

UNIVERSIDAD PÚBLICA DE EL ALTO
CARRERA INGENIERIA DE SISTEMAS



PROYECTO DE GRADO

“SISTEMA WEB PARA EL REGISTRO Y CONTROL DE OPERACIONES DEL
LABORATORIO TECNICO DE SUELOS”

CASO: SOCIEDAD DE INGENIEROS DE BOLIVIA DEPARTAMENTAL LA PAZ

(SIB LA PAZ)

Para optar al título de Licenciatura en Ingeniería de Sistemas

Mención: “GESTION Y PRODUCCION”

Postulante: Univ. Adilson Eusebio Condori Felipes

Tutor Metodológico: Ing. Marisol Arguedas Balladares

Tutor Especialista: Ing. Rolando Alarcon Choquehuanca

Tutor Revisor: Lic. Margarita Bernarda López Mariaca

EL ALTO-BOLIVIA

2020

INDICE DE CONTENIDO

1. Marco preliminar	1
1.1. Introducción	1
1.2. Antecedentes	3
1.2.1. Antecedentes de la institución.	3
1.2.2. Antecedentes afines al proyecto.	4
1.3. Planteamiento del problema.....	7
1.3.1. Problemática.	7
1.3.2. Problema principal.....	8
1.3.3. Problemas Secundarios.....	8
1.4. Objetivos	9
1.4.1. Objetivo General.....	9
1.4.2. Objetivos Específicos.	9
1.5. Justificación.....	10
1.5.1. Técnica.....	10
1.5.2. Económica.	10
1.5.3. Social.	11
1.6. Metodología	11

1.7.	Herramientas	12
1.7.1.	Lenguaje de modelado unificado (UML).	12
1.7.2.	Elección de bases de datos.	13
1.7.3.	Elección de lenguaje de programación.	15
1.7.4.	Cuadro comparativo de servidores web APACHE y IIS.	17
1.8.	Límites y alcances	18
1.8.1.	Límites.	18
1.8.2.	Alcances.	18
1.9.	Aportes	19
2.	Marco teórico	20
2.1.	Sistema	20
2.2.	Sistema De Información.....	21
2.2.1.	Actividades De Un Sistema De Información.....	22
2.2.2.	Ciclo De Vida De Los Sistemas De Información.	24
2.3.	Gestión De Información.....	26
2.4.	Web	28
2.5.	Operaciones.....	28
2.6.	Metodología de Desarrollo.....	29

2.6.1.	El Proceso Unificado Racional.....	29
2.6.2.	Características esenciales.....	30
2.6.3.	Estructura del proceso.....	40
2.6.4.	Unified Modeling Language (UML).	40
2.6.5.	Diagramas del UML.	42
2.6.6.	Las restricciones de las operaciones: la programación por contrato. ..	48
2.7.	Tecnologías de software.....	49
2.7.1.	MySQL.	50
2.7.2.	PHP.	50
2.7.3.	Framework Bootstrap	54
2.7.4.	Framework CakePHP.	55
2.8.	Métricas de calidad	58
2.8.1.	NORMA ISO 9126.....	59
2.9.	Costos y/o beneficios	64
2.9.1.	COCOMO.....	64
2.10.	Seguridad.....	67
2.10.1.	Seguridad física.....	67
2.10.2.	Seguridad lógica.	68

3. Marco aplicativo	72
3.1. Introducción	72
3.2. Fase de inicio.....	73
3.2.1. Modelado del negocio.....	73
3.3. Fase de elaboración	76
3.3.1. Determinación y análisis de requerimientos.....	76
3.3.2. Descripción de los usuarios del sistema.	77
3.3.3. Diagramas de casos de uso del sistema.	78
3.3.4. Diagrama De Secuencia.....	86
3.3.5. Diagrama de actividades.....	90
3.4. Fase de diseño	94
3.4.1. Diagrama de clases.	95
3.4.2. Diagrama Físico.....	96
3.4.3. Diagrama De Despliegue.....	96
3.4.4. Diagrama De Paquetes.....	97
3.4.5. Diseño De Interfaz.....	98
3.5. Fase de transición.....	102
3.5.1. Implementación.	102

3.5.2.	Restricciones del sistema.....	106
3.5.3.	Capacitación.....	107
4.	Métricas de Calidad.....	108
4.1.	Introducción.....	108
4.2.	Funcionalidad.....	108
4.3.	Mantenibilidad.....	114
4.4.	Portabilidad.....	116
4.5.	Usabilidad.....	117
4.6.	Confiabilidad.....	118
4.7.	Eficiencia.....	119
4.8.	Resultado final.....	119
5.	Evaluación de costo-beneficio.....	121
5.1.	Introducción.....	121
5.2.	Análisis de costos.....	121
5.3.	Análisis de beneficios.....	125
5.4.	Costo /Beneficio.....	128
5.5.	Tasa interna de retorno.....	128
6.	Seguridad del Sistema.....	130

6.1. Introducción	130
6.2. Seguridad física	131
6.3. Seguridad lógica.....	131
7. Conclusiones y Recomendaciones.....	132
7.1. Conclusiones	132
7.2. Recomendaciones.....	133
Bibliografía	135

INDICE DE TABLAS

Tabla 1 Cuadro de ventajas y desventajas de PostgreSQL.....	13
Tabla 2 Cuadro de ventajas y desventajas de MySQL	14
Tabla 3 Cuadro de ventajas y desventajas de ASP	15
Tabla 4 Cuadro de ventajas y desventajas de PHP	16
Tabla 5 Cuadro de ventajas y desventajas de APACHE	17
Tabla 6 Cuadro de ventajas y desventajas de IIS.....	17
Tabla 7 Descripción del caso de uso solicitud de ensayo	74
Tabla 8 Descripción del caso de uso registro de ensayo.....	75
Tabla 9 Descripción de caso de uso entrega de ensayo.....	75
Tabla 10 Funciones del sistema.....	77
Tabla 11 Descripción Caso de Uso - Adicionar usuario.....	80
Tabla 12 Descripción Caso de Uso - Elimina usuario.....	81
Tabla 13 Descripción caso de uso- Edita usuario.....	82
Tabla 14 Descripción de caso de uso- Ensayo.....	84
Tabla 15 Descripción de caso de uso-Reporte de ensayos.....	86
Tabla 16 Cálculo de Punto Función.....	110
Tabla 17 Valores de ajuste de complejidad.....	110
Tabla 18 Punto Función.....	111
Tabla 19 Información requerida por el IMS.....	115
Tabla 20 Evaluación de preguntas para calcular la usabilidad.....	117
Tabla 21 Evaluación de preguntas para calcular la eficiencia.....	119

Tabla 22 Resultado total.....	120
Tabla 23 Conversión de puntos función.....	122
Tabla 24 Relación de valores del modelo COCOMO	124
Tabla 25 Cálculo del VAN	126
Tabla 26 Criterio de interpretación del VAN.....	127
Tabla 27 Determinación del TIR.....	129

INDICE DE FIGURAS

Figura 1 Los Casos de Uso integran el trabajo	31
Figura 2 Trazabilidad a partir de los Casos de Uso	31
Figura 3 Los modelos se completan, la arquitectura no cambia drásticamente.....	33
Figura 4 Una iteración RUP.....	35
Figura 5 Esfuerzo en actividades según fase del proyecto	36
Figura 6 Caso de uso "Realizar Ensayo"	43
Figura 7 Diagrama de actividades, que muestra la manera en que el personal del laboratorio hace sus ensayos	45
Figura 8 Diagrama de clases	46
Figura 9 Diagrama de clases	48
Figura 10 Modelo de casos de uso del negocio- Caso registro ensayo.....	73
Figura 11 Modelo de caso de uso- General	79
Figura 12 Modelo de caso de uso-administra usuarios.....	80
Figura 13 Modelo de caso de uso- Registra solicitud.....	84
Figura 14 Modelo de caso de uso- Generar reportes	85
Figura 15 Diagrama de secuencia-Adicionar usuario.....	87
Figura 16 Diagrama de secuencia- Elimina Usuario	88
Figura 17 Diagrama de secuencia- Modifica usuario	88
Figura 18 Diagrama de secuencia- Ensayo.....	89
Figura 19 Diagrama de secuencia-Busca orden.....	89

Figura 20 Diagrama de secuencia-Generar reportes.....	90
Figura 21 Diagrama de actividades-Adicionar Usuario	91
Figura 22 Diagrama de actividades-Modifica o Elimina.....	91
Figura 23 Diagrama de actividades-Registrar Solicitud.....	92
Figura 24 Diagrama de actividades-Busca Orden	93
Figura 25 Diagrama de actividades-Generar reportes	94
Figura 26 Diagrama de clases del sistema.....	95
Figura 27 Modelo físico del sistema.....	96
Figura 28 Diagrama de despliegue del sistema	97
Figura 29 Diagrama de paquetes del sistema.....	98
Figura 30 Autenticación en el sistema	99
Figura 31 Pantalla principal del sistema.....	100
Figura 32 Pantalla de administración de usuarios	100
Figura 33 Pantalla de registro nuevo usuario.....	101
Figura 34 Pantalla de agregar Ensayo.....	102

1. Marco preliminar

1.1. Introducción

La tecnología de la información y la comunicación (TIC) viene jugando un papel central en nuestra sociedad, las cuales se desarrollan a partir de avances científicos producidos en los ámbitos de la informática y las telecomunicaciones. Las TIC son un conjunto de tecnologías que permiten el acceso, producción, tratamiento y comunicación de toda la información presentada en diferentes códigos. El elemento más representativo de las nuevas tecnologías es el ordenador y más concretamente el Internet. Lo que supone un salto de gran magnitud, cambiando y redefiniendo los modos de conocer y relacionarse del hombre.

Dentro del Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia, el tratamiento que se daba a la información hace años atrás correspondía a ser atendido de manera manual, esto debido a la poca demanda que existía para entonces de los servicios que presta. La demanda de estos servicios fue incrementando al pasar del tiempo debido a nuevas exigencias de los entes reguladores como la Alcaldía, los cuales exigen contar con certificaciones dentro de la construcción para ofrecer seguridad dentro de la misma. Esto conlleva a una desorganización total, la cual ocasiona pérdidas económicas, todo esto debido a que el sistema de atención del Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia no se encuentra preparado para un incremento excesivo de clientes. Las soluciones informáticas ayudan a la solución de este tipo de problemas ya que permiten

un reajuste de todo el sistema actual mejorando procesos que quedaron obsoletos. Es notorio que estas medidas tecnológicas se ajustan al tamaño de cada problema, logrando dar solución a distintas áreas de trabajo.

El Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia, tiene como finalidad prestar atención de servicio a toda obra estatal o privada, esto para ayudar a cumplir los estándares solicitados por los entes reguladores de todas las obras de construcción. Para dicho fin es necesario obtener estudios “in situ” dentro de cada obra, el cual nos permitirá abstraer los datos e información correctos para cada estudio a realizarse. Así mismo en la actualidad se realiza muestreos y dicha información se la toma en papeletas (formularios de papel) o toma una muestra del elemento para su ensayo cuando corresponda. Estos ensayos se realizan con instrumentos de alta precisión y cada medición se registra en papeletas que luego se ingresan a planillas Excel en donde se obtienen los cálculos necesarios para poder realizar el informe que certifique dicho elemento muestreado, para luego ser entregado al cliente. La confección del informe toma tiempo y corre riesgo de ser elaborada con errores y manipulada por otros funcionarios.

Para facilitar el registro y control de operaciones del Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia, surge la idea de realizar el desarrollo de un sistema web, con el fin de automatizar los registros y controles de la información respecto a cada operación, además ayudara a incrementar la productividad y eliminara la imprecisión causada por varias copias de planillas Excel inexactas, también se lograra centralizar la información en una base de datos de laboratorios realizados, para clientes que así lo

requieran a futuro el cual ayudara a obtener los reportes y las estadísticas cuando sea necesario.

El proyecto descrito, estará basado en la metodología RUP, esto debido a que es la que se adapta al planteamiento de dicho sistema y resolución de problemas, además que para la base de datos se usara MySQL, como lenguaje de programación PHP y APACHE como servidor, estas herramientas se eligen por medio de comparaciones para lograr las especificaciones más adecuadas.

1.2. Antecedentes

1.2.1. Antecedentes de la institución.

El Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia, es un laboratorio de Suelos. Este está encargado de evaluar y controlar la calidad, para la construcción de todo tipo de obras privadas o estatales. Se encuentra en funcionamiento desde enero del año 2012, pudiendo notarse sencillamente más de 8 años de experiencia el campo de la construcción, con innumerables estudios en obras civiles.

Cuenta al presente, con un cuerpo técnico completo y con amplia experiencia en la realización de ensayos de laboratorio, no solo por una sólida formación profesional, sino que también como resultado de muchos años de ejercicio permanente de esta actividad.

El Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia está ubicado en la Calle Resequin esq. Aspiazu # 2001, zona Sopocachi, La Paz. El parque de

equipos y maquinaria es renovado permanentemente esto debido al desgaste, por lo que, disponen de máquinas para la ejecución de cualquier obra civil en las mejores condiciones dentro de un rango de competitividad económica.

A continuación, se citan los principales clientes de la empresa:

- Gobiernos autónomos departamentales.
- Gobiernos autónomos municipales.
- Empresas privadas.
- Empresas sin fines de lucro.

1.2.2. Antecedentes afines al proyecto.

Las aplicaciones en el campo de la informática hoy en día son muy numerosas en la gestión empresarial, administrativa y comercial. Cabe mencionar que los laboratorios empresariales en el campo de la construcción han ido también adoptando un computador como ayuda dentro del desarrollo de sus actividades, es decir estas fueron adoptadas no solamente para registro de datos, sino que también para las actividades de producción, administrativa y comercialización, para enfrentarse a ser competitivo en cuanto al uso de la tecnología.

Sin embargo, el simple uso de un ordenador no cumple todas las necesidades, requerimientos y/o obligaciones que el Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia departamental la paz, requiere una solución informática. De esta

forma surge la idea de desarrollar un Sistema Web Para El Registro y Control De Operaciones del Laboratorio Técnico De Suelos.

En el transcurso de esta investigación, se pudo obtener y rescatar anteriores estudios nacionales e internacionales, relacionados con él tema como ser:

1.2.2.1. Internacionales.

“Sistematización de los procesos de construcción”, realizado por García Arenas (2003) en el “Instituto Pol. Nacional”, este sistema se realizó con el objetivo de organizar las construcciones y así lograr una mejora en cuanto a la eficiencia dentro las construcciones en general.

“Sistematización de los servicios externos para el laboratorio de ensayo de materiales de la universidad Ricardo Palma” realizado por Camayo Chavarría (2011) en la “Universidad Ricardo Palma”, la sistematización nace con el objetivo de encontrar una estrategia en la entrega de mejores resultados de los ensayos que realiza, de esta forma por el tal hecho recurrió a la tecnología informática para lograr una alta ventaja competitiva en el mercado.

“Software de física para experimentación e investigación en cienytec”, realizado por “cienytec Ltda.” (2015), este tipo de software está orientado a realizar de manera eficiente experimentos de laboratorio de forma segura y confiable.

“Diseño Del Sistema Documental Del Laboratorio De Suelos Y Materiales”, realizado por Valencia Salinas en la “Universidad de El Salvador”, este diseño se hizo con

el fin de conducir y operar una organización en forma exitosa para impulsar a la organización a ser más competitiva y garantizar el mejor desempeño al satisfacer y exceder las expectativas del cliente, como evidencia de su mejor competencia.

1.2.2.2. Nacionales.

“sistema presupuestario de obras y ejecución de materiales de construcción civil caso de estudio” (2015), fue realizado por Salas Jiménez en la “universidad salesiana de Bolivia”, lo que se pretende mostrar en el trabajo es que el control de ejecución presupuestaria de materiales de construcción juega un papel importante en las empresas constructoras, desarrollando un sistema para la elaboración del presupuesto de obra, controlar la ejecución presupuestaria de materiales de construcción para la empresa Plugins Computer Center.

Estos sistemas tratan de coadyuvar de alguna manera en el área de la administración de laboratorios común y también de construcción, no obstante, el siguiente proyecto de grado va a la integración con el sistema de control y seguimiento de procesos dentro el laboratorio.

1.3. Planteamiento del problema

1.3.1. Problemática.

En el Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia Departamental La Paz, para cada trabajo, se registra a detalle la información pertinente de los solicitantes y solicitudes, posteriormente se realiza los ensayos, luego en determinados tiempos se obtienen los datos de las muestras, realizadas de cada ensayo, se copian a planillas Excel, de los cuales se puede abstraer la información necesaria para dar una respuesta concreta sobre el caso de estudio. Como resultado de las operaciones realizadas se genera una biblioteca física la cual no cuenta con información que describa su existencia y a corto plazo se ve probables pérdidas de información.

El procesamiento de la información en estas operaciones está sujeta a errores por la redundancia de datos y el acceso a la misma por cualquiera de los técnicos, como resultado se obtiene información desordenada, no confiable, con errores de cálculo lo que ocasiona pérdida de tiempo en la atención a los clientes y a la vez afecta directamente en la obra por información inexacta, lo que puede ocasionar problemas a corto y largo plazo, como desniveles en el terreno, desprendimiento de material de taludes, etc. Y puede desencadenar pérdidas de vida, daños a vías de comunicación, etc. Además, también de ocasionar problemas legales.

1.3.2. Problema principal.

¿Cómo mejorar el proceso de registro y control de operaciones para el Laboratorio Técnico de Suelos, de la Sociedad de Ingenieros de Bolivia Departamental La Paz (SIB LA PAZ) que le permita realizar un trabajo eficiente y prestar un mejor servicio?

1.3.3. Problemas Secundarios.

- Se tiene deficiencias en la manipulación de datos, debido a la redundancia de los mismos, ya que esto es usado para la formalidad del Laboratorio Técnico de Suelos de la Sociedad de Ingenieros de Bolivia, lo cual genera excesivas planillas de control lo cual retarda el procesamiento de registros para la base de datos física.
- No se cuenta con una base de planillas Excel actualizadas lo cual hace que se tenga errores frecuentes de cargado de datos e información por usar distintas versiones de planillas. Esto tiene como consecuencias la mala elaboración de resultados y una inexacta administración de la información, y además genera cansancio en el personal.
- La actual manera de trabajar da acceso libre a todos los funcionarios, dando mucha independencia a niveles que no corresponde, lo cual genera planillas desordenadas o modificadas, y por lo tanto ocasiona más revisiones de lo correspondiente, por valores fuera de parámetro o valores irracionales.

- No se tiene actualmente una información organizada correctamente, lo cual genera demoras en los tramites que corresponden a la institución.

1.4. Objetivos

1.4.1. Objetivo General.

Desarrollar un Sistema Web para el registro y control de operaciones del laboratorio Técnico de suelos, de la Sociedad de Ingenieros de Bolivia S.I.B. La Paz, con el fin de optimizar las operaciones de servicios externos e internos, dentro del laboratorio y así obtener información confiable, oportuna y la correcta toma de decisiones.

1.4.2. Objetivos Específicos.

- Se diseñará un módulo de control de datos, para evitar la redundancia de los mismos, y se unificará las planillas de control de tal forma de frenar el retardo del procesamiento de registros en la base de datos esto ya que no existirá redundancias.
- Se elaborará un módulo de repositorio de planillas Excel esto para evitar la manipulación de fórmulas y ecuaciones, además se contará con las versiones se actualizadas. El cual mejorará la elaboración de resultados, y evitará el cansancio humano y se tendrá una correcta administración de los datos e información.

- Se mejorarán los niveles de seguridad para cada cargo específico de esta manera se evitará errores de cálculo posteriores.

1.5. Justificación

1.5.1. Técnica.

Es importante contar con un sistema de esta índole ya que la misma permitirá un mejor desempeño y aporte a la labor del personal administrativo, de esta manera se eliminará muchas tareas manuales, y se mejorará la productividad de manera eficaz y dinámica. Las mejoras se verán a corto plazo debido a que se podrá registrar todo el historial del laboratorio y a la par se podrán guardar los nuevos avances.

1.5.2. Económica.

Al contar con un sistema de estas características se utilizarán equipos ya existentes en el laboratorio, y también se pretende utilizar solo software libre para la programación, ya que este tipo de software no tiene costo. De esta forma también se evitarán consultas personales dentro del despacho del laboratorio ya que al sistema se podrá acceder mediante internet desde otros dispositivos. También el sistema planteado, reducirá los errores más comunes y así permitirá no mal gastar tiempo en la emisión de informes inexactos,

minimizará el papeleo común y maximizará el rendimiento, lo que significará evitar pérdidas económicas para el laboratorio.

1.5.3. Social.

Al contar con un sistema de esta índole, la calidad del servicio que esta brinde facilitará y mejorará las condiciones de trabajo realizadas por el personal administrativo, así también beneficiará con el acceso inmediato a la información necesaria a toda la población involucrada en la construcción, así mismo incidirá positivamente en la rápida atención de los servicios que se prestan, logrando de esta forma satisfacer al personal como también a sus clientes. Además de que el sistema puede emitir Informes del historial para cualquier percance que tuviese algún constructor.

