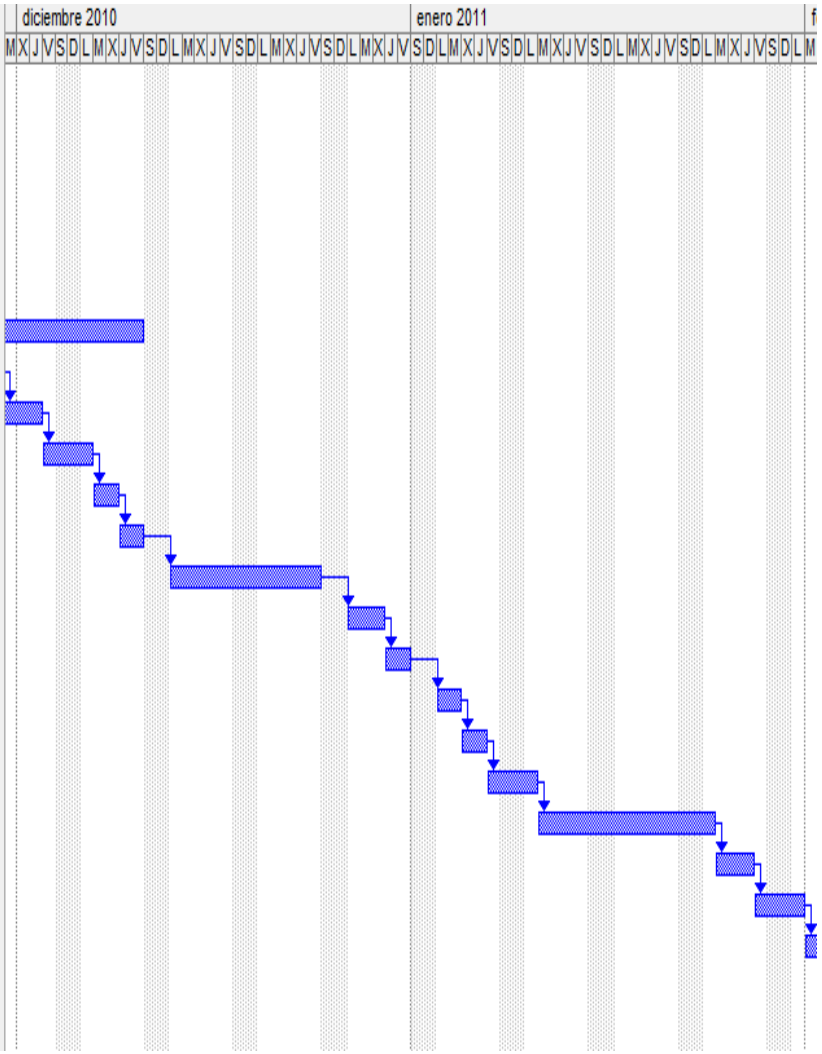
Historia de Usuario		Tiempo Estimado	Interacción Asignada			Entrega Asignada		
	Días Estimado	Horas/Hombre	1	2	3	1	2	3
01-Especificación de Requisitos.	41	328						
02-Datos del Usuario	3	24	x			x		
03-Datos del Departamento	2	16	x			x		
04-Datos de la Persona	2	16	x			x		
05-Datos del Tipo Documento	2	16	x			x		
06-Datos de Los Documentos Enviados	5	40		x		x		
07-Datos de los Documentos Recibidos	5	40		x		x		
08-Reportes	3	24			x			x

3.1.11.6 Plan Estratégico de Planificación

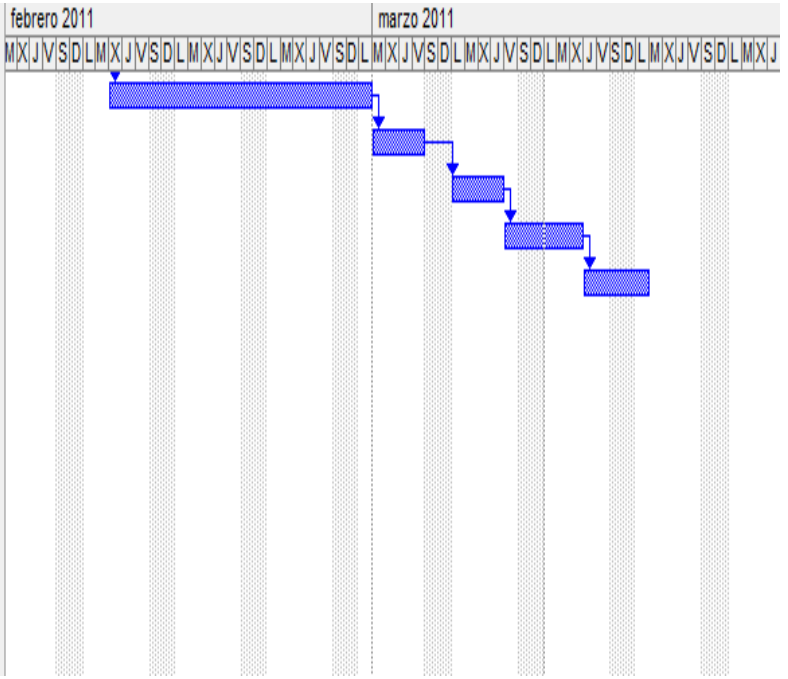
	Nombre de tarea	Duración	Comienzo	Fin	Predeces
1	ANALISIS	21 días?	vie 01/10/10	vie 29/10/10	
2	HISTORIAS DEL USUARIO	4 días?	vie 01/10/10	mié 06/10/10	
3	UML	17 días?	jue 07/10/10	vie 29/10/10	2
4	Diagrama de Caso de Uso	4 días?	jue 07/10/10	mar 12/10/10	
5	Diagrama de Caso de Uso Expandido	4 días?	mié 13/10/10	lun 18/10/10	4
6	Diagrama de caso de Uso de Secuencia	4 días?	mar 19/10/10	vie 22/10/10	5
7	Diagrama de Clases	5 días?	lun 25/10/10	vie 29/10/10	6
8					
9	DISEÑO	79 días?	lun 01/11/10	jue 17/02/11	7
10					
11	DATOS DEL USUARIO	5 días?	lun 01/11/10	vie 05/11/10	3
12	Diseño de la interfaz Datos Usuario	1 día?	lun 01/11/10	lun 01/11/10	1
13	Validacion de datos de la interfaz datos del Usuario	1 día?	mar 02/11/10	mar 02/11/10	12
14	Almacenar Datos del Usuario en la base de datos	1 día?	mié 03/11/10	mié 03/11/10	13
15	Pruebas de aceptacion de los Datos Usuario	1 día?	jue 04/11/10	jue 04/11/10	14
16	Documentacion de los datos de Aceptacion	1 día?	vie 05/11/10	vie 05/11/10	15
17	DATOS DEL DEPARTAMENTO	5 días?	lun 08/11/10	vie 12/11/10	16
18	Diseño de la interfaz Datos Departamento	1 día?	lun 08/11/10	lun 08/11/10	11
19	Validacion de datos de la interfaz datos del Departamento	1 día?	mar 09/11/10	mar 09/11/10	18
20	Almacenar Datos del Departamento en la base de datos	1 día?	mié 10/11/10	mié 10/11/10	19
21	Pruebas de aceptacion de los Datos del Dapartamento	1 día?	jue 11/11/10	jue 11/11/10	20
22	Documentacion de los datos de Aceptacion	1 día?	vie 12/11/10	vie 12/11/10	21



	Nombre de tarea	Duración	Comienzo	Fin	Prede
23	**DATOS DE LA PERSONA**	10 días?	lun 15/11/10	vie 26/11/10	22
24	Diseño de la interfaz Datos de la Persona	1 día?	lun 15/11/10	lun 15/11/10	17
25	Validacion de datos de la interfaz datos de la Persona	3 días?	mar 16/11/10	jue 18/11/10	24
26	Almacenar Datos de la Persona en la base de datos	2 días?	vie 19/11/10	lun 22/11/10	
27	Pruebas de aceptacion de los Datos de la Persona	2 días?	mar 23/11/10	mié 24/11/10	26
28	Documentacion de los datos de Aceptacion	2 días?	jue 25/11/10	vie 26/11/10	27
29	**DATOS DEL TIPO DOCUMENTO**	10 días?	lun 29/11/10	vie 10/12/10	28
30	Diseño de la interfaz Datos del tipo documento	1 día?	lun 29/11/10	lun 29/11/10	23
31	Validacion de datos de la interfaz datos del tipo documento	3 días	mar 30/11/10	jue 02/12/10	30
32	Almacenar Datos del tipo documento en la base de datos	2 días	vie 03/12/10	lun 06/12/10	31
33	Pruebas de aceptacion de los Datos del tipo documento	2 días	mar 07/12/10	mié 08/12/10	32
34	Documentacion de los datos de Aceptacion	2 días	jue 09/12/10	vie 10/12/10	33
35	**DATOS DE LOS DOCUMENTOS ENVIADOS**	10 días	lun 13/12/10	vie 24/12/10	34
36	Diseño de la interfaz Datos de los documentos enviados	3 días	lun 27/12/10	mié 29/12/10	35
37	Validacion de datos de la interfaz datos de los documentos enviados	2 días	jue 30/12/10	vie 31/12/10	36
38	Almacenar Datos de los documentos enviados en la base de datos	2 días	lun 03/01/11	mar 04/01/11	37
39	Pruebas de aceptacion de los Datos de los documentos enviados	2 días	mié 05/01/11	jue 06/01/11	38
40	Documentacion de los datos de Aceptacion	2 días	vie 07/01/11	lun 10/01/11	39
41	**DATOS DE LOS DOCUMENTOS REGISTRADOS**	10 días	mar 11/01/11	lun 24/01/11	40
42	Diseño de la interfaz Datos de los documentos registrados	3 días	mar 25/01/11	jue 27/01/11	41
43	Validacion de datos de la interfaz datos de los documentos registrados	2 días	vie 28/01/11	lun 31/01/11	42
44	Almacenar Datos de los documentos registrados en la base de datos	2 días	mar 01/02/11	mié 02/02/11	43
45	Pruebas de aceptacion de los Datos de los documentos registrados	2 días	jue 03/02/11	vie 04/02/11	44
46	Documentacion de los datos de Aceptacion	2 días	lun 07/02/11	mar 08/02/11	45



	Nombre de tarea	Duración	Comienzo	Fin	Prede
47	REPORTES	14 días	mié 09/02/11	lun 28/02/11	46
48	Diseño de la interfaz Datos par los reportes	4 días	mar 01/03/11	vie 04/03/11	47
49	Validacion de datos de la interfaz datos de los reportes	4 días	lun 07/03/11	jue 10/03/11	48
50	Pruebas de aceptacion de los Datos para los reportes	4 días	vie 11/03/11	mié 16/03/11	49
51	Documentacion de los Datos de Aceptacion	3 días	jue 17/03/11	lun 21/03/11	50



Figuras 3.1: Plan Estratégico de Tiempos

3.1.12 FASE DE ANÁLISIS

3.2.2.1 Especificación de Requerimientos (XP)

Las Historias de Usuarios son tarjetas escritas por el cliente en un lenguaje natural para este, indicando las necesidades que el sistema debe satisfacer.

A continuación se describen las historias de usuario finales, las mismas que han sido agrupadas en los respectivos componentes del sistema. Algunas de estas historias fueron eliminadas o cambiadas a lo largo del proyecto, a medida que cambiaban los requisitos del cliente o se tenía una concepción más clara del proyecto.

3.2.2.2 Primera Interacción

Administración de altas

Tabla 3.4. Historia de los Datos del usuario

Historia de Usuario	
Número:00 1	**Usuario:** Usuario
Nombre Historia: Datos del Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 24	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere tener usuarios con su respectivas Autorizaciones para el manejo del sistema, el usuario debe proveer sus datos básicos como el número de Tipo de usuario, Login y Contraseña. Además se realizarán las funciones de altas, bajas, modificación y consultas del usuario	
Observaciones: Esta ficha no está vinculada a ninguna historia	

Tabla 3.5. Historia de los Departamentos

Historia de Usuario	
Número: 002	**Usuario:** Departamento
Nombre Historia: Datos de los Departamentos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 26	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere que cada usuario seleccione a su respectivo departamento de trabajo para que se guarden los documentos realizados y registrados en cada uno de los departamentos para tener un mejor control.	
Observaciones Esto ficha no está vinculada a ninguna historia	

Tabla 3.6. Historia de las Personas de los Departamentos

Historia de Usuario	
Número: 002	**Personas de los Departamentos**
Nombre Historia: Datos de las Personas de los Departamentos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 26	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere tener las personas que trabajan en los departamentos para que el usuario seleccione las Personas de los Departamentos que van en el documentó que va a ser realizado los mismos que deben proveer sus datos como son: Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo.	

Además se realizarán las funciones de altas, bajas, modificación y consultas del usuario
Observaciones Esto ficha no está vinculada a ninguna historia

Tabla 3.7. Historia de los Tipos de Documentos

Historia de Usuario	
Número: 002	**Tipo documento**
Nombre Historia: Datos de los Tipos de documentos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 26	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere tener los documentos que más se utilizan en los departamentos para que el usuario seleccione el tipo de documentó que va a realizar para enviarlo los mismos que deben proveer sus datos como son: Fecha actual, No del documento, De, Para, Asunto, Tipo de Documento, Ano, Cargo De, Cargo Para, Contenido, Adjunto, elaborado por y revisado, firma.	
Observaciones Esto ficha no está vinculada a ninguna historia	

Tabla 3.8. Historia de los Datos de los Documentos Enviados

Historia de Usuario	
Número: 002	**Usuario:** Documentos Enviados
Nombre Historia: Datos de los Documentos Enviados	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Horas estimadas: 26	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere realizar un documento de cualquier tipo en el cual el usuario podrá seleccionar que tipo de documento va a realizar al escoger el tipo de documento que va a realizar el documento deberá proveer los siguientes datos como son: Fecha actual, No del documento, De, Para, Asunto, Tipo de Documento, Ano, Cargo De, Cargo Para, Contenido, Adjunto, elaborado por y revisado, firma. Además se realizarán las funciones de altas, bajas, modificación y consultas del Documento Realizado	
Observaciones Esto ficha no está vinculada a ninguna historia	

Tabla 3.9. Historia de los Documentos recibidos

Historia de Usuario	
Número: 002	**Usuario:** Documentos Recibidos
Nombre Historia: Datos de los Documentos Recibidos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 26	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: se requiere almacenar todos los documentos que ingresan a dicho departamento para poder mantener un control de todos los documentos que ingresan a ese departamento, el mismo documento que ingresa debe proveer los siguientes datos para poder registrar dicho documento: Fecha actual, No del Tramite, No de Documento, Dependencia, Tipo de Documento, hora, Ano, Fecha del Documento, Recepción, asunto, Para: y De:. Además se realizarán las funciones de altas, bajas, modificación y consultas	
Observaciones: : Esto ficha no está vinculada a ninguna historia	

Tabla 3.10. Historia de Reportes

Historia de Usuario	
Número: 002	**Usuario:** Reportes
Nombre Historia: Reportes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Horas estimadas: 24	**Integración asignada:**1
Programador responsable: Equipo X.P	
Descripción: Se requiere obtener reportes para una clara obtención de los Documentos de Realizados y Recibidos donde el sistema permitirá realizar los reportes individuales, generales y por fechas.	
Observaciones: Esta ficha no está vinculada con ninguna historia. .	

3.2.2.3 Formulario de Entrada

3.1.12.3.1 Ingreso de Datos del Administrador

DATOS INFORMATIVOS DEL ADMINISTRADOR
Login: Contraseña:

3.1.12.3.2 Ingreso del Usuario

DATOS INFORMATIVOS DEL USUARIO
Usuario: Contraseña:

3.2.2.4 DIAGRAMA DE CASOS DE USO DE LOS ACTORES

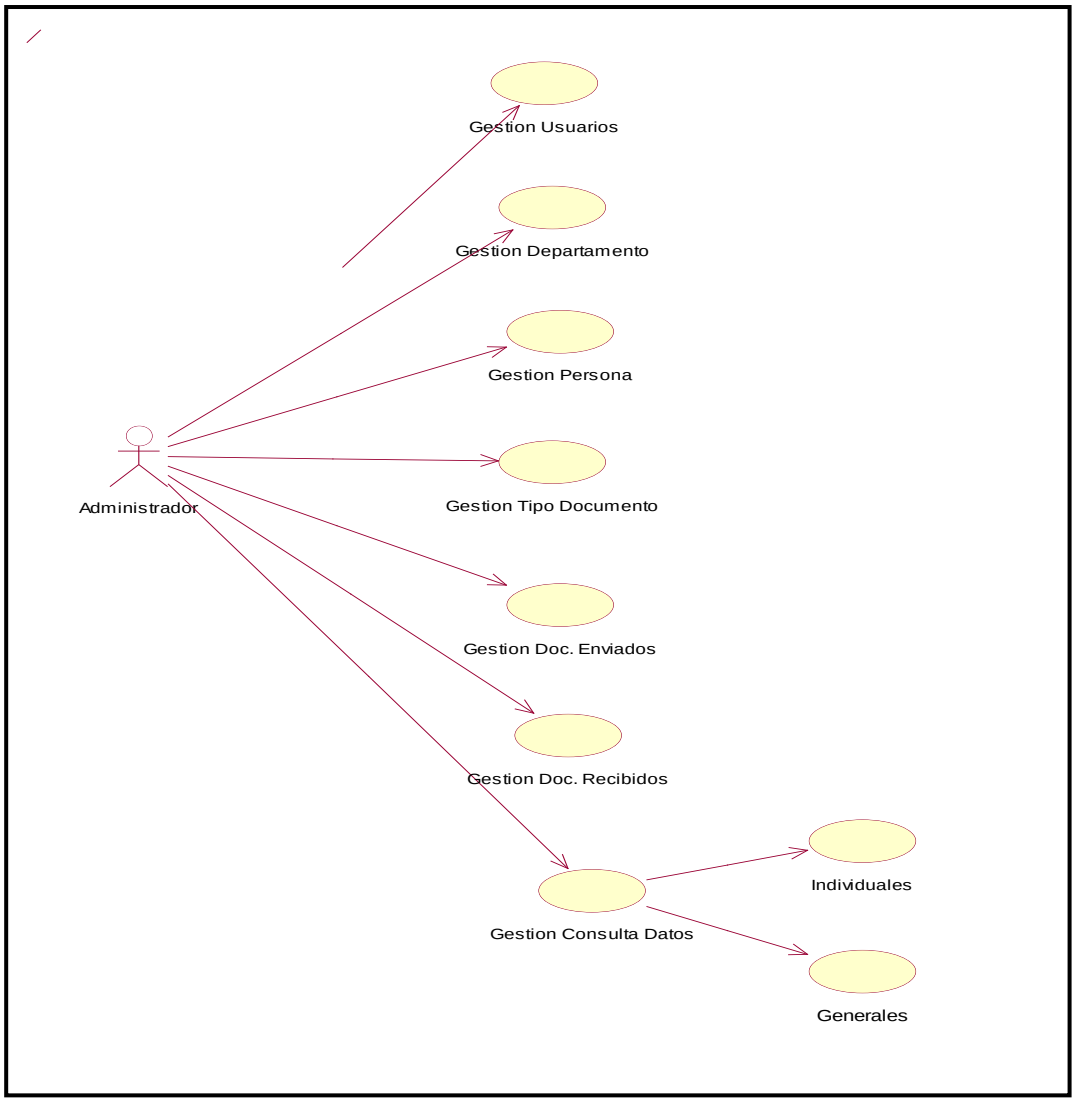
En esta sección hemos incluido los diagramas de casos de uso de nuestro sistema, desarrollados con la herramienta Rational Rose 2000.

Diagrama de los Actores principales:

a) Administrador

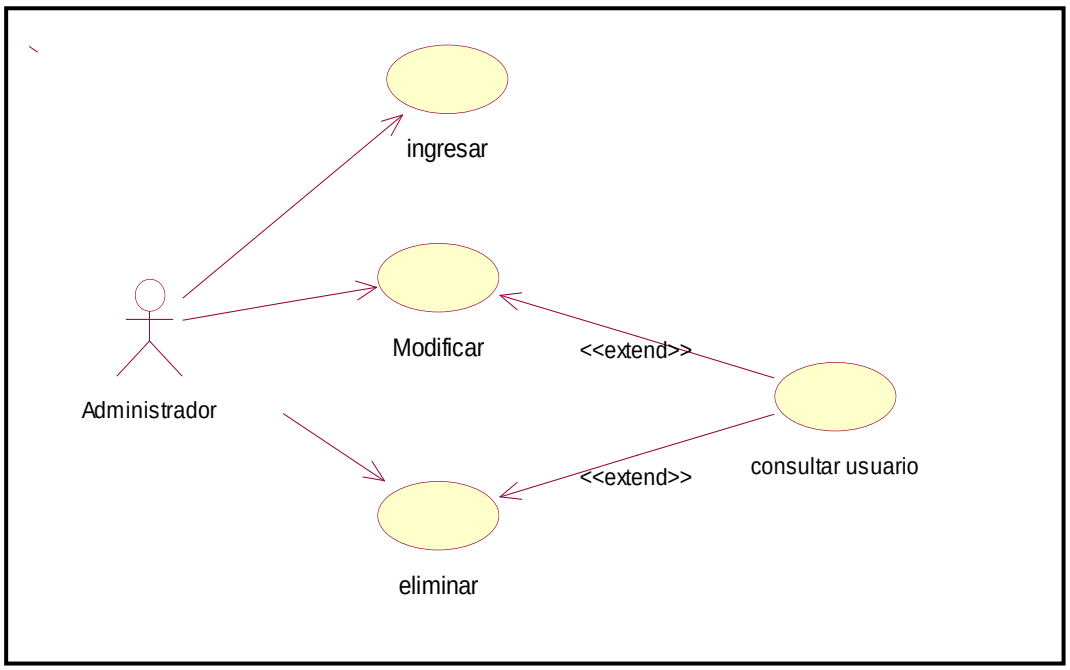
b) Usuario

3.1.12.4.1 DIAGRAMA DE CASO DE USO ADMINISTRADOR



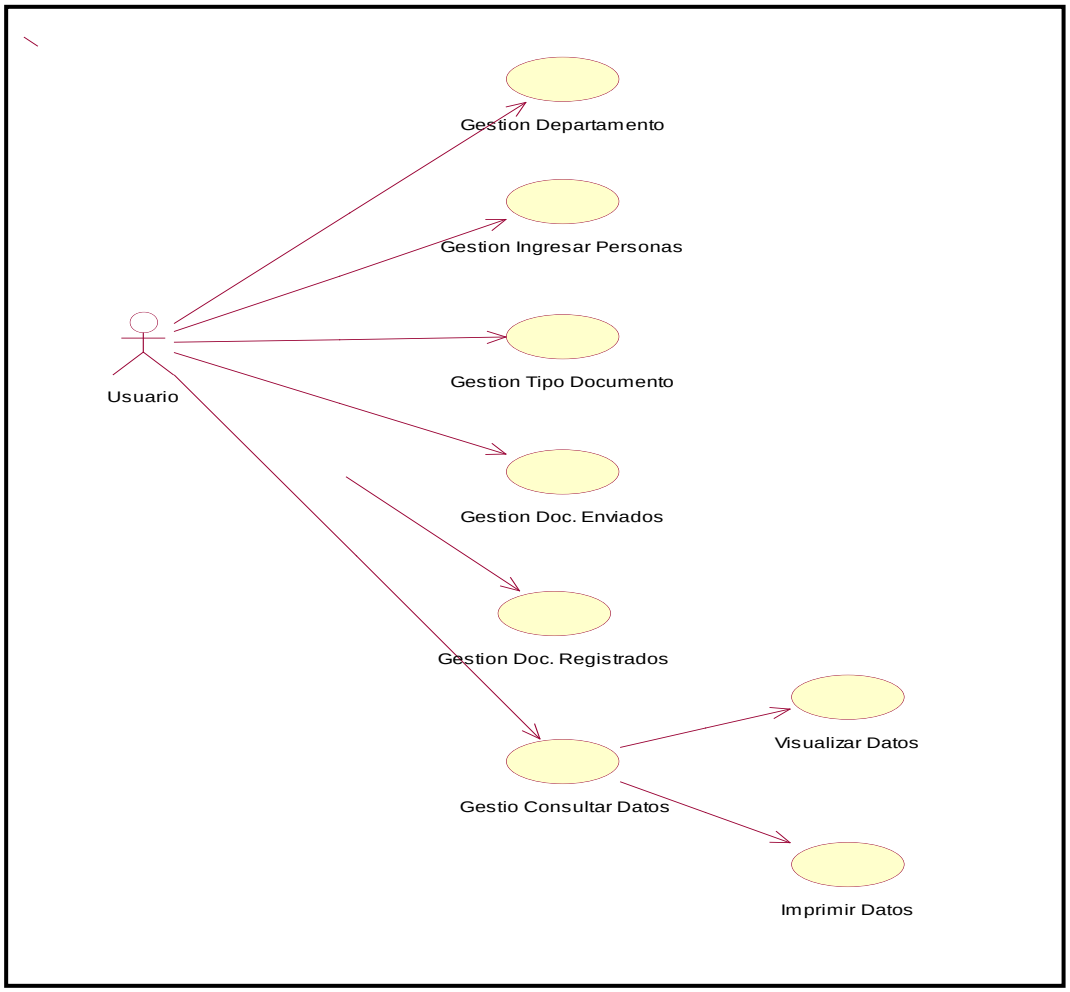
Figuras 3.2: Diagrama de caso de uso Administrador