1.6. Metodología

Un proyecto de software exitoso, es desarrollado de forma económica y puntual, y es resistente al cambio y la adaptación. El siguiente proyecto está basado en la ingeniería de software, que nos da lineamientos relacionado al proceso, herramientas y procedimientos. El proceso de desarrollo, mediante Proceso Unificado de Rational RUP (Rational Unified Process en inglés), es la metodología que mejor se desenvuelve, con las características de creación del sistema planteado.

RUP. –

El Proceso Unificado Racional (RUP, por las siglas de Rational Unified Process) es un ejemplo de un modelo de proceso moderno que se derivó del trabajo sobre el UML y el proceso asociado de desarrollo de software unificado. Conjunta elementos de todos los modelos de proceso genéricos, ilustra la buena práctica en especificación y diseño y apoya la creación de prototipos y entrega incremental.

El RUP reconoce que los modelos de proceso convencionales presentan una sola visión del proceso. En contraste, el RUP por lo general se describe desde tres perspectivas:

1. Una perspectiva dinámica que muestra las fases del modelo a través del tiempo.
2. Una perspectiva estática que presenta las actividades del proceso que se establecen.
3. Una perspectiva práctica que sugiere buenas prácticas a usar durante el proceso.

La mayoría de las descripciones del RUP buscan combinar las perspectivas estática y dinámica en un solo diagrama

(Kruchten, 2004, pág. 19).

1.7. Herramientas**1.7.1. Lenguaje de modelado unificado (UML).**

El Unified Modeling Language (UML) es un modelo para la construcción de software orientado a objetos que ha sido propuesto como estándar de ISO por el OMG. Consta de un conjunto de tipos de diagramas interrelacionados, dentro de los cuales se

utilizan elementos del modelo, que sirven para describir distintos aspectos de la estructura y la dinámica del software

(Campderrich, 2003, pág. 32).

El UML brinda la tecnología necesaria para apoyar la práctica de la ingeniería de software orientada a objetos, pero no da la estructura del proceso que guíe a los equipos del proyecto cuando aplican la tecnología. En los siguientes años, Jacobson, Rumbaugh y Booch desarrollaron el proceso unificado, estructura para la ingeniería de software orientado a objetos que utiliza UML. Actualmente, el proceso unificado (PU) y el UML se usan mucho en proyectos de toda clase orientados a objetos. El modelo iterativo e incremental propuesto por el PU puede y debe adaptarse para que satisfaga necesidades específicas del proyecto

(Pressman R. , 2010, pág. 46).

1.7.2. Elección de bases de datos.

Comparación de gestores de bases de datos “PostgreSQL” y “MySQL”, mediante cuadros esto con la finalidad de hallar el mejor gestor de bases de datos.

Tabla 1
Cuadro de ventajas y desventajas de PostgreSQL

Ventajas	Desventajas
<ul style="list-style-type: none"> • Ampliamente popular, Ideal para tecnologías Web. 	<ul style="list-style-type: none"> • Sin experticia, configurar llega a ser un caos.

<ul style="list-style-type: none"> • Fácil de Administrar. • Su sintaxis SQL es estándar y fácil de aprender. • Footprint bajo de memoria, bastante poderoso con una configuración adecuada. • Multiplataforma. • Capacidades de replicación de datos. • Soporte empresarial disponible. 	<ul style="list-style-type: none"> • Es fácil de vulnerar sin protección adecuada. • El motor MyISAM es instalado por defecto y carece de capacidades de integridad relacional. • innoDB genera mucho footprint al indizar. • El toolset empresarial tienen un costo adicional por suscripción anual. • Realizar revisiones llega a ser una labor manual tediosa para el DBA. • Reducir cantidad de tipos de datos.
--	---

(Universidad de Asuay, 2006)

Tabla 2
Cuadro de ventajas y desventajas de MySQL

Ventajas	Desventajas
<ul style="list-style-type: none"> • MySQL software es Open Source. • Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento. • Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una maquina con escasos recursos sin ningún problema. • Facilidad de configuración e instalación. • Soporta gran variedad de sistemas operativos. • Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está. 	<ul style="list-style-type: none"> • Un gran porcentaje de las utilidades de MySQL no están documentadas. • No es intuitivo, como otros programas (ACCESS).

-
- Su conectividad, velocidad, y seguridad hacen que MySQL Server altamente apropiado para acceder bases de datos en internet.
 - El software MySQL usa la licencia GPL.
-

(Universidad de Asuay, 2006)

Por las tablas comparativas mostradas anteriormente se eligió y/o prefirió usar MySQL como gestor de base de datos.

1.7.3. Elección de lenguaje de programación.

Comparación de los lenguajes de programación “ASP” y “PHP”, mediante cuadros esto con la finalidad de hallar el mejor lenguaje e implementarlo.

Tabla 3
Cuadro de ventajas y desventajas de ASP

Ventajas	Desventajas
<ul style="list-style-type: none"> • Cuenta con códigos prediseñados lo cual da una mayor facilidad a la hora de diseñar una página web. 	<ul style="list-style-type: none"> • La principal desventaja es que solo funciona en plataforma Windows. • El programa no lleva el control de las aplicaciones porque ya están prediseñadas. • Es de licencia propietaria y el costo es muy elevado.

(Universidad de Allcante, 2002)

Tabla 4
Cuadro de ventajas y desventajas de PHP

Ventajas	Desventajas
<ul style="list-style-type: none"> • La principal que veo en el uso de PHP es que puede funcionar en un servidor Windows y en Linux. • Además, tiene muchas características para su facilidad de manejo y aprendizaje ya que utiliza instrucciones sencillas que cualquier programador promedio puede entender, además tiene funciones incorporadas para manejadores de bases de datos como MySQL, POSTGRES y hasta ODBC. • Otra de las ventajas es que no tienen ningún costo y se pueden descargar fácilmente desde páginas de internet. • PHP ejecuta más rápido las operaciones matemáticas que ASP. • Capacidad de conexión con la mayoría de los manejadores de bases de datos. • Permite las técnicas de programación orientada a objetos. 	<ul style="list-style-type: none"> • Se dificulta más a la hora de programar ya que no cuenta con códigos prediseñados como los tiene ASP.

(Universidad de Allcante, 2002)

Teniendo en cuenta las ventajas de PHP, se observó que este dará mejor ventaja al sistema.

1.7.4. Cuadro comparativo de servidores web APACHE y IIS.

Tabla 5
Cuadro de ventajas y desventajas de APACHE

Ventajas	Desventajas
<ul style="list-style-type: none"> • Altamente configurable • Se desarrolla dentro del proyecto http. • Tiene amplia aceptación en la red. • Posee licencia freeware gracias a su amplio nivel de capacitación, su costo y su compatibilidad con los sistemas operativos. • Posee código abierto y es fácil de conseguirlo ayuda/soporte (es popular). 	<ul style="list-style-type: none"> • Esta incluye formatos de configuración no estándar. • No cuenta con una buena administración. • Falta de integración.

(Aula Mentor Gobierno de España Ministerio de Educación, Cultura y Deporte, 2012)

Tabla 6
Cuadro de ventajas y desventajas de IIS

Ventajas	Desventajas
<ul style="list-style-type: none"> • Es confiable, seguro y administrable en internet. • Proporciona capacidades de servidor web integrado. • Al momento de la instalación permite elegir sobre que servidor web va a correr (apache o IIS). • Desarrolla y es compatible con las aplicaciones beneficiándose con un único entorno de alojamiento de aplicaciones integrado con compatibilidad total. 	<ul style="list-style-type: none"> • Tiende a limitarse en las versiones que no son de la familia “server”. • Posee vulnerabilidades. • Este servidor no es multiplataforma, solo funciona bajo Windows.

(Aula Mentor Gobierno de España Ministerio de Educación, Cultura y Deporte, 2012)

Teniendo en cuenta las comparaciones realizadas anteriormente se vio que el servidor APACHE nos da mayores ventajas respecto a otros servidores, por eso se decidió trabajar con este servidor para el presente proyecto.

1.8. Límites y alcances

1.8.1. Límites.

El presente proyecto se limitará en la construcción del sistema planteado. Tomando en cuenta que este, estará presto a ser modificado si en un futuro así se propone.

El sistema no contemplará modelado de redes, ni protocolos que se manejan dentro de la red.

Se trabajará con un Interfaz amigable, que sea fácil de manejar para un usuario. También se desarrollará una base de datos necesaria para el registro que el sistema requiera.

1.8.2. Alcances.

Creación de módulos:

- Se realizará el análisis, diseño y creación de la base de datos para la administración de la información.
- Se hará la implementación de seguridad para restringir el acceso al sistema a usuarios ajenos a la institución.
- Se creará un módulo permita descargar las planillas exactas con versiones actualizadas.
- Se creará los módulos de impresión de solicitudes e informes.

1.9. Aportes

El aporte principal del presente proyecto, es proveer una herramienta de apoyo al laboratorio, que permita el registro y control de las operaciones, lo cual implica que la información será ordenada y precisa, se podrá abarcar más ensayos en un determinado tiempo y además de esto la información lograra estar correctamente guardada y respaldada, para su posterior uso, así mismo esta herramienta protegerá la información generada. Todo lo mencionado coadyuvara en los objetivos del laboratorio.

2. Marco teórico

2.1. Sistema

El termino sistema es universalmente usado. Hablamos sobre sistemas informáticos, sistemas operativos, sistemas de pago, el sistema educacional, el sistema de gobierno, etcétera. Estos son obviamente usos bastante diferentes de la palabra sistema, aunque coinciden en que, de algún modo, el sistema es más que simplemente la suma de sus partes (Sommerville, 2005, pág. 20).

En sentido amplio, un sistema es un conjunto de componentes que interaccionan entre sí para lograr un objetivo común'. Siguiendo esta propuesta, podemos decir que un sistema es la organización de partes interactuantes e interdependientes que se encuentran unidas y relacionadas para formar una célula compleja.

Cuando hablamos de sistemas nos referimos a un concepto que rebasa por mucho el campo de la informática, por lo que es necesario conocer el desarrollo de la teoría general de los sistemas (TGS), cuyos planteamientos han sido adoptados en muchos campos del conocimiento

(Dominguez, 2012, pág. 10).

2.2. Sistema De Información

Los sistemas de información se desarrollan para distintos fines, dependiendo de las necesidades de los usuarios humanos y la empresa. Los sistemas de procesamiento de transacciones (TPS) funcionan en el nivel operacional de la organización; los sistemas de automatización de oficinas (OAS) y los sistemas de trabajo de conocimiento (KWS) brindan soporte para el trabajo a nivel del conocimiento. Entre los sistemas de nivel superior se encuentran los sistemas de información administrativa (MIS) y los sistemas de soporte de decisiones (DSS)*. Los sistemas expertos aplican la experiencia de los encargados de tomar decisiones para resolver problemas específicos y estructurados. En el nivel estratégico de la administración se encuentran los sistemas de soporte para ejecutivos (ESS). Los sistemas de soporte de decisiones en grupo (GDSS) y los sistemas de trabajo colaborativo asistido por computadora (CSCWS), que se describen en forma más general, ayudan en el proceso de toma de decisiones, a nivel de grupo, de la variedad semiestructurada o no estructurada (Kendall & Kendall, 2011, pág. 2).

El objetivo principal de las empresas en el mundo es satisfacer las necesidades del cliente, y esta actividad es más eficiente gracias a las nuevas tecnologías y sistemas de la información. Actualmente los consumidores, requieren servicios rápidos y consistentes, esperando una atención personalizada. Bajo estas condiciones las tecnologías y sistemas de la información se conformarán cada vez más en un elemento estratégico dentro del esquema de muchos servicios

(Dominguez, 2012, pág. 10).

2.2.1. Actividades De Un Sistema De Información.

Un sistema de información ejecuta cuatro actividades básicas las cuales son: entrada, almacenamiento, procesamiento y salida de información.

Entrada de Información: Es el proceso mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfases automáticas.

Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de diskette, los códigos de barras, los escáner' s, la voz, los monitores sensibles al tacto, el teclado y el mouse, entre otras.

Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).

Procesamiento de Información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general de un año base.

Salida de Información: La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso, también existe una interfase automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interfase automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procesales de los clientes

(Laudon & Laudon, 2016, pág. 17).

2.2.2. Ciclo De Vida De Los Sistemas De Información.

El desarrollo de sistemas, un proceso formado por el análisis y el diseño, empieza cuando la administración o algunos miembros del personal encargado en desarrollar sistemas detectan un sistema de la empresa que necesita mejoras.

El ciclo de vida de un sistema está determinado por el conjunto de actividades que los analistas, diseñadores y beneficiarios realizan para el desarrollo e implementación de un sistema de información. El ciclo de vida de sistemas comprende seis fases:

- **Investigación preliminar.** La necesidad de recibir ayuda de un sistema de información puede surgir por diversas razones; sin importar cuales sean éstas, el proceso se inicia siempre con la petición de una persona.
- **Determinación de los requerimientos del sistema.** Lo fundamental del análisis de sistemas es comprender todas las fases importantes de la empresa que se encuentra bajo estudio. Los investigadores, al trabajar con los empleados y administradores, deben saber los procesos de una empresa para dar respuesta a las siguientes preguntas claves: ¿Qué es lo que hace?, ¿Cómo se hace?, ¿Con qué frecuencia se hace?, ¿Qué tan grande es el volumen de transacciones o decisiones?, ¿Cuál es el grado de eficiencia con el que se efectúa las tareas? y ¿Existe algún problema?

- **Diseño del sistema.** El diseño del sistema de información establece la forma en la que el sistema efectuará las obligaciones descritas durante la fase de análisis. Los técnicos en sistemas se refieren con frecuencia a esta etapa como el diseño lógico, en oposición al desarrollo del programa, el cual recibe el nombre de diseño físico.
- **Desarrollo del software.** Los encargados de desarrollar programas pueden instalar software comprado a terceros o escribir programas diseñados a la medida de la solicitud. La decisión depende del costo de cada alternativa, del tiempo disponible para escribir el programa y de la disponibilidad de los programadores.
- **Prueba del sistema.** Consiste en probar el sistema de manera experimental para comprobar si el software no tiene fallas, es decir, se trata de que el sistema llegue a funcionar de acuerdo con las especificaciones y en la forma en que los usuarios esperen que lo haga.
- **Implantación y evaluación.** La implantación es el proceso de instalar nuevo equipo, preparar a los usuarios para usar el sistema, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla. Cuando se han instalado, estas aplicaciones se emplean durante muchos años. Dado que las organizaciones y los usuarios cambian con el paso del tiempo, es necesario evaluar el sistema periódicamente

(Dominguez, 2012, pág. 57).

2.3. Gestión De Información

La finalidad de la Gestión de la información es ofrecer mecanismos que permitieran a la organización adquirir, producir y transmitir, al menor coste posible, datos e informaciones con una calidad, exactitud y actualidad suficientes para servir a los objetivos de la organización. En términos perfectamente entendibles sería conseguir la información adecuada, para la persona que lo necesita, en el momento que lo necesita, al mejor precio posible para toma la mejor de las decisiones. En el momento actual parece indiscutible que el éxito de la empresa no dependerá únicamente de cómo maneje sus activos materiales, sino también de la gestión de los recursos de información. La importancia de este recurso es tal que algunos autores estiman que las organizaciones deben ser consideradas como sistemas de información. Es frecuente confundir un sistema de información con la tecnología que lo soporta. Las Tecnologías de la información han supuesto una auténtica revolución en la capacidad de manejo de los recursos de información, permitiendo un rápido y eficiente proceso de adquisición, enriquecimiento y acceso a la misma, aunque nunca hay que olvidar que un Sistema de Gestión de Información va más allá de las propias herramientas utilizadas. El Sistema de Gestión de Información es el encargado de seleccionar, procesar y distribuir la información procedente de los ámbitos interno, externo y corporativo.

- **Información interna.** La producida en la actividad cotidiana de la institución.
- **Información externa.** La adquirida por la institución para disponer de información sobre los temas de su interés.
- **Información corporativa o pública.** La que la institución emite al exterior.

Las funciones de la Gestión Información abarcarían desde:

- Determinar las necesidades de información en correspondencia a sus funciones y actividades.
- Mejora de los canales de comunicación y acceso a la información.
- Mejora de los procesos informativos.
- Empleo eficiente de los recursos.

En este contexto, la información es considerada un recurso, un producto y un activo.

- La información como activo tiene un coste y debe tener un rendimiento.
- La información como producto deberá tener unas exigencias de calidad.
- La información como activo implica que la organización se preocupe por poseerla, gestionarla y utilizarla

(Arevalo, 2007, pág. 9).

2.4. Web

World Wide Web es un servicio proporcionado por Internet, que utiliza estándares aceptados en forma universal para almacenar, recuperar y mostrar información en un formato de página en Internet. Las páginas Web contienen gráficos, animaciones, sonidos y video, y están enlazadas con otras páginas Web. Al hacer clic en palabras resaltadas o botones en una página Web, usted puede enlazarse con las páginas relacionadas para encontrar información adicional y enlaces o vínculos hacia otras ubicaciones en Web. La Web puede servir como la base para los nuevos tipos de sistemas de información, como el sistema de rastreo de paquetes basado en Web de UPS (Laudon & Laudon, 2016, pág. 22).

2.5. Operaciones

Las operaciones definen el comportamiento de un objeto. Aunque existen muchos tipos distintos de operaciones, por lo general se dividen en cuatro categorías principales: 1) operaciones que manipulan datos en cierta manera (por ejemplo, los agregan, eliminan, editan, seleccionan, etc.), 2) operaciones que realizan un cálculo, 3) operaciones que preguntan sobre el estado de un objeto y 4) operaciones que vigilan un objeto en cuanto a la ocurrencia de un evento de control.

Estas funciones se llevan a cabo con operaciones sobre los atributos o sobre asociaciones de éstos. Por tanto, una operación debe tener “conocimiento” de la naturaleza de los atributos y de las asociaciones de la clase.

(Pressman R. , 2010, pág. 146).

2.6. Metodología de Desarrollo

2.6.1. El Proceso Unificado Racional.

El Proceso Unificado Racional (RUP, por las siglas de Rational Unified Process) (Krutchen, 2003) es un ejemplo de un modelo de proceso moderno que se derivó del trabajo sobre el UML y el proceso asociado de desarrollo de software unificado. Conjunta elementos de todos los modelos de proceso genéricos, ilustra la buena práctica en especificación y diseño y apoya la creación de prototipos y entrega incremental.

El RUP reconoce que los modelos de proceso convencionales presentan una sola visión del proceso. En contraste, el RUP por lo general se describe desde tres perspectivas:

1. Una perspectiva dinámica que muestra las fases del modelo a través del tiempo.
2. Una perspectiva estática que presenta las actividades del proceso que se establecen.
3. Una perspectiva práctica que sugiere buenas prácticas a usar durante el proceso.

La mayoría de las descripciones del RUP buscan combinar las perspectivas estática y dinámica en un solo diagrama

(Kruchten, 2004, pág. 19)

2.6.2. Características esenciales.

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales:

- proceso dirigido por los Casos de Uso.
- proceso centrado en la arquitectura.
- proceso iterativo e incremental.

2.6.2.1. Proceso dirigido por Casos de Uso.

Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar. Se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los Casos de Uso representan los requisitos funcionales del sistema.

En RUP los Casos de Uso no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba. Los Casos de Uso constituyen un elemento integrador y una guía del trabajo como se muestra en la Figura 1.

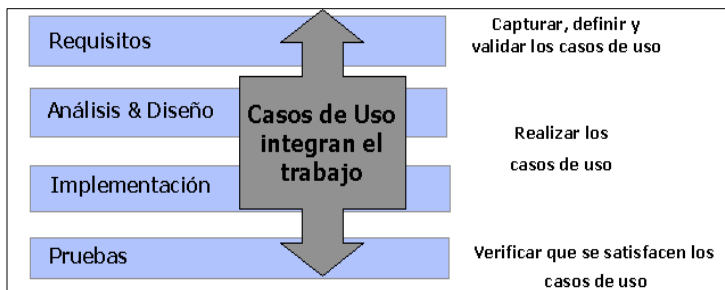


Figura 1 Los Casos de Uso integran el trabajo

(Kruchten, 2004)

Los Casos de Uso no sólo inician el proceso de desarrollo, sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

Como se muestra en la Figura 2, basándose en los Casos de Uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada Caso de Uso. Todos los modelos deben estar sincronizados con el modelo de Casos de Uso.