3.1.12.4.2 DIAGRAMA DE GESTION DE SUARIO



Figuras 3.3: Diagrama de Gestión de Usuario

3.1.12.4.3 DIAGRAMA DE CASO DE USO USUARIO



Figuras: 3.4: Diagrama de caso de uso Usuario

3.2.2.5 CASOS DE USOS EXPANDIDOS

3.1.12.5.1 MODELO DE CASO DE USO DEL ACTOR DMINISTRADOR

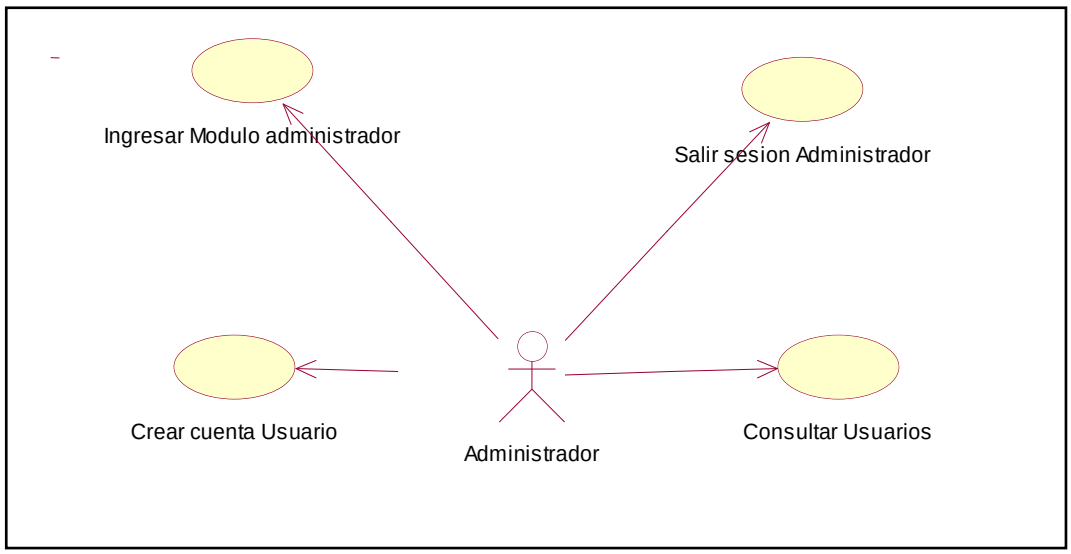


Figura 3.5: Ingresar Modulo Administrador

3.1.12.5.2 MODELO DE CASO DE USO INGRESAR MODULO DMINISTRADOR

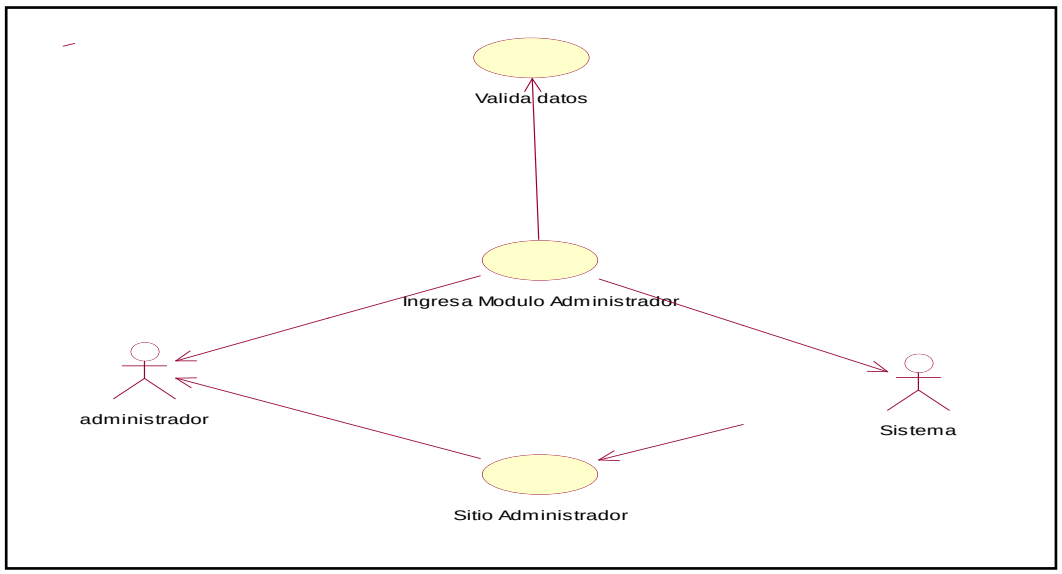


Figura 3.6: Caso de uso Ingresar Modulo Administrador

Tabla 3.11. Caso de uso Ingresar Modulo Administrador

Caso de Uso	Ingresar Modulo Administrador
Actores	Administrador (iniciador)
Propósito	Permitir a un Administrador Ingrese al sitio de Administradores.
Tipo	Primario Real
Visión General	El Administrador solicita la operación Ingresar Modulo administrador, el sistema presenta ventana de ingreso de modulo del Administrador, el administrador ingresa su login y password el sistema valida los datos si son correctos el administrador podrá acceder al sitio de administradores, el sistema confirma operación.
Referencias	

Curso Típicos de Eventos

1. El administrador solicita la operación Ingresa Modulo Administrador.
2. El sistema presenta ventana de Ingreso a Modulo Administrador.
3. El Administrador ingresa su login y password.
4. El sistema valida los datos ingresados por el Administrador y confirma operación

Curso Alternativos

Curso Típico 2: No existe ventana de ingreso a Modulo de Administradores, termina caso de uso.

Curso Típico 4: Los Datos no están bien ingresados, termina el caso de uso

3.1.12.5.3 CREAR CUENTA USUARIO

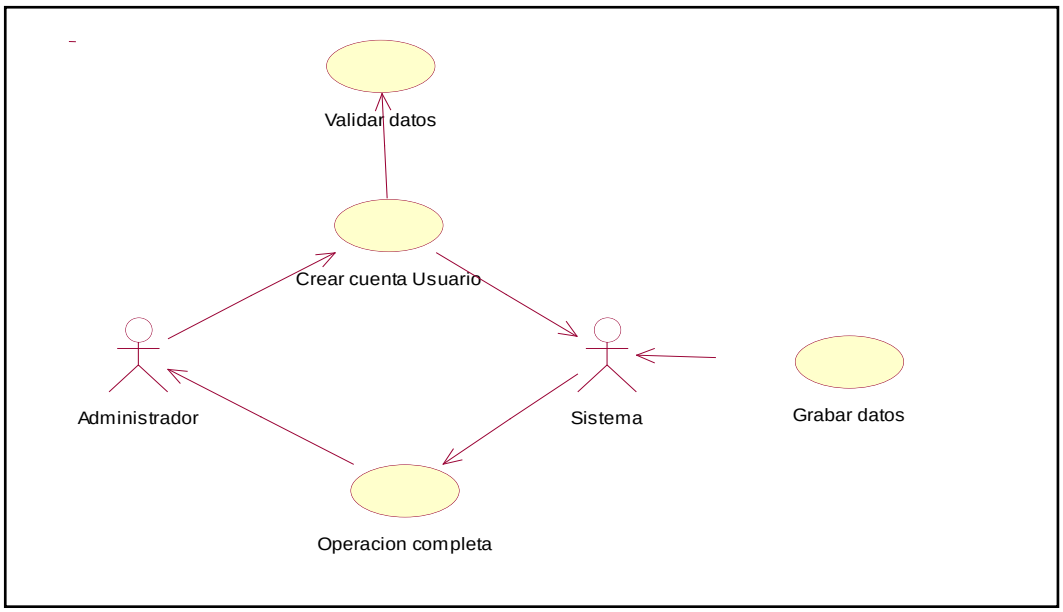


Figura 3.7: Caso de Uso Crear Cuenta Usuario

Tabla 3.12. Caso de uso crear cuenta usuario

Caso de Uso	Crear cuenta usuario
Actores	Administrador (iniciador)
Propósito	Crear una nueva cuenta de Usuario para el sistema
Tipo	Primario Real
Visión General	El administrador solicita la operación Crear Cuenta usuario, el sistema presenta formulario de registro de un nuevo Usuario, el administrador ingresa uno a uno los datos necesarios para la creación de una nueva cuenta de usuario, el sistema almacena los datos del usuario y confirmar operación
Referencias	

Curso Típicos de Eventos

5. El caso de uso comienza cuando el Administrador solicita la operación Crear Cuenta Usuario.
6. El sistema presenta formulario de registro de una nueva cuenta de usuario
7. El Administrador ingresa los datos solicitados por el sistema.
8. El Sistema almacena los datos del usuario y confirma operación.

Curso Alternativos

Curso Típico 2: No existe formulario de registro de una nueva cuenta de usuario, termina caso de uso.

Curso Típico 4: Los datos no están bien ingresados termina el caso de uso.

3.1.12.5.4 CONSULTAR USUARIOS

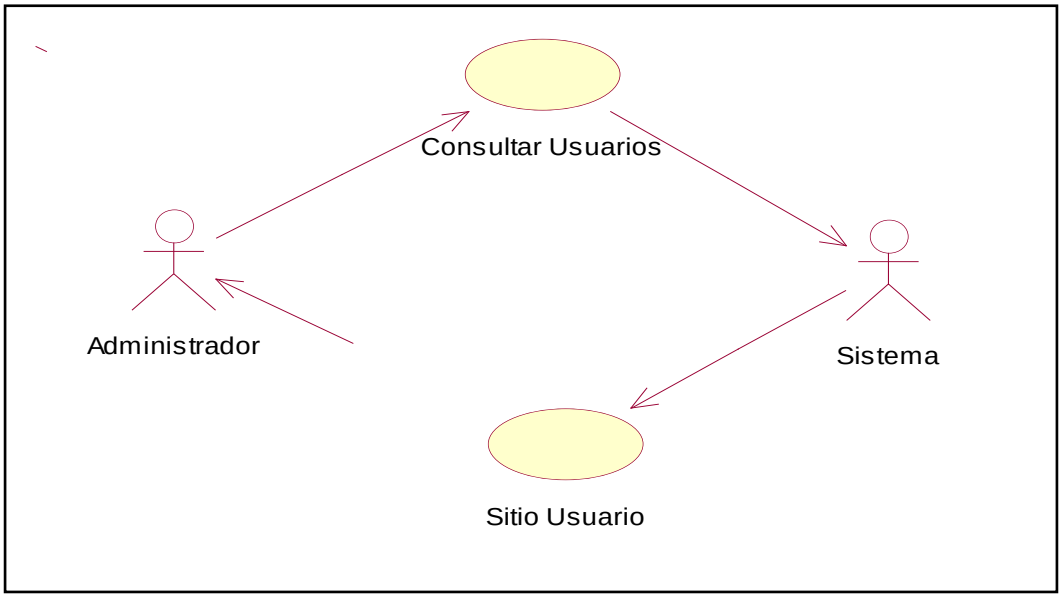


Figura 3.8: Caso de Uso Consultar Usuario

Tabla 3.13. Caso de uso consultar usuarios

Caso de Uso	Consultar Usuarios
Actores	Administrador (iniciador)
Propósito	Observar los usuarios registrados en el sistema
Tipo	Primario Real
Visión General	El administrador solicita la operación Consultar usuarios, el sistema presenta una lista de los usuarios registrados en el sistema y confirma operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Administrador solicita la operación Consultar Usuarios.
2. El sistema presenta una lista de los usuarios en el sistema.
3. El Sistema confirma operación.

Curso Alternativos

Curso Típico 2: No existe formulario de registro de una nueva cuenta de usuario, termina caso de uso.

3.1.12.5.5 SALIR SESION ADMINISTARDOR

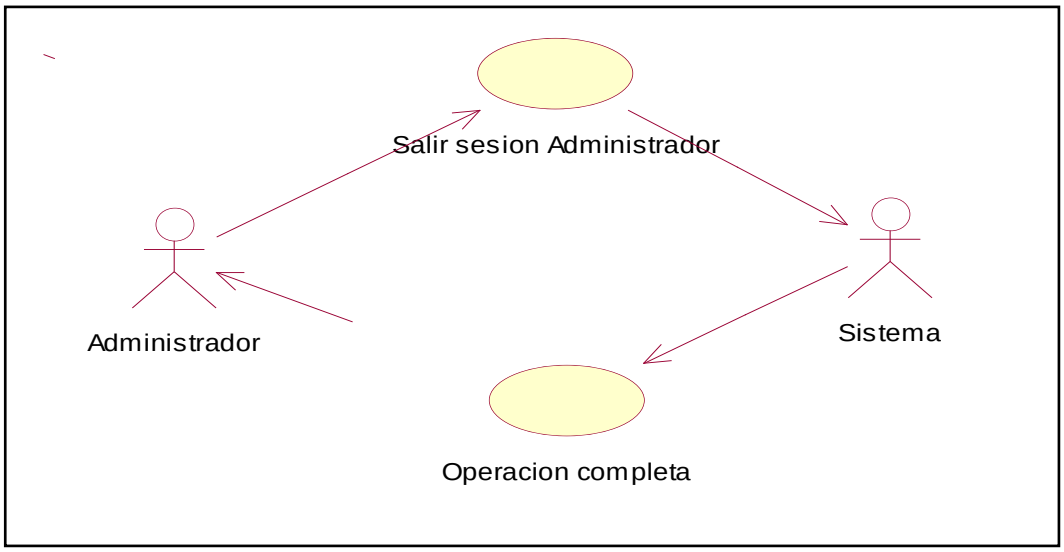


Figura 3.9: Caso de Uso Salir Sesión Administrador

Tabla 3.14. Caso de uso salir sesión administrador.

Caso de Uso	Salir sesión Administrador
Actores	Administrador (iniciador)
Propósito	Permitir a un Administrador abandonar el sitio Administrador.
Tipo	Primario Real
Visión General	El administrador solicita la operación Salir sesión Administrador, el sistema confirma operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Administrador solicita la operación salir Sesión Administrador.
2. El Sistema confirma operación.

3.1.12.5.6 MODELO DE CASO DE USO DEL ACTOR USUARIO

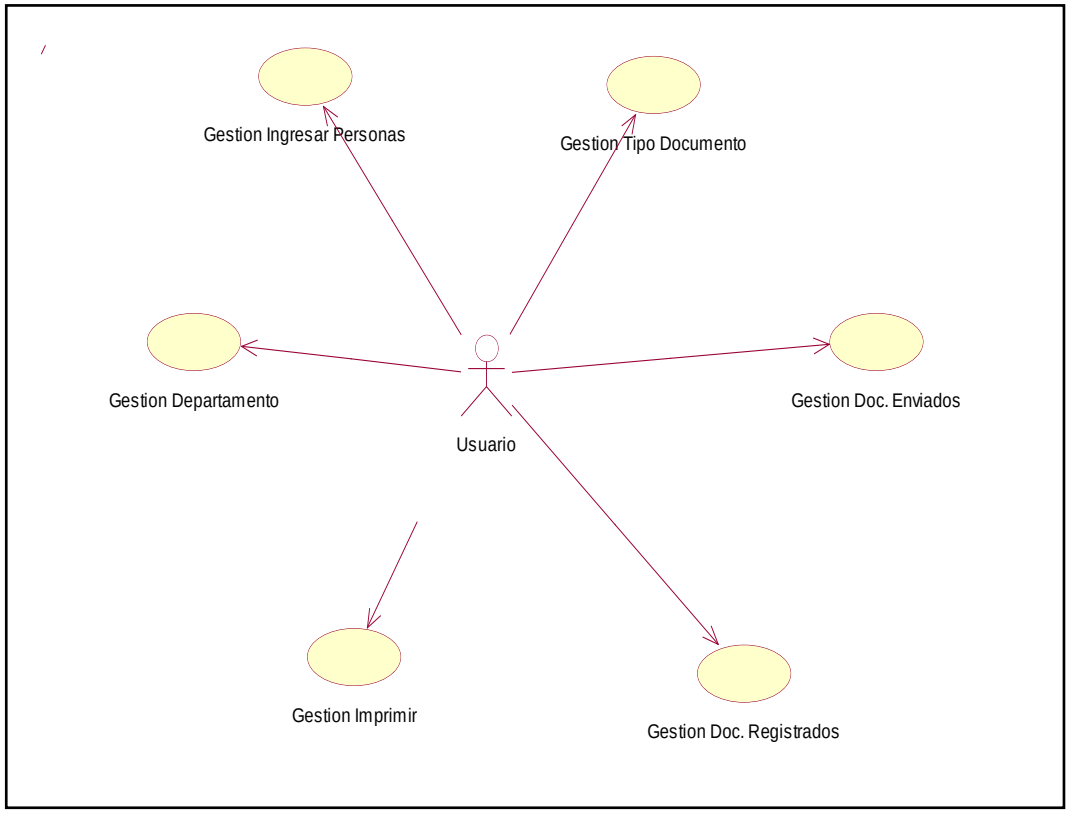


Figura 3.10: Caso de Uso Actor Usuario

3.1.12.5.7 INGRESAR MODULO USUARIO

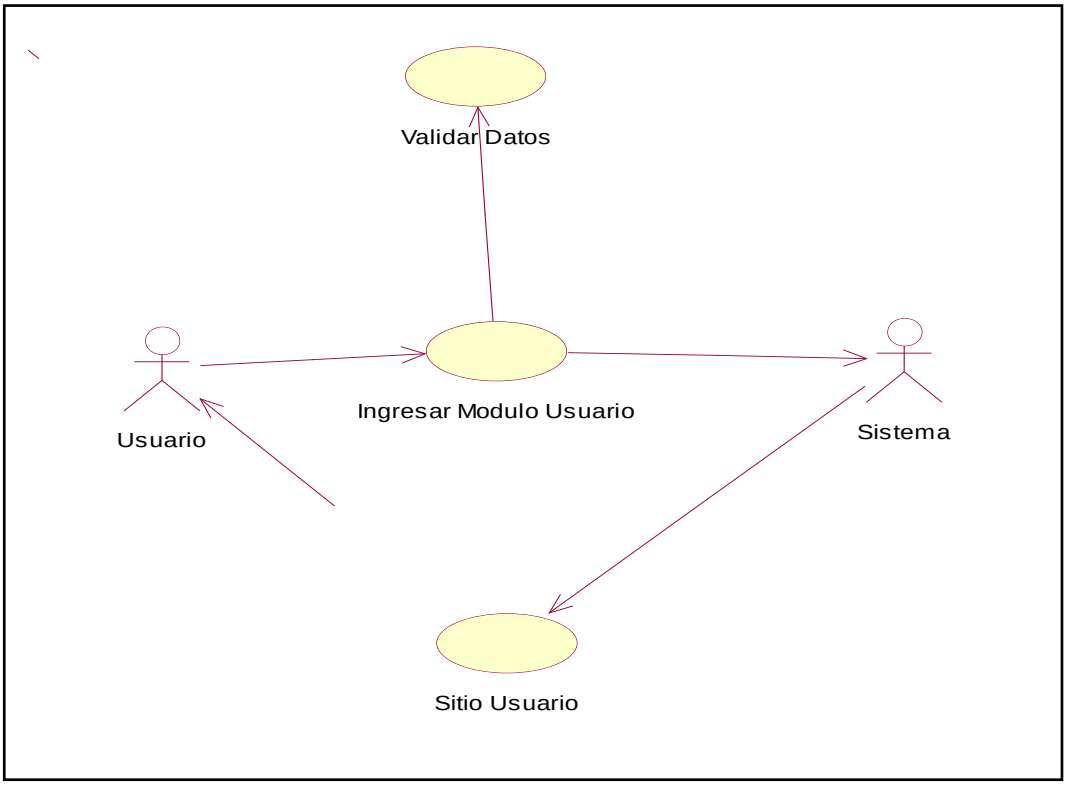


Figura 3.11 Caso de Uso Ingresar Modulo Usuario

Tabla 3.15. Caso de uso Ingresar Modulo Usuario

Caso de Uso	Ingresar Modulo Usuario
Actores	Usuario (iniciador)
Propósito	Permitir a un Usuario Ingresar al sitio de Usuarios
Tipo	Primario Real
Visión General	El Usuario solicita la operación Ingresa Modulo Usuario, el sistema presenta ventana de ingreso de modulo del Usuario, el usuario ingresa su login y password el sistema valida los datos si son correctos el usuario podrá

	acceder al Sitio de Usuarios, el sistema confirma operación.
Referencias	

Curso Típicos de Eventos

1. El Usuario solicita la operación Ingresar Modulo Usuario.
2. El sistema presenta ventana de ingreso a Modulo Usuario
3. El Usuario ingresa su login y password.
4. El sistema valida los datos ingresados por el usuario y confirma operación.

Curso Alternativos

Curso Típico 2: No existe ventana de ingreso a Modulo de usuario, termina caso de uso.

Curso Típico 4: Los datos no están bien ingresados termina el caso de uso.

3.1.12.5.8 SELECCIONAR DEPARTAMENTO

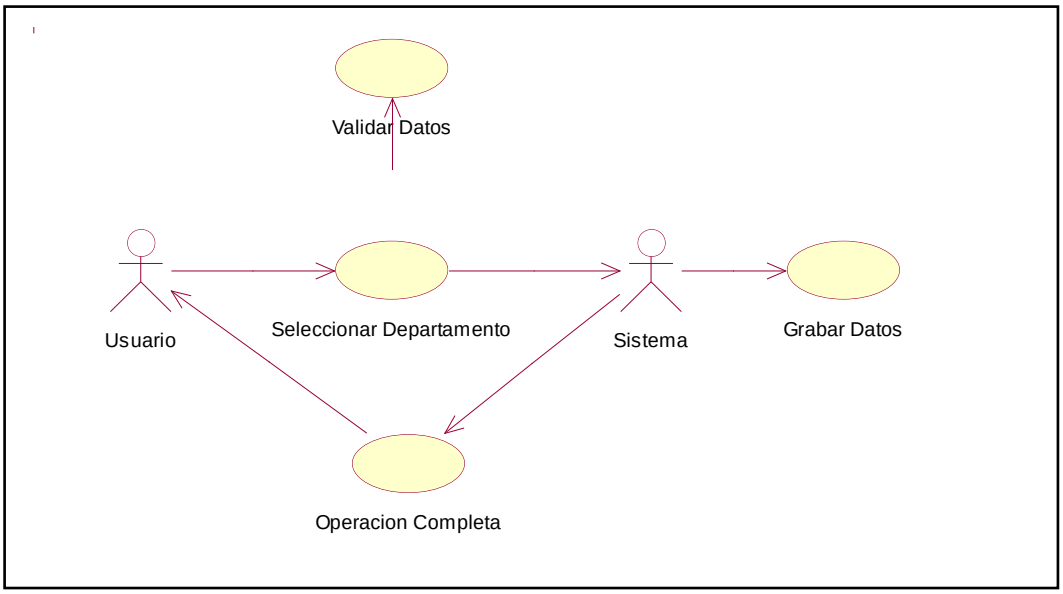


Figura 3.12: Caso de Uso Crear Departamento.

Tabla 3.16. Caso de uso Seleccionar Departamento

Caso de Uso	Seleccionar Departamento
Actores	Persona (iniciador)
Propósito	Permitir al Usuario Seleccionar El tipo de Departamento al que pertenece.
Tipo	Primario Esencial
Visión General	El Usuario selecciona la operación de seleccionar Departamento. El sistema presenta el formulario de edición de Seleccionar el Departamento, el Usuario podrá seleccionar el tipo de departamento al que pertenece, el sistema almacena los datos seleccionados Del departamento y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Seleccionar Departamento.
2. El sistema presenta el formulario de edición de departamento.
3. El Usuario tendrá la opción de seleccionar al departamento que pertenece para realizar los documentos y registrar.
4. El sistema almacena el Departamento seleccionado y confirma operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, termina el caso de uso.