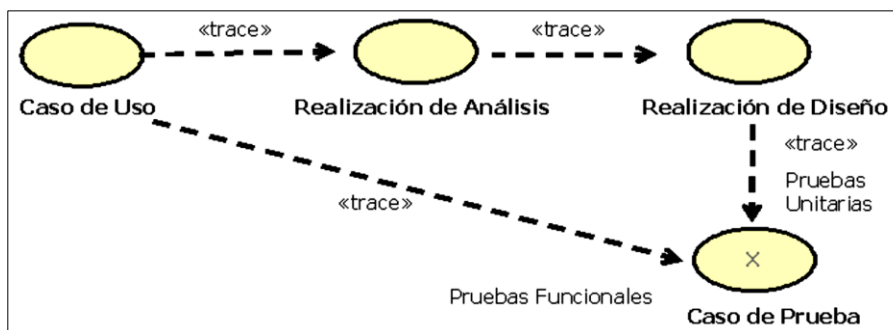


Figura 2 Trazabilidad a partir de los Casos de Uso

(Kruchten, 2004)

2.6.2.2. *Proceso centrado en la arquitectura.*

La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden. Además, la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo. La arquitectura se ve influenciada por la plataforma software, sistema operativo, gestor de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados. Muchas de estas restricciones constituyen requisitos no funcionales del sistema.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben

encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

(Kruchten, 2004, pág. 20).

Es conveniente ver el sistema desde diferentes perspectivas para comprender mejor el diseño por lo que la arquitectura se representa mediante varias vistas que se centran en aspectos concretos del sistema, abstrayéndose de los demás. Para RUP, todas las vistas juntas forman el llamado modelo 4+1 de la arquitectura, el cual recibe este nombre porque lo forman las vistas lógicas, de implementación, de proceso y de despliegue, más la de Casos de Uso que es la que da cohesión a todas.

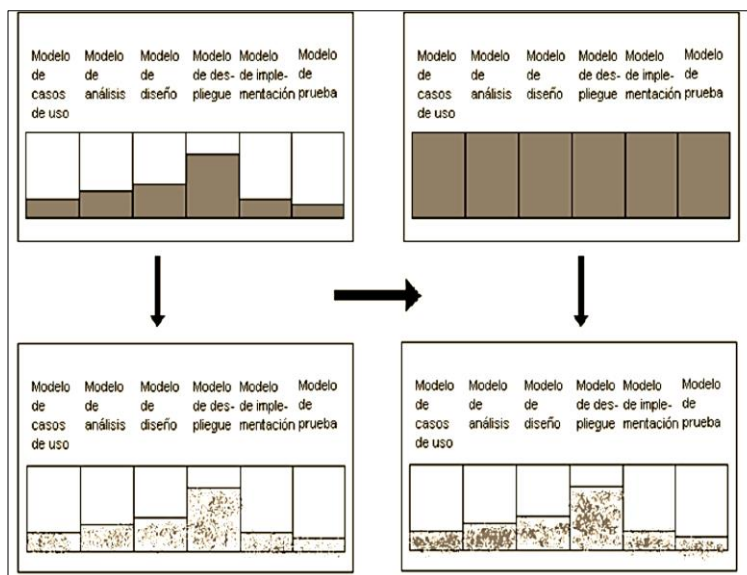


Figura 3 Los modelos se completan, la arquitectura no cambia drásticamente

(Kruchten, 2004)

Al final de la fase de elaboración se obtiene una *baseline*1 de la arquitectura donde fueron seleccionados una serie de Casos de Uso arquitectónicamente relevantes (aquellos que ayudan a mitigar los riesgos más importantes, aquellos que son los más importantes para el usuario y aquellos que cubran las funcionalidades significativas).

Como se observa en la Figura 4, durante la construcción los diversos modelos van desarrollándose hasta completarse (según se muestra con las formas rellenas en la esquina superior derecha). La descripción de la arquitectura, sin embargo, no debería cambiar significativamente (abajo a la derecha) debido a que la mayor parte de la arquitectura se decidió durante la elaboración. Se incorporan pocos cambios a la arquitectura (indicados con mayor densidad de puntos en la figura inferior derecha) (Kruchten, 2004, pág. 22).

2.6.2.3. Proceso iterativo e incremental.

El equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos

de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la Figura 5 Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores (Kruchten, 2004, pág. 23).

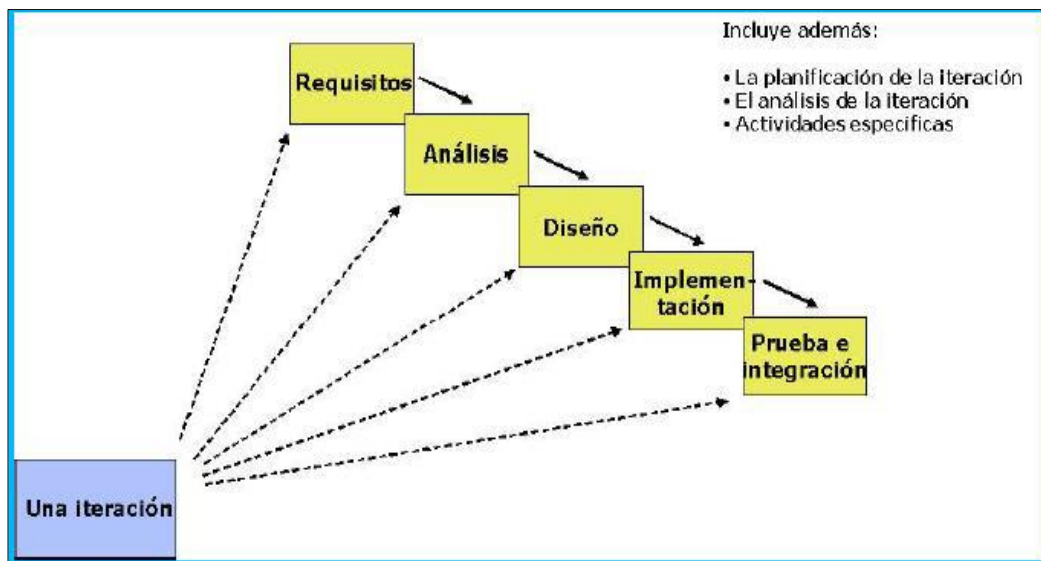


Figura 4 Una iteración RUP

(Kruchten, 2004)

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto

(Kruchten, 2004, pág. 24).

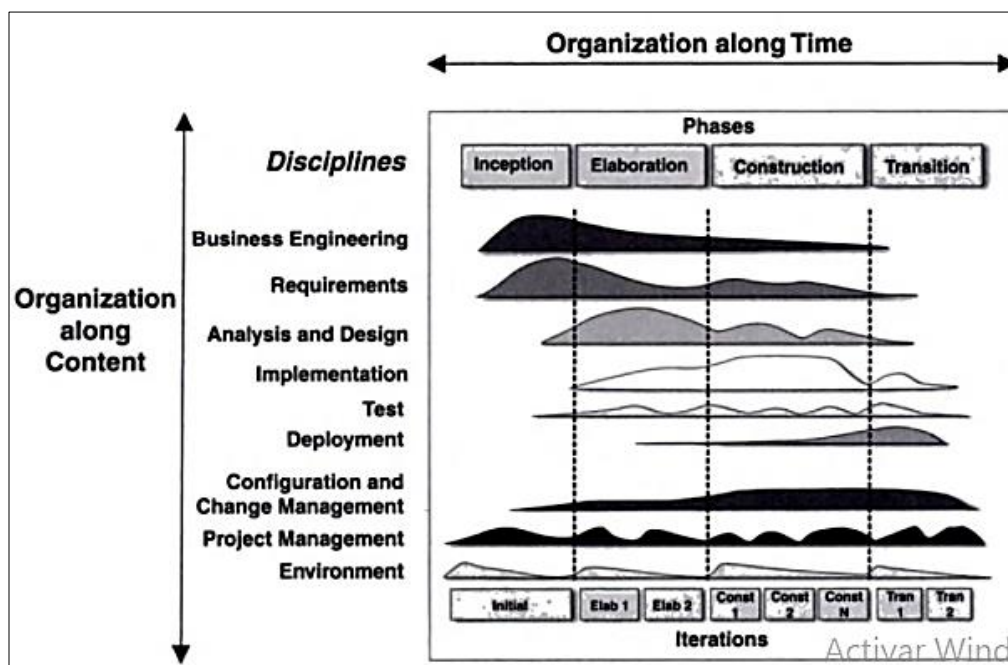


Figura 5 Esfuerzo en actividades según fase del proyecto

(Kruchten, 2004)

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. En la Figura 5 se muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una base line de la arquitectura.

Durante la fase de inicio las iteraciones hacen poner mayor énfasis en actividades modelado del negocio y de requisitos.

La fase de elaboración, las iteraciones se orientan al desarrollo de la base line de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la base line de la arquitectura.

En la fase de construcción, se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios. Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

La iteración con el RUP se apoya en dos formas. Cada fase puede presentarse en una forma iterativa, con los resultados desarrollados incrementalmente. Además, todo el conjunto de fases puede expresarse de manera incremental, como se muestra en la flecha en curva desde transición hasta concepción.

La visión estática del RUP se enfoca en las actividades que tienen lugar durante el proceso de desarrollo. Se les llama flujos de trabajo en la descripción RUP. En el proceso se identifican seis flujos de trabajo de proceso centrales y tres flujos de trabajo de apoyo centrales. El RUP se diseñó en conjunto con el UML, de manera que la descripción del flujo de trabajo se orienta sobre modelos UML asociados, como modelos de secuencia, modelos de objeto, etcétera.

La ventaja en la presentación de las visiones dinámica y estática radica en que las fases del proceso de desarrollo no están asociadas con flujos de trabajo específicos. En principio, al menos, todos los flujos de trabajo RUP pueden estar activos en la totalidad de las etapas del proceso. En las fases iniciales del proceso, es probable que se use mayor esfuerzo en los flujos de trabajo como modelado del negocio y requerimientos y, en fases posteriores, en las pruebas y el despliegue.

El enfoque práctico del RUP describe las buenas prácticas de ingeniería de software que se recomiendan para su uso en el desarrollo de sistemas. Las seis mejores prácticas fundamentales que se recomiendan son:

- **Desarrollo de software de manera iterativa:** Incrementar el plan del sistema con base en las prioridades del cliente, y desarrollar oportunamente las características del sistema de mayor prioridad en el proceso de desarrollo.

- **Gestión de requerimientos:** Documentar de manera explícita los requerimientos del cliente y seguir la huella de los cambios a dichos requerimientos. Analizar el efecto de los cambios sobre el sistema antes de aceptarlos.
- **Usar arquitecturas basadas en componentes:** Estructurar la arquitectura del sistema en componentes.
- **Software modelado visualmente:** Usar modelos UML gráficos para elaborar representaciones de software estáticas y dinámicas.
- **Verificar la calidad del software:** Garantizar que el software cumpla con los estándares de calidad de la organización.
- **Controlar los cambios de Software:** Gestione los cambios de software usando un sistema de gestión de cambios y procedimientos y herramientas de gestión de configuraciones.

El RUP no es un proceso adecuado para todos los tipos de desarrollo, por ejemplo, para desarrollo de software embebido. Sin embargo, sí representa un enfoque que potencialmente combina los tres modelos de proceso genéricos. Las innovaciones más importantes en el RUP son la separación de fases y flujos de trabajo, y el reconocimiento de que el despliegue del software en un entorno del usuario forma parte del proceso. Las fases son dinámicas y tienen metas. Los flujos de trabajo son estáticos y son actividades técnicas que no se asocian con una sola fase, sino que pueden usarse a lo largo del desarrollo para lograr las metas de cada fase (Sommerville, 2005, pág. 78).

2.6.3. Estructura del proceso

El proceso puede ser descrito en dos dimensiones o ejes:

- **Eje horizontal:** Representa el tiempo y es considerado el eje de los aspectos dinámicos del proceso. Indica las características del ciclo de vida del proceso expresado en términos de fases, iteraciones e hitos.
- **Eje vertical:** Representa los aspectos estáticos del proceso. Describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

2.6.4. Unified Modeling Language (UML).

El Unified Modeling Language (UML) es un modelo para la construcción de software orientado a objetos que ha sido propuesto como estándar de ISO por el OMG. Consta de un conjunto de tipos de diagramas interrelacionados, dentro de los cuales se utilizan elementos del modelo, que sirven para describir distintos aspectos de la estructura y la dinámica del software (Campderrich, 2003, pág. 32).

UML es el resultado de una cierta unificación de los modelos utilizados en tres métodos preexistentes de desarrollo de software orientado a objetos hechos por sus autores en colaboración. Estos métodos son los siguientes:

- El método de Grady Booch.

- El OMT, de Jim Rumbaugh y otros.
- El OOSE, de Ivar Jacobson.

2.6.4.1. Evolución del modelo UML.

Los primeros pasos hacia el modelo unificado se dieron en el año 1994, cuando Booch y Rumbaugh, trabajando en Rational Software Corporation, comenzaron la unificación de los modelos respectivos, y en octubre de 1995 se publicó la versión provisional 0.8 del entonces denominado Unified Method.

El mismo año, Jacobson se incorporó con su empresa al equipo mencionado y a Rational y, como resultado del trabajo de los tres autores, en 1996 salieron las versiones 0.9 y 0.91 del UML. El OMG emitió en aquella época una Request For Proposal, para un modelo de este tipo, y entonces Rational, para responderle, constituyó un consorcio con otras organizaciones, con el resultado de que en enero de 1997 se presentó en la OMG la versión 1.0 del UML.

Otras empresas que habían presentado también respuestas de manera independiente se añadieron al consorcio y se publicó la versión 1.1, que fue aceptada por el OMG en noviembre de 1997 (hubo otra propuesta, la del modelo OML, que tenía y todavía tiene un número importante de partidarios). El OMG encargó una revisión, cuyo resultado fue una versión 1.2, no publicada, y la versión 1.3, ya publicada como estándar. La versión 1.4 se publicó oficialmente en septiembre de 2001.

Con el UML se ha llegado a un modelo orientado a objetos único como modelo oficial, pero eso no quiere decir que se haya alcanzado un método único de desarrollo orientado a objetos; la verdad es que por el momento parece que falta bastante para llegar al mismo, si es que alguna vez se consigue (Campderrich, 2003, pág. 33).

2.6.5. Diagramas del UML.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos.

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

A continuación, se describirán brevemente los diagramas:

2.6.5.1. Diagramas de caso de uso.

Que exponen las interacciones entre un sistema y su entorno. Los diagramas de casos de uso son el equivalente del arte rupestre moderno. Los símbolos principales de un

caso de uso son el actor y el óvalo del caso de uso figura 6. Los diagramas de casos de uso son responsables principalmente de documentar los macro requisitos del sistema. Piense en los diagramas de casos de uso como la lista de las capacidades que debe proporcionar el sistema

(Kimmel, 2008, pág. 7).

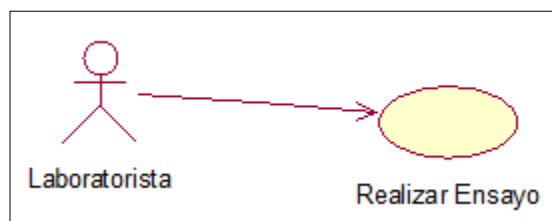


Figura 6 Caso de uso "Realizar Ensayo"

(Elaboración Propia)

2.6.5.2. Diagrama de actividades.

Que muestran las actividades incluidas en un proceso o en el procesamiento de datos. Un diagrama de actividades es la versión UML de un diagrama de flujo. Los diagramas de actividades se usan para analizar los procesos y si es necesario, volver a realizar la ingeniería de los procesos como se muestra en la figura 7.

Un diagrama de actividades es una herramienta excelente para analizar problemas que, al final, el sistema deberá resolver. Como una herramienta de análisis, no queremos empezar resolviendo el problema en un nivel técnico mediante la asignación de clases, pero

podemos usar los diagramas de actividades para entender el problema e incluso refinar los procesos que comprenden el problema

(Kimmel, 2008, pág. 8).

Un diagrama de actividad UML muestra el comportamiento dinámico de un sistema o de parte de un sistema a través del flujo de control entre acciones que realiza el sistema. Es similar a un diagrama de flujo, excepto porque un diagrama de actividad puede mostrar flujos concurrentes.

El componente principal de un diagrama de actividad es un nodo acción, representado mediante un rectángulo redondeado, que corresponde a una tarea realizada por el sistema de software. Las flechas desde un nodo acción hasta otro indican el flujo de control; es decir, una flecha entre dos nodos acción significa que, después de completar la primera acción, comienza la segunda acción. Un punto negro sólido forma el nodo inicial que indica el punto de inicio de la actividad. Un punto negro rodeado por un círculo negro es el nodo final que indica el fin de la actividad.

Un tenedor (fork) representa la separación de actividades en dos o más actividades concurrentes. Se dibuja como una barra negra horizontal con una flecha apuntando hacia ella y dos o más flechas apuntando en sentido opuesto. Cada flecha continua representa un flujo de control que puede ejecutarse de manera concurrente con los flujos correspondientes a las otras flechas continuas. Dichas actividades concurrentes pueden realizarse en una computadora, usando diferentes hebras o incluso diferentes computadoras (Pressman R. , 2010, pág. 735).

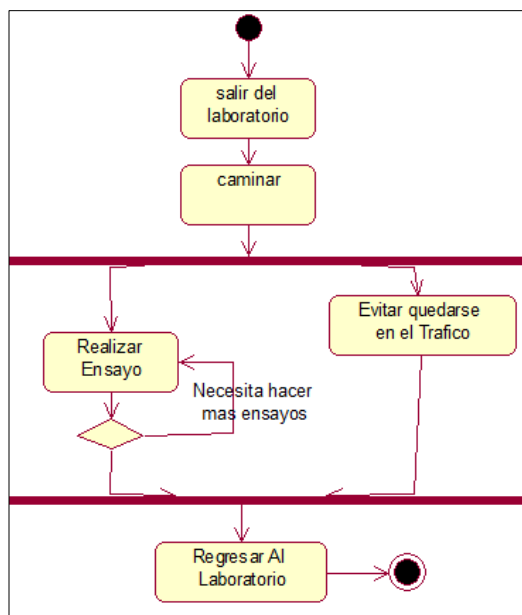


Figura 7 Diagrama de actividades, que muestra la manera en que el personal del laboratorio hace sus ensayos

(Elaboración Propia)

2.6.5.3. Diagramas de clase.

Que revelan las clases de objeto en el sistema y las asociaciones entre estas clases.

Los diagramas de clases se usan para mostrar las clases de un sistema y las relaciones entre ellas figura 8. Una sola clase puede mostrarse en más de un diagrama de clases y no es necesario mostrar todas las clases en un solo diagrama monolítico de clases. El mayor valor es mostrar las clases y sus relaciones desde varias perspectivas, de una manera que ayudará a transmitir la comprensión más útil.

Los diagramas de clases muestran una vista estática del sistema; no describen los comportamientos o cómo interactúan los ejemplos de las clases. Para describir los

comportamientos y las interacciones entre los objetos de un sistema, podemos revisar los diagramas de interacción

(Kimmel, 2008, pág. 102).

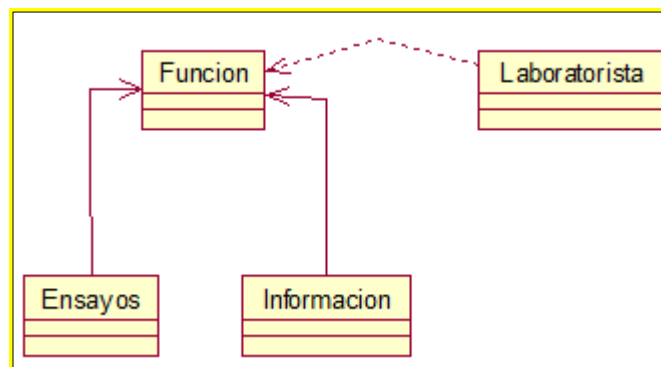


Figura 8 Diagrama de clases

(Elaboración Propia)

2.6.5.4. Diagramas de interacción.

Existen dos tipos de diagramas de interacción: la secuencia y la colaboración. Ambos transmiten la misma información, empleando una perspectiva un poco diferente. Los diagramas de secuencia muestran las clases a lo largo de la parte superior y los mensajes enviados entre esas clases, modelando un solo flujo a través de los objetos del sistema.

a. **Los diagramas de colaboración**

Usan las mismas clases y mensajes, pero organizados en una disposición espacial.

La figura 9 muestra un ejemplo sencillo de diagrama de secuencia.

b. **Diagramas de secuencias**

Muestran las interacciones entre los actores y el sistema, y entre los componentes del sistema, diagrama de secuencia implica un ordenamiento en el tiempo al seguir la secuencia de mensajes desde arriba a la izquierda hasta abajo a la derecha. Debido a que en el diagrama de colaboración no se indica en forma visual un ordenamiento en el tiempo, numeramos los mensajes para indicar el orden en el cual se presentan. Algunas herramientas convertirán de manera automática los diagramas de interacción entre secuencia y colaboración, pero no es necesario crear los dos tipos de diagramas. En general, se percibe que un diagrama de secuencia es más fácil de leer y más común (Kimmel, 2008, pág. 82).

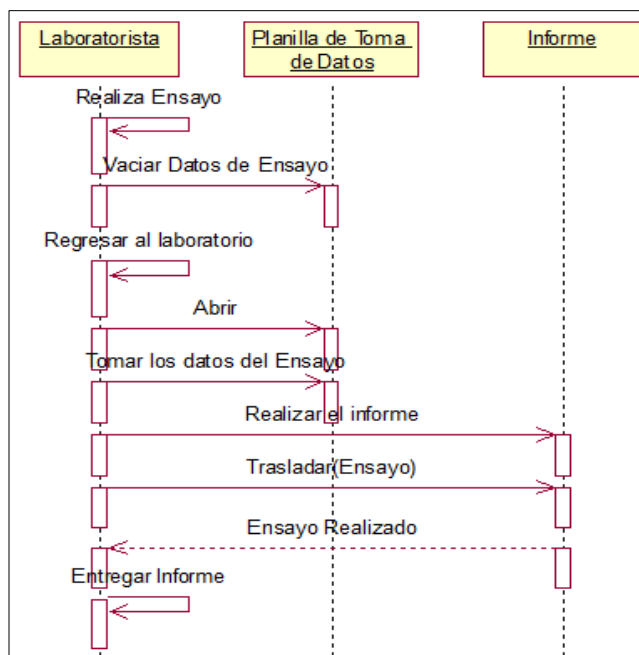


Figura 9 Diagrama de clases

(Elaboración Propia)

2.6.6. Las restricciones de las operaciones: la programación por contrato.

En el UML hay tres tipos de restricciones relativas a las operaciones: precondiciones, postcondiciones e invariantes.

- Las precondiciones son restricciones que se deben cumplir antes de ejecutar una operación. Su cumplimiento nos garantiza que la operación se ejecuta partiendo de un estado correcto del sistema.

- Las postcondiciones se comprueban al acabar la ejecución de una operación, y garantizan que cuando esté terminada la operación, el sistema vuelva a situarse en un estado correcto.
- Las invariantes son condiciones que se deben cumplir en todo momento. Se tienen que comprobar al inicio de cualquier operación –excepto los constructores y al acabar.

Las restricciones pueden servir para diseñar con vistas a hacer programación por contrato, que se basa en unas condiciones sobre operaciones y objetos denominadas aserciones, las cuales se pueden expresar en forma de restricciones del UML (Campderrich, 2003, pág. 68).

2.7. Tecnologías de software

Las herramientas CASE son software de apoyo al desarrollo, mantenimiento y documentación informatizados de software.

Quedan, pues, principalmente las herramientas que ayudan a aplicar técnicas concretas de desarrollo y mantenimiento de software y por eso gestionan información sobre los elementos y conceptos que se utilizan en los métodos de desarrollo.

El gestor de base de datos que utilizaremos para desarrollar el sistema será:

2.7.1. MySQL.

MySQL es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. MySQL compite con sistemas RDBMS propietarios conocidos, como Oracle, SQL Server y DB2. MySQL incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger y hacer volcados de datos. Puede desarrollar sus propias aplicaciones de base de datos en la mayor parte de los lenguajes de programación utilizados en la actualidad y ejecutarlos en casi todos los sistemas operativos, incluyendo algunos de los que probablemente no ha oído nunca hablar. MySQL utiliza el lenguaje de consulta estructurado (SQL). Se trata del lenguaje utilizado por todas las bases de relacionales. Este lenguaje permite crear bases de datos, así como agregar, manipular y recuperar datos en función de criterios específicos (Gilfillan, 2003, pág. 39).

2.7.2. PHP.

Tres letras que juntas constituyen el nombre de uno de los lenguajes de programación más populares para el desarrollo de Web, el Preprocesador de Hipertexto PHP. Y mientras tal vez sonrías por lo insulso y reiterativo del acrónimo, te diré que las

estadísticas indican que PHP no debe tomarse a la ligera: actualmente este lenguaje se utiliza en más de 20 millones de sitios Web y en más de un tercio de los servidores Web en todo el mundo; no es algo despreciable, especialmente cuando se considera que el lenguaje ha sido desarrollado por completo por una comunidad de voluntarios repartida en todo el mundo y está disponible en Internet ¡sin costo alguno!

Durante los últimos años, PHP se ha convertido, de facto, en la opción para el desarrollo de aplicaciones Web orientadas a bases de datos, sobre todo por su escalabilidad, facilidad, uso y el amplio soporte para diferentes bases de datos y formatos de éstos. Este primer capítulo te presentará una introducción amigable al mundo de PHP con un recorrido relámpago por su historia y sus características, y luego te guiará por la escritura y ejecución de tu primer programa PHP

(Vaswani, 2010, pág. 4).

2.7.2.1. Características únicas.

- **Rendimiento** Los scripts escritos en PHP se ejecutan más rápido que los escritos en otros lenguajes de creación de scripts; numerosos estudios comparativos independientes ponen este lenguaje por encima de sus competidores como JSP, ASP.NET y Perl. El motor de PHP 5.0 fue completamente rediseñado con un manejo óptimo de memoria para mejorar su rendimiento y es claramente más veloz que las versiones previas. Además, están disponibles aceleradores de terceros que pueden mejorar aún más el rendimiento y el tiempo de respuesta.

- **Portabilidad** PHP está disponible para UNIX, Microsoft Windows, Mac OS y OS/2 y los programas escritos en PHP se pueden transportar de una plataforma a otra. Como resultado, las aplicaciones PHP desarrolladas en Windows, por ejemplo, se ejecutarán en UNIX sin grandes contratiempos. Esta capacidad de desarrollar fácilmente para múltiples plataformas es muy valiosa, en especial cuando se trabaja en un ambiente corporativo de varias plataformas o cuando se intenta atacar diversos sectores del mercado.
- **Fácil de usar** “La sencillez es la mayor sofisticación”, dijo Leonardo da Vinci y, de acuerdo con ello, PHP es un lenguaje de programación extremadamente sofisticado. Su sintaxis es clara y consistente y viene con una documentación exhaustiva para las más de 5 000 funciones incluidas en la distribución principal. Esto reduce de manera importante la curva de aprendizaje tanto para los desarrolladores novatos como para los expertos, y es una de las razones por las que PHP es favorecido como una herramienta rápida para la creación de prototipos que permitan el desarrollo de aplicaciones basadas en Web.
- **Código libre** PHP es un proyecto de código libre; el lenguaje es desarrollado por un grupo de programadores voluntarios distribuidos por todo el mundo, quienes ponen a disposición gratuita el código fuente a través de Internet, y puede ser utilizado sin costo, sin pagos por licencia y sin necesidad de grandes inversiones en equipo de cómputo ni programas. Con ello se reduce el costo del desarrollo de programas sin afectar la flexibilidad ni la confiabilidad de los productos. La naturaleza del código libre implica que cualquier desarrollador,

donde quiera que se encuentre, puede inspeccionar el árbol de código, detectar errores y sugerir posibles correcciones; con esto se produce un producto estable y robusto, en que las fallas, una vez descubiertas, se corrigen rápidamente, en algunas ocasiones, ¡horas después de ser descubiertas!

- **Soporte comunitario** Una de las mejores características de los lenguajes a los que da soporte una comunidad, como PHP, es el acceso que ofrece a la creatividad e imaginación de cientos de desarrolladores ubicados en diferentes partes del mundo. En la comunidad PHP, los frutos de esta creatividad pueden ser encontrados en PEAR (PHP Extension and Application Repository), el repositorio de extensiones y aplicaciones de PHP (<http://pear.php.net>), y en PECL (PHP Extension Community Library), la biblioteca de la comunidad de extensiones PHP, que contienen cientos de soluciones y extensiones que los desarrolladores pueden ocupar para añadir sin esfuerzo nuevas funcionalidades a sus aplicaciones PHP. Utilizar estas soluciones suele ser una mejor opción en tiempo y costo, en vez de desarrollar desde cero tu propio código.
- **Soporte a aplicaciones de terceros** Una de las fortalezas históricas de PHP ha sido su soporte a una amplia gama de diferentes bases de datos, entre las cuales se incluyen MySQL, PostgreSQL, Oracle y Microsoft SQL Server. PHP 5.3 soporta más de quince diferentes motores de bases de datos, e incluye una API (interfaz de programación de aplicaciones) común para el acceso a base de datos. El soporte para XML facilita la lectura (y escritura) de documentos XML como si fueran estructuras de datos nativas de PHP; es posible acceder a colecciones de nodos

XML utilizando XPath y transformar código XML en otros formatos con las hojas de estilo XSLT. Y no termina aquí. La arquitectura extensible de PHP permite que los desarrolladores escriban sus propias adiciones personalizadas al lenguaje, de manera que hoy en día los desarrolladores de PHP pueden hacer que sus aplicaciones lean y registren imágenes en formato GIF, JPEG y PNG; enviar y recibir correos electrónicos utilizando protocolos SMTP, IMAP y POP3; colaborar con servicios Web utilizando protocolos SOAP y REST; validar datos de entrada utilizando expresiones regulares de Perl, además de crear y manipular documentos PDF. Más aún, PHP puede acceder a las bibliotecas de C, las clases de Java y los objetos COM

(Vaswani, 2010, pág. 4).

2.7.3. Framework Bootstrap

Bootstrap es uno de los frameworks más populares y utilizados del mercado para la creación de páginas responsive, habiendo sido desarrollado por el equipo de Twitter. Entre los navegadores soportados se encuentran Chrome, Firefox, Opera, Safari e Internet Explorer a partir de la versión 8 (aunque en la versión 7 también funciona correctamente). Está preparado para funcionar tanto en navegadores de PCs y portátiles con cualquier tamaño de pantalla, así como para tablets y smartphones de tamaños mucho más reducidos. Para conseguir que una misma web se pueda visualizar correctamente en todos esos tamaños de pantalla ha diseñado un avanzado sistema de rejilla dividido en columnas para

el posicionamiento de los elementos de nuestra web. Además, incorpora otras muchas utilidades y complementos (formularios, botones, barras de navegación, etc.) para simplificar el desarrollo de una web responsive (Gallego, 2018, pág. 8).

2.7.4. Framework CakePHP.

CakePHP es un framework libre, de código abierto, para el desarrollo rápido de aplicaciones para PHP. Es una estructura fundamental para la ayudar a los programadores a crear aplicaciones web. Nuestro objetivo principal es permitirte trabajar de forma estructurada y rápida y sin pérdida de flexibilidad. CakePHP pone a tu disposición todas las herramientas que necesita para empezar a programar lo que realmente hay que hacer: la lógica específica de tu aplicación. En lugar de reinventar la rueda cada vez que te sientas a hacer un nuevo proyecto, obtén una copia de CakePHP y empieza con el verdadero corazón de tu aplicación. CakePHP tiene un equipo de desarrollo activo y una comunidad muy viva, lo que le da un gran valor al proyecto. Además de no tener que reinventar la rueda, usar CakePHP significa que el núcleo de la aplicación estará bien probado y está siendo constantemente mejorado.

He aquí una lista rápida de las características que disfrutarás al utilizar CakePHP:

- Licencia flexible.
- Compatible con las versiones de PHP 5.2.6 y superiores.
- Contiene CRUD para la interacción de la base de datos.

- Andamiaje de código.
- Generación automática de código.
- Arquitectura MVC URLs personalizadas Función de Validación.
- Plantillas rápidas y flexibles (La sintaxis de PHP, con ayudantes).
- Ayudantes para AJAX, JavaScript, formularios HTML y más.
- Componentes de Email, Cookie, Seguridad, Sesión y otros.
- ACL flexible.
- Sanitización de Datos.
- Poderoso Caché.
- Localización e Internacionalización.
- Funciona desde cualquier directorio de sitios web, con poca o ninguna configuración adicional

(Cake Software Foundation, 2020).

2.7.4.1. MVC: Modelo - Vista – Controlador.

CakePHP sigue el patrón diseño de software llamado MVC22. Programar usando MVC separa tu aplicación en tres partes principalmente:

- **La Capa del Modelo**, el modelo representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos convirtiéndolos en conceptos significativos para la

aplicación, así como su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos. A primera vista los objetos del modelo pueden ser considerados como la primera capa de la interacción con cualquier, base de datos que podría estar utilizando tu aplicación. Pero en general representan los principales conceptos en torno a los cuales se desea implementar un programa. En el caso de una red social, la capa de modelo se haría cargo de tareas tales como guardar datos del usuario, almacenamiento de asociaciones con amigos, el almacenamiento y la recuperación de fotos de los usuarios, encontrar sugerencias de nuevos amigos, etc. Mientras que los objetos del modelo pueden ser considerados como «Amigo», «Usuario», «Comentario» y «Foto».

- **La capa de la Vista**, la vista hace una presentación de los datos del modelo estando separada de los objetos del modelo. Es responsable del uso de la información de la cual dispone para producir cualquier interfaz de presentación de cualquier petición que se presente. Por ejemplo, como la capa de modelo devuelve un conjunto de datos, la vista los usaría para hacer una página HTML que los contenga. O un resultado con formato XML para que otras aplicaciones puedan consumir. La capa de la Vista no se limita únicamente a HTML o texto que represente los datos, sino que puede ser utilizada para ofrecer una amplia variedad de formatos en función de sus necesidades tales como videos, música, documentos y cualquier otro formato que puedas imaginar.
- **La capa del Controlador**, La capa del controlador gestiona las peticiones de los usuarios. Es responsable de responder la información solicitada con la ayuda

tanto del modelo como de la vista. Los controladores pueden ser vistos como administradores cuidando de que todos los recursos necesarios para completar una tarea se deleguen a los trabajadores más adecuados. Espera peticiones de los clientes, comprueba su validez de acuerdo a las normas de autenticación o autorización, delega la búsqueda de datos al modelo y selecciona el tipo de respuesta más adecuado según las preferencias del cliente. Finalmente delega este proceso de presentación a la capa de la Vista

(Cake Software Foundation, 2020).

2.8. Métricas de calidad

Las métricas de calidad del software, como las métricas de productividad, se enfocan en el proceso, el proyecto y el producto. Al desarrollar y analizar una línea de referencia de métricas para la calidad, una organización puede corregir aquellas áreas del proceso de software que sean la causa de los defectos de software.

La medición da como resultado un cambio cultural. La recopilación de datos, cálculo de métricas y análisis de métricas son los tres pasos que deben implementarse para comenzar un programa de métricas. En general, un enfoque dirigido a metas ayuda a una organización a enfocarse en las métricas correctas para su empresa. Al crear una línea de referencia de métricas, una base de datos que contenga mediciones de proceso y producto, los ingenieros de software y sus gerentes pueden obtener mejor comprensión del trabajo que hacen y del producto que elaboran.

Hasta los desarrolladores de software exhaustivos están de acuerdo en que es importante crear software de alta calidad. Pero, ¿Cómo se define la calidad? En el sentido más amplio, calidad del software es el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente (Pressman R. , 2010, pág. 590).

La calidad del software se define también, en términos de ausencia de errores en el funcionamiento del sistema. El ajuste a las necesidades del usuario, el sistema debe ser flexible u susceptible a modificaciones que se puedan realizar de manera rápida y oportuna. El sistema debe alcanzar un desempeño apropiado en términos de tiempo, volumen y espacio. Un sistema debe cumplir de la mejor forma los estándares internacionales establecidos, en lo que a la calidad de software se refiere a:

2.8.1. NORMA ISO 9126.

Esta norma Internacional fue publicada en 1992, la cual es usada para la evaluación de la calidad de software, llamado “Information technology – Software product evaluation - Quality characteristics and guidelines for their use”; o también conocido como ISO 9126 (o ISO/IEC 9126).

El estándar ISO 9126 se desarrolló con la intención de identificar los atributos clave del software de cómputo. Este estándar describe 6 características generales, y son definidas transcribiéndolas de su fuente original así:

- **Funcionalidad** En este grupo se conjunta una serie de atributos que permiten calificar si un producto de software maneja en forma adecuada el conjunto de funciones que satisfagan las necesidades para las cuales fue diseñado. Para este propósito se establecen los siguientes atributos:
 - **Adecuación.** Se enfoca a evaluar si el software cuenta con un conjunto de funciones apropiadas para efectuar las tareas que fueron especificadas en su definición.
 - **Exactitud.** Este atributo permite evaluar si el software presenta resultados o efectos acordes a las necesidades para las cuales fue creado.
 - **Interoperabilidad.** Permite evaluar la habilidad del software de interactuar con otros sistemas previamente especificados.
 - **Conformidad.** Evalúa si el software se adhiere a estándares, convenciones o regulaciones en leyes y prescripciones similares.
 - **Seguridad.** Se refiere a la habilidad de prevenir el acceso no autorizado, ya sea accidental o premeditado, a los programas y datos.
- **Confiablez** Aquí se agrupan un conjunto de atributos que se refieren a la capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido. Las sub características que el estándar sugiere son:

- Nivel de Madurez. Permite medir la frecuencia de falla por errores en el software.
- Tolerancia a fallas. Se refiere a la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software o de cometer infracciones de su interfaz específica.
- Recuperación. Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla, así como al tiempo y el esfuerzo necesarios para lograrlo.
- **Usabilidad** Consiste de un conjunto de atributos que permiten evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema.
 - Comprensibilidad. Se refiere al esfuerzo requerido por los usuarios para reconocer la estructura lógica del sistema y los conceptos relativos a la aplicación del software.
 - Facilidad de Aprender. Establece atributos del software relativos al esfuerzo que los usuarios deben hacer para aprender a usar la aplicación.
 - Operabilidad. Agrupa los conceptos que evalúan la operación y el control del sistema.
- **Eficiencia** Esta característica permite evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados. Los aspectos a evaluar son:
 - Comportamiento con respecto al Tiempo. Atributos del software relativos a los tiempos de respuesta y de procesamiento de los datos.

- Comportamiento con respecto a Recursos. Atributos del software relativos a la cantidad de recursos usados y la duración de su uso en la realización de sus funciones.
- **Mantenibilidad** Se refiere a los atributos que permiten medir el esfuerzo necesario para realizar modificaciones al software, ya sea por la corrección de errores o por el incremento de funcionalidad. En este caso, se tienen los siguientes factores:
 - Capacidad de análisis. Relativo al esfuerzo necesario para diagnosticar las deficiencias o causas de fallas, o para identificar las partes que deberán ser modificadas.
 - Capacidad de modificación. Mide el esfuerzo necesario para modificar aspectos del software, remover fallas o adaptar el software para que funcione en un ambiente diferente.
 - Estabilidad. Permite evaluar los riesgos de efectos inesperados debidos a las modificaciones realizadas al software.
 - Facilidad de Prueba. Se refiere al esfuerzo necesario para validar el software una vez que fue modificado.
- **Portatibilidad** En este caso, se refiere a la habilidad del software de ser transferido de un ambiente a otro, y considera los siguientes aspectos:
 - Adaptabilidad. Evalúa la oportunidad para adaptar el software a diferentes ambientes sin necesidad de aplicarle modificaciones.

- Facilidad de Instalación. Es el esfuerzo necesario para instalar el software en un ambiente determinado.
- Conformidad. Permite evaluar si el software se adhiere a estándares o convenciones relativas a portabilidad.
- Capacidad de reemplazo. Se refiere a la oportunidad y el esfuerzo usado en sustituir el software por otro producto con funciones similares.

El mundo globalizado exige cada vez más la aplicación de estándares internacionales que garanticen la calidad de los productos.

Por esta razón, es necesario que todo aquel que se dedica al desarrollo de software incluya en sus procesos, estándares de calidad que permitan certificarse en alguno de los modelos.

Aquí se ha presentado un estándar, el ISO-9126, el cual establece una guía para la evaluación de la calidad del software, sin embargo, es necesario que cada empresa dedicada a producir software trabaje en establecer su modelo de calidad que le permita valorar el nivel de excelencia de sus productos, en el que deberán incluirse instrumentos de medición que permitan calificar cuantitativamente cada una de las características aquí presentadas. Es importante mencionar, que dependiendo de los distintos tipos de aplicaciones las métricas podrán variar, ya que, aunque las características expuestas son comunes a la totalidad de los productos, cada software particular requiere una evaluación específica

(Abud, 2012).

2.9. Costos y/o beneficios

2.9.1. COCOMO.

El modelo COCOMO es un modelo algorítmico, el cual ayuda a dar valoraciones del software. El cual se divide en tres jerarquías.

El primer nivel, COCOMO básico es escaso de factores debido a la temprana solicitud del mismo debido a lo cual, pero su exactitud debe limitarse a su carencia de factores que expliquen la diferencia en las calidades del proyecto (Conductores del costo). COCOMO intermedio toma estos conductores del costo en consideración y COCOMO detallado explica además la influencia de las fases del proyecto individual.