3.1.12.5.9 CREAR PERSONA

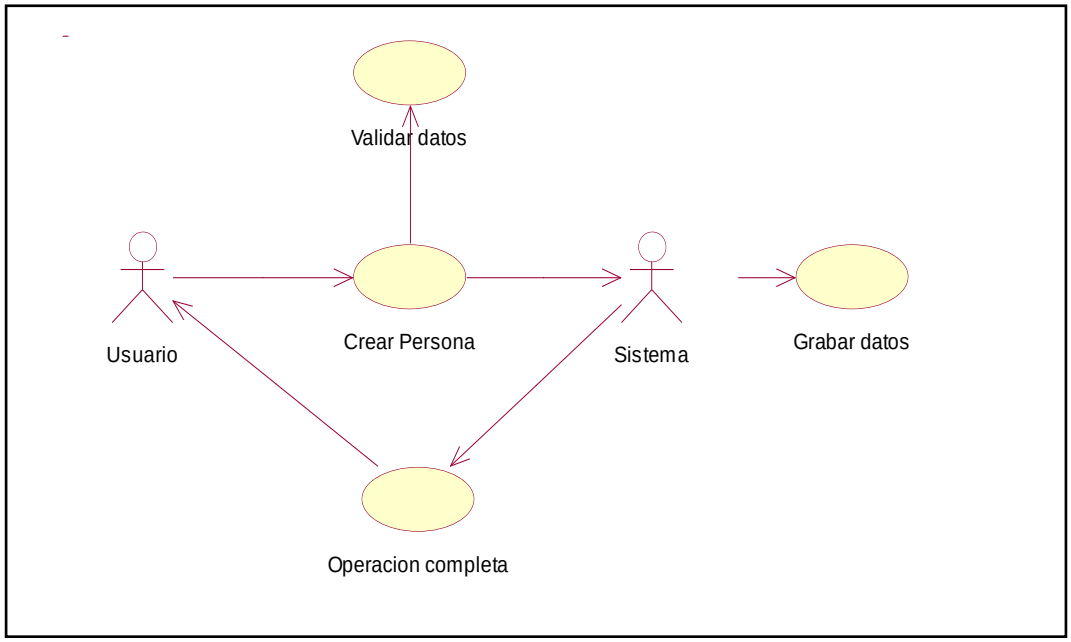


Figura 3.13: Caso de Uso Crear Persona.

Tabla 3.17. Caso de uso Crear Personal del Departamento

Caso de Uso	Crear Personal del Departamento
Actores	Personal del Departamento (iniciador)
Propósito	Crear una nueva persona para el sistema
Tipo	Primario Real
Visión General	El Usuario solicita la operación Crear Persona del Departamento, el sistema presenta el formulario de registro de nueva persona, el usuario ingresa los campos (Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo) al sistema, el sistema almacena los datos y confirma operación
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación crear persona
2. El sistema presenta formulario de registro de una nueva persona.
3. El Usuario ingresa los campos (Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo) al sistema
4. El sistema almacena los datos de grado y confirma operación

Curso Alternativos

Curso Típico 2: No existe Formulario de registro de una nueva persona, termina caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso.

3.1.12.5.10 MODIFICAR PERSONA

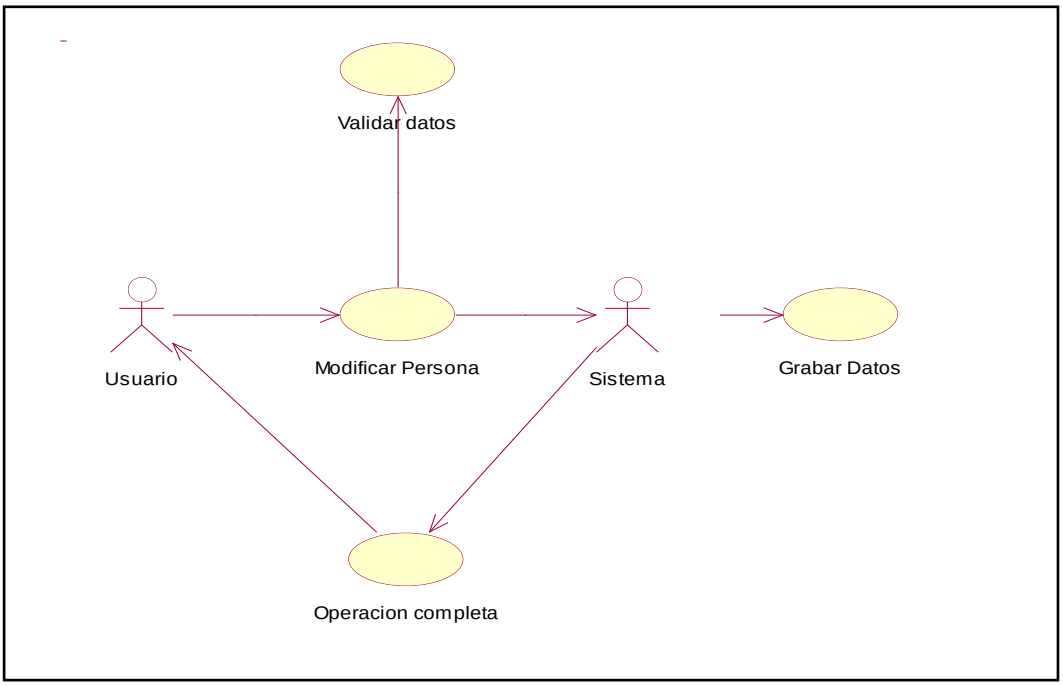


Figura 3.14: Caso de Uso Modificar Persona.

Tabla 3.18. Caso de uso Modificar Persona

Caso de Uso	Modificar Tipo de Persona
Actores	Persona (iniciador)
Propósito	Permitir al Usuario Modificar los datos de Tipo de Persona.
Tipo	Primario Esencial
Visión General	El Usuario selecciona la operación de modificar Persona. El sistema presenta el formulario de edición de Persona, el Usuario podrá modificar los campos (Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo) del Tipo de Persona, el sistema almacena los datos modificados de la Persona y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Modificar Tipo de Persona.
2. El sistema presenta el formulario de edición de Tipo de Persona.
3. El Usuario tendrá la opción de modificar los campos (Cédula id Departamento, Apellidos, Nombres, Teléfono, Cargo y Nominativo).
4. El sistema almacena los datos modificados y confirma operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, termina el caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso

3.1.12.5.11 SELECCIONAR TIPO DOCUMENTO

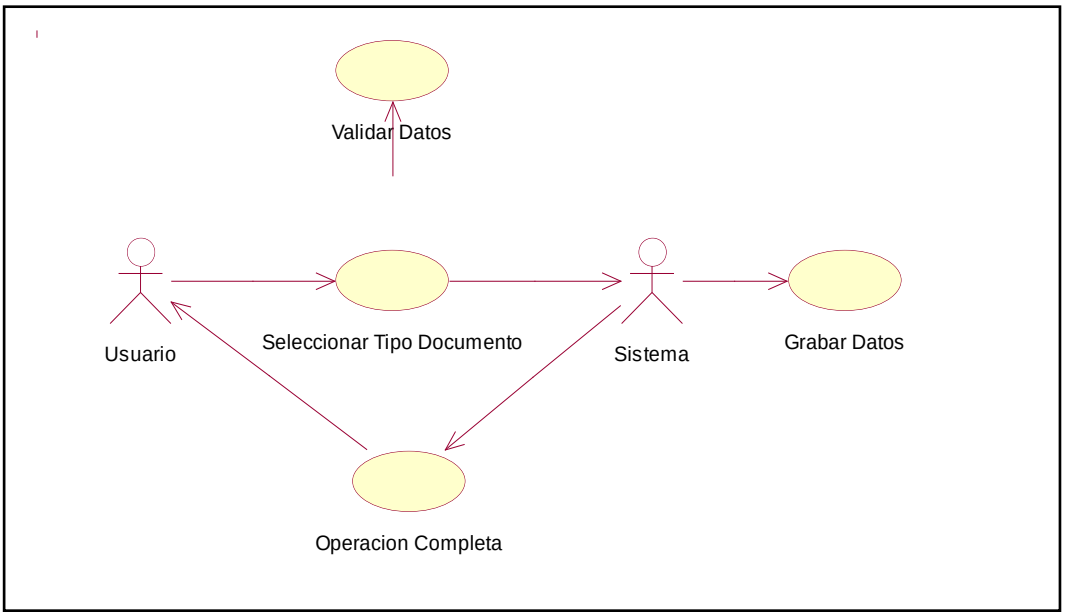


Figura 3.15: Caso de Uso Tipo Documento.

Tabla 3.19: Caso de uso Tipo Documento

Caso de Uso	Tipo de Documento
Actores	Persona (iniciador)
Propósito	Permitir al Usuario a Seleccionar El tipo de documento que va a realizar.
Tipo	Primario Esencial
Visión General	El Usuario selecciona la operación de seleccionar Tipo Documento. El sistema presenta el formulario de edición de Tipo Documento, el Usuario podrá seleccionar el tipo de documento que va a realizar, el sistema almacena los datos seleccionados del Tipo de Documento y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Seleccionar Tipo de Documento.
2. El sistema presenta el formulario de edición de Tipo de Documento.
3. El Usuario tendrá la opción de seleccionar que tipo de documento va a realizar.
4. El sistema almacena el Tipo de Documento seleccionado y confirma operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, termina el caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso

3.1.12.5.12 CREAR DOCUMENTO

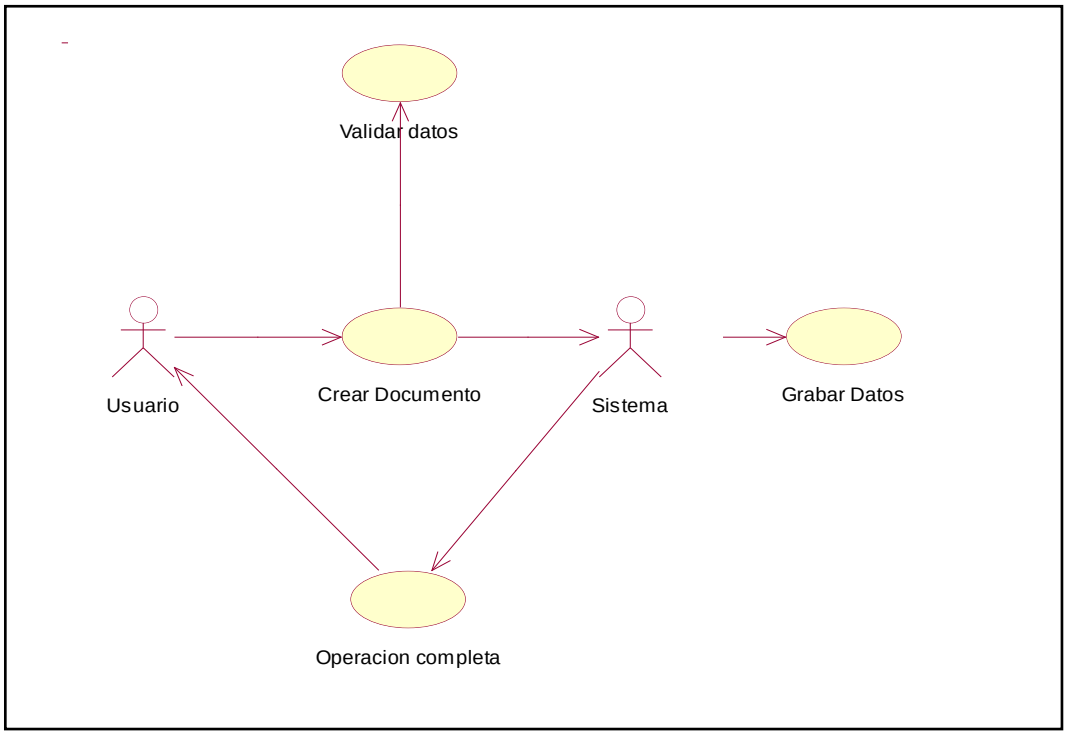


Figura 3.16. Caso de Uso Crear Documento.

Tabla 3.20: Caso de uso Crear Documento

Caso de Uso	Crear Documento
Actores	Documento (iniciador)
Propósito	Crear un nuevo documento para el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de crear documento, el sistema presenta el formulario de crear nuevo documento, el usuario elige que tipo de documento va a crear, y luego llena los campos siguientes: Fecha actual, No del documento, De, Para, Asunto, Tipo de Documento, Ano, Cargo De, Cargo Para, Contenido, Adjunto, elaborado por y revisado, firma.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Crear Documento.
2. El sistema presenta el formulario de crear un nuevo documento.
3. El Usuario ingresa los siguientes capos: Fecha actual, No del documento, De, Para, Asunto, Tipo de Documento, Ano, Cargo De, Cargo Para, Contenido, Adjunto, elaborado por y revisado, firma. al sistema
4. El sistema almacena los datos del documento creado y confirma la operación

Curso Alternativos

Curso Típico 2: No existe Formulario de crear un nuevo documento, termina caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso.

3.1.12.5.13 MODIFICAR DOCUMENTO

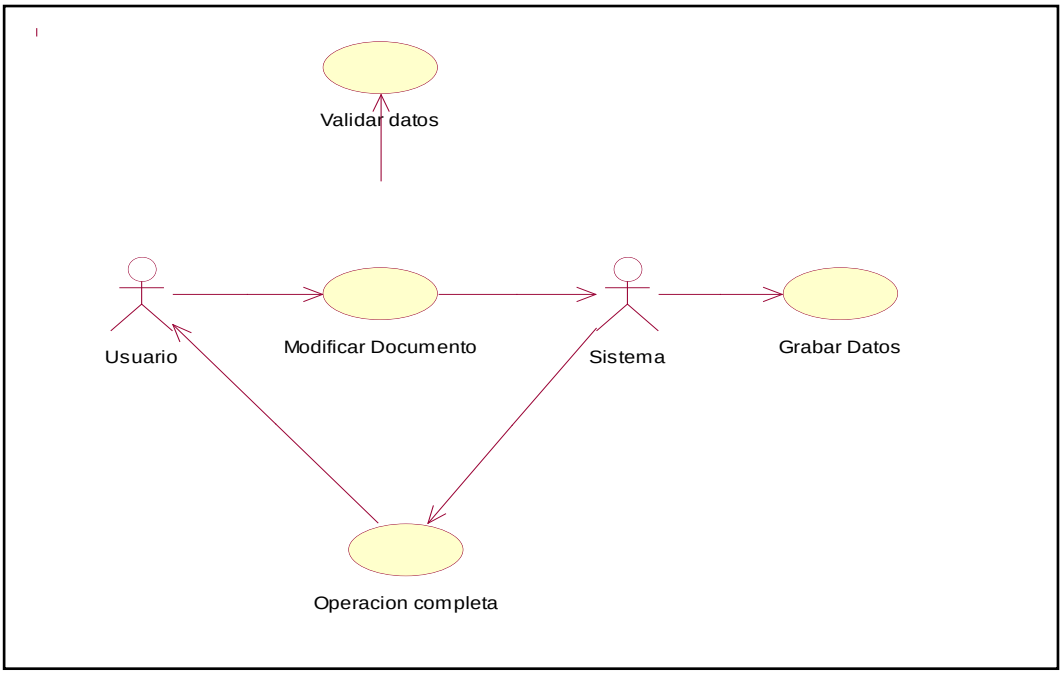


Figura 3.17: Caso de Uso Modificar Documento.

Tabla 3.21. Caso de uso Modificar Documento

Caso de Uso	Modificar Documento
Actores	Documento (iniciador)
Propósito	Permitir al Usuario Modificar los Documentos Enviados.
Tipo	Primario Esencial
Visión General	El Usuario selecciona la operación de Modificar el Documento. El sistema presenta el formulario de edición de Documento, el Usuario podrá modificar los campos (DE: PARA:, Texto del documento, Quien revisa y quien realiza) del Documento, el sistema almacena los datos modificados del

	Documento y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Modificar Documento.
2. El sistema presenta el formulario de edición de Documento.
3. El Usuario tendrá la opción de modificar los campos (DE:, PARA:, Texto del documento, Quien revisa y quien realiza).
4. El sistema almacena los datos modificados y confirma operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, termina el caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso

3.1.12.5.14 CASO DE USO CONSULTA INDIVIDUAL DE LOS DOCUMENTOS ENVIADOS

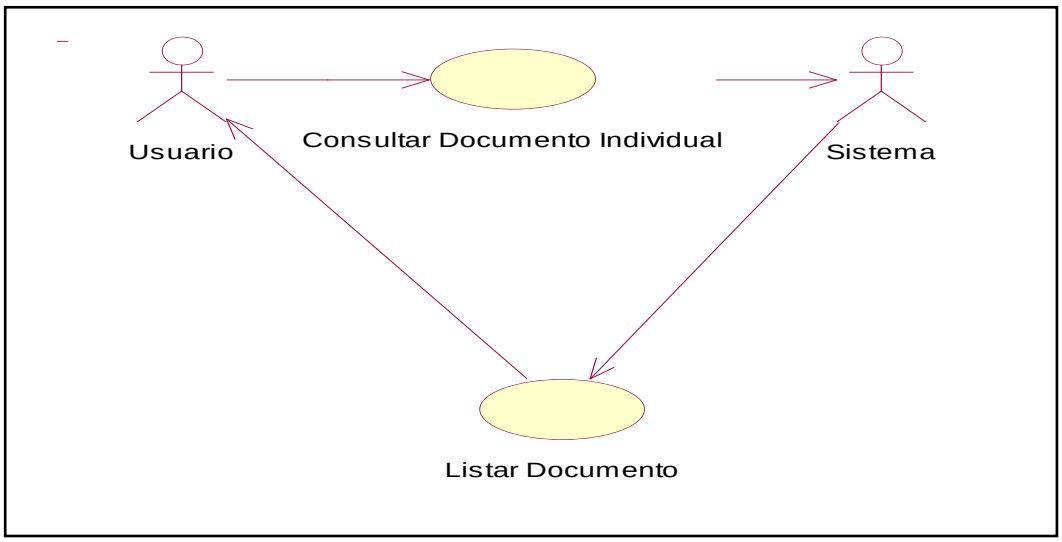


Figura 3.18: Caso de Uso Consulta Individual de los Documentos Enviados

Tabla 3.22. Caso de uso Consulta Individual de los Documentos Enviados

Caso de Uso	Consulta Individual de los Documentos Enviados
Actores	Documento Enviados(iniciador)
Propósito	Observa cuales son los Documentos Enviados en el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de consulta Individual de los documentos Enviados. El sistema presenta la lista de documentos registrados en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Consulta Individual de los Documentos Enviados.
2. El sistema presenta la lista y selecciona El Documento Enviado.
3. El sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe Documentos Enviados, y termina el caso de uso.

3.1.12.5.15 CASO DE USO CONSULTA GENERAL DE LOS DOCUMENTOS ENVIADOS

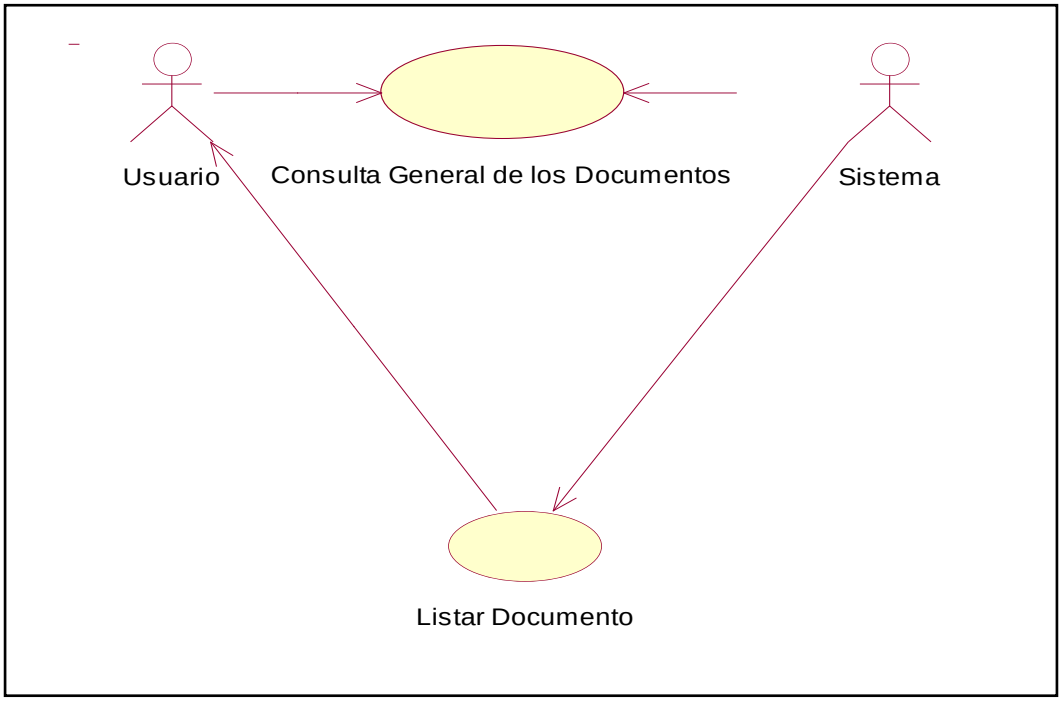


Figura 3.19: Caso de Uso Consulta General de los Documentos Enviados.

Tabla 3.23. Caso de uso Consulta General de Los Documentos Enviados

Caso de Uso	Consulta General De los Documentos Enviados
Actores	Usuario (iniciador)
Propósito	Observa de forma general cuales son los Documentos Enviados en el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de consultar datos generales de Documentos Enviados. El sistema presenta la lista general de los

	Documentos Enviados en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Consulta General de los Documentos Enviados.
2. El sistema presenta la lista general de los Documentos Enviados en el sistema.
3. El sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe Documentos Enviados, y termina el caso de uso.

3.1.12.5.16 CASO DE USO IMPRIMIR DOCUMENTO ENVIADO

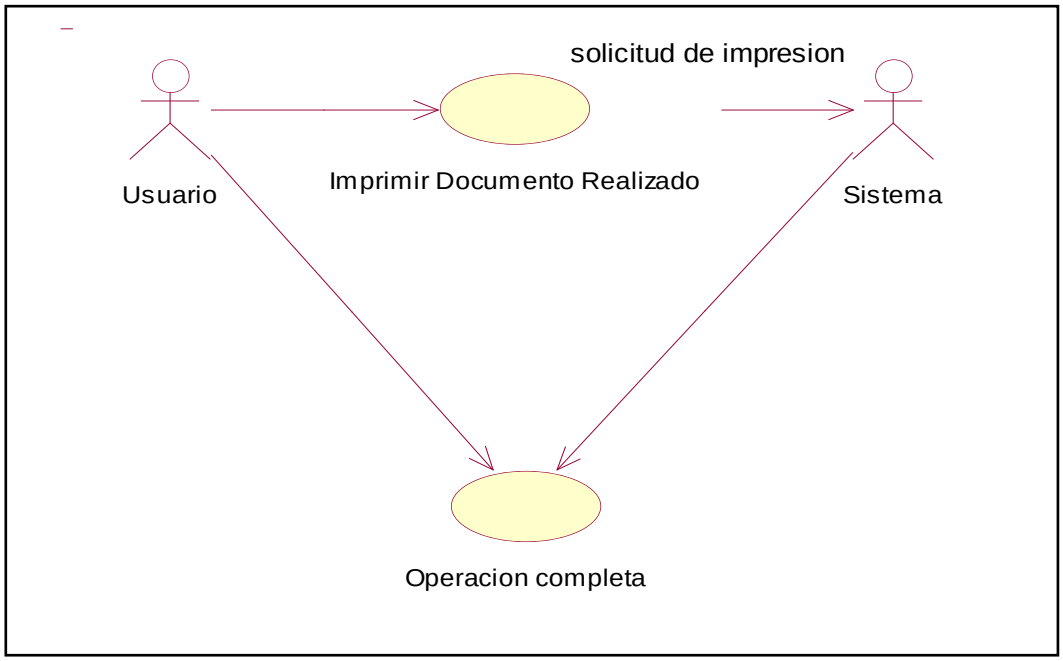


Figura 3.20: Caso de Uso imprimir Documento Enviado.

Tabla 3.24. Caso de uso Imprimir Documento Enviado

Caso de Uso	Imprimir Documentos Enviados
Actores	Usuario (iniciador)
Propósito	Que el sistema permita imprimir los documentos Enviados con la información respectiva de cada uno de los Documentos Enviados
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de imprimir documentos enviados. El sistema presenta el documento donde el Usuario puede visualizar el contenido del documento enviado y que pueda imprimir, El Usuario selecciona en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Imprimir Los documentos enviados.
2. El sistema presenta el formulario de Imprimir documentos enviados.
3. El Usuario selecciona los documentos enviados que desea imprimir, el sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, y termina el caso de uso.