- **COCOMO básico.** Realiza el cálculo del esfuerzo y el costo del desarrollo en función del tamaño del programa estimado en LDC, por lo cual no es tan exacto debido a la escasez de factores.
- **COCOMO intermedio.** Realiza prácticamente una estimación de COCOMO básico, y además un conjunto de conductores de costo que incluyen la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.
- **COCOMO detallado.** Reúne las peculiaridades de la versión intermedia de COCOMO y lleva a cabo una evaluación del impacto de los conductores de costo en cada fase (análisis, desarrollo, etc.) del proceso.

2.9.1.1. *Modelo COCOMO II.*

La jerarquía de modelos de estimación de COCOMO II, abarcan lo siguiente:

- **Modelo de composición de aplicación.** Se usa en las primeras etapas de la ingeniería de software, cuando se ven elaboraciones de prototipos de interfases de usuario, la consideración de la interacción del software y el sistema, la valoración del rendimiento y la evaluación de la madurez de la tecnología.
- **Modelo de etapa temprana de diseño.** Se usa una vez fijados los requisitos y establecida la arquitectura básica del software.
- **Modelo de etapa postarquitectónica.** Se utiliza en la interacción de creación del Software.

Como todos los modelos de estimación para software, los modelos COCOMO II requieren información sobre dimensionamiento. Como parte de la jerarquía del modelo, están disponibles tres diferentes opciones de dimensionamiento: puntos objeto, puntos de función y líneas de código fuente.

Fórmula para hallar el factor de complejidad TCF:

$$TCF = (0.65 + 0.01 \times PF)$$

El procesamiento de datos del punto función se basa en la fórmula siguiente:

$$PF = Cuenta\ Total \times TCF$$

Factor LDC/PF se calcula con la fórmula:

$$LDC = PF \times Factor \frac{LDC}{PF}$$

Número estimado de líneas de código distribuidas en miles se calcula con la siguiente fórmula:

$$KLCD = \frac{LDC}{1000}$$

Las ecuaciones del COCOMO básico tienen la siguiente forma:

$$E = a_b (KLDC)^{bb}$$

$$D = c_b D^{db}$$

Donde:

E: Esfuerzo aplicado en personas por mes.

D: Tiempo de desarrollo en meses cronológicos.

KLDC: Número estimado de líneas de código distribuidas (en miles).

El número de programadores (N° Prog) se obtiene con la siguiente fórmula:

$$N^{\circ} Prog = \frac{E}{D}$$

Costo del software desarrollado por persona = Numero de programadores * salario de un programador

(Elaboración Propia).

2.10. Seguridad

La seguridad del software es una actividad de aseguramiento de la calidad del software que se enfoca en la identificación y evaluación de los peligros potenciales que pueden afectar negativamente al software.

(Sommerville, 2005, pág. 50).

La seguridad es una habilidad para protegerse de posibles ataques externos, que podrían ser accidentales o deliberados. Esto debido a que existen mayores números de personas conectados mediante dispositivos, lo cual genera mayor probabilidad de ataques externos

(Elaboración Propia).

2.10.1. Seguridad física.

Se trata de aplicar medidas de protección físicas e instrucciones de control, como medidas de prevención y contramedidas para proteger los recursos e información confidencial.

Este tipo de seguridad se enfoca a cubrir las amenazas ocasionadas tanto por el hombre como por la naturaleza del medio físico en que se encuentra ubicado el sistema. Algunas amenazas son desastres naturales, incendios accidentales y cualquier variación producida por las condiciones ambientales, amenazas ocasionadas por el hombre como robos o sabotajes. Disturbios internos y externos deliberados, etc.

La función primordial del mismo es la seguridad física. Tener controlado el ambiente y acceso físico permite disminuir adversidades y tener los medios para luchar contra accidentes

(Elaboración propia).

2.10.2. Seguridad lógica.

Es la aplicación de barreras y procedimientos que protejan el acceso a los datos y a la información contenida en el sistema.

A la par de la seguridad física la seguridad lógica no debe ser tomado a la ligera ya que protege al activo más importante es la información.

La seguridad lógica trata de conseguir, restringir el acceso a los programas y archivos, asegurar que los usuarios puedan trabajar sin supervisión y no puedan modificar los programas ni los archivos que no correspondan, asegurar que se estén utilizados los datos, archivos y programas correctos en y por el procedimiento correcto, verificar que la información transmitida sea recibida sólo por el destinatario al cual ha sido enviada y que la información recibida sea la misma que la transmitida, Disponer de pasos alternativos de emergencia para la transmisión de información.

2.10.2.1. Seguridad a nivel sistema operativo.

Todos los sistemas operativos incorporan características de seguridad principalmente para mantener a los usuarios no autorizados fuera del acceso de los recursos de su ordenador.

La seguridad de sistema operativo se basa en dos principios:

- El sistema operativo proporciona acceso a una serie de recursos, directa o indirectamente, como los archivos en un disco local, las llamadas privilegiadas del sistema, la información personal sobre los usuarios, y los servicios ofrecidos por los programas que se ejecutan en el sistema.
- El sistema operativo es capaz de distinguir entre algunos solicitantes de estos recursos que están autorizados o se permite para acceder a los recursos, y otros que no están autorizados o prohibido. Aunque algunos sistemas solo puede distinguir entre privilegiados y no privilegiados, los sistemas suelen tener una forma de identidad solicitante, tales como un nombre de usuario.

Además de permitir que el modelo de seguridad, en un sistema operativo con un alto nivel de seguridad también se ofrecen opciones de auditoria. Esto permitirá el seguimiento de las solicitudes de acceso a recursos tales como “que ha estado leyendo este archivo”. La seguridad del sistema operativo más se puede dividir en dos subsecciones en lo que respecta a los solicitantes:

- Seguridad Interna – un programa en ejecución. En algunos sistemas, un programa una vez que se ejecuta no tiene limitaciones. Sin embargo, más

comúnmente, el programa tiene una identidad que se guarda y se utiliza para comprobar todas sus solicitudes de recursos.

- Seguridad Externa – una nueva solicitud desde fuera de la computadora como un login en una consola conectada o algún tipo de conexión de red. Para establecer la identidad, puede haber un proceso de autenticación. A menudo, un nombre de usuario debe ser citado y cada usuario puede tener una contraseña.

Otros métodos de autenticación, tales como tarjetas magnéticas o los datos biométricos podrán utilizarse en su lugar. En algunos casos, especialmente con las conexiones de una red, los recursos se pueden acceder si autenticación en absoluto.

La seguridad del sistema operativo ha sido durante mucho tiempo una preocupación por los datos altamente sensibles celebra en equipos de carácter personal, comercial, e incluso militares. Es por eso que los programadores del sistema operativo prestan especial atención a la seguridad de los sistemas operativos que están desarrollando. Ellos quieren asegurarse de que los datos delicados contenidos en un sistema se mantienen como privado y solo se le permite ser visto por aquellos que están autorizados a hacerlo.

2.10.2.2. Seguridad a nivel de Base de Datos.

Es la capacidad del sistema para proteger Datos, Servicios y Recursos de usuarios no autorizados. El fin de la seguridad es garantizar la protección o estar libre de todo peligro y/o daño, y que en cierta manera es infalible.

- **Confidencialidad:** nos dice que los objetos de un sistema han de ser accedidos únicamente por elementos autorizados a ello, y que esos elementos autorizados no van a convertir esa información en disponible para otras entidades.
- **Integridad:** Significa que los objetos solo pueden ser modificados por elementos autorizados, y de una manera controlada.
- **Disponibilidad:** indica que los objetos del sistema tienen que permanecer accesibles a elementos autorizados; es el contra
- **Encriptación** La mayor parte de bases de datos contiene la información sensible, propia, y/o probada.

Esto puede incluir información de cliente, salarios de empleado, registros de pacientes, números de la tarjeta de crédito. La llave al mantenimiento de esta información en una manera segura es la confidencialidad – y las empresas que no pueden asegurar la seguridad (el valor) para la vergüenza del riesgo confidencial de la información, penas financieras, y a veces aun el negocio mismo

(Fuente Propia)

2.10.2.3. Seguridad a nivel del Software.

La seguridad del software aplica los principios de la seguridad de la información específica. Esto se refiere a la seguridad de información contra el acceso desautorizado y la modificación de información la cual el sistema está encargado de utilizar

(Fuente propia).

3. Marco aplicativo

3.1. Introducción

El desarrollo del “Sistema Web Para El Registro Y Control De Operaciones Del Laboratorio Técnico De Suelos” Caso: Sociedad De Ingenieros De Bolivia Departamental La Paz (Sib La Paz), sigue todo lo descrito en el capítulo anterior como ser la metodología del Proceso Unificado de Desarrollo de Software (RUP), utilizando el lenguaje de Modelado Unificado (UML), la base de datos MySQL, frameworks Bootstrap y CakePhp que nos ayudaran a desarrollar de manera más certera lo propuesto anteriormente.

En la fase de inicial se modelará, el funcionamiento del sistema actual y describiremos los actores que intervienen en el negocio. En la fase de elaboración analizaremos y determinaremos los requerimientos más importantes y en base a ellos partiremos a la construcción de los diagramas de caso de uso del sistema, los diagramas de secuencia y los diagramas de actividades.

En la fase de diseño construiremos el diagrama de clases que nos mostrara como se relacionan los componentes del software, también veremos los diagramas de despliegue, diagramas de paquete y finalmente mostraremos el diseño de interfaz del sistema. Posteriormente en la fase de transición se realizarán pruebas en el sistema, pruebas de seguridad y restricciones al sistema.

3.2. Fase de inicio

3.2.1. Modelado del negocio.

El modelado del negocio nos ayudara a comprender la estructura y la dinámica de la organización, también comprender los problemas actuales e identificar posibles mejoras.

a) Modelado de casos de uso del negocio

A continuación, se describen los procesos que se realizan actualmente en la parroquia Santísima Trinidad y los actores que intervienen en cada uno de los procesos. Como se muestra en la figura 10.

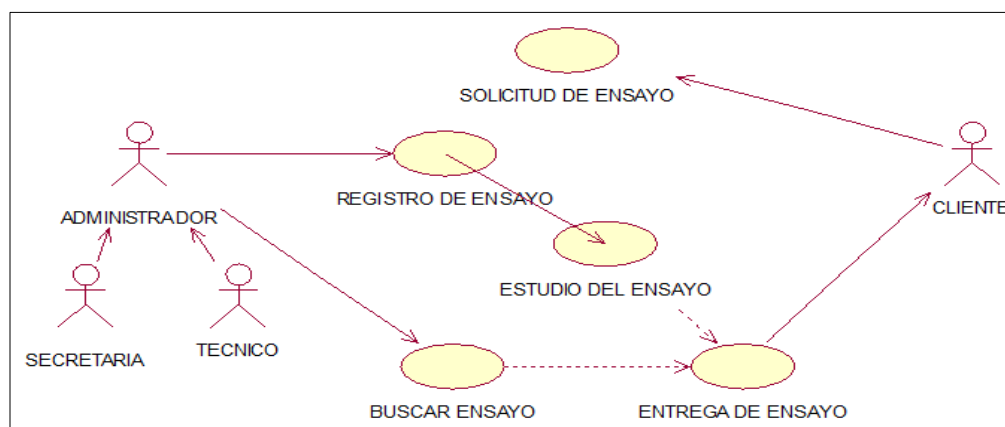


Figura 10 Modelo de casos de uso del negocio- Caso registro ensayo

(Elaboración Propia)

b) Descripción de los actores del negocio

Administrador: Es la persona encargada de registrar y entregar los ensayos. El administrador cumple dos roles, puede ser la secretaria o el técnico quien registre y entregue los ensayos de cualquier solicitud.

Cliente: Es la persona que solicita diferentes ensayos para poder cumplir las normativas de la construcción.

c) Descripción de los casos de uso del negocio

El flujo de información del caso de uso solicitud de ensayo se describe en la tabla 7. Considerando todas las opciones que se ven cuando una persona solicita un ensayo.

Tabla 7

Descripción del caso de uso solicitud de ensayo

Caso de uso	Solicitud de ensayo	
Actores	Cliente	
Propósito	Solicitar el registro de un ensayo	
Descripción	El cliente solicita ensayos y se procede a registrar su orden	
Flujo Principal	Eventos actor	Eventos Sistema
	1. El cliente acude al laboratorio para realizar una solicitud 4. El cliente obtiene la copia de su orden	2. Se procede a registrar su orden 3. Se toman los datos para realizar dicho ensayo

(Elaboración Propia)

El flujo de información del caso de uso registro de ensayo se describe en la tabla 8. Para ver de qué forma se procede al registro del ensayo con el actual sistema que maneja el laboratorio.

Tabla 8

Descripción del caso de uso registro de ensayo

Caso de uso	Registro de ensayo	
Actores	Cliente y administrador	
Propósito	Registrar los ensayos que el cliente ordeno	
Descripción	El Administrador registra al cliente, previo análisis de sus solicitudes.	
Flujo Principal	Eventos actor	Eventos Sistema
	1. El administrador verifica que la solicitud este dentro de los parámetros que trabaja el laboratorio 5. El administrador entrega comprobante	2.Los datos del cliente son registrados para un determinado ensayo 3.Se verifica que no haya algún error 4.Se guardan los datos

(Elaboración Propia)

El flujo de información del caso de uso entrega de ensayo se describe en la tabla xx de forma detallada para ver que actores intervienen y de qué forma responde el sistema actual que tiene el laboratorio.

Tabla 9

Descripción de caso de uso entrega de ensayo

Caso de uso	Entrega de ensayo
Actores	Administrador y cliente

Propósito	Entrega de ensayo solicitado	
Descripción	El cliente recoge la solicitud que realizo como constancia de haber realizado una orden.	
Flujo Principal	Eventos actor	Eventos Sistema
	1. El cliente solicita respaldo de la solicitud 4. El administrador entrega el respaldo de solicitud	2.El sistema verifica los datos 3.El sistema emite respaldo de solicitud

(Elaboración propia)

3.3. Fase de elaboración

En la fase de elaboración se detalla en detalle la mayoría de los diagramas de casos de uso, secuencia y estados más representativos que median en el sistema.

3.3.1. Determinación y análisis de requerimientos.

La determinación y análisis de requerimientos forma el punto de partida del sistema, con ella se busca percibir los requerimientos del usuario, conceptualizando los datos al interior del sistema, realizando la presentación grafica de los procesos.

Una vez elaborado el análisis de requerimientos, en base a entrevistas, revisando la forma de trabajo y la evaluación de archivos existentes, se pudo abstraer los módulos a desarrollar.

Los requerimientos más importantes identificados en el laboratorio se deben considerar para su respectivo análisis estos se detallan en la tabla 10.

Tabla 10

Funciones del sistema

REF.	FUNCION	CATEGORIA
R1	Módulo de registro de usuarios	Evidente
R2	Módulo de registro de clientes	Evidente
R3	Módulo de registro de secretarios	Evidente
R4	Módulo de registro de técnicos	Evidente
R5	Módulo de registro de ensayos	Evidente
R6	Módulo de registro de ordenes	Evidente
R7	Emisión de registro de ordenes	Evidente
R8	Generación de reportes y búsqueda de ensayos pasados	Evidente
R9	Control de acceso de los usuarios al sistema	Oculto

(Elaboración Propia)

3.3.2. Descripción de los usuarios del sistema.

Es necesario identificar a todos los participantes, para proveer de forma efectiva un software de calidad, afines a las necesidades de los usuarios. También es necesario identificar a los usuarios del sistema y asegurarse que el conjunto de colaboradores los representa debidamente.

En esta sección se mostrará el perfil de los usuarios envueltos en el proyecto, así como los problemas más importantes que estos observan, para enfocar la solución propuesta hacia ellos.

Administrador: Encargado de administrar el Sistema Informático realizando el seguimiento administrativo, contemplando acciones como registrar, actualizar, eliminar y revisar documentos para su archivado. También realiza la administración de la base de datos.

- **Secretaria:** Este usuario se encarga del llenado de los documentos como ser: el registro, modificación y eliminación de registros. También puede sacar algunos reportes.
- **Técnico:** Este usuario se encarga de la obtención de datos para el respectivo ensayo, y posterior llenado de planillas, y subsanación de las mismas.

3.3.3. Diagramas de casos de uso del sistema.

El diagrama de casos de uso representa, como un actor opera con el sistema, además de los elementos que interactúan con el mismo.

Para poder definir los casos de uso, se considerará los requerimientos del sistema definidos con anterioridad. En la figura 11 se describe el sistema en forma clara y consistente.

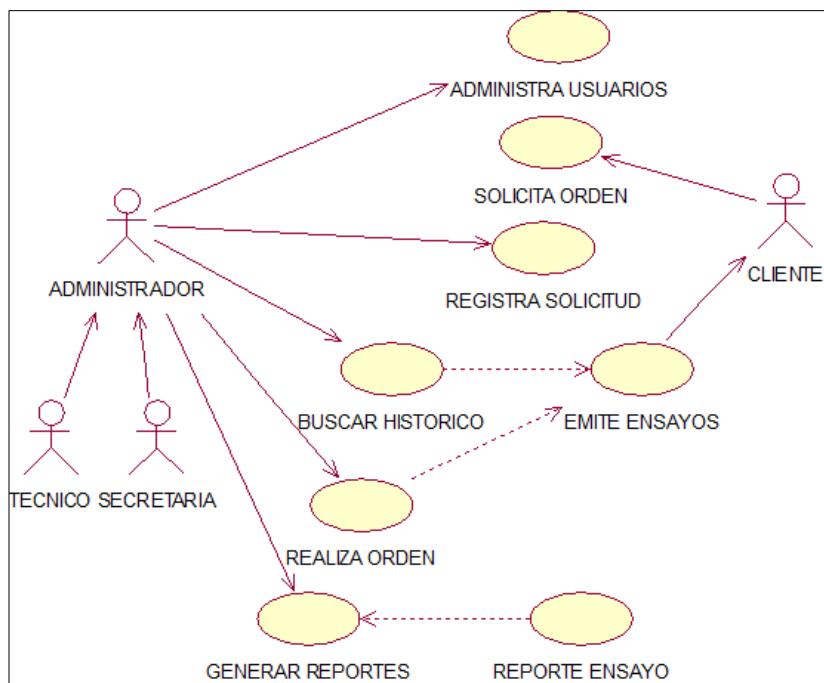


Figura 11 Modelo de caso de uso- General

(Elaboración Propia)

3.3.3.1. Caso de uso expandido.

El caso de uso expandido muestra a detalle partes del caso de uso general, teniendo en cuenta los requerimientos del sistema, la figura 12 muestra en detalle el caso de uso de administra usuario.

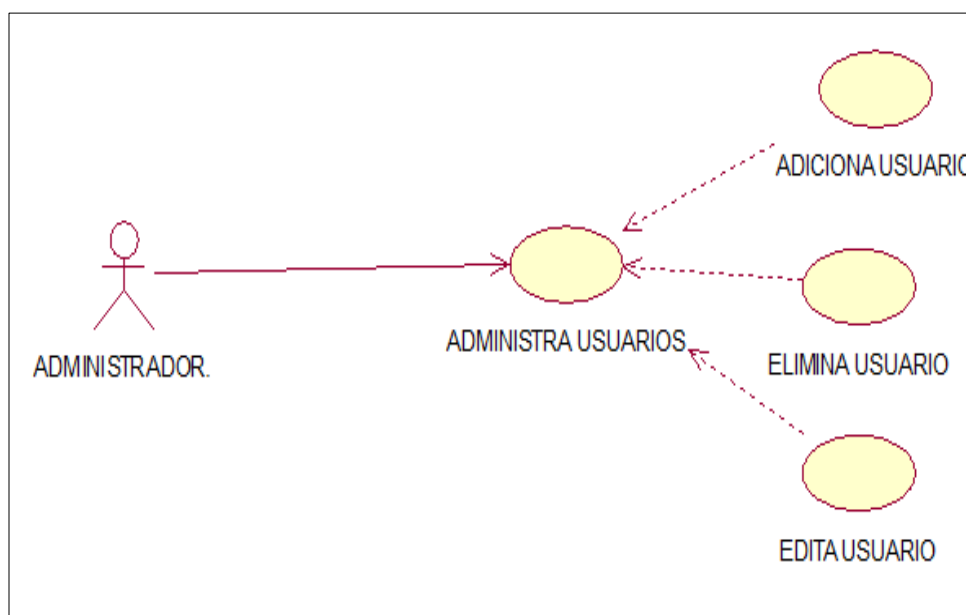


Figura 12 Modelo de caso de uso-administra usuarios

(Elaboración propia)

El flujo de información del caso de uso registro de usuario nuevo se describe en la tabla 11 de forma detallada.