Cuando no existe papel en la impresora

3.1.12.5.17 REGISTRAR DOCUMENTO

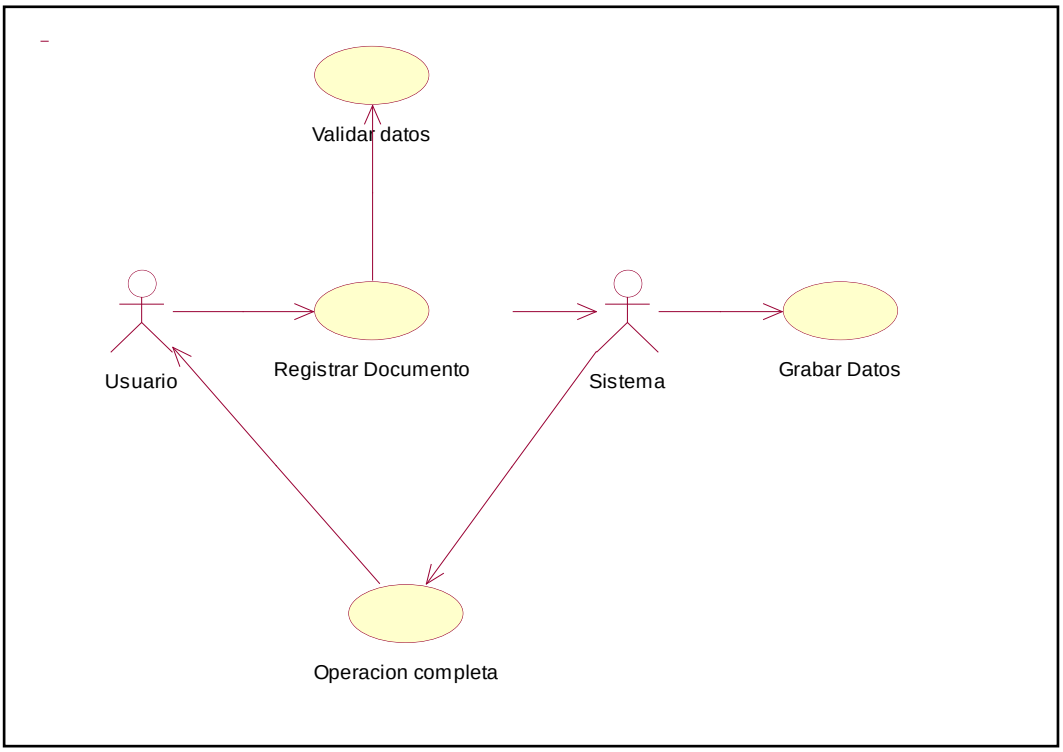


Figura 3:21: Caso de Uso Registrar Documento.

Tabla 3.25. Caso de uso Registrar Documento

Caso de Uso	Registrar Documento
Actores	Documento (iniciador)
Propósito	Registrar un nuevo documento para el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de registrar nuevo documento, el sistema presenta el formulario de registrar nuevo documento, el usuario ingresa los campos (Fecha actual, No del Tramite, No de Documento, Dependencia, Tipo de Documento, hora, Ano, Fecha del Documento, Recepción, asunto, Para: y De:) al sistema, el sistema almacena los

	datos del documentó y confirma operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Registrar Documento.
2. El sistema presenta el formulario de registrar un nuevo documento.
3. El Usuario ingresa los campos (Fecha actual, No del Tramite, No de Documento, Dependencia, Tipo de Documento, hora, Ano, Fecha del Documento, Recepción, asunto, Para: y De:) al sistema
4. El sistema almacena los datos del documento registrado y confirma la operación

Curso Alternativos

Curso Típico 2: No existe Formulario de registro de un nuevo documento, termina caso de uso.

Curso Típico 4: Los datos no están bien ingresados, termina el caso de uso.

3.1.12.5.18 CASO DE USO CONSULTA INDIVIDUAL DE LOS DOCUMENTO REGISTRADOS

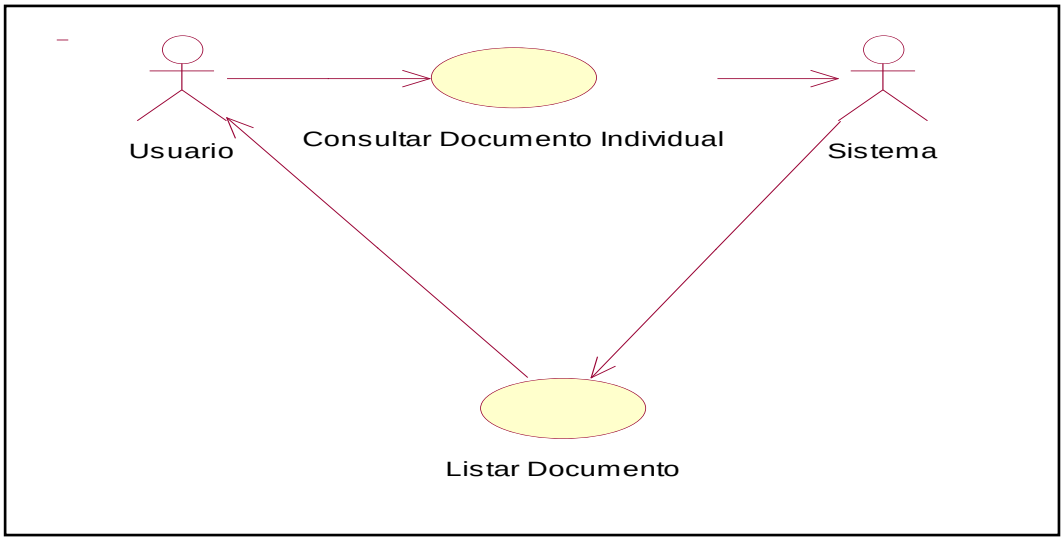


Figura 3.22: Caso de Uso Consulta Individual de los Documentos Registrados

Tabla 3.26. Caso de uso Consulta Individual de los Documentos Registrados

Caso de Uso	Consulta Individual de los Documentos Registrados
Actores	Documento (iniciador)
Propósito	Observa cuales son los Documentos Registrados en el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de consulta Individual de los documentos Registrados. El sistema presenta la lista de documentos registrados en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Consulta Individual de los Documentos registrados.
2. El sistema presenta la lista y selecciona El Documento Registrado.
3. El sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe Documentos Registrado, y termina el caso de uso.

3.1.12.5.19 CASO DE USO CONSULTA GENERAL DE LOS DOCUMENTOS REGISTRADOS

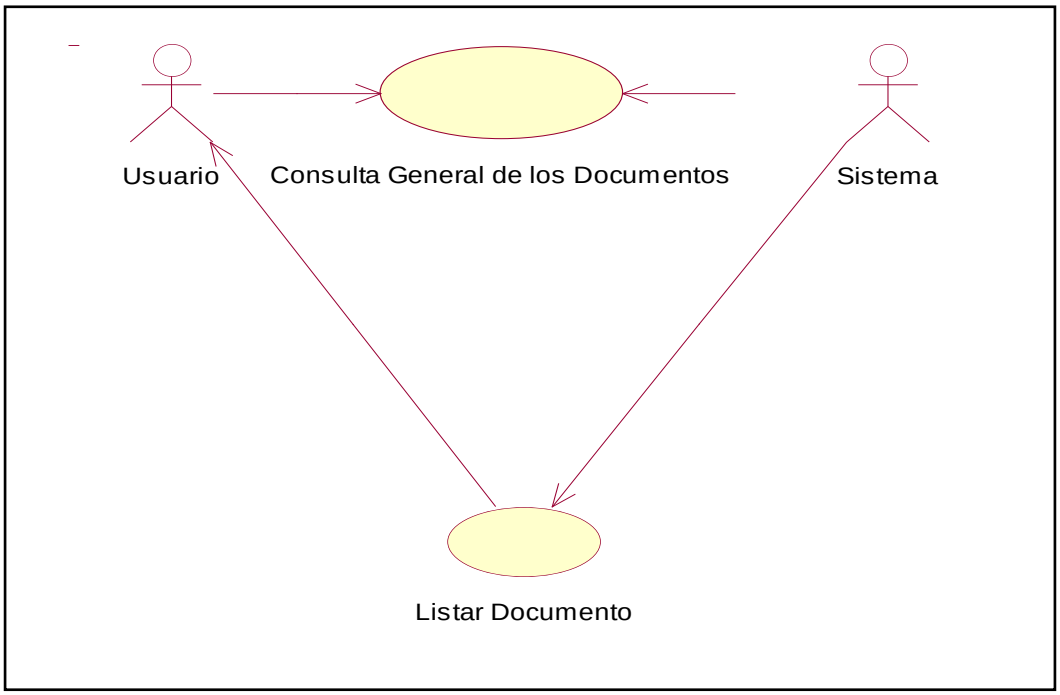


Figura 3.23: Caso de Uso Consulta General de los Documentos Registrados.

Tabla 3.27. Caso de uso Consulta General de Los Documentos Registrados.

Caso de Uso	Consulta General De los Documentos Registrados
Actores	Usuario (iniciador)
Propósito	Observa de forma general cuales son los Documentos Registrados en el sistema.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de consultar datos generales de Documentos Registrados. El sistema presenta la lista

	general de los Documentos Registrados en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Consulta General de los Documentos Registrados.
2. El sistema presenta la lista general de los Documentos Registrados en el sistema.
3. El sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe Documentos Enviados, y termina el caso de uso.

3.1.12.5.20 CASO DE USO IMPRIMIR DOCUMENTO REGISTRADO

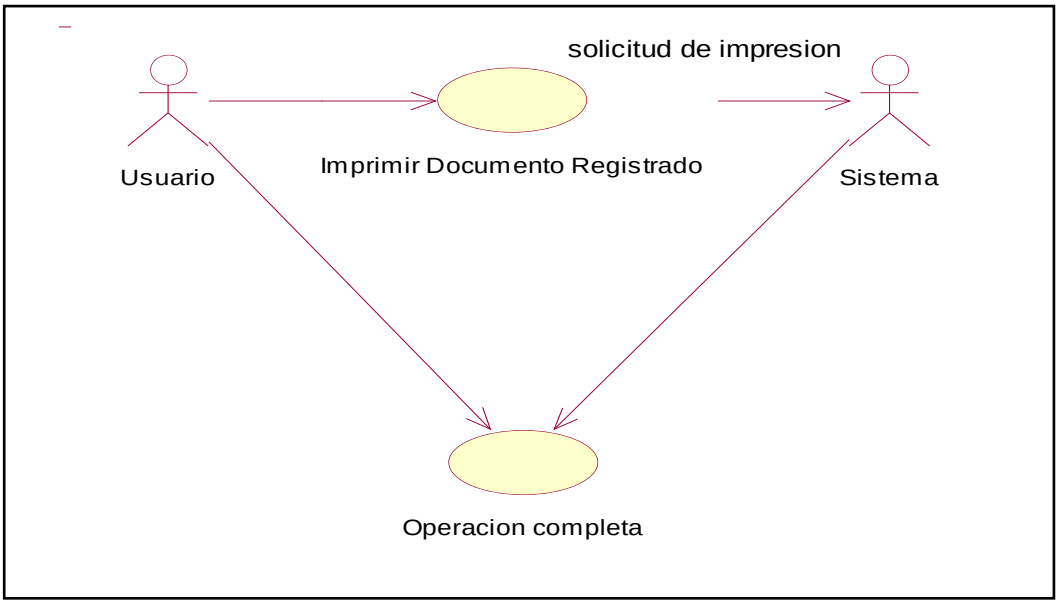


Figura 3.24: Caso de Uso imprimir Documento Registrado.

Tabla 3.28. Caso de uso Imprimir Documento Registrado

Caso de Uso	Imprimir Documentos Registrado
Actores	Usuario (iniciador)
Propósito	Que el sistema permita imprimir los documentos Registrados con la información respectiva de cada uno de los Documentos Registrados.
Tipo	Primario Esencial
Visión General	El Usuario solicita la operación de imprimir documentos registrados. El sistema presenta el documento donde el Usuario puede visualizar el contenido del documento registrado y que pueda imprimir, El Usuario selecciona en el sistema y confirma la operación.
Referencias	

Curso Típicos de Eventos

1. El caso de uso comienza cuando el Usuario solicita la operación Imprimir Los documentos registrados.
2. El sistema presenta el formulario de Imprimir documentos realizados.
3. El Usuario selecciona los documentos realizados que desea imprimir, el sistema confirma la operación.

Cursos Alternativos

Curso Típico 2: No existe formulario, y termina el caso de uso.

Cuando no existe papel en la impresora

3.2 FASE DE DISEÑO

3.2.2 DIAGRAMAS DE SECUENCIA

3.2.2.1 DIAGRAMA DE SECUENCIA ACTOR ADMINISTRADOR

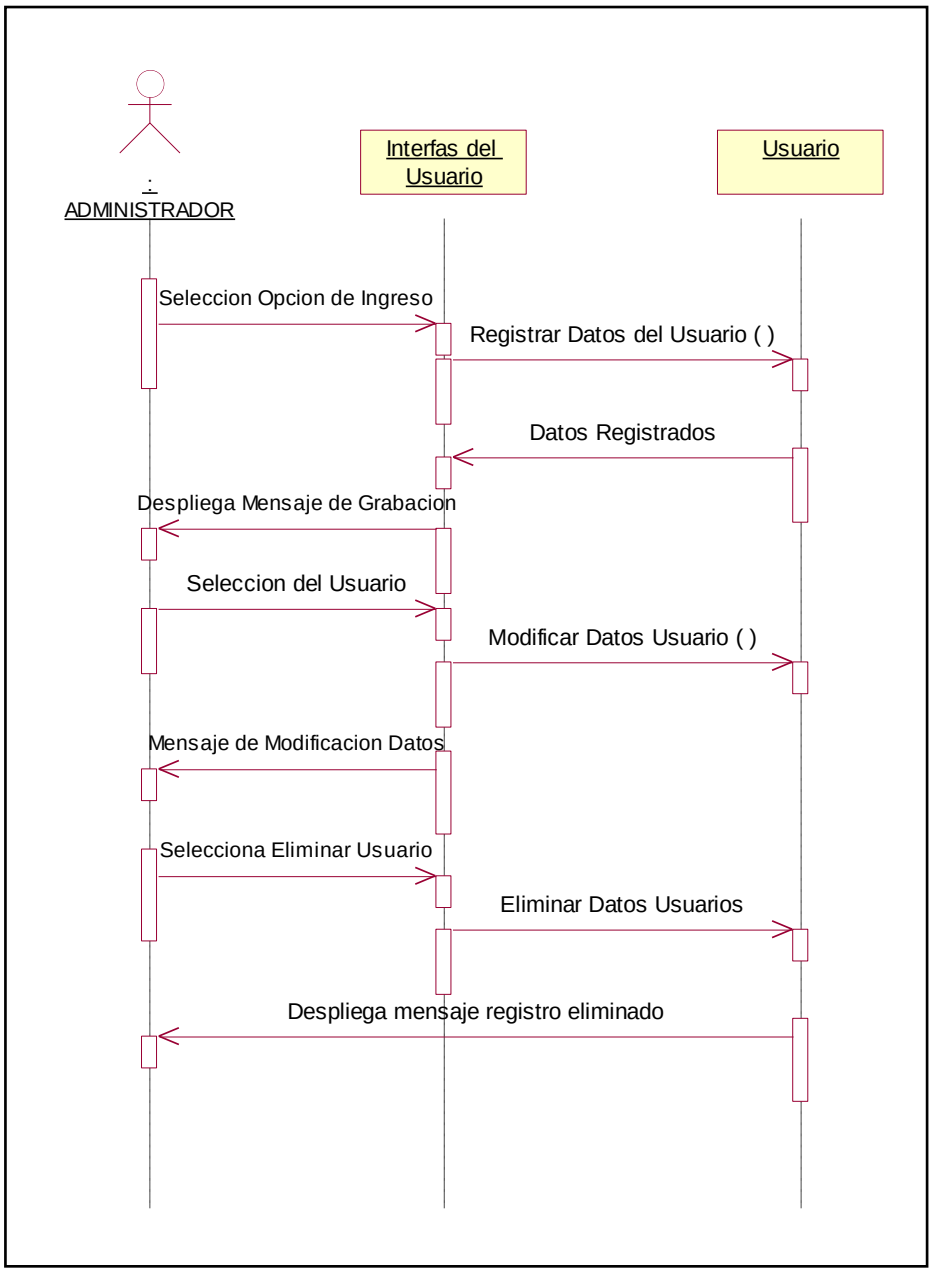


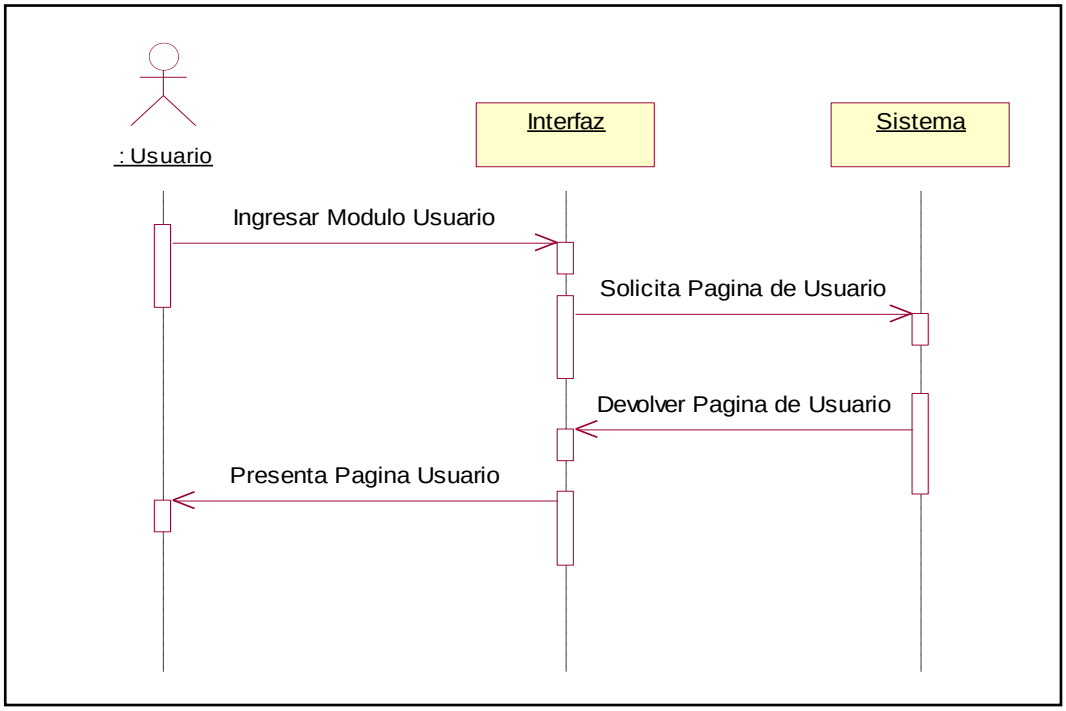
Figura 3.25 Diagrama De Secuencia Actor Administrador

Tabla 3.29. Contrato De Operación Administrar Usuarios

Nombre:	Administrar Usuarios
Responsabilidades:	Deberá validar los datos del usuario y tipo de datos al momento de ingresarlos, modificarlos y eliminarlos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Administrar Usuarios
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Ingresar, modificar, eliminar y consultar
Postcondiciones:	Validar Datos

3.2.3 DIAGRAMAS DE SECUENCIA

3.2.3.1 INGRESAR MODULO USUARIO

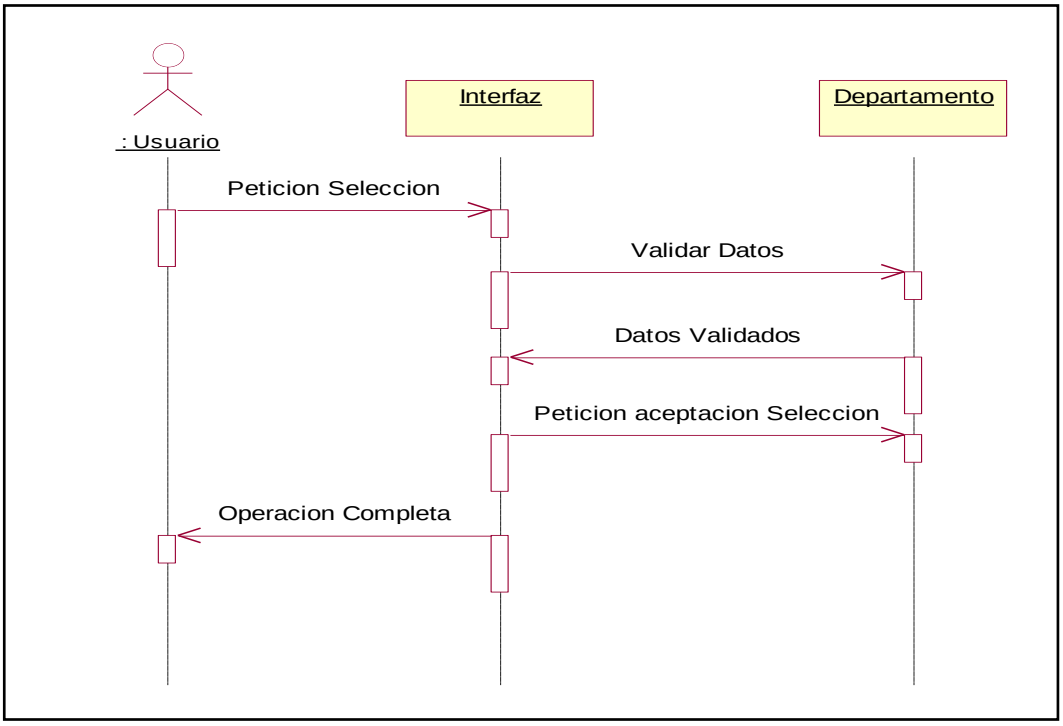


Figuras 3.26: Diagrama De Secuencia Ingreso Modulo Usuario

Tabla 3.30. Contrato De Operación Ingresar Modulo Usuario

Nombre:	Ingresar Modulo Usuario
Responsabilidades:	Deberá validar los Datos de Usuario y tipo de los datos al momento de ingresar.
Tipo:	SISCERDESPE-L
Caso de Uso:	Ingresar Modulo Usuario
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Ingresar Datos
Postcondiciones:	Datos Validados

3.2.3.2 SELECCIONAR DEPARTAMENTO

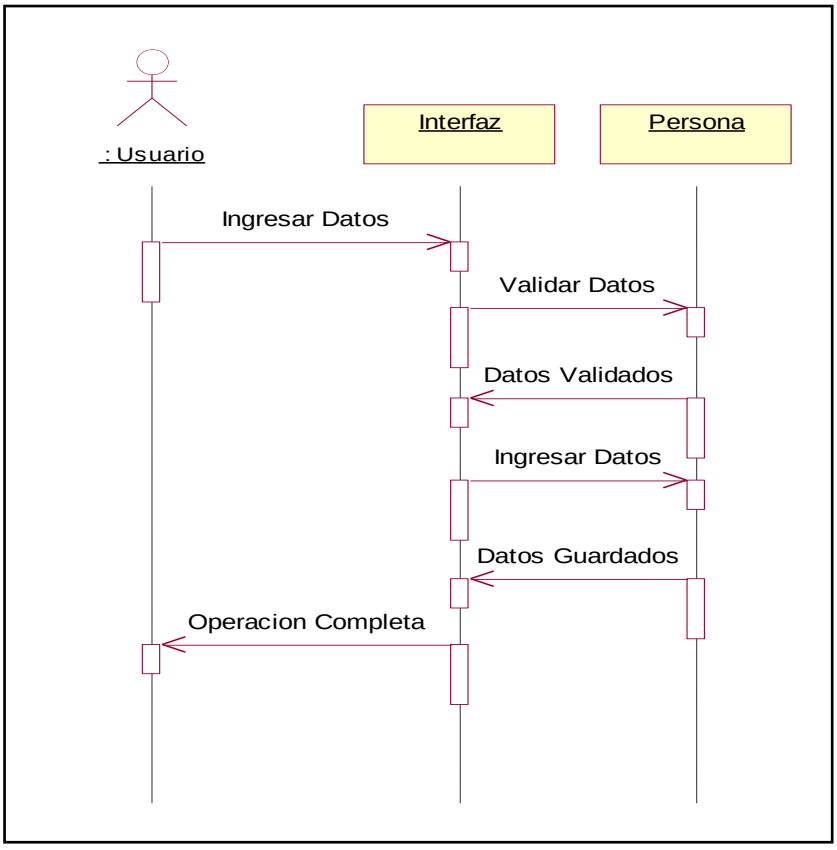


Figuras 3.27: Diagrama De Secuencia Seleccionar Departamento

Tabla 3.31. Contrato De Operación Seleccionar Departamento

Nombre:	Verificar Departamento
Responsabilidades:	Deberá Verificar si los datos del Tipo de documento a Seleccionar tienen o no otros datos que dependan de ellos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Seleccionar Departamento
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Petición Seleccionar
Postcondiciones:	Selección Verificada

3.2.3.3 CREAR PERSONA

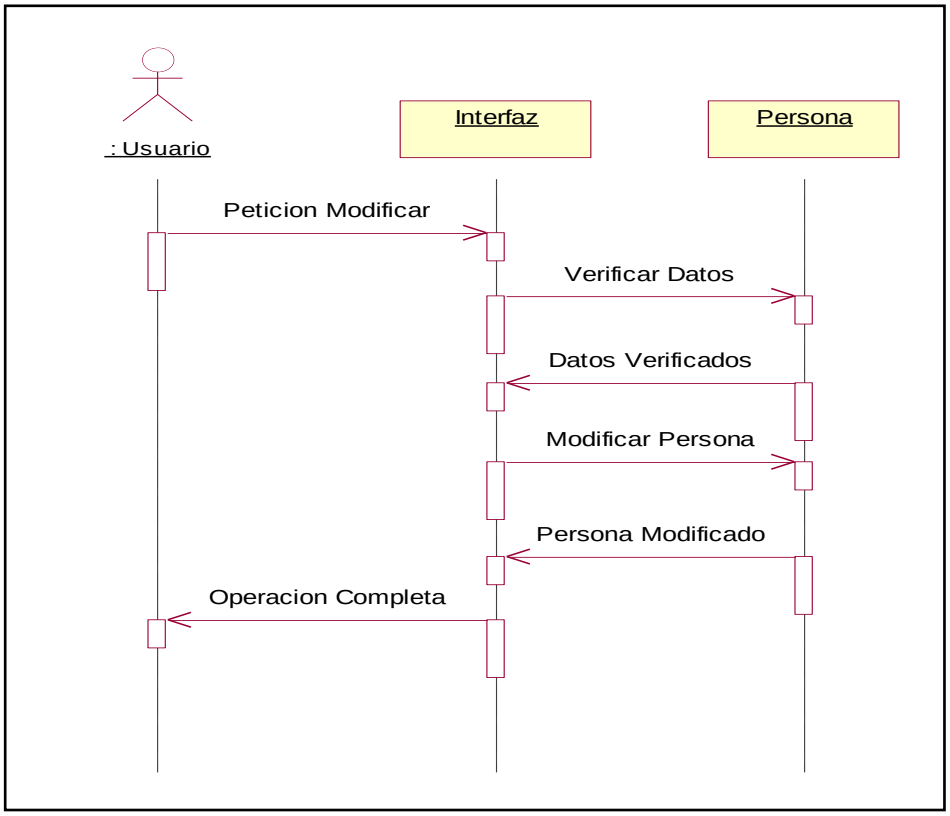


Figuras: 3.28: Diagrama De Secuencia Crear Persona

Tabla 3.32. Contrato De Operación Crear Persona

Nombre:	Validar datos
Responsabilidades:	Deberá validar El formato y tipo de los datos al momento de ingresarlos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Crear Persona
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Ingresar Datos
Postcondiciones:	Datos Validados

3.2.3.4 MODIFICAR PERSONA

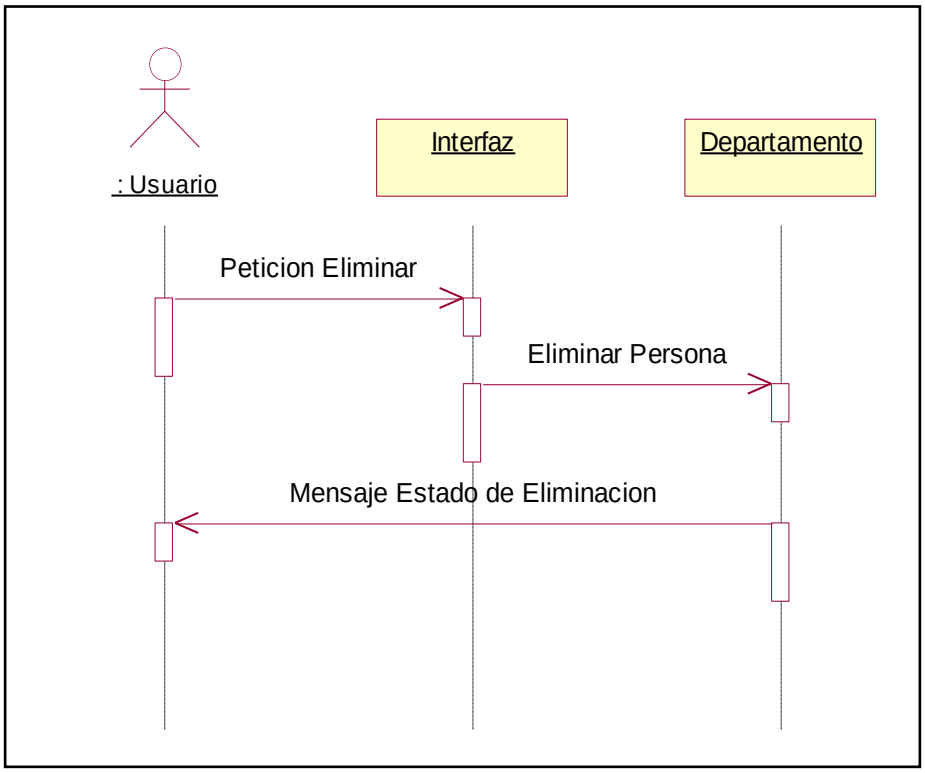


Figuras 3.29: Diagrama De Secuencia Modificar Persona

Tabla 3.33. Contrato De Operación Modificar Persona

Nombre:	Verificar Persona
Responsabilidades:	Deberá Verificar si los datos de la Persona a Modificar tienen o no otros datos que dependan de ellos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Modificar Persona
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Petición modificar
Postcondiciones:	Persona Verificado

3.2.3.5 ELIMINAR PERSONA

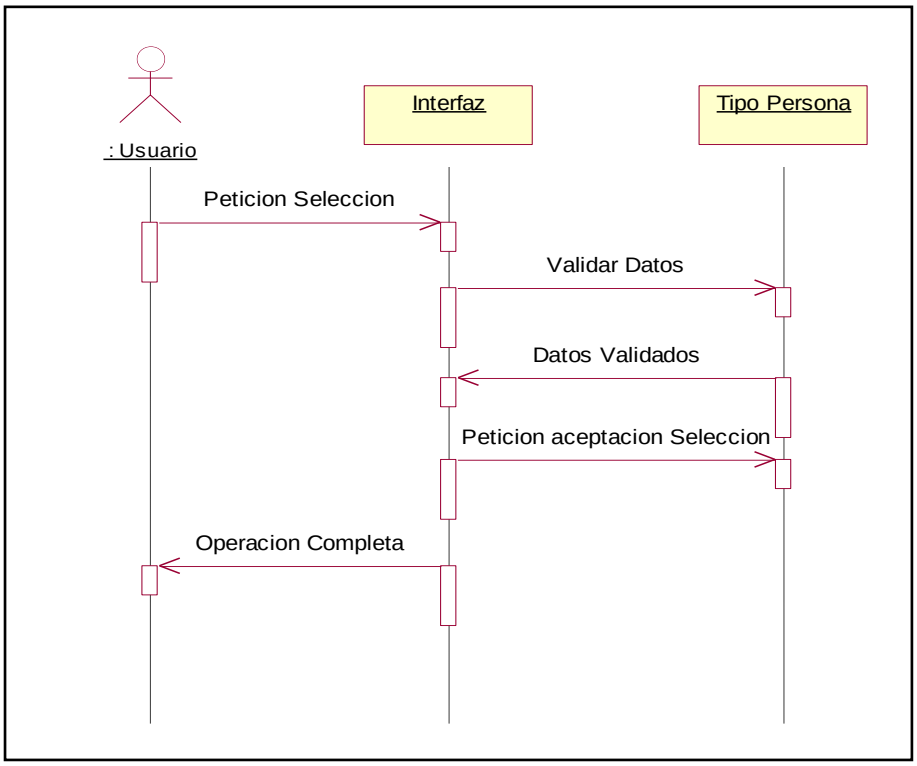


Figuras 3.30: Diagrama De Secuencia Eliminar Persona

Tabla 3.34. Contrato De Operación Eliminar Persona

Nombre:	Eliminar Persona
Responsabilidades:	Deberá Verificar si los datos de la persona a Eliminar tienen o no otros datos que dependan de ellos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Eliminar Persona
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Petición Eliminar
Postcondiciones:	Persona Verificado

3.2.3.6 SELECCIONAR TIPO DOCUMENTO

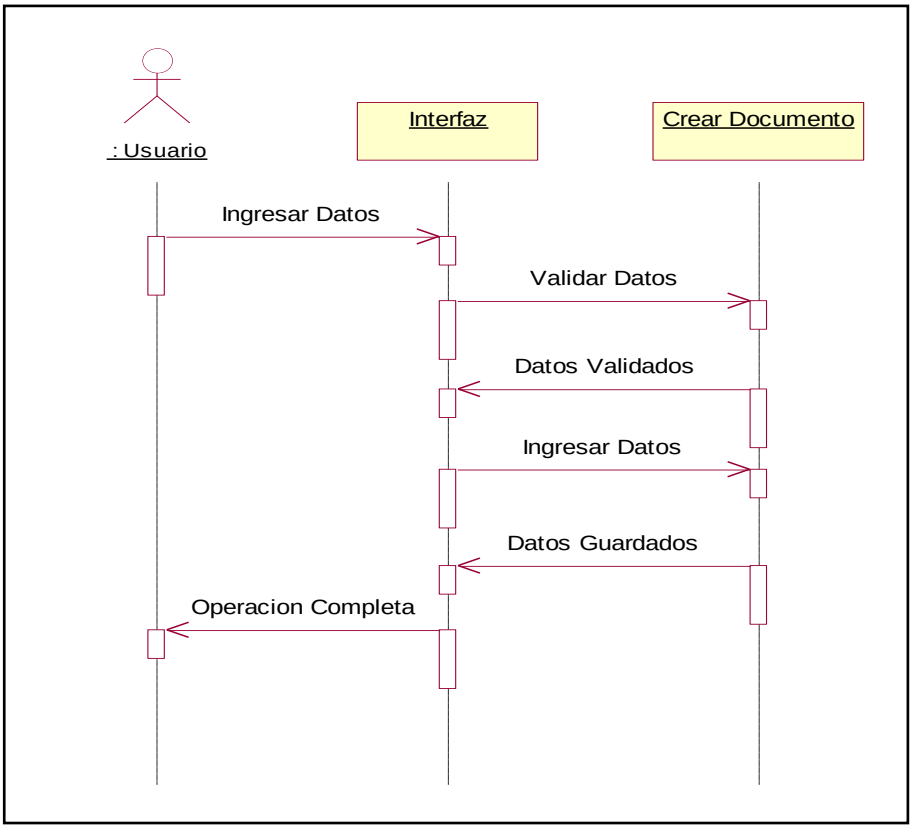


Figuras 3.31: Diagrama De Secuencia Seleccionar Tipo Documento

Tabla 3.35. Contrato De Operación Seleccionar Tipo De Documento

Nombre:	Verificar Tipo Documento
Responsabilidades:	Deberá Verificar si los datos del Tipo de documento a Seleccionar tienen o no otros datos que dependan de ellos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Seleccionar Tipo Documentó
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Petición Seleccionar
Postcondiciones:	Selección Verificada

3.2.3.7 CREAR DOCUMENTO

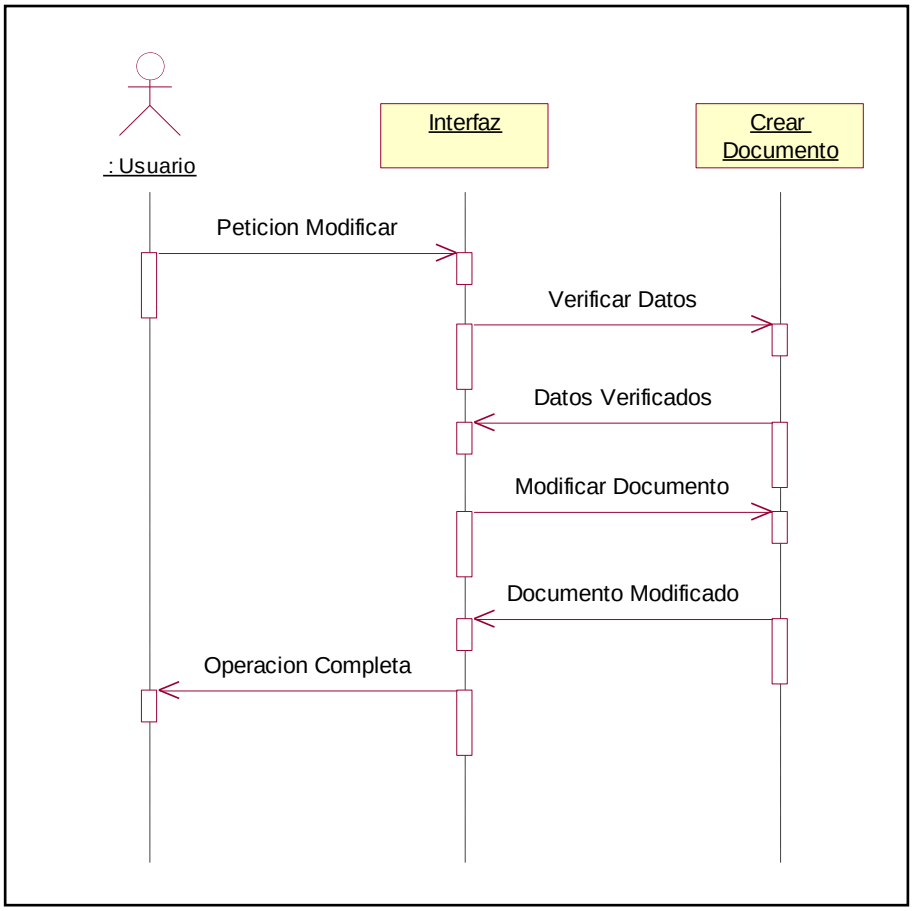


Figuras 3.32: Diagrama De Secuencia Crear Documento

Tabla 3.36. Contrato De Operación Crear Documento

Nombre:	Validar datos
Responsabilidades:	Deberá validar El formato y tipo de los datos al momento de ingresarlos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Crear Documento
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Ingresar Datos
Postcondiciones:	Datos Validados

3.2.3.8 MODIFICAR DOCUMENTO CREADO

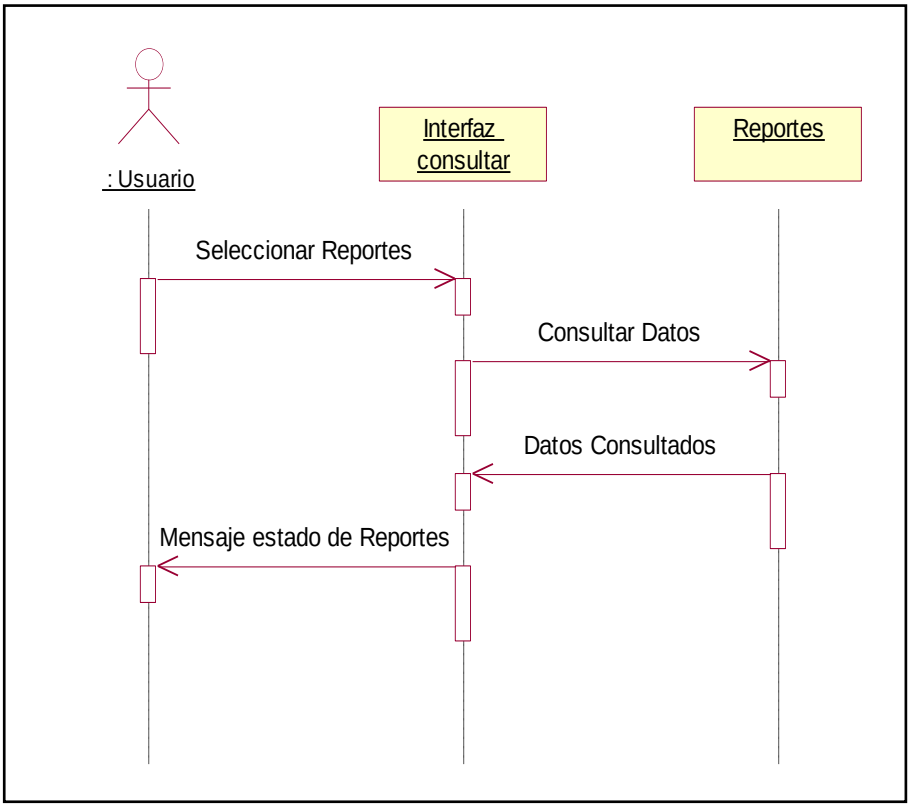


Figuras 3.33: Diagrama De Secuencia Modificar Documento Creado

Tabla 3.37. Contrato De Operación Modificar Documento Creado

Nombre:	Verificar Documento
Responsabilidades:	Deberá Verificar si los datos del Documento Creado a Modificar tienen o no otros datos que dependan de ellos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Modificar Documento Creado
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Petición modificar
Postcondiciones:	Documento Creado Verificado

3.2.3.9 CONSULTAR DOCUMENTO CREADO

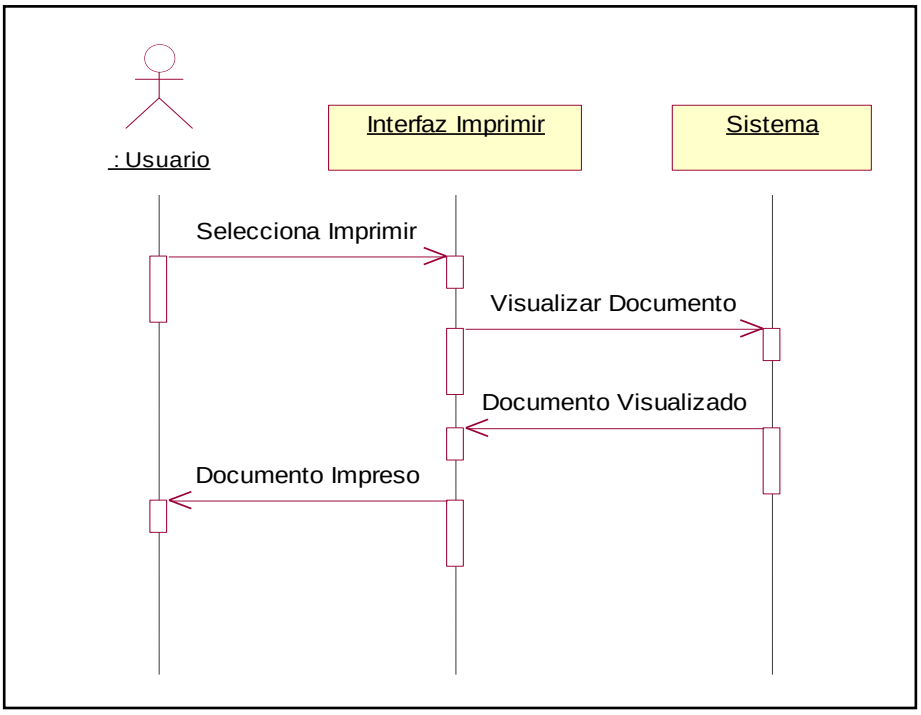


Figuras 3.34: Diagrama De Secuencia Consultar Documento Creado

Tabla 3.38. Contrato De Operación Consultar Datos

Nombre:	Consultar Datos
Responsabilidades:	Deberá consultar los datos de los Documentos dependiendo de los parámetros solicitado
Tipo:	SISCERDESPE-L
Caso de Uso:	Consultar Datos
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Consultar Datos
Postcondiciones:	

3.2.3.10 IMPRIMIR DOCUMENTO CREADO

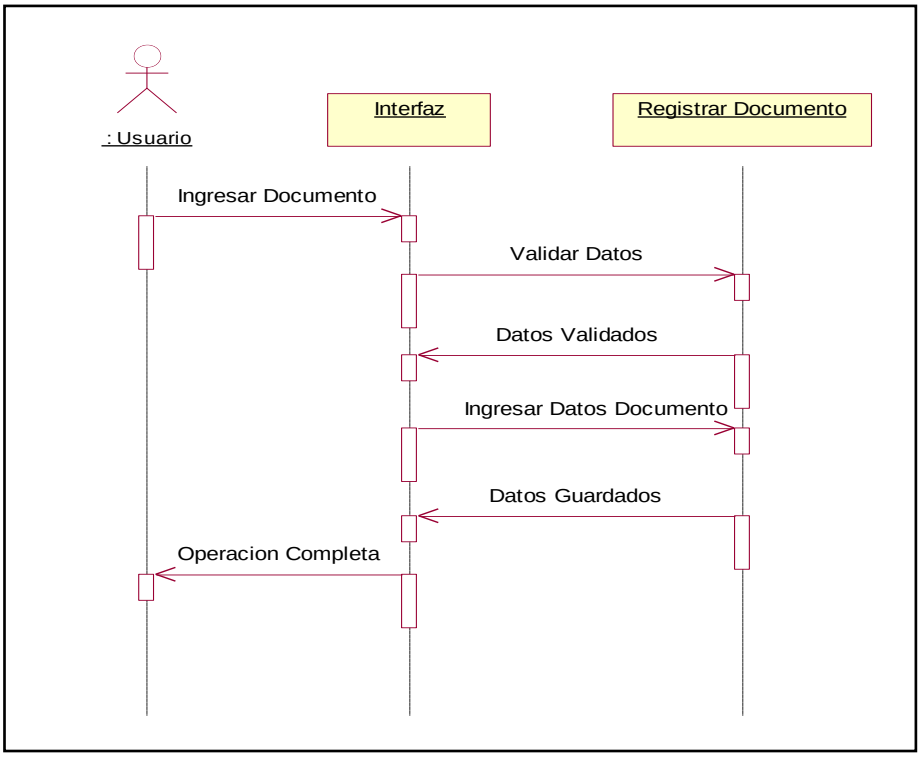


Figuras 3.35: Diagrama De Secuencia Imprimir Documento Creado

Tabla 3.39. Contrato de Operación Imprimir Documento Creado

Nombre:	Imprimir Documento Creado
Responsabilidades:	Debe imprimir el documento creado por el usuario
Tipo:	SISCERDESPE-L
Caso de Uso:	Imprimir Documento
Notas:	
Excepciones:	
Salidas:	Documento Creado
Precondiciones:	Almacenar datos para imprimir
Postcondiciones:	Impresión del Documento Creado

3.2.3.11 REGISTRAR DOCUMENTO

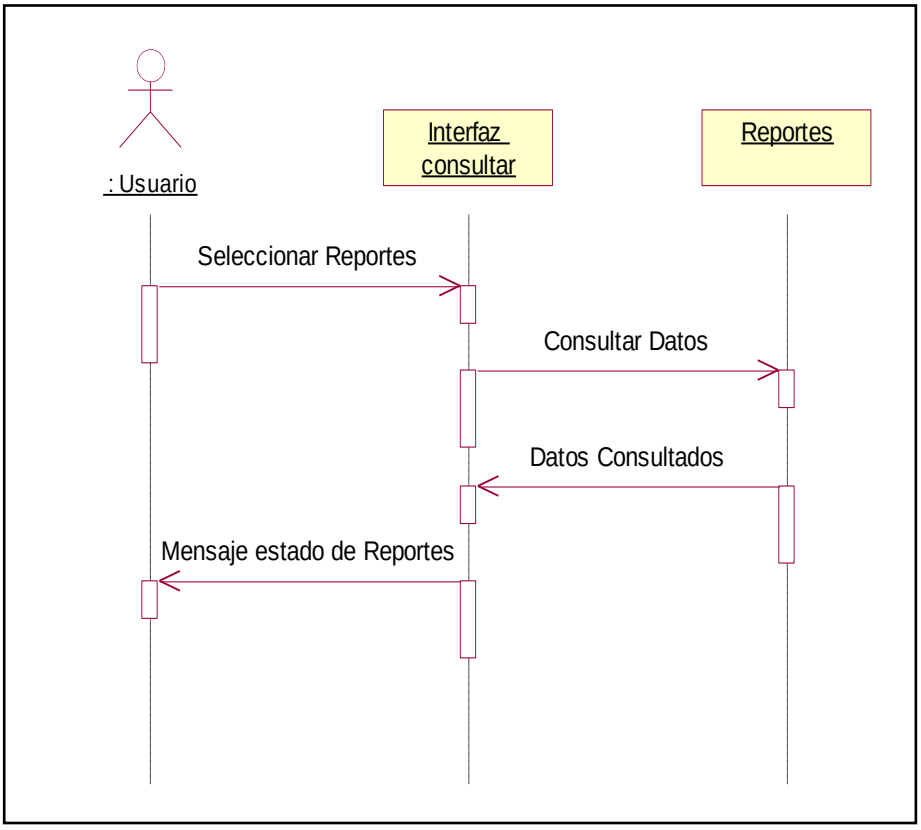


Figuras 3.36: Diagrama De Secuencia Registrar Documento

Tabla 3.40. Contrato De Operación Registrar Documento

Nombre:	Validar datos
Responsabilidades:	Deberá validar El formato y tipo de los datos al momento de ingresarlos.
Tipo:	SISCERDESPE-L
Caso de Uso:	Registrar Documento
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Ingresar Datos
Postcondiciones:	Datos Validados