Tabla 11

Descripción Caso de Uso - Adicionar usuario

Caso de uso	Adicionar usuario	
Actores	Administrador (técnico, secretaria)	
Propósito	Registrar nuevos usuarios	
Descripción	El administrador podrá registrar a un nuevo usuario y asignarle su contraseña respectiva	
	Eventos actor	Eventos Sistema

Flujo Principal	<p>1. El administrador ingresa al sistema con su nombre de usuario y contraseña.</p> <p>3. Selecciona la opción nuevo usuario.</p> <p>5. Llenan el formulario con los datos requeridos</p> <p>6. Acepta guardar el registro del nuevo usuario</p>	<p>2. El sistema autentifica el usuario y despliega el menú principal</p> <p>4. Despliega formulario para el nuevo usuario</p> <p>7. El sistema inserta el nuevo registro en la base de datos.</p>
Flujo Alternativo	<p>2. El usuario ingresa su nombre de usuario o contraseña errónea</p> <p>5. No se llena el formulario de manera adecuada.</p>	
Precondición	El administrador ha realizado correctamente el ingreso al sistema	
Postcondición	El sistema debe registrar al nuevo usuario	

(Elaboración Propia)

El flujo de información del caso de uso eliminar usuario se describe en la tabla 12

Tabla 12

Descripción Caso de Uso - Elimina usuario

Caso de uso	Elimina usuario	
Actores	Administrador (técnico, secretaria)	
Propósito	Eliminar usuario	
Descripción	El administrador podrá eliminar un usuario que ya no sea parte del laboratorio	
	Eventos actor	Eventos Sistema

Flujo Principal	1. El administrador ingresa al sistema con su nombre de usuario y contraseña.	2. El sistema autentifica el usuario y despliega el menú principal.
	3. Selecciona la opción eliminar usuario	4. Busca en los registros el usuario a eliminar
	5. Selecciona el usuario a eliminar	7. El sistema elimina al usuario seleccionado
	6. Presiona la tecla eliminar usuario	
	8. Verifica que ya no exista el usuario	
Flujo Alternativo	1. El usuario ingresa su nombre de usuario o contraseña errónea.	
Precondición	El usuario existe en la base de datos	
Postcondición	El usuario ha sido eliminado	

(Elaboración propia)

El flujo de información del caso de uso modificar usuario se describe en la tabla 13.

Tabla 13

Descripción caso de uso- Edita usuario

Caso de uso	Edita usuario	
Actores	Administrador (técnico, secretaria)	
Propósito	Edita los datos de un usuario existente	
Descripción	El administrador podrá modificar los datos de un usuario para cualquier actualización	
	Eventos actor	Eventos Sistema

Flujo Principal	<p>1. El administrador ingresa al sistema con su nombre de usuario y contraseña.</p> <p>3. Selecciona la opción modificar usuario.</p> <p>5. Elige el usuario a modificar</p> <p>7. Cambia los datos del usuario</p> <p>8. Acepta guardar el registro del usuario modificado.</p>	<p>2. El sistema autentifica el usuario y despliega el menú principal.</p> <p>4. Despliega la lista de todos los usuarios</p> <p>6. Despliega el registro del usuario</p> <p>9.El sistema guarda los datos en la base de datos.</p>
Flujo Alternativo	<p>2. El usuario ingresa su nombre de usuario o contraseña errónea.</p> <p>5. No se llena el formulario de manera adecuada.</p>	
Precondición	El administrador ha realizado correctamente el ingreso al sistema.	
Postcondición	Mediante el sistema se edita al usuario seleccionado.	

(Elaboración Propia)

La figura 13 muestra en forma detallada el caso de uso registra sacramento.

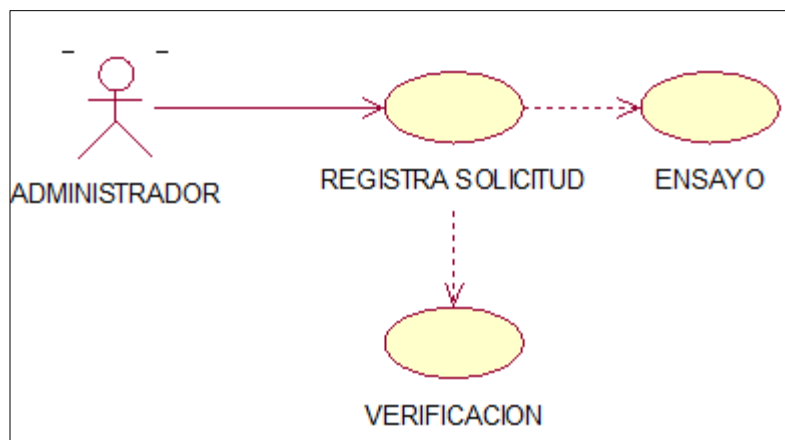


Figura 13 Modelo de caso de uso- Registra solicitud
(Elaboración Propia)

El flujo de información del caso de uso ensayo se describe en la tabla 14.

Tabla 14

Descripción de caso de uso- Ensayo

Caso de uso	Ensayo	
Actores	Administrador (técnico, secretaria)	
Propósito	Agregar nuevas órdenes de ensayos	
Descripción	El caso de uso especifica el registro de ordenes de ensayo que provienen de una solicitud	
Flujo Principal	Eventos actor	Eventos Sistema
	1.El administrador ingresa al sistema con su nombre de usuario y contraseña. 3. Selecciona el módulo registro nuevo cliente	2. El sistema autentifica el usuario y despliega el menú principal. 5. Verifica que no exista registros del cliente. 6. Despliega opciones de los ensayos

	4.Solicita al cliente sus datos personales 7. Elige registrar nueva orden 9. Llena los datos faltantes 10. Acepta guardar la nueva orden	8. Despliega formulario de registro con todos los campos llenos 11. El sistema guarda los datos en la base de datos.
Flujo Alternativo	9. No se llena el formulario de manera adecuada.	
Precondición	Los registros no existen en el sistema	
Postcondición	Sistema registra la nueva orden	

(Elaboración propia)

En la figura 14 podemos observar de forma detallada el caso de uso genera reportes.

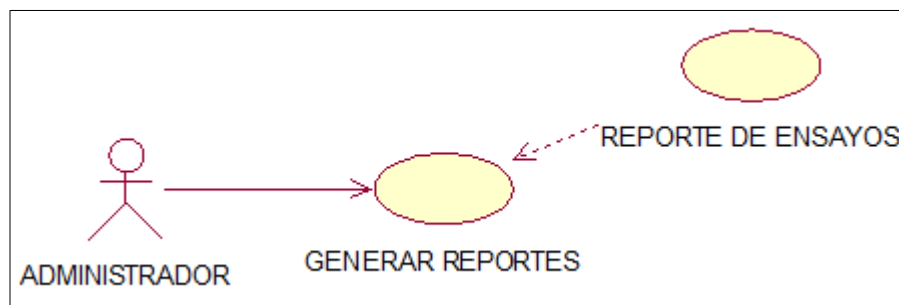


Figura 14 Modelo de caso de uso- Generar reportes

(Elaboración Propia)

El flujo de información del caso de uso reporte de ensayos se describe en la tabla 15.

Tabla 15

Descripción de caso de uso-Reporte de ensayos

Caso de uso	Reporte de ensayos	
Actores	Administrador (técnico, secretaria)	
Propósito	Generar reportes de los ensayos	
Descripción	El administrador podrá generar reportes de todos los ensayos registrados	
Flujo Principal	Eventos actor	Eventos Sistema
	1.- El administrador ingresa al sistema con su nombre de usuario y contraseña. 3.- Selecciona la opción de reportes 5.- Selecciona un conjunto de registros para el reporte	2.- El sistema autentifica el usuario y despliega el menú principal. 4.- Despliega todos los Registro de las ordenes 6.- Genera reporte
Flujo Alternativo	Ninguno	
Precondición	Las ordenes se encuentran almacenadas	
Postcondición	Se genera el reporte según lo seleccionado por el administrador	

(Elaboración Propia)

3.3.4. Diagrama De Secuencia.

El diagrama de secuencia muestra la manera en que se comunican los objetos al transcurrir el tiempo en el orden de las llamadas/eventos del sistema. El evento del sistema es una entrada externa que origina una operación del sistema como respuesta al evento,

representados en secuencias, el detalle del diagrama depende de la fase en la que estemos, lo que pretendamos contar con el diagrama y a quien.

A continuación, se muestran los diagramas de secuencia correspondientes al sistema:

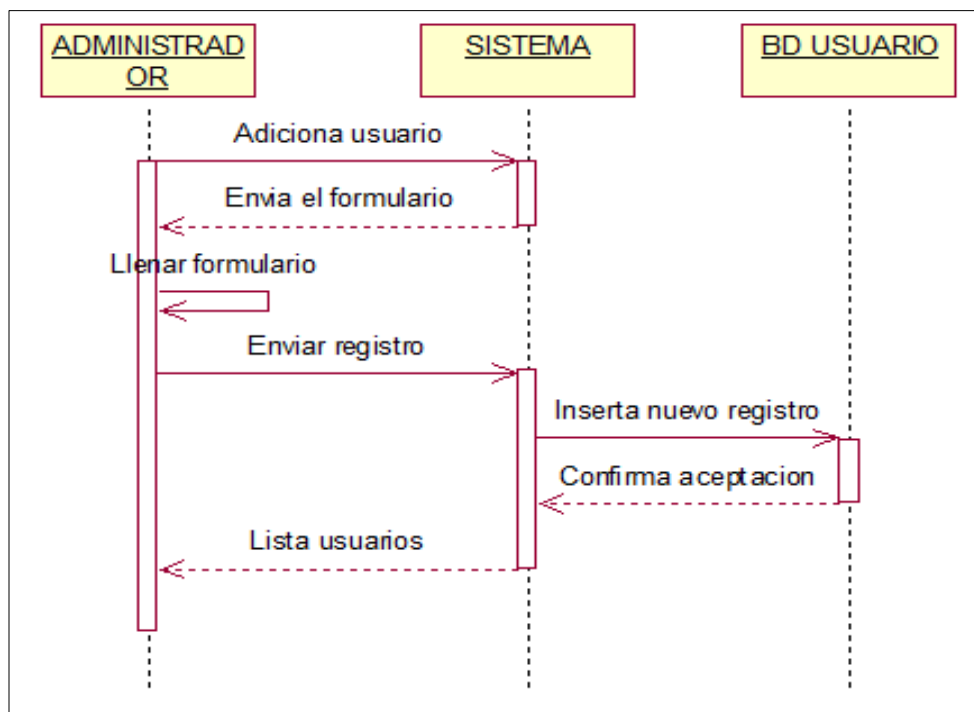


Figura 15 Diagrama de secuencia-Adicionar usuario
(Elaboración Propia)

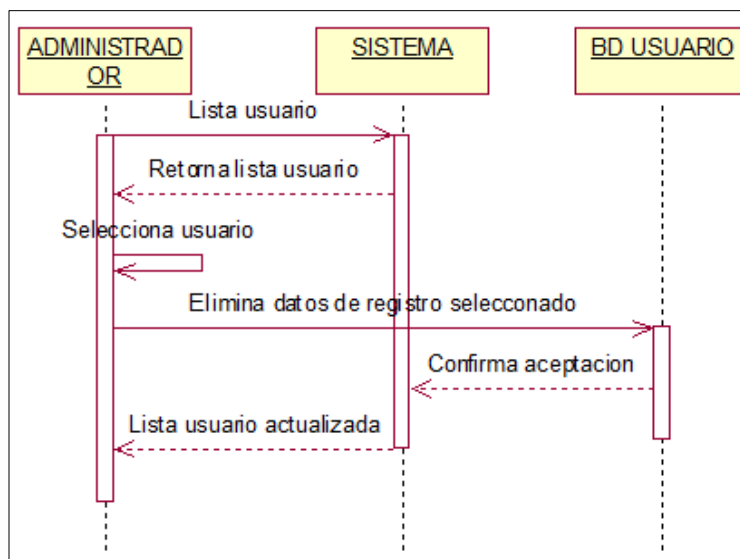


Figura 16 Diagrama de secuencia- Elimina Usuario
(Elaboración Propia)

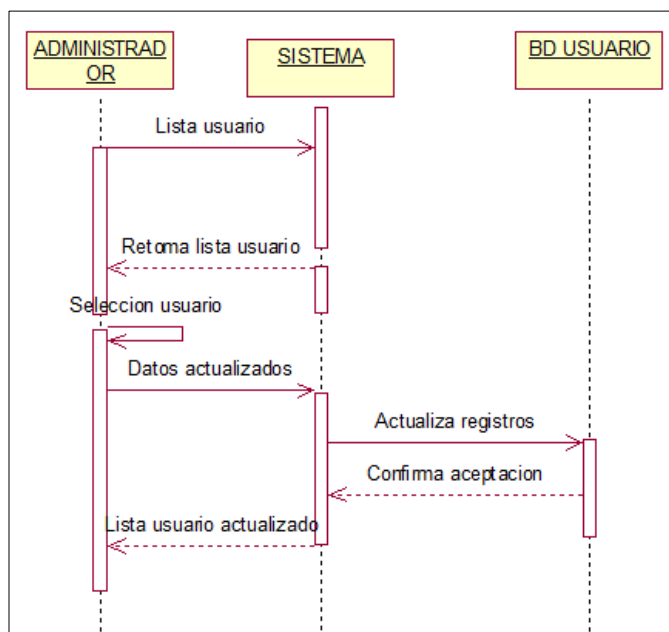


Figura 17 Diagrama de secuencia- Modifica usuario
(Elaboración Propia)

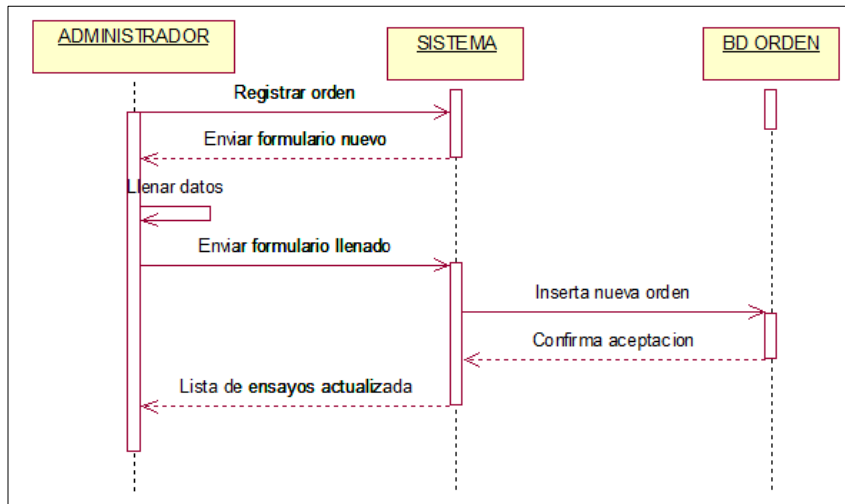


Figura 18 Diagrama de secuencia- Ensayo
(Elaboración Propia)

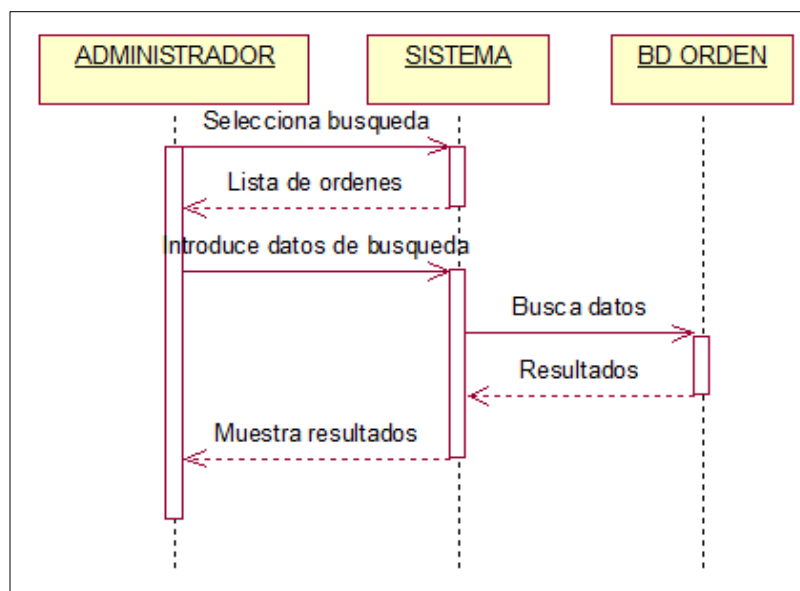


Figura 19 Diagrama de secuencia-Busca orden
(Elaboración Propia)

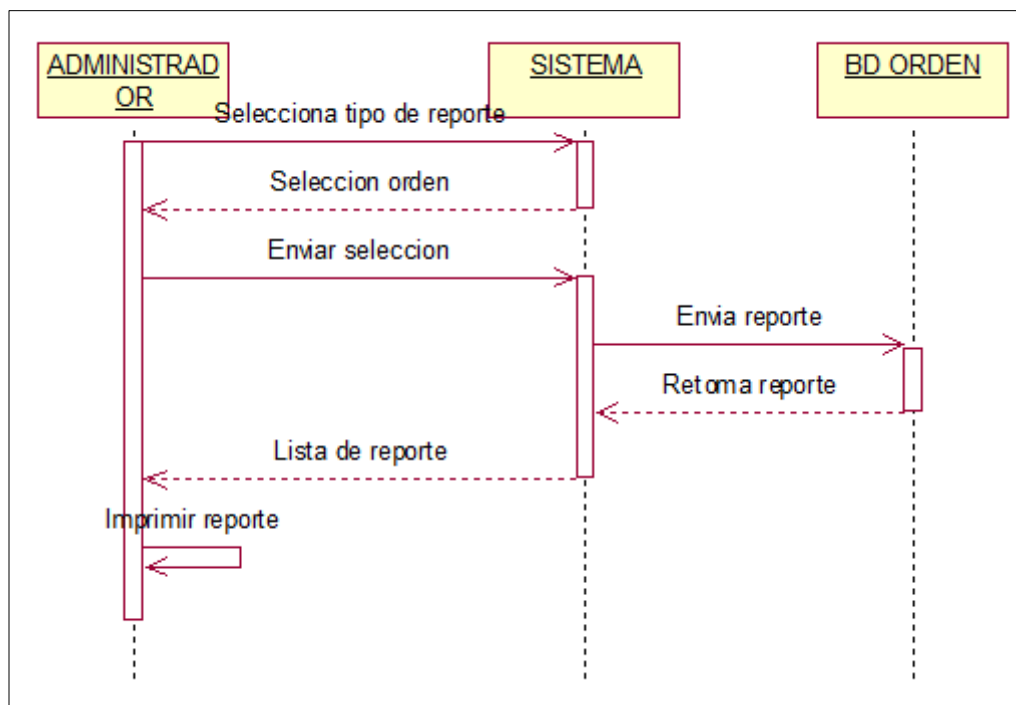


Figura 20 Diagrama de secuencia-Generar reportes

(Elaboración Propia)

3.3.5. Diagrama de actividades.

A continuación, se muestran los diagramas de actividades que nos ayudaran a analizar los problemas, los cuales iremos resolviendo como se muestran en las siguientes figuras.