3.2.3.12 CONSULTAR DOCUMENTOS REGISTRADOS

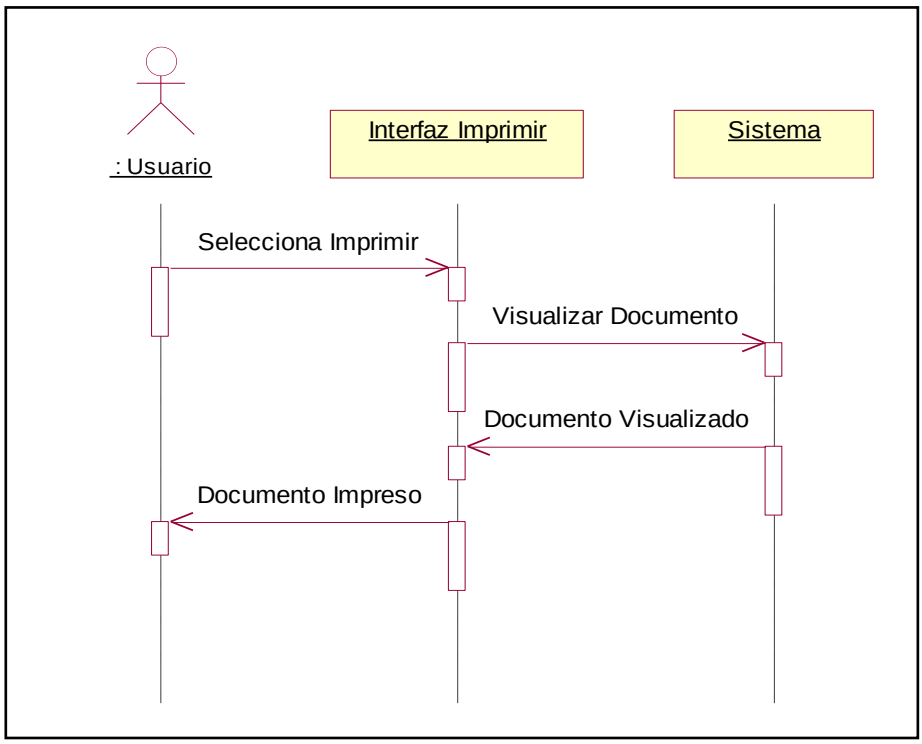


Figuras 3.37: Diagrama De Secuencia Consultar Documento Creado

Tabla 3.41. Contrato De Operación Consultar Datos

Nombre:	Consultar Documentos Registrados
Responsabilidades:	Deberá consultar los datos de los Documentos Registrados dependiendo de los parámetros solicitado
Tipo:	SISCERDESPE-L
Caso de Uso:	Consultar Documentos Registrados
Notas:	
Excepciones:	
Salidas:	
Precondiciones:	Consultar Documentos Registrados
Postcondiciones:	

3.2.3.13 IMPRIMIR DOCUMENTOS REGISTRADOS



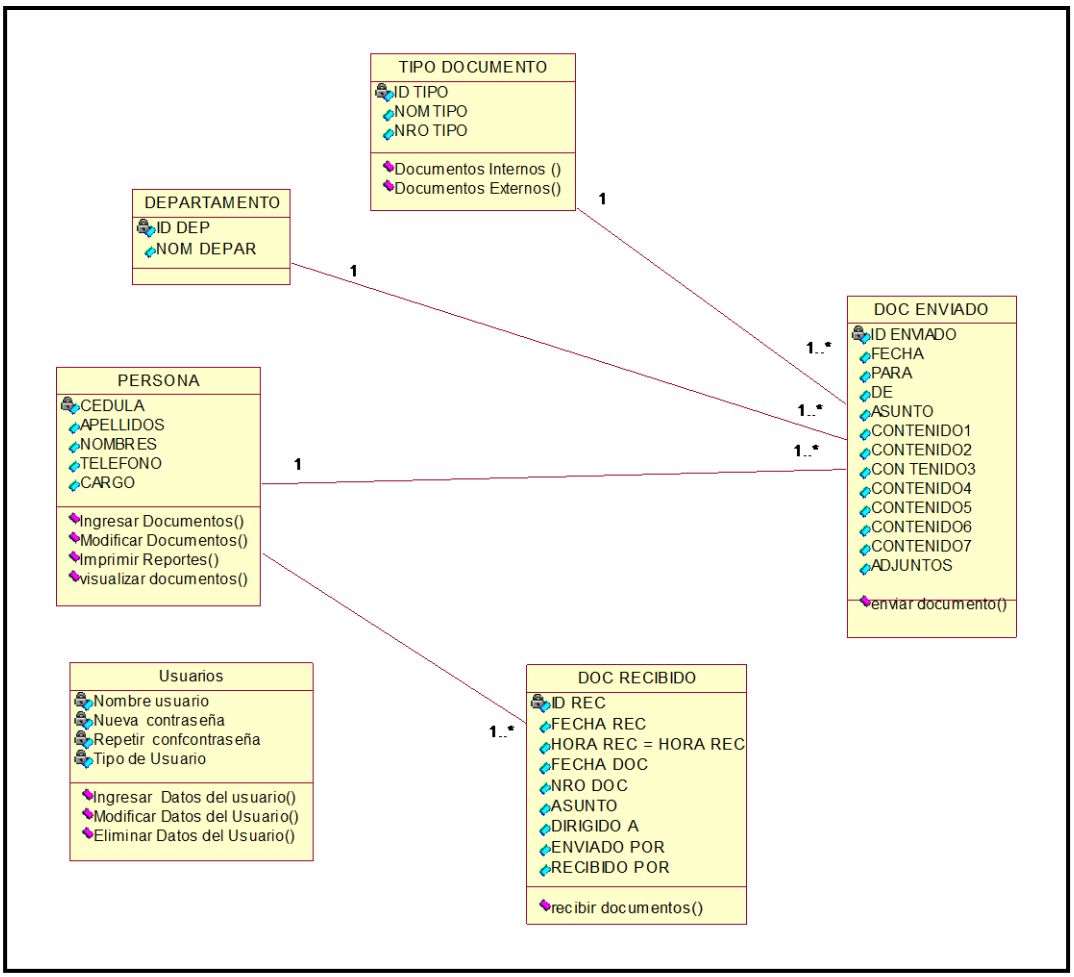
Figuras 3.38: Diagrama De Secuencia Imprimir Documento Registrado

Tabla 3.42. Contrato de Operación Imprimir Datos

Nombre:	Imprimir Documentos Registrados
Responsabilidades:	Deber imprimir todos los documentos registrados por el usuario
Tipo:	SISCERDESPE-L
Caso de Uso:	Imprimir Documentos Registrados
Notas:	
Excepciones:	
Salidas:	Documento Registrado
Precondiciones:	Almacenar datos para imprimir
Postcondiciones:	Impresión de los Documentos Registrados

3.2.3.14 DIAGRAMA DE CLASES

Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos, Métodos y las relaciones entre ellos. El diagrama de clases es utilizado durante el proceso de análisis y diseño del sistema, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.



Figuras 3.39: Diagrama De Clases

3.3 IMPLEMENTACIÓN Y PRUEBA

3.3.1 IMPLEMENTACIÓN

Para la implementación y construcción del programa se baso en la utilización de las UML y como los casos de uso, diagrama de secuencia y diagrama de clase. Para mayor facilidad y entendimiento en la construcción y diseño de las tablas.

RUP (Proceso Unificado de Rational) es una metodología que nos va a decir, el cómo tenemos que hacer las cosas. En el desarrollo del software.

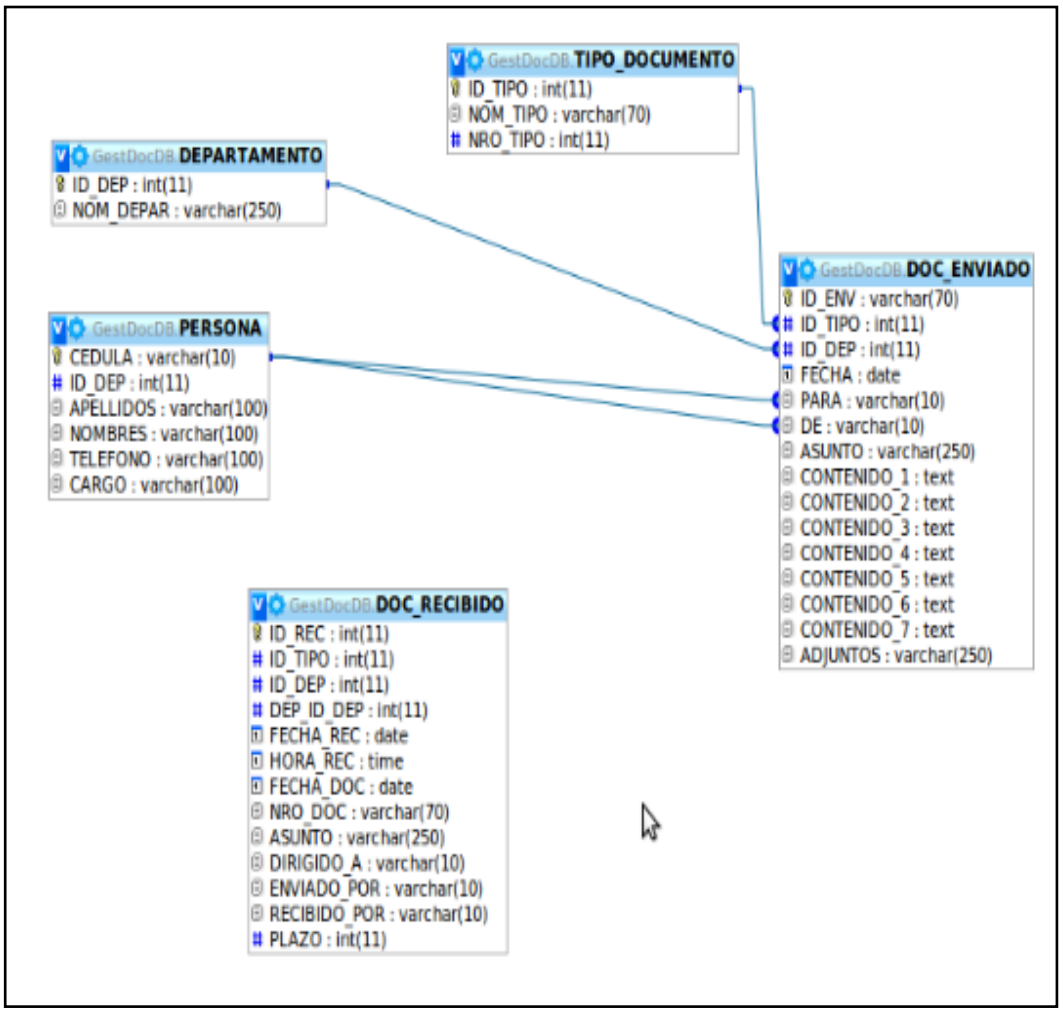
UML (Lenguaje unificado de modelado) es un lenguaje de modelado de datos, que nos va a servir para modelar el sistema.

3.3.1.1 SEGUIMIENTO Y EJECUCIÓN DEL PROGRAMA (SISCERDESPE-L)

Luego de haber planificado las iteraciones se procede al seguimiento y ejecución de las tareas. Hay que tomar en cuenta, que la planificación inicial puede ser distinta a la real, debido a que las fechas de inicio y fin pueden variar durante la realización del proyecto.

En cada iteración por realizarse en el proyecto, se implementará las historias de usuario involucradas, controlando que la ejecución de estas se encuentre de acuerdo a los permisos de acceso al sistema establecido.
En el transcurso de una iteración el equipo de desarrollo como primer punto debe ejecutar las tareas y concluir con el control de las mismas.

3.3.1.2 BASE DE DATOS



Figuras 3.40: De La Base De Datos

3.3.1.3 PROTOTIPO DE INTERFACES

3.3.1.3.1 PÁGINA DE INICIO

Figura 3.41: Pantalla de la página de inicio

3.3.1.3.2 GALERÍA DE IMÁGENES

Figura 3.42: Pantalla de la galería de imágenes

FORO

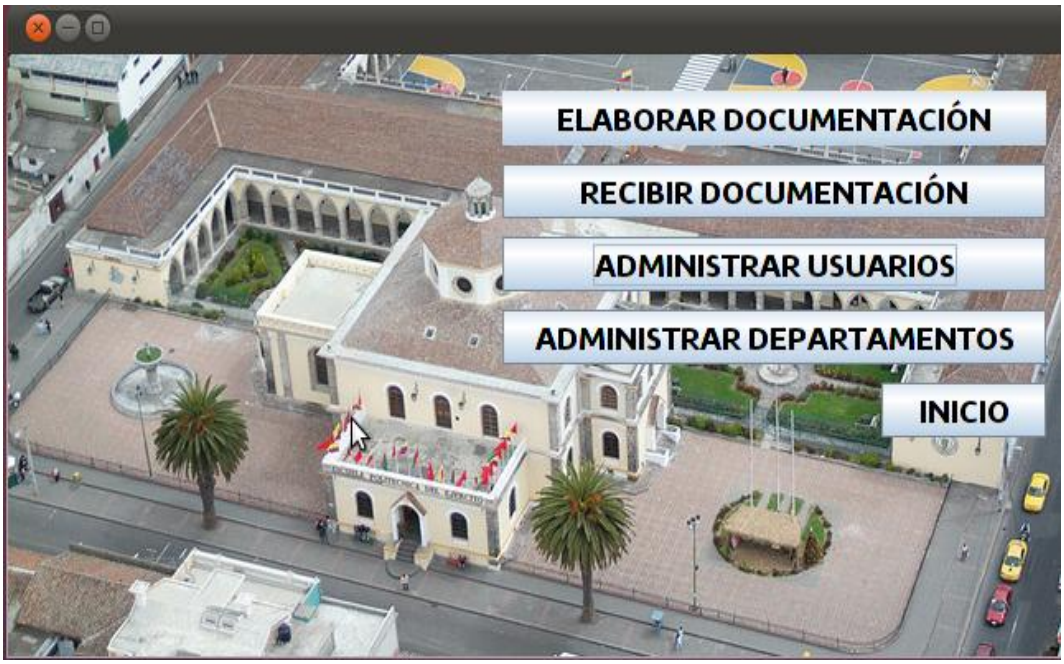


Figura 3.43: Pantalla de foro

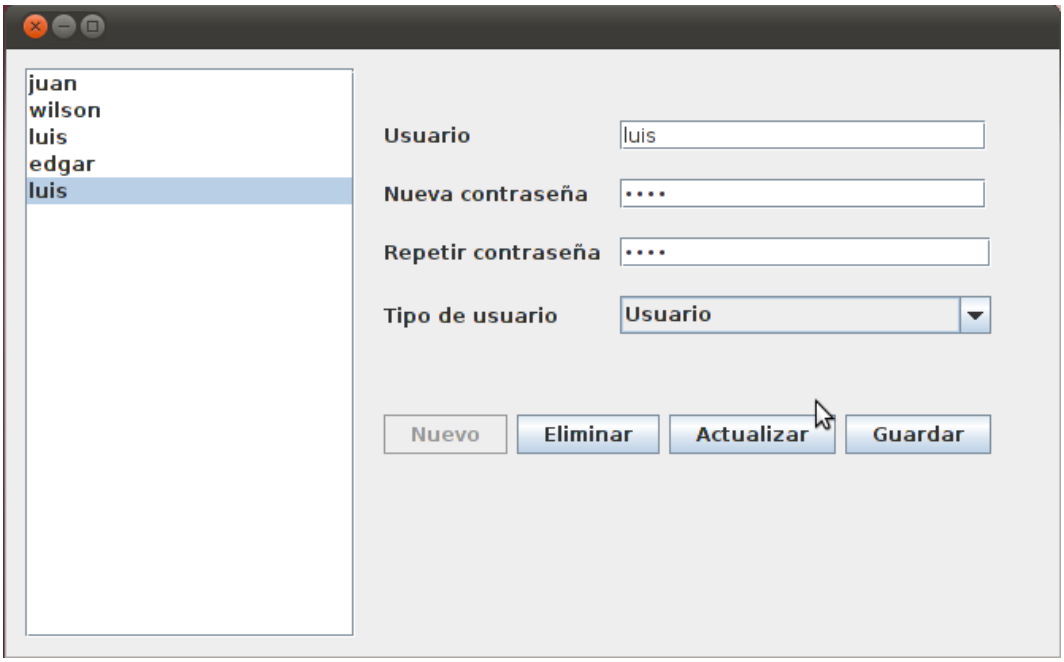


Figura 3.44: Pantalla de foro

Figura 3.45: Pantalla de foro

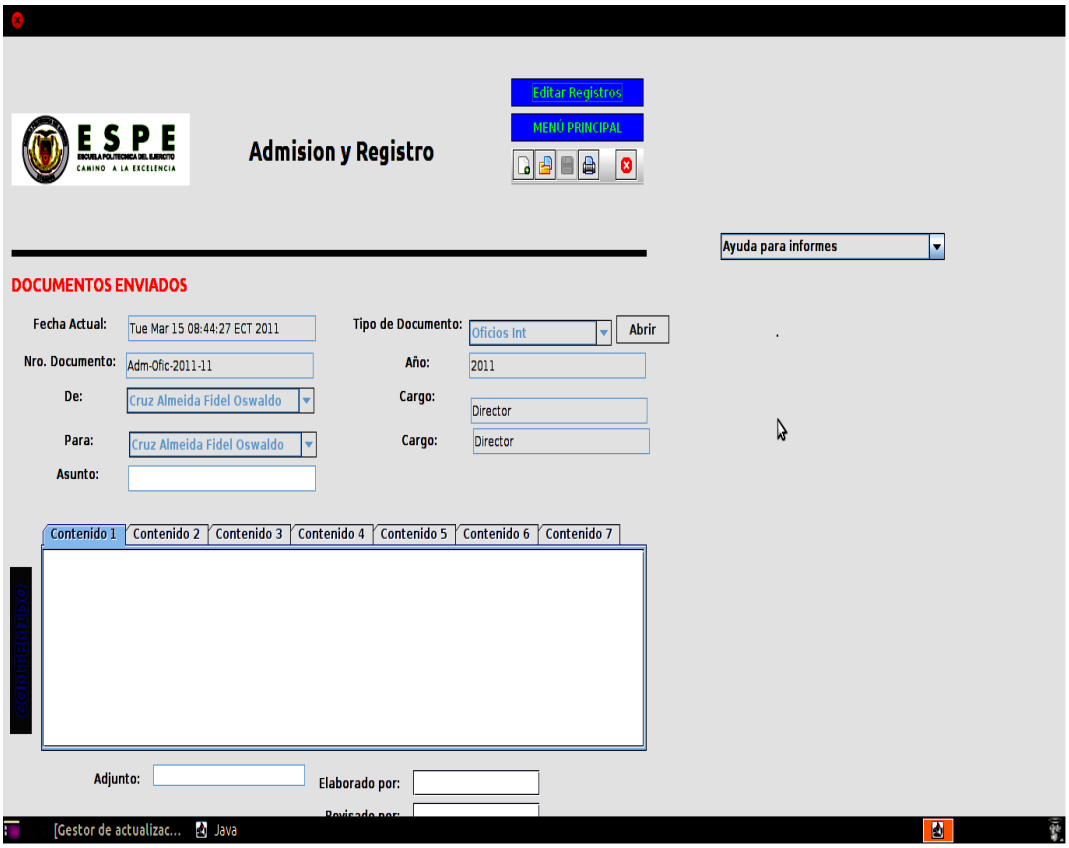
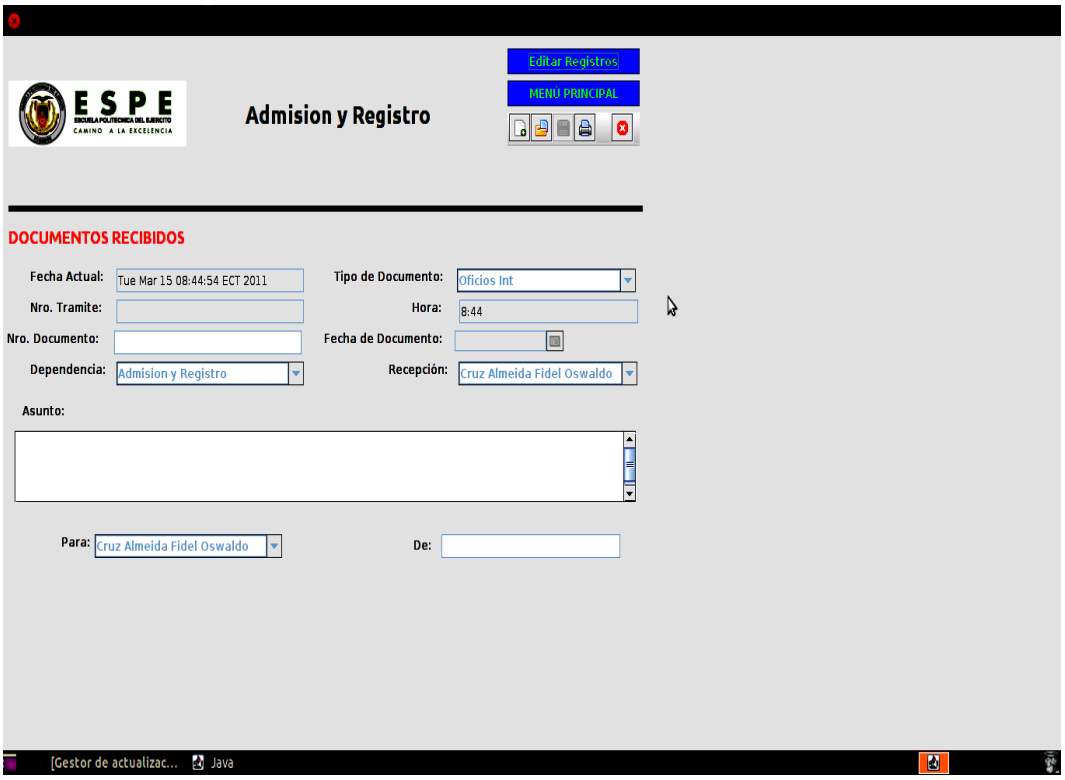


Figura 3.46: Pantalla de foro



3.3.2 TRÁFICO MENSUAL

El flujo de datos se calcula a través del número de Documentos enviados y Documentos Registrados que se ingresan al sistema, por la cantidad de visualizaciones de las páginas esperadas por mes.

Para el **SISCERDESPE-L** el primer registro de datos mensual, es un aproximado de 5 a 10 Documentos Enviados y Documentos Registrados de visualizaciones por mes.

Una vez conocidos estos datos se está en la posibilidad de escoger un servicio de alojamiento de acuerdo a las necesidades que presente los diferentes departamentos de la ESPE-L Extensión Latacunga.

3.3.3 UTILIZACIÓN DE HERRAMIENTAS

La utilización de las Herramientas Rational Rose, Java-NetBeans, Mysql y el Internet de la ESPE-L a posibilitado que la creación y implementación del SISCERDESPE-L se haya podido hacer de una manera rápida y cómoda, ha evitado que tengamos que realizar las funcionalidades con dificultades, como también la ayuda del Director y Codirector de tesis de la misma.

A continuación se presenta de manera un poco más detallada la forma en que se diseño y desarrollo la implementación, aunque sin entrar en detalles de cómo se realizo la codificación.

3.3.4 CREACIÓN DEL MODELO SISCERDESPE-L

Para poder realizar cualquier labor SISCERDESPE-L que se ha definido en la etapa del diseño, para los cuales trabaja para el SISCERDESPE-L vaya cobrando forma. Esta construcción se ha realizado en paralelo con el diseño del resto de desarrollo del ciclo. Ya que ningunas de las decisiones que se tomen después del diseño van a influir.

Para poder realizar cualquier labor en SISCERDESPE-L que se ha definido en la etapa del diseño, para los cuales trabaja para el SISCERDESPE-L vaya cobrando forma. Esta construcción se ha realizado en paralelo con el diseño del resto de desarrollo del ciclo. Ya que ningunas de las decisiones que se tomen después del diseño van a influir.

Figura 3.47: De La Instalación Donde Se Implementara El SISCERDESPE-L

La implementación de este ciclo de desarrollo ha sido bastante rápida, ya que la gran mayoría de Hardware y Software se encontraban implementados por lo que ha hecho posible la implantación del sistema.

Sin embargo, se ha dedicado más tiempo a la realización de las pruebas para comprobar el funcionamiento, el hecho que se realizó en Open Source ha facilitado el desarrollo e implementación del sistema durante esta fase.

Donde se ha debido de hacer la mayoría de cambios, fue en la parte de dar privilegios a los usuarios, Realizar el diseño de los Documentos Para Enviar: teniendo en cuenta que la aplicación cumple el siguiente esquema del Figuras.

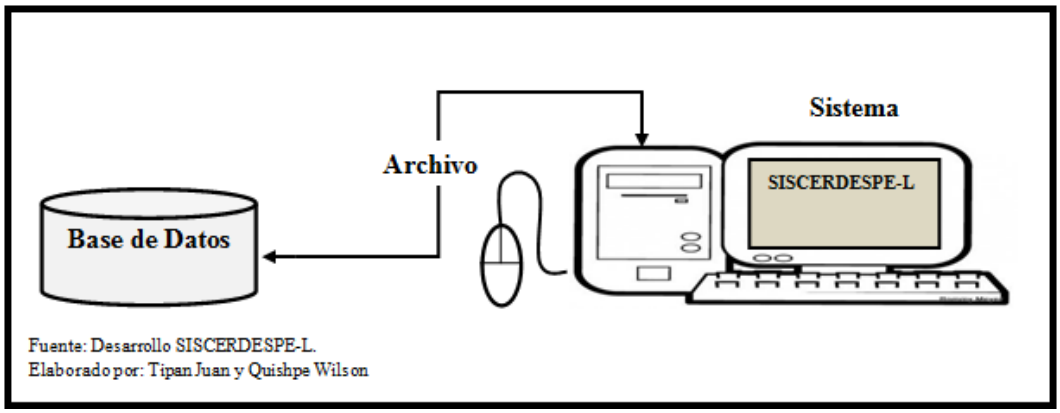


Figura 3.48: Esquema de despliegue del SISCERDESPE-L.

El sistema SISCERDESPE-L se ha implementado luego de varias pruebas ejecutadas en el mismo, y en la base de datos Mysql y finalmente implantado en los departamentos de la Escuela Politécnica del Ejército Extensión Latacunga.

CAPITULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

Al finalizar el presente trabajo de investigación podemos concluir lo siguiente:

- La realización de este Proyecto, permitió elaborar un Sistema Automatizado para controlar el envío y recepción de documentos de la ESPE-L, fue desarrollado como una manera de darle solución a las fallas que se presentan al cumplir con esas funciones en forma manual, cuyo planteamiento del problema se originó de la observación directa de las tareas y actividades que se cumplen en los diferentes departamentos de la ESPE-L.
- Se soluciono las fallas encontradas en forma precisa y metodológica, se planteó el objetivo general y los objetivos específicos, los cuales se cumplieron a cabalidad.
- En las bases teóricas se explicó detalladamente el software tanto del manejador de la base de datos, como el que se utilizó para el desarrollo del sistema. Así como también, se plasmaron conceptos básicos para el entendimiento del lenguaje que se utiliza en la realización del proyecto.
- Para el diseño, el nivel de investigación y para elaborar otros conceptos, se consideró las consultas en internet y la bibliografía obtenida en la biblioteca de la ESPE-L.
- Una vez, culminado todos los puntos propuestos para el desarrollo del proyecto, se llega a la conclusión que fue factible lograr un sistema que se adapte a los usuarios de los diferentes departamentos de la ESPE-L, al haberse elaborado el Sistema que permitirá el Control en la Emisión y Recepción de Documentos para la ESPE-L, el cual puede ser susceptible de efectuarle ajustes, de acuerdo a los nuevos

requerimientos que se tengan en el proceso de puesta en marcha del sistema.

4.2 RECOMENDACIONES

Para mejorar este sistema se puede tomar en cuenta la siguiente acotación:

- Utilizar software libre, ya que a más de no necesitar licencia, son robustos, estables, y además no obligan a usar una plataforma en particular, como ocurren con otros tipos de software.
- Realizar pruebas en cada una de las fases de Desarrollo del Proyecto con el fin de comprobar el buen funcionamiento de la aplicación.
- Utilizar una metodología orientada a objetos al desarrollar un software ya que representa un ahorro de tiempo, permitiendo tener un enfoque claro del software desde el principio.
- Adiestramiento a los usuarios que van a utilizar el sistema, para un mejor funcionamiento del mismo.
- Mantenimiento constante al sistema, como a la base de datos.

REFERENCIAS BIBLIOGRAFICAS

LIBROS BIBLIOTECA ESPE-L:

- Java 2 Interfaces graficas aplicaciones para internet Autor Fco. Javier Ceballos
- Programación Orientada a Objetos Segunda edición Autor Luis Joyanes Aguilar
- UML y Patrones Introducción al Análisis y Diseño orientado a Objetos Autor Craig Larman
- JIM BUYENS(aprenda a desarrollo de base de datos web edición primera 2000)
- PAUL DUBOIS (edición especial de MSQL del 2001)
- http://www.scribd.com/doc/16600726/Guia-Como-Programar-Java-Con-Netbeans-PDF
- http://www.scribd.com/doc/967380/Tutorial-Netbeans
- http://dev.mysql.com/doc/refman/5.0/es/creating-a-spatially-enabled-mysql-database.html
- http://dev.mysql.com/doc/refman/5.0/es/tutorial.html
- http://www.webestilo.com/mysql/intro.phtml

DIRECCIONES ELECTRONICAS

- http://www.google.com
- http://www.wikipedia.com
- http://www.monografias.com
- http://www.programacion.com

GLOSARIO

LINUX.- Es un Sistema Operativo basado en UNIX, por lo que se trata de un sistema operativo muy seguro y poderoso, con ventajas adicionales: es gratuito y abierto.

EXTREME PROGRAMMING.- Es el más destacado de los procesos agiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

OPEN SOURCE.- Libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

JAVA.- es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

NETBEANS.- es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java.

MySQL.- es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

UML.- por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group).

IDE.-Un entorno de desarrollo integrado (en inglés intégrate development environment) es un programa informático compuesto por un conjunto de herramientas de programación.

JDK.- Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.

ANEXOS

ANEXO 1

Manual de Usuario

Información C D # 1

MANUAL DE USUARIOS

DEL SISTEMA DE CONTROL PARA LA EMISION Y RECEPCION DE LA DOCUMENTACIÓN DE LA ESCUELA POLITECNICA DEL EJERCITO EXTENSIÓN LATACUNGA

1 VENTANA DE INGRESO COMO ADMINISTRADOR O USUARIO

Lo primero que debemos realizar es ejecutar el programa luego nos muestra una ventana de verificación de ingreso para el Administrador o Usuario

Pasos

Ingresamos el Nombre del Administrador o del Usuario

Ingresar la contraseña de ingreso

2 VENTANA DE INGRESO AL SISTEMA

Al momento de ingresar como Administrador o Usuario el sistema nos muestra una Ventana con las siguientes opciones:

a) Seleccionar Tipo de Departamento

b) Menú de Inicio

3 ICONO DE SELECCION DE DEPARTAMENTOS

Al seleccionar el icono de Departamentos nos despliega el numero de departamentos que se encuentran registrados y poder realizar la elección a que departamento pertenecemos, como muestra en la siguiente ventana.

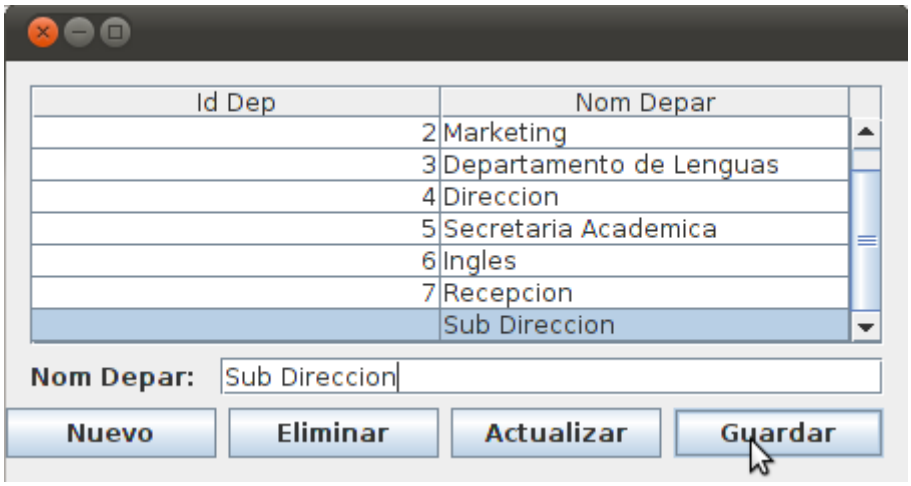
4 ICONO DE MENU INICIO

Al seleccionar el icono de menú inicio nos muestra una ventana con las **siguientes opciones.**

5 OPCION ADMINISTRAR DEPARTAMENTO

Al seleccionar el icono de Administrar Departamento nos muestra la siguiente venta donde podemos realizar algunas tareas como crear departamentos que no estén registrados y eliminar departamentos no deseados

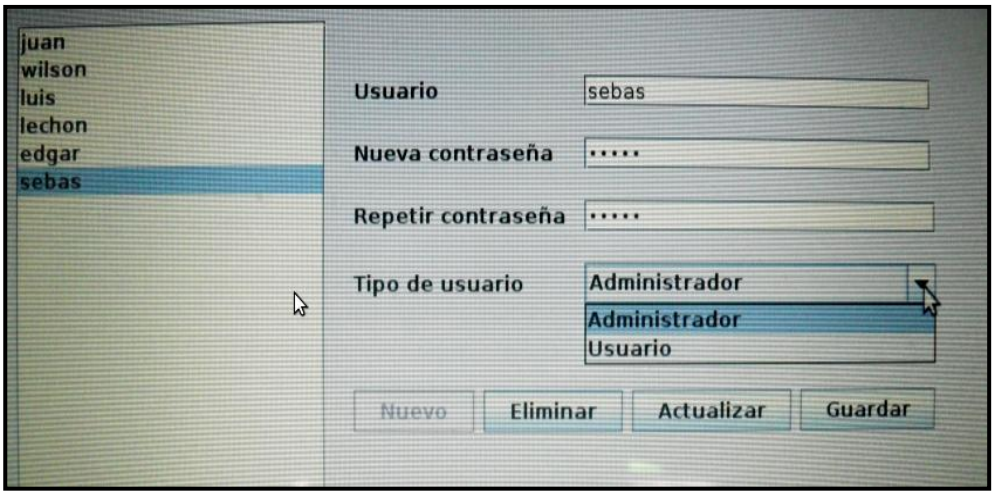
Para crear un departamento primero seleccionamos el icono Nuevo escribimos en los campos el nombre del departamento seleccionamos el icono guardar y actualizar



Para **eliminar un departamento** seleccionamos al departamento que deseamos eliminar damos un clip en el icono eliminar y listo

6 OPCION ADMINISTRAR USUARIOS

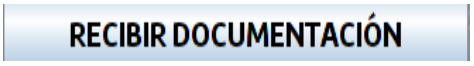
Al seleccionar el icono de Administrar Usuarios nos muestra la siguiente venta donde podemos realizar algunas tareas como crear Administradores y Usuarios que no estén registrados y También Podemos eliminarlos los no deseados



Para crear un departamento primero seleccionamos el icono Nuevo escribimos en los siguientes campos el nombre del Administrador o Usuario, Nueva contraseña y Repetir contraseña seleccionamos que tipo de usuario vamos a crear Administrador o Usuario.

Para **eliminar un Administrador o Usuario** seleccionamos el Administrador o Usuario a eliminar damos clip en el icono eliminar y listo

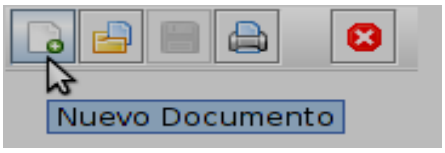
7 OPCION RECIBIR DOCUMENTACION



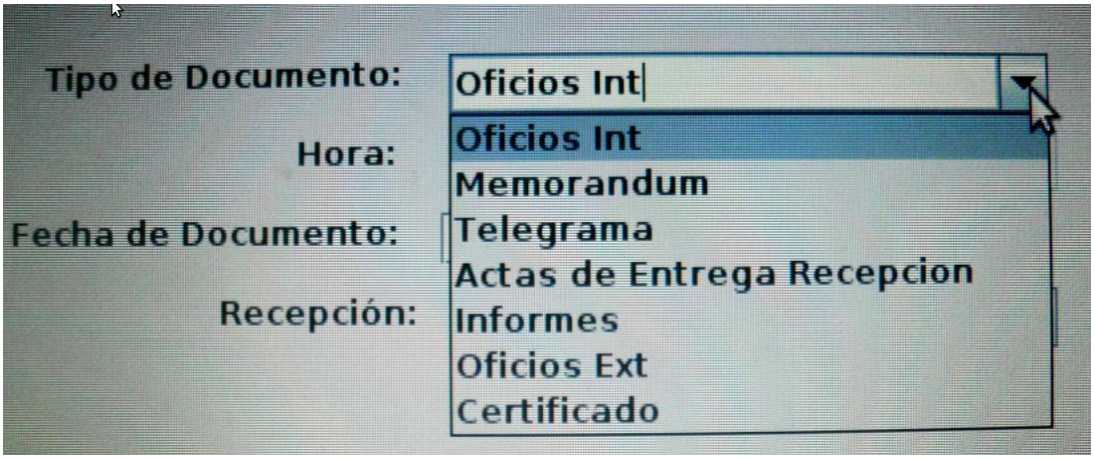
Al seleccionar el icono de Documentos Recibidos nos muestra la siguiente venta donde podemos realizar algunas tareas como crear un nuevo registro o eliminar un registro, almacenar el nombre de una nueva

persona en la base de datos o eliminarla, realizar consultas de los registros de los documentos recibido, imprimir los registros

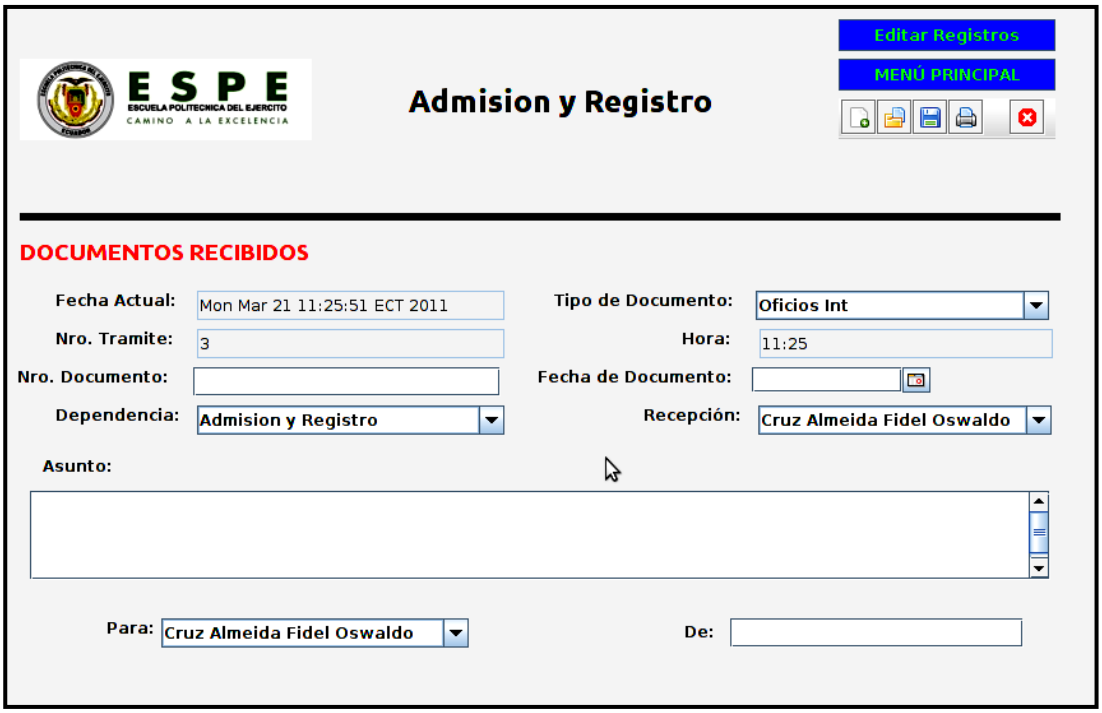
Para crear un nuevo documento de registro seleccionamos el icono nuevo documento.



Seleccionamos que tipo de documento vamos a registrar.



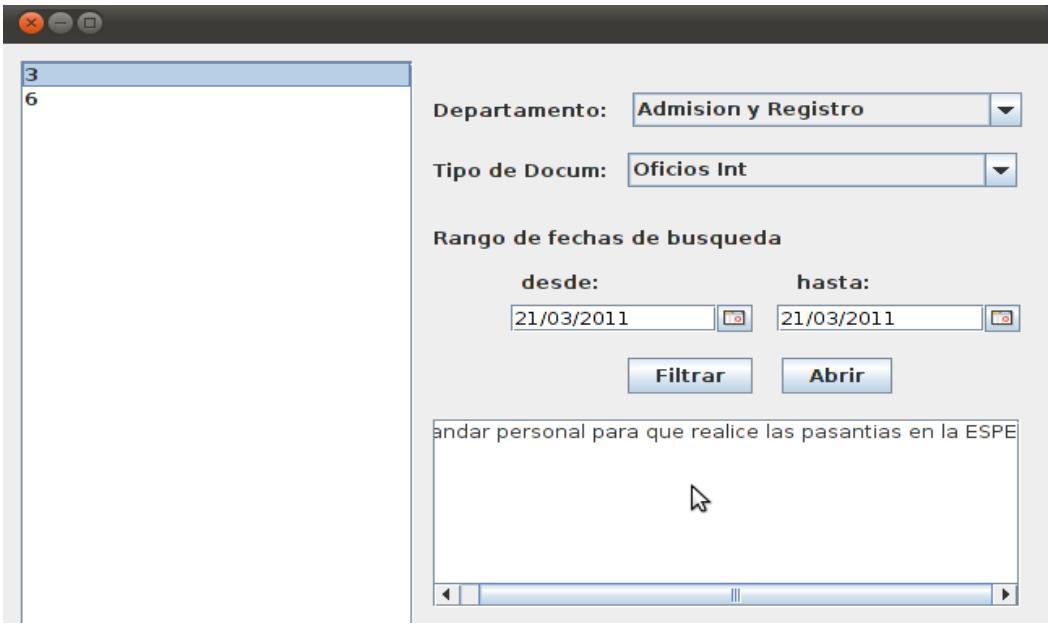
Llenamos los campos vacíos y seleccionamos los campos de selección que nos muestra la siguiente ventana



Para guardar el documento registrado Seleccionamos el icono guardar y li

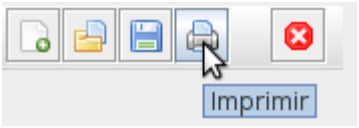
Para realizar una consulta de todos los documentos guardados escogemos el siguiente icono de consulta.

Luego nos muestra la siguiente venta de consulta

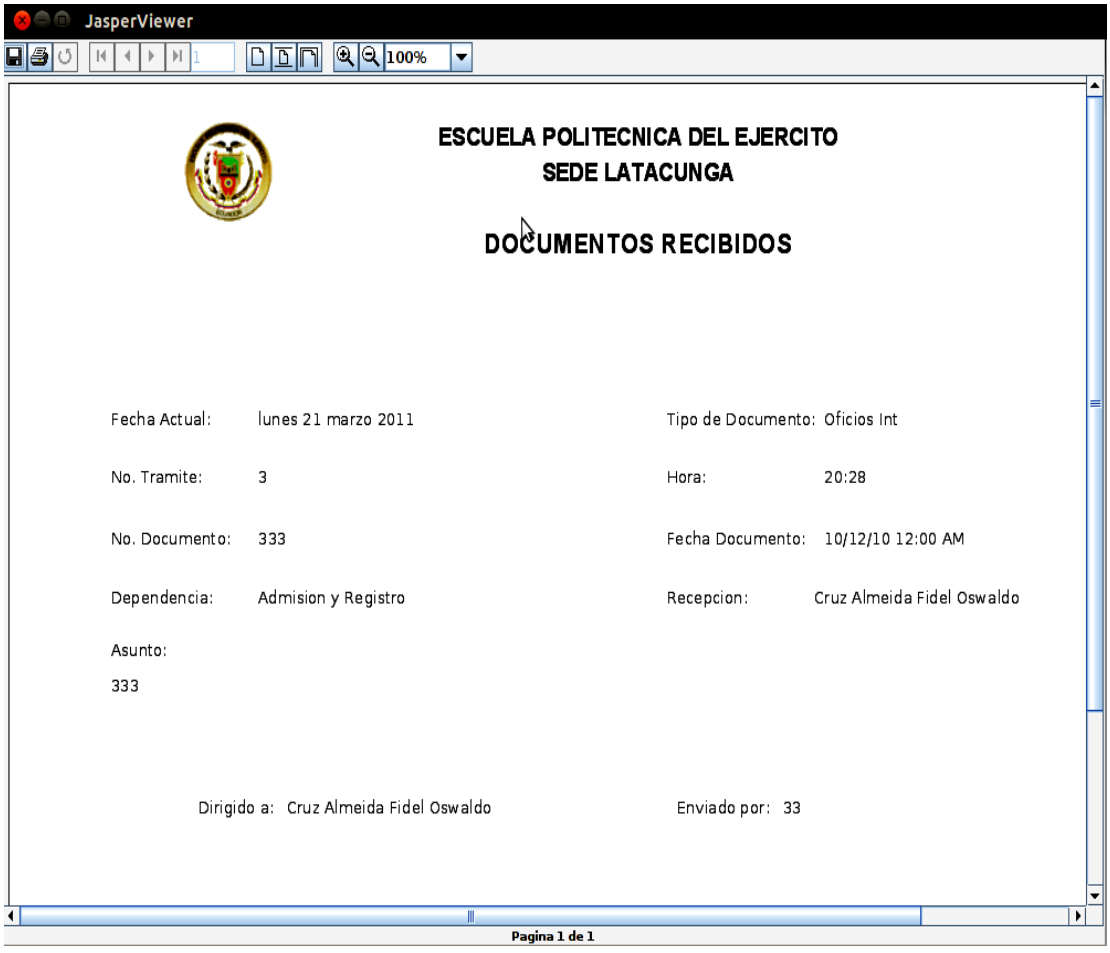


Para realizar una consulta debemos seleccionar en los siguientes campos tipo de departamento, tipo de documento y rango de fechas de búsqueda, una vez llenados estos campos damos clip en el icono filtrar y listo.

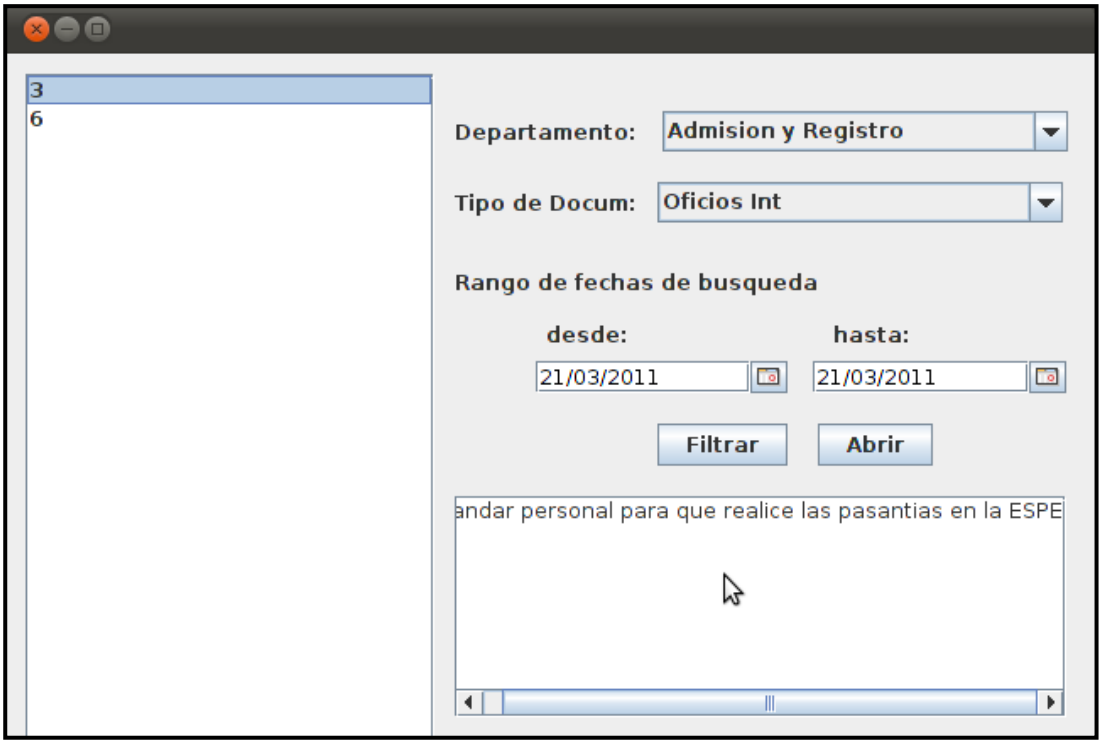
Para poder imprimir el documento seleccionamos el icono imprimir.

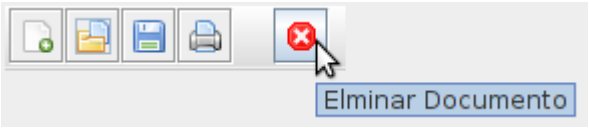


Luego nos visualiza el documento que vamos a imprimir como nos muestra en la siguiente ventana.



Para poder eliminarla un documento registrado primero realizamos una consulta de todos los documentos luego seleccionamos el documento que vamos a eliminar damos clip en el icono abrir el documento.