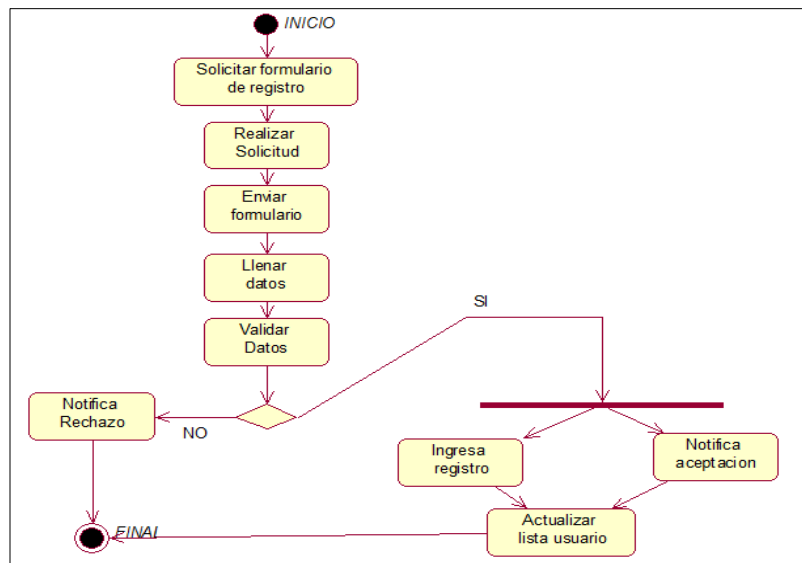


Figura 21 Diagrama de actividades-Adicionar Usuario
(Elaboración Propia)

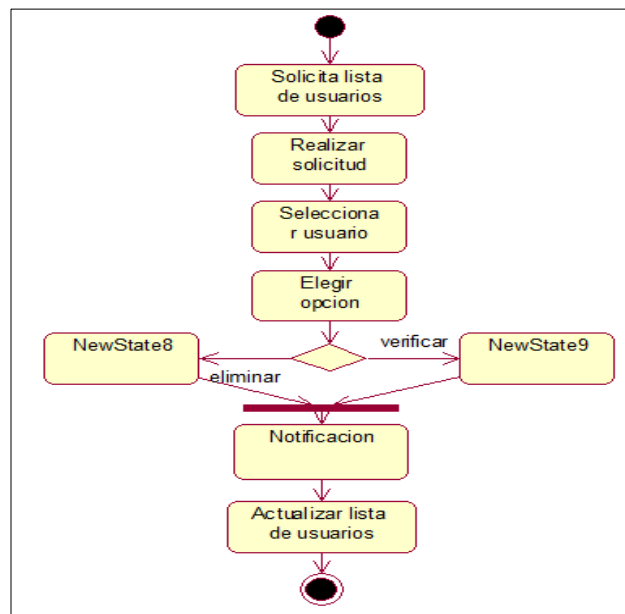


Figura 22 Diagrama de actividades-Modifica o Elimina
(Elaboración Propia)

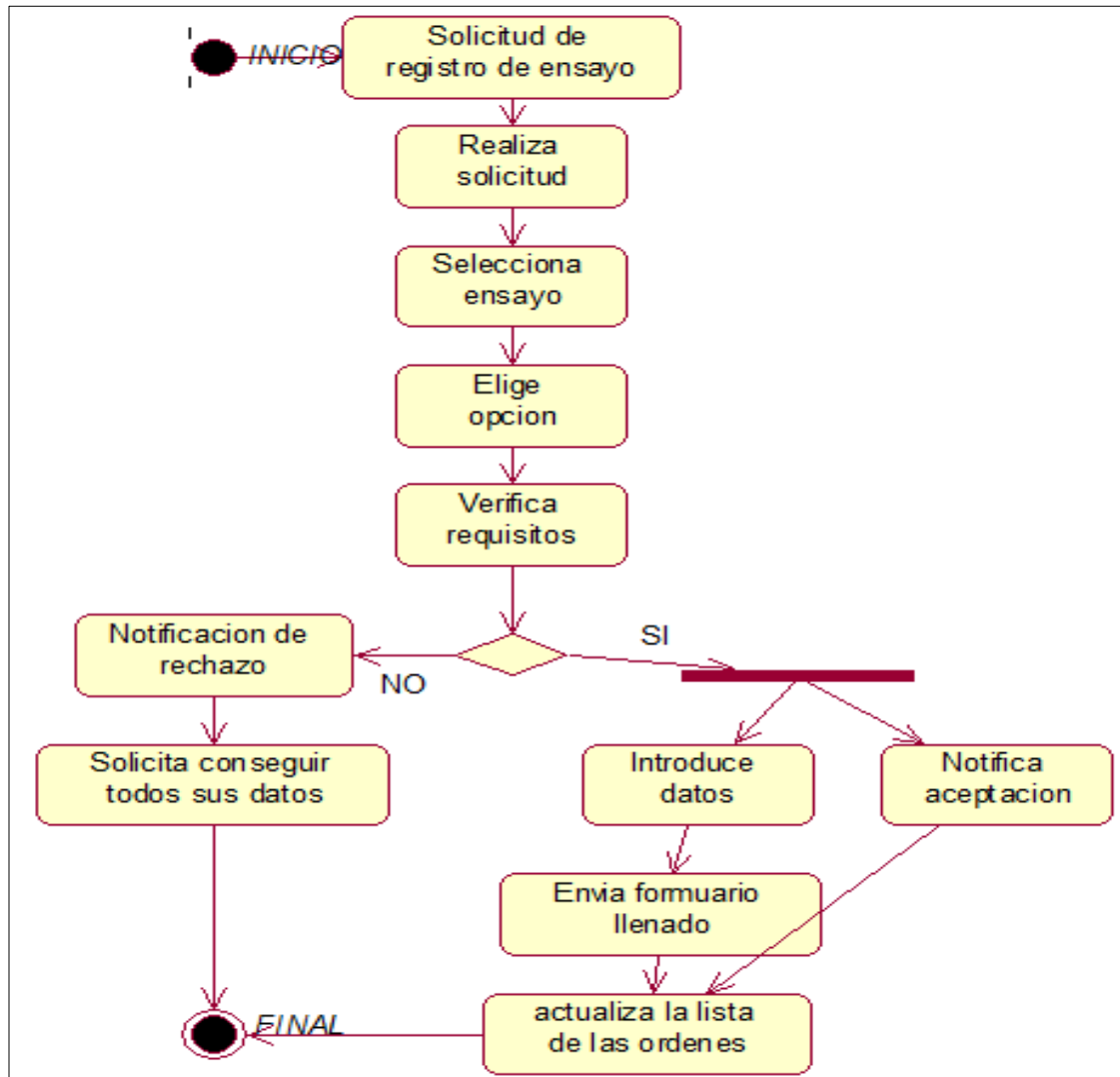


Figura 23 Diagrama de actividades-Registrar Solicitud

(Elaboración propia)

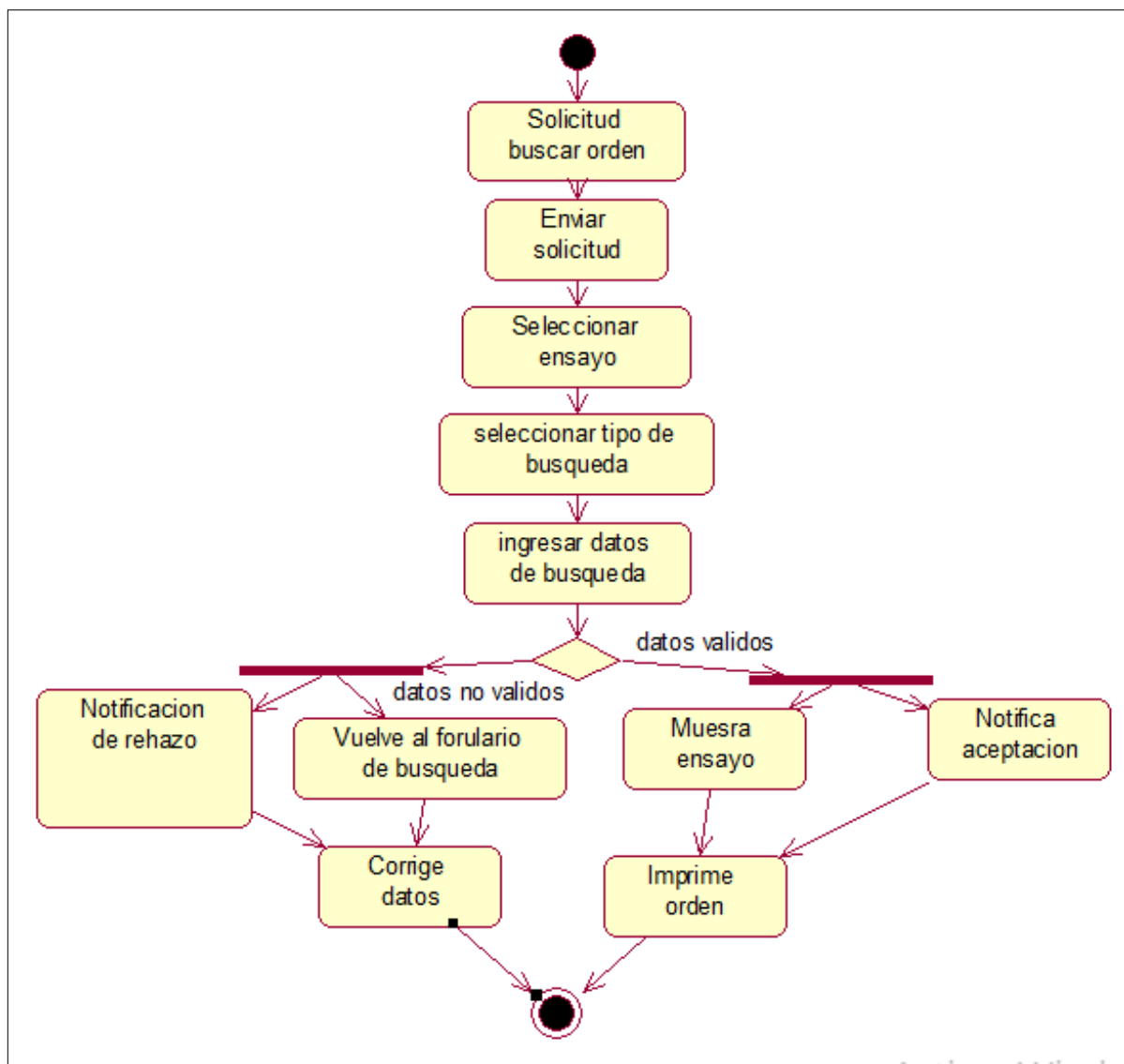


Figura 24 Diagrama de actividades-Busca Orden

(Elaboración Propia)

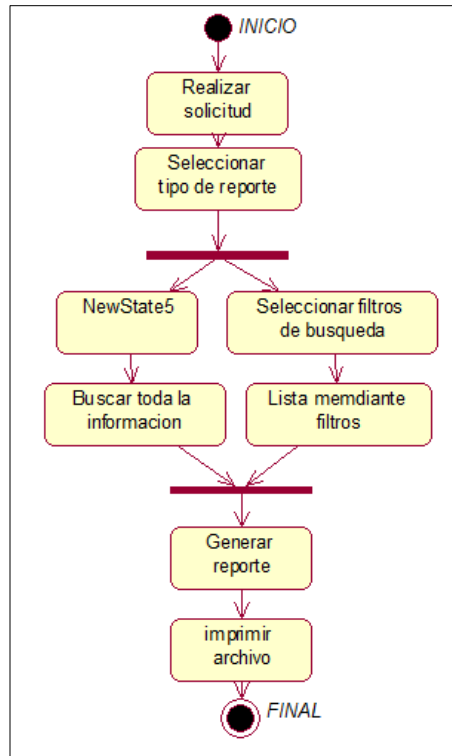


Figura 25 Diagrama de actividades-Generar reportes
(Elaboración Propia)

3.4. Fase de diseño

El modelo de diseño refleja decisiones en cuanto a asignación de responsabilidades entre las operaciones, muestra cómo se afectan componentes de software para resolver el problema planteado es el paso previo a la implementación.

3.4.1. Diagrama de clases.

Estos diagramas representan el o los objetos esenciales del sistema, es decir los que percibe el usuario y con los que espera tratar para completar su tarea en vez de objetos del sistema o de un modelo de programación.

Este es el diagrama principal para el análisis y diseño, en la figura se representan las relaciones entre las clases, atributos y sus operaciones para representar la información del sistema. Después de haber realizado el análisis para la base de datos e identificar todas las entidades que intervienen en el sistema, se elabora el diagrama de clases como se muestra en la figura 26

La meta de esta fase es producir un esquema físico donde muestre las llaves primarias y los atributos respectivos, que sea eficiente para las operaciones de consulta y actualización en el sistema.

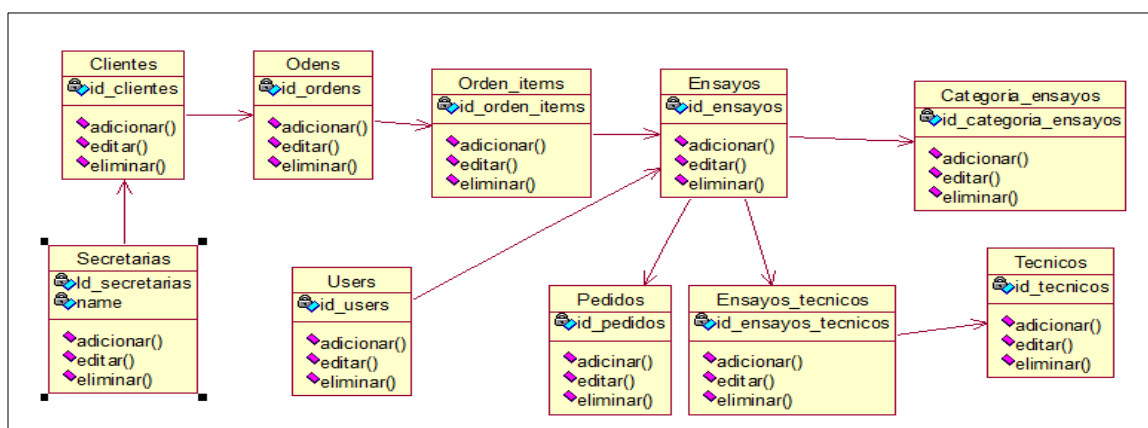


Figura 26 Diagrama de clases del sistema

(Elaboración Propia)

3.4.2. Diagrama Físico.

En la figura 27 se muestra el modelo físico del sistema.

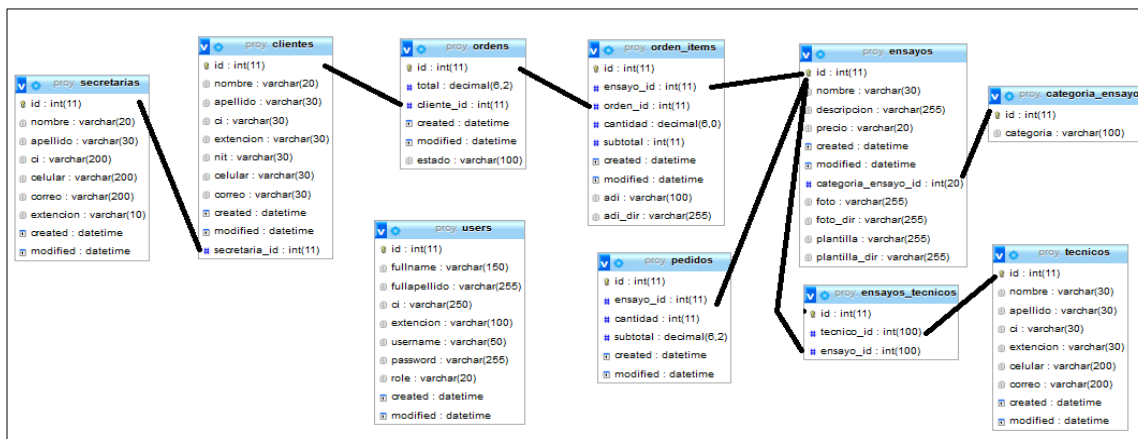


Figura 27 Modelo físico del sistema

(Elaboración Propia)

3.4.3. Diagrama De Despliegue.

El diagrama de despliegue ajusta la arquitectura en lapso de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra los elementos y artefactos del software.

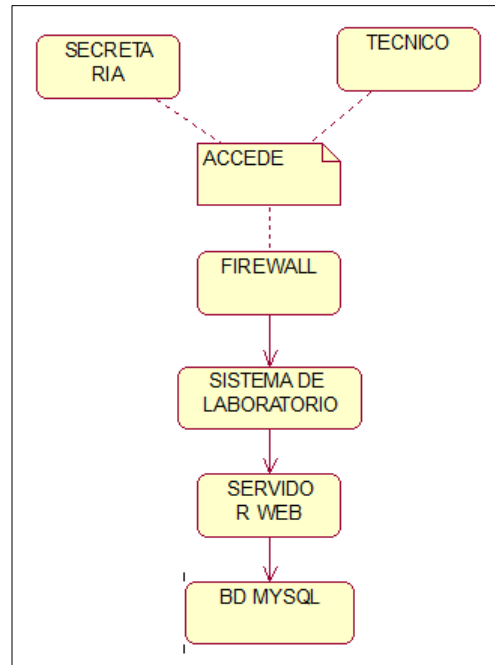


Figura 28 Diagrama de despliegue del sistema
(Elaboración Propia)

3.4.4. Diagrama De Paquetes.

Los diagramas de paquetes se usan para reflejar la organización de paquetes y sus componentes. Cuando se usan para representaciones, los diagramas de paquete de los elementos de clase se usan para proveer una visualización de los espacios de nombres. La figura 29 muestra de forma detallada lo mencionado, los cuales son diagramas de paquetes del sistema.

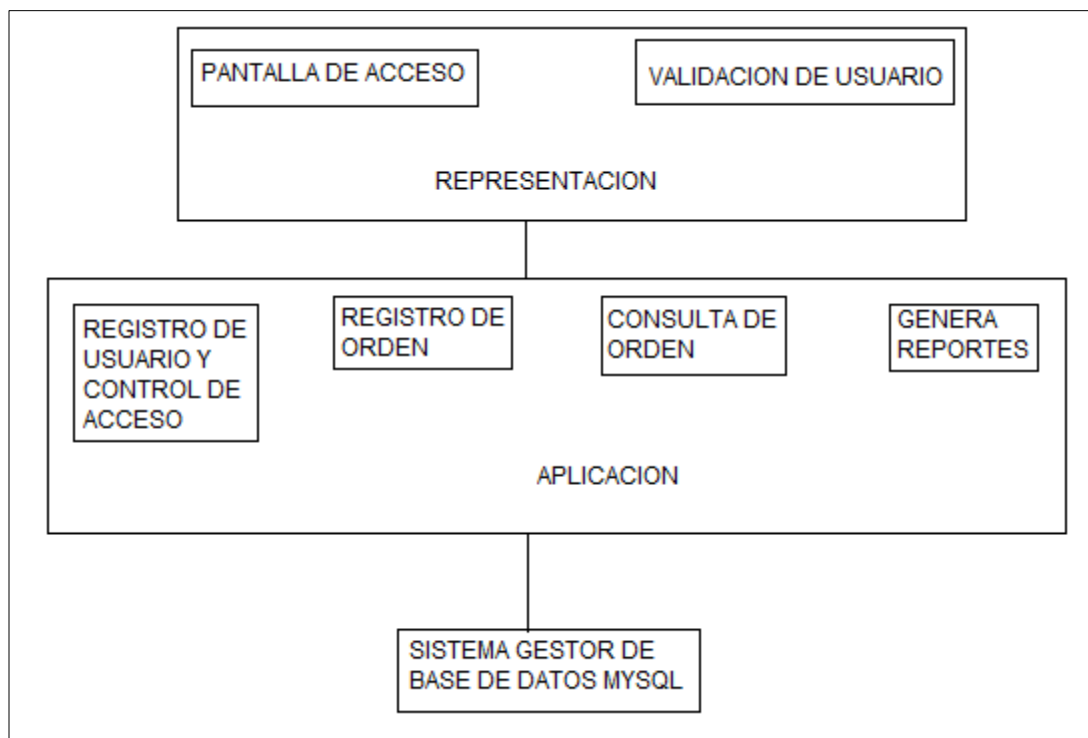


Figura 29 Diagrama de paquetes del sistema

(Elaboración Propia)

3.4.5. Diseño De Interfaz.

A continuación, se muestra las principales interfaces del sistema.

- Inicio de sesión

El administrador que quiera acceder al sistema debe tener su cuenta de usuario y contraseña, como se muestra en la figura 30.



SIB LA PAZ

Usuario

Contraseña

Acceder

SOCIEDAD DE INGENIEROS DE BOLIVIA DEPARTAMENTAL LA PAZ

S.I.B.
La Paz

SISTEMA WEB PARA EL REGISTRO Y CONTROL DE OPERACIONES DEL LABORATORIO TECNICO DE SUELOS

Figura 30 Autenticación en el sistema

(Elaboración Propia)

- Pantalla principal del sistema

En la figura 31 podemos observar la pantalla principal del sistema donde muestra todas las opciones a las cuales tiene acceso el administrador del sistema el cual puede ser el técnico o una secretaria.



Figura 31 Pantalla principal del sistema

(Elaboración Propia)

- Pantalla de administración de usuarios

En la figura 32 podemos observar la pantalla donde se administra a los usuarios en el cual muestra todas las opciones a las cuales tiene acceso el administrador del sistema.