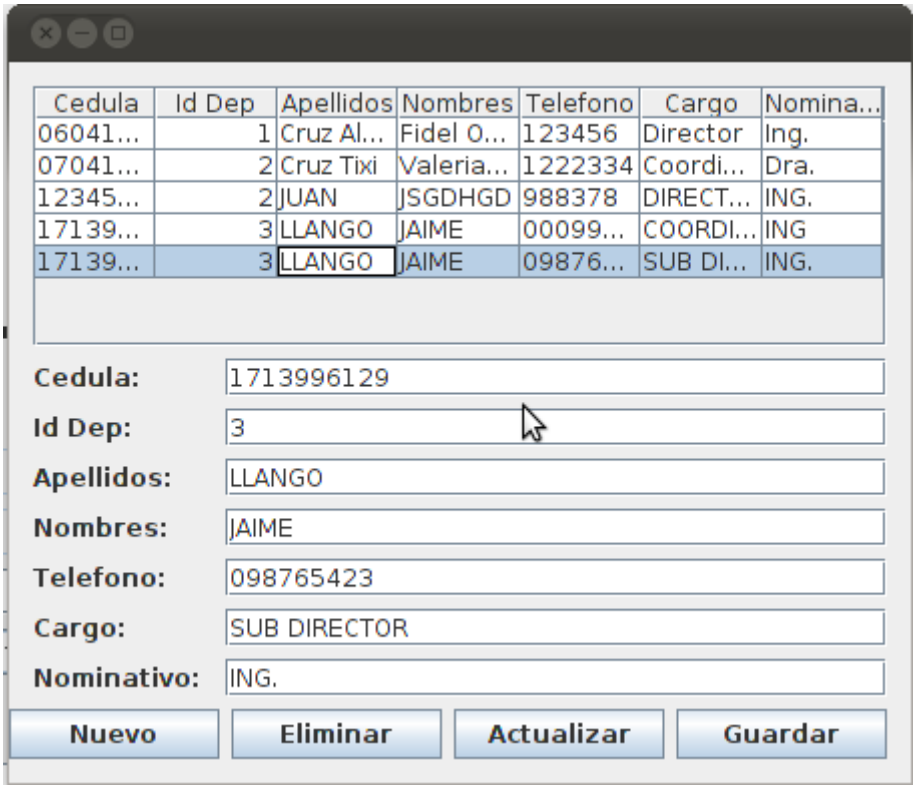




Damos un clip en el icono eliminar y listo.

Para registrar a la personas que no se encuentran en la base de datos seleccionamos el icono Editar registro.

Luego nos muestra la siguiente ventana.

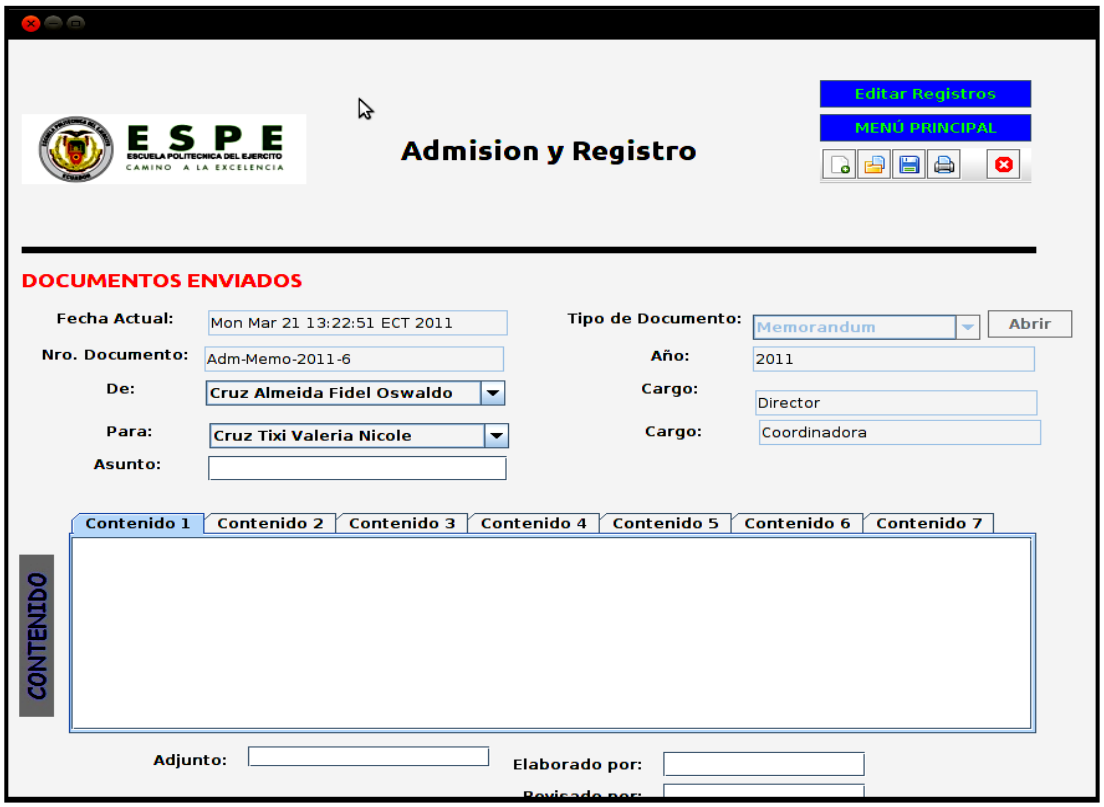


Para poder registrar a una nueva persona damos clip en el icono Nuevo llenamos los campos vacíos con la información necesaria luego damos clip en el icono guardar y actualizar

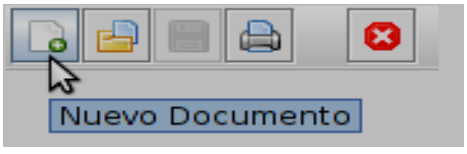
Para eliminar una persona registrada seleccionamos del listado a la persona que deseamos eliminar damos clip en el icono eliminar y listo

8 OPCION DOCUMENTOS ENVIADOS

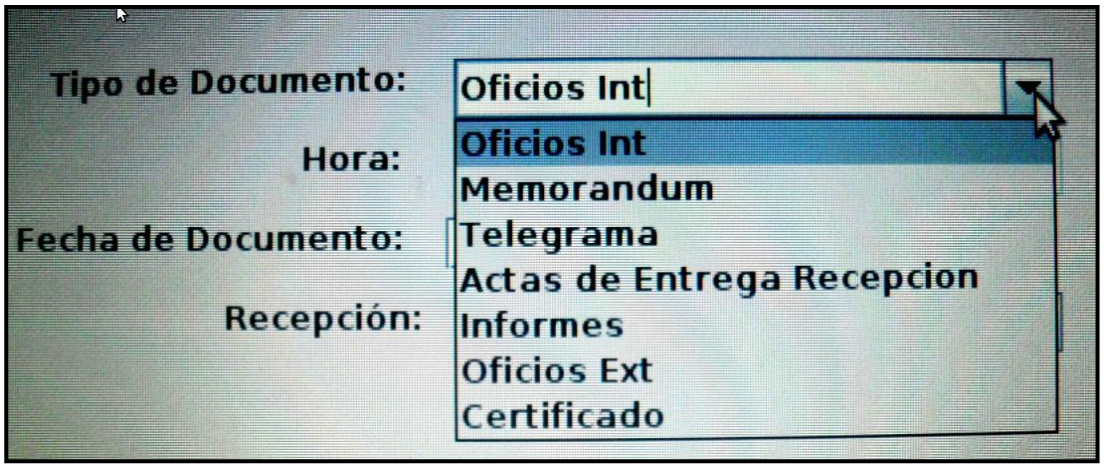
Al seleccionar el icono Documentos enviados nos muestra la siguiente venta donde podemos realizar algunas tareas como crear un nuevo documento o eliminar un documento, almacenar el nombre de una nueva persona en la base de datos o eliminarla, realizar consultas de los documentos creados, imprimir los documentos creados.



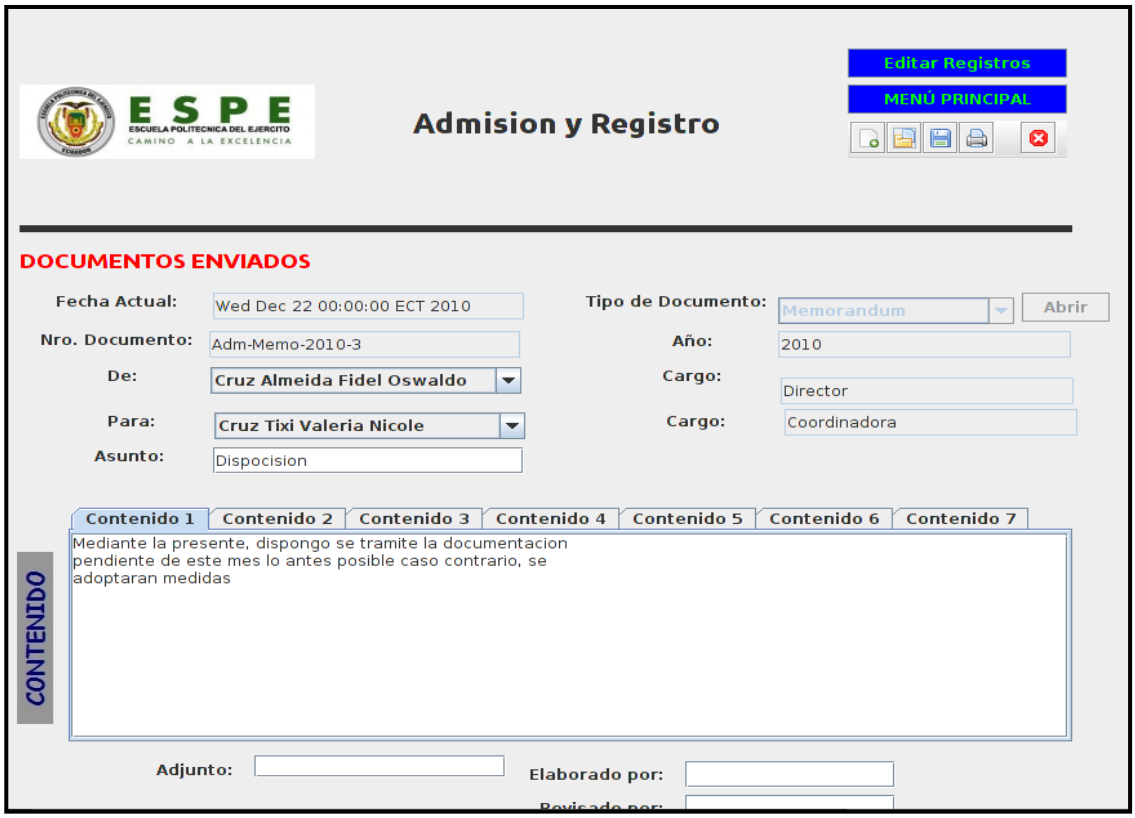
Para crear un nuevo documento seleccionamos el icono nuevo documento.



Seleccionamos que tipo de documento vamos a realizar y damos clip en el icono abrir documento.

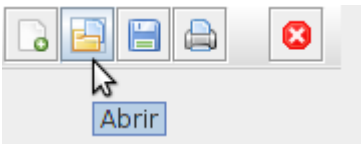


Llenamos los campos vacíos y seleccionamos los campos de selección que nos muestra la siguiente ventana

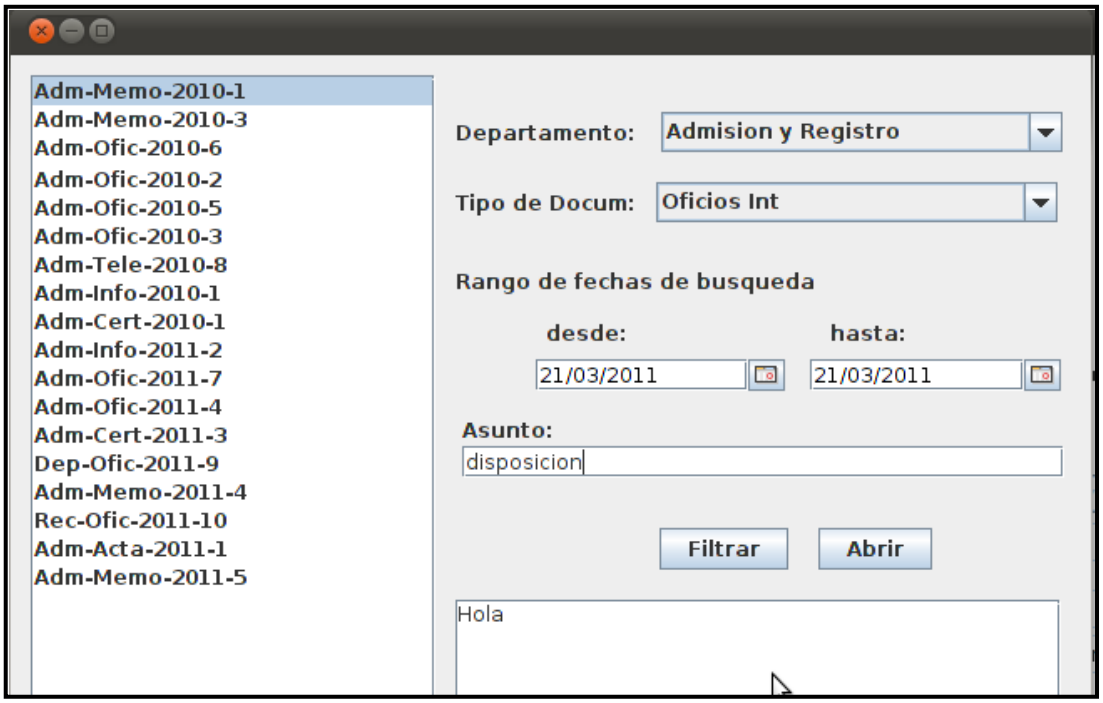


Para guardar el documento creado Seleccionamos el icono guardar y listo

Para realizar una consulta de todos los documentos guardados escogemos el siguiente icono de consulta.

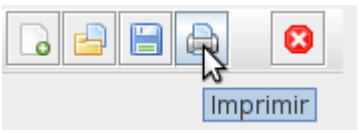


Luego nos muestra la siguiente venta de consulta

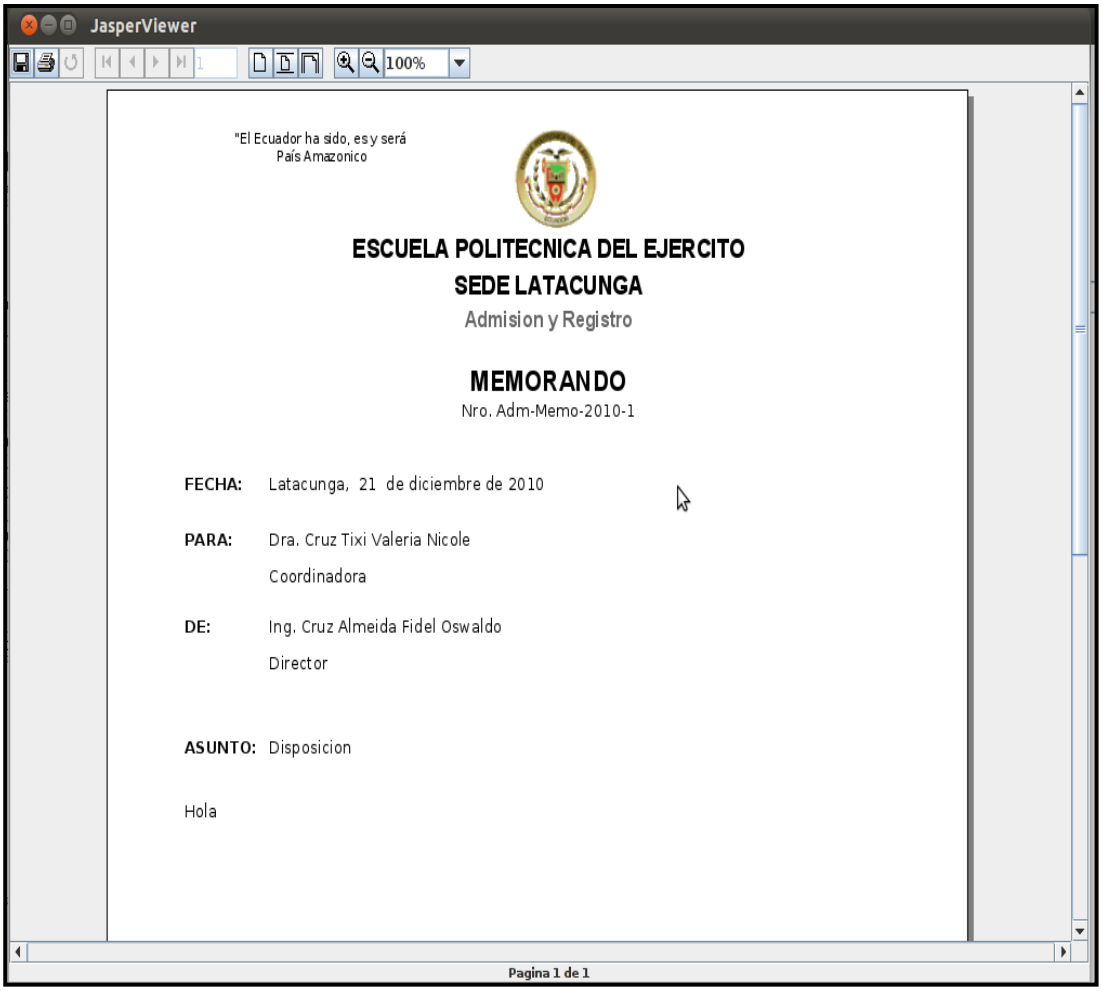


Para realizar una consulta debemos seleccionar en los siguientes campos tipo de departamento, tipo de documento y rango de fechas de búsqueda, una vez llenados estos campos damos clip en el icono filtrar y listo.

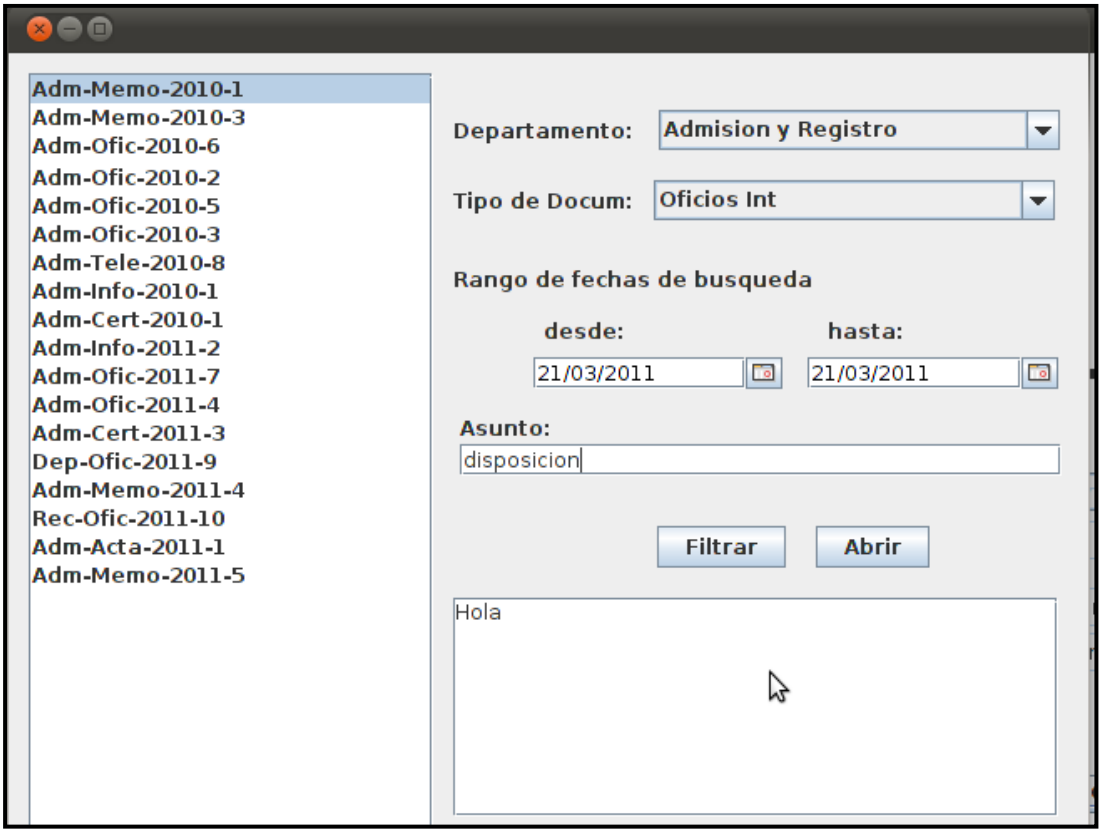
Para poder imprimir el documento seleccionamos el icono imprimir.



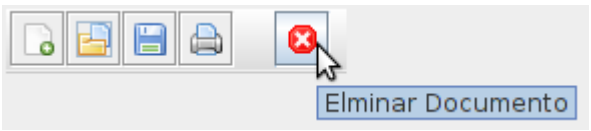
Luego nos visualiza el documento que vamos a imprimir como nos muestra en la siguiente ventana mandamos a imprimir y listo.



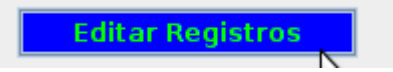
Para poder eliminarla un documento registrado primero realizamos una consulta de todos los documentos luego seleccionamos el documento que vamos a eliminar damos clip en el icono abrir el documento como nos muestra en la ventana siguiente.



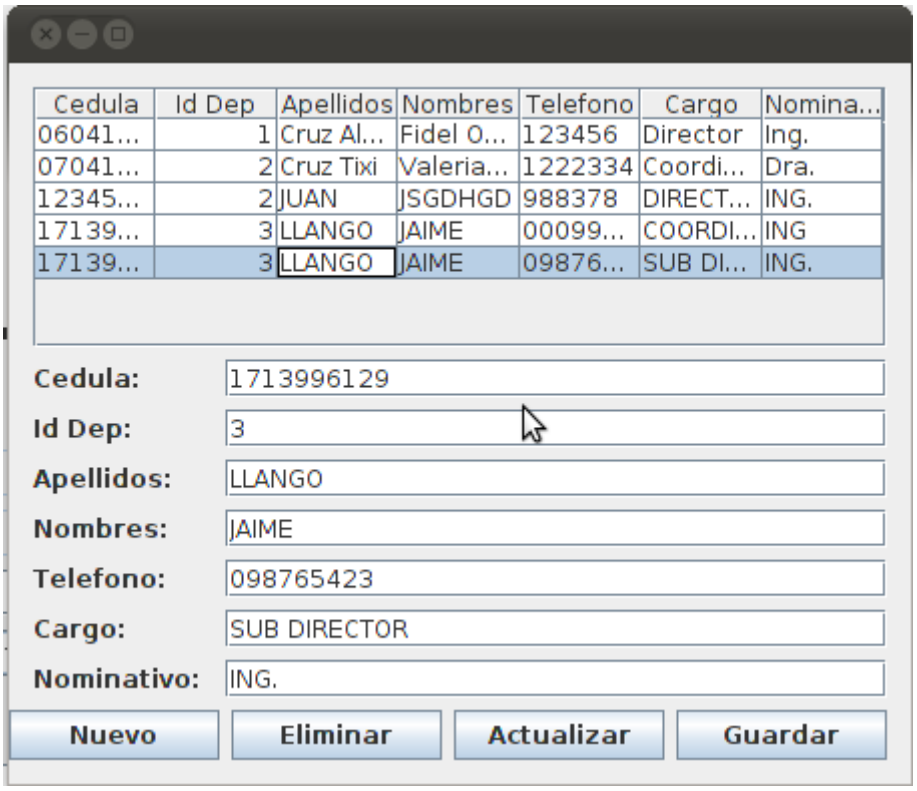
Una vez que ya esté abierto el documento solo damos clip en el icono eliminar documento y listo.



Para registrar a personas que no se encuentran en la base de datos seleccionamos el icono Editar registro.



Luego nos muestra la siguiente venta.



Para poder registrar a una nueva persona damos clip en el icono Nuevo llenamos los campos vacíos con la información necesaria luego damos clip en el icono guardar y actualizar

Para eliminar una persona registrada seleccionamos del listado a la persona que deseamos eliminar damos clip en el icono eliminar y listo

NOTA: AL INGRESAR COMO USUARIO SE RESTRINGE LAS SIGUIENTES FUNCIONES:

Crear Administradores y Usuario

Crear Departamentos

Eliminar Documentos Recibidos y Enviados

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por Wilson Homero Quishpe Cahueñas y Juan Pablo Tipán Simbaña, bajo nuestra supervisión.

_____

Ing. Javier Montaluisa

CODIRECTOR DEL PROYECTO

_____

Ing. José Carrillo

DIRECTOR DEL PROYECTO

_____

Ing. José Luis Carrillo

CORDINADOR DE LA CARRERA DE SISTEMAS E INFORMÁTICA

_____

Dr. Rodrigo Vaca

SECRETARIO ACADÉMICO ESPE-L

Latacunga, Marzo del 2011