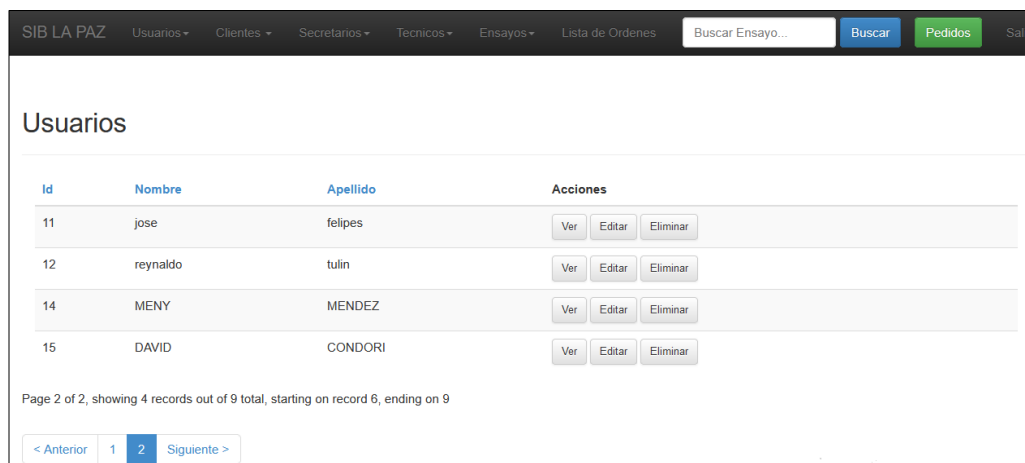
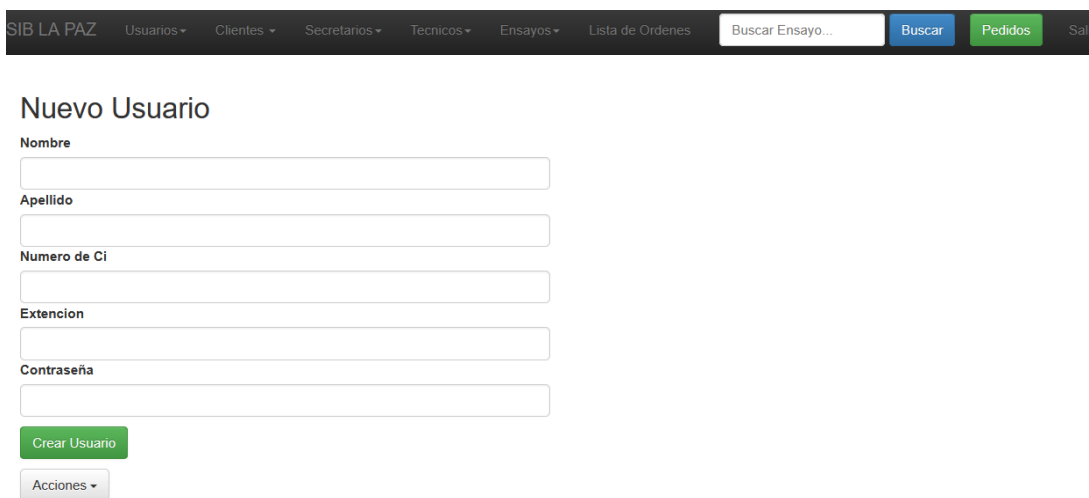


Figura 32 Pantalla de administración de usuarios

(Elaboración Propia)

- Pantalla de registro de nuevo usuario

En la figura 33 podemos observar cómo se registra un nuevo usuario



The screenshot shows a web application interface with a dark navigation bar at the top. The navigation bar contains the text 'SIB LA PAZ' and several menu items: 'Usuarios', 'Clientes', 'Secretarios', 'Tecnicos', 'Ensayos', and 'Lista de Ordenes'. To the right of these items are three buttons: 'Buscar Ensayo...', 'Buscar', and 'Pedidos', followed by a 'Salir' link. Below the navigation bar, the main content area is titled 'Nuevo Usuario'. It contains five text input fields, each with a label to its left: 'Nombre', 'Apellido', 'Numero de Ci', 'Extencion', and 'Contraseña'. Below the input fields are two buttons: a green 'Crear Usuario' button and a grey 'Acciones' button with a downward arrow.

Figura 33 Pantalla de registro nuevo usuario


(Elaboración Propia)

- Pantalla de agregar Ensayo

En la figura 34 podemos observar cómo se registra la orden.

SIB LA PAZ Usuarios Cientes Secretarios Tecnicos Ensayos Lista de Ordenes Buscar Ensayo... Buscar Pedidos Salir

GRANULOMETRIA



GRANULOMETRIA
Bs 100
Agregar a Pedido
Descripción: tamaño de los agregados
Creado: 2020-11-22 08:40:53
Modificado: 2020-11-22 08:40:53
Categoría: SUELOS
Acciones -

Tecnicos Relacionados

Id	Nombre	Apellido	Ci	Extencion	Acciones
3	MIGUEL	TORREZ	23463856	SC	Ver Editar Eliminar

Plantilla de Ensayo:
[Descargar Ejemplar](#)

Figura 34 Pantalla de agregar Ensayo
(Elaboración Propia)

3.5. Fase de transición

La fase de transición se dirige en liberar el sistema en producción para clausurar esta fase se cumple, con la aprobación y visto bueno del diseño e implementación del sistema que será por parte de los usuarios.

3.5.1. Implementación.

Para esto se aplica pruebas extensivas a lo largo de esta fase, una buena afinación del proyecto tiene lugar aquí, incluyendo modificaciones a los defectos significativos, se

destaca las pruebas de caja blanca y caja negra, con el objeto de descubrir defectos que tiene el sistema.

3.5.1.1. Pruebas del sistema.

El motivo de la elección de este método, es la utilidad que tiene para probar software. La prueba de caja negra intenta descubrir errores en las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en la estructura de datos o acceso a base de datos externas.
- Errores de rendimiento.
- Errores en la inicialización y terminación.

El mismo consiste en catalogar las entradas y salidas del sistema, primeramente, en forma global y después en forma consecutiva en cada uno de sus componentes que se encuentran elaborados.

Esto se lo realiza en forma independiente y la manera en que estas son transformadas, para tener un mejor y buen entendimiento del objetivo de cada uno de los distintos componentes.

- Se realizó pruebas unitarias, es decir cada módulo en el momento de la construcción previamente puesta en producción.

- La prueba de integración se realiza cuando todos los módulos, están diseñados y desarrollados, para luego integrarlos y posteriormente realizar pruebas de todo el sistema.
- Las pruebas del software se realizan una vez integrado los módulos, se procede a las pruebas con datos reales.
- Las pruebas de implantación, se realiza en forma incremental, es decir que ya parte del sistema está en plena producción acorde a la metodología RUP.
- Las pruebas de aceptación, después del periodo de pruebas, se espera que el sistema sea aceptado por los usuarios finales.

3.5.1.2. Pruebas de seguridad.

La seguridad es responsabilidad de todos aquellos que están en contacto con el sistema. La cual lo enfocamos desde tres aspectos interrelacionados entre sí: físico, lógico y conductual.

Seguridad física

- El servidor debe ser manejado por el personal autorizado, en este caso sería el párroco encargado de la parroquia, para una buena administración de la base de datos. En caso de mantenimiento técnico, este deberá consultar al administrador del sistema.

- A las oficinas de las distintas capillas que tiene la parroquia, se debe restringir, el acceso físico de otras personas, que nos son propios de esta oficina, para una mayor seguridad.
- El personal de limpieza debe estar consciente de las restricciones de seguridad.

Seguridad lógica

Se especifica, al acceso de los usuarios, a los recursos y a la información que dispone en un ordenador, como también al uso correcto de los mismos. Cuando se utilizan, permiten al usuario entrar al sistema o a una parte particular de una base de datos con una contraseña correcta, entre las cuales se destacan:

- Seguridad del motor MySQL, contraseña de acceso a base de datos.
- Seguridad del sistema, mediante el manejo de usuarios del sistema y contraseñas.
- Se puede utilizar Back Up de la base de datos, solo por usuarios con este rol de manejo de sistemas.
- Sistema de encriptación de contraseñas, para que el usuario confíe en el sistema.
- Se puede realizar restauración del sistema en caso de eventualidades, es decir recuperar una copia anterior.

Seguridad conductual

- Crear conciencia en el manejo de información a través de capacitaciones planificadas, para que los mismos reconozcan el valor que tiene la información.
- Identificación de personal que eventualmente tendrán acceso a la computadora, datos e información para asegurar que sus intereses son consistentes con los intereses del laboratorio y que entienden por completo la importancia de llevar a cabo los procedimientos de seguridad.
- Se crea un manual de funciones, que describan las políticas con respecto a la seguridad, para que los administradores estén totalmente conscientes de las expectativas y responsabilidades que se deben asumir.

3.5.2. Restricciones del sistema.

- El sistema funciona dentro del entorno de Windows 7 adaptable a sus siguientes versiones.
- El sistema funciona en un entorno de red intranet, pero se tiene previsto, para que pueda ser utilizado desde la web, una vez se adquiera su respectivo dominio.

3.5.3. Capacitación.

La capacitación del personal de la Parroquia Santísima Trinidad, que maneja el sistema para el registro de certificados parroquiales, se realiza de la siguiente manera:

- Instalación del sistema.
- Back Up y restauración de la base de datos del sistema.
- Altas de usuarios.
- Ubicación física de equipos.
- Políticas de manejo de usuarios.
- Ingreso al sistema.
- Registro, modificación y eliminación de usuarios.
- Registro, modificación y eliminación de solicitudes y ordenes
- Búsqueda de órdenes.
- Generación de reportes.

4. Métricas de Calidad

4.1. Introducción

La calidad del software es la concordancia con los requerimientos funcionales y de rendimiento directamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se esperan de todo software desarrollado. Un producto de alta calidad requiere menos mantenimiento y facilita tanto el desarrollo como el mantenimiento de la productividad.

A la conclusión del desarrollo del sistema, se tiene la necesidad de conocer la calidad obtenida del sistema siendo este un factor importante, por tanto, existen dos tipos de medición directa e indirecta. En el presente proyecto se aplicará las medidas indirectas planteada por la norma ISO 9126.

Si bien no se llega a obtener la calidad perfecta se desea lograr una calidad necesaria o suficiente para el momento cuando el usuario así lo requiera.

4.2. Funcionalidad

La funcionalidad no se mide directamente, por tanto, no es necesario evaluar un conjunto de características y capacidades del sistema.

Para el cálculo de la funcionalidad utilizaremos la métrica de “**Punto Función**” (PF), para esto se debe determinar cinco características de dominios de información necesarias para el cálculo de la misma.

1. **Número de Entradas del Usuario.** Se cuenta cada entrada del usuario que proporciona diferentes datos al sistema, en el caso del sistema se identificaron 19 entradas del usuario.
2. **Número de Salidas del Usuario.** Se cuenta cada salida que proporciona la información del usuario, estas pueden ser informes, reportes, etc. Se identificaron 8 salidas del usuario en el sistema.
3. **Número de Peticiones del Usuario.** Se cuentan la cantidad de entradas interactivas que producen la generación de respuestas (salidas) inmediatas del sistema. Se apreciaron 6 peticiones de usuario.
4. **Número de Archivos.** Se cuenta cada archivo maestro lógico es decir un grupo lógico de datos que sean parte de la base de datos, o archivos independientes. Se contaron 10 archivos.
5. **Número de Interfaces Externas.** Se cuenta todas las interfaces legibles por la máquina (por ejemplo: archivos de datos de disco) que se utilizan para transmitir información a otros sistemas. En este caso existen 3 interfaces externas.

La tabla 16 muestra las cinco características con el factor de ponderación medio para el cálculo de punto función

Tabla 16

Cálculo de Punto Función

Parámetros de medición	Factor de Ponderación				Resultado
	Cuenta	Simple	Medio	complejo	
Nro. de entradas de usuario	19	3	4	6	76
Nro. de salidas de usuario	8	4	5	7	40
Nro. de peticiones de usuario	6	3	4	6	24
Nro. de archivos	10	7	10	15	100
Nro. de interfaces externas	3	5	7	10	21
Cuenta Total					261

(Elaboración Propia)

Los valores de la variable F_i , se obtienen tomando en cuenta la ponderación que se muestra en la tabla 17.

Tabla 17

Valores de ajuste de complejidad

Complejidad	<i>Escala</i>
Sin importancia	0
Incidencia	1
Moderado	2
Medio	3
Significativo	4

Esencial	5
----------	---

(Elaboración Propia)

Respondiendo a las preguntas en la tabla 18.

Tabla 18

Punto Función

	Factor	Ponderación
1	¿Requiere el sistema copias de seguridad y recuperación flexible?	5
2	¿Se requiere comunicación de datos?	3
3	¿Existen funciones del procedimiento distribuido?	4
4	¿Es crítico el rendimiento?	3
5	¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?	5
6	¿Requiere el sistema entrada interactiva?	4
7	¿Requiere la entrada de datos interactiva que las transiciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	3
8	¿Se actualiza los archivos maestros de forma interactiva?	4

9	¿Son complejos las entradas, las salidas, los archivos y las peticiones?	3
10	¿Es complejo el procedimiento interno?	3
11	¿Se ha diseñado el código para ser reutilizable?	4
12	¿Están incluidas en el diseño la conversión y la instalación?	4
13	¿Se ha desarrollado el sistema para soportar múltiples instalaciones en diferentes organizaciones?	4
14	¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizado por el usuario?	4
	Total	53

(Elaboración Propia)

PF es una métrica que mide el tamaño y la complejidad del sistema, el software en términos de las funciones del usuario y se define con la formula siguiente.

$$PF = \text{Cuenta}_{\text{total}} \times [0.65 + 0.01 \times \sum fi]$$

Donde:

- **PF** = Medida de Funcionalidad.
- **Cuenta Total** = Es la suma de valor de las entradas, salidas, peticiones, interfaces externas y archivos.

- **Grado de Confiabilidad** = Es la confiabilidad estimada del sistema.
- **Tasa de error** = Probabilidad subjetiva estimada del dominio de la información, este error estimado es de 1%.
- **Fi** = Son valores de ajuste de complejidad que toman los valores de la tabla 17 y que dan respuesta a las preguntas de la tabla 18.

Entonces

$$\sum fi = 53$$

Para calcular los puntos de función se utiliza la siguiente relación, para un nivel de confianza del 65%.

$$PF_{Real} = Cuentatotal \times [0.65 + 0.01 \times \sum fi]$$

$$PF_{Real} = 261 \times [0.65 + 0.01 \times 53]$$

$$PF_{Real} = 307,98$$

Comparando con los intervalos:

$$300 < PF - \text{Optimo}$$

$$200 < PF < 300 - \text{Bueno}$$

$$100 < PF \leq 200 - \text{Suficiente}$$

$$PF \leq 100 - \text{Deficiente}$$

Comparamos los valores de funcionalidad del sistema con el punto de función máximo que se puede alcanzar es:

$$PF_{Esperado} = Cuentatotal \times [0.65 + 0.01 \times 70]$$

$$PF_{Esperado} = 261 \times [0.65 + 0.01 \times 70]$$

$$PF_{Esperado} = Cuentatotal \times [0.65 + 0.01 \times 70]$$

$$PF_{Esperado} = 352.35$$

Con estos dos últimos obtenemos el porcentaje de funcionalidad del sistema:

$$\%PF = PF_{Real} / PF_{Esperado}$$

$$\%PF = 307,98 / 352.35$$

$$\%PF = 0.87$$

Entonces la funcionalidad del sistema alcanza un 87%

4.3. Mantenibilidad

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

Para medir la mantenibilidad del sistema se utilizan los índices de madurez del software (IMS).

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

Dónde:

MT = Número de módulos en la versión actual.

Fc = Número de módulos en la versión actual que han cambiado.

Fa = Número de módulos en la versión actual añadido.

Fe = Número de módulos en la versión anterior que se ha borrado.

Entonces

Tabla 19

Información requerida por el IMS

Información	Valores Obtenidos
MT	6
Fc	1
Fa	0
Fe	0

(Elaboración Propia)

Remplazando en la ecuación tenemos:

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

$$IMS=5/6$$

$$IMS=(6-(1+0+0))/6 \quad IMS=0.83$$

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

75 % <= IMS <= 100 % Optima.

50 % <= IMS <= 75 % Buena.

25 % <= IMS <= 50 % Suficiente.

0 % <= IMS <= 25 % Deficiente.

Entonces vemos que el valor obtenido se encuentra en el primer intervalo, con esto se puede afirmar que el sistema tiene una mantenibilidad optima del 83 %.

4.4. Portabilidad

La portabilidad se refiere al esfuerzo necesario para transferir el programa de un entorno ya sea de hardware y/o software a otro, es una característica deseable de todo software.

La portabilidad tiene la siguiente fórmula:

$$P=1-EP/EI$$

Dónde:

P = Portabilidad

EP = Número de días para portar el sistema

EI = Número de días para implementar el sistema

Entonces tenemos que:

EP=0.5 días

EI= 2,5 días

Remplazando en la formula tenemos:

$$P=1-0.5/2.5$$

$$P=0.80$$

Lo que significa que existe un 80% de que el usuario instale el software con éxito, entonces la portabilidad es óptima.

4.5. Usabilidad

La usabilidad o facilidad de uso (FU), se calcula con la siguiente formula.

$$FU = ((\sum x / m) \times 100) / n$$

En la tabla 20, calcularemos, $\sum x_i$ utilizando la escala de evaluación:

Tabla 20

Evaluación de preguntas para calcular la usabilidad.

Nro.	Preguntas	Evaluación (xi)
1	¿El sistema satisface los requerimientos de manejo de información?	4
2	¿Las salidas del sistema están de acuerdo a sus requerimientos?	4
3	¿Cómo considera el ingreso de datos del sistema?	5
4	¿Cómo considera los formularios que elabora el sistema?	5
5	¿El sistema facilita el trabajo que realiza?	5
$\sum x_i$		23

Calculando FU:

$$FU = (23/5 \times 100) / 5 \quad FU = 92$$

Por lo tanto, la facilidad de uso es del 92%.

4.6. Confiabilidad

La confiabilidad es la capacidad del software de mantener su nivel de rendimiento bajo las condiciones establecidas por un periodo de tiempo.

La confiabilidad del Software se mide con la siguiente formula:

$$R(t) = e^{-\lambda t}$$

Dónde:

R(t)= Confiabilidad del Sistema

λ = Error de tasa constante de fallas

t= Tiempo de operación del sistema (meses)

La tasa de error o la probabilidad de error que puede tener el sistema es del 0.5%, si consideramos un tiempo de 12 meses para el funcionamiento del sistema.

Remplazando los valores en la fórmula:

$$R(t) = e^{-\lambda t}$$

$$R(t) = e^{-0.005\% \times 12}$$

$$R(t) = 0.94$$

Por lo tanto, la confiabilidad del sistema es del 94% lo cual significa que en 12 meses el sistema mantendrá un rendimiento óptimo.

4.7. Eficiencia

La eficiencia es el conjunto de atributos que se relacionan con el nivel de performance del software y de la cantidad de recursos usados, bajo las condiciones establecidas, en tiempo y en recursos.

Tabla 21

Evaluación de preguntas para calcular la eficiencia

Nro.	Pregunta	Evaluación
1	¿La respuesta es rápida al utilizar las funciones?	94
2	¿Tiene rendimiento de acuerdo a los factores que utiliza?	95
3	¿Responde adecuadamente cuando utiliza las funciones?	96
	Total	95

(Elaboración Propia)

El sistema se considera eficiente por la óptima utilización de los recursos en un 95%.

4.8. Resultado final

En la tabla 22 se muestra el resultado final de los factores de la norma ISO 9126

Tabla 22

Resultado total

Factor	Resultado
Funcionalidad	88 %
Mantenibilidad	83 %
Portabilidad	80 %
Usabilidad	92 %
Confiabilidad	94 %
Eficiencia	95 %
Evaluación de la calidad total	88.6

(Elaboración Propia)

Entonces podemos ver que de un usuario que acceda al sistema tendrá una satisfacción del 87 % al utilizarla.

5. Evaluación de costo-beneficio

5.1. Introducción

Unas de las tareas más importantes en la planificación de los proyectos de software es la estimación, el cual consiste en determinar, con cierto grado de certeza, los recursos de hardware y software, costos, tiempo y esfuerzo necesarios para el desarrollo de los mismos en el presente capítulo examinaremos estos términos costos – beneficios, mediante COCOMO II, modelo de estimación de costos, mediante el obtendremos el esfuerzo, tiempo y personal necesarios para el desarrollo del software, también utilizaremos los modelos del VAN, TIR para obtener los beneficios a los que incurriría el laboratorio.

5.2. Análisis de costos

Se debe calcular todos los costos pronosticados asociados al sistema. Para establecer el costo del software desarrollado, se utiliza el modelo constructivo COCOMO II, que están orientados a los puntos de función. De la tabla 16 que vimos en el anterior capítulo tenemos que:

Cuenta total=261

Así también de la tabla 16 vimos que el factor punto función es:

Factor punto función=53

Para el cálculo del factor de complejidad técnica TCF, se toma en cuenta los datos de la tabla 16, para considerar la siguiente fórmula:

$$TCF=(0.65+0.01 \times 54)$$

$$TCF=1.19$$

El procesamiento de datos del punto función se basa en la formula siguiente:

$$PF=Cuenta\ Total \times TCF$$

$$PF=261 \times 1,19$$

$$PF=310,59$$

Conversión de los puntos de función a KLDC

Para determinar el esfuerzo nominal en el modelo COCOMO II los puntos función no ajustados tienen que ser convertidos a miles de líneas de código fuente considerando el lenguaje de implementación que se muestra en la tabla 23.

Tabla 23

Conversión de puntos función

LENGUAJE	NIVEL	FACTOR LDC/PF
C	2.5	128
Ansi Basic	5	64
Java	6	53
Ansi Cobol	3	107
Visual Basic	7	46
ASP	9	36
PHP	11	29
Visual C++	9.5	34

(Elaboración Propia)

Así con el valor que extraemos de la tabla 23 del valor del factor LDC/PF tenemos que:

$$\text{LDC} = \text{PF} \times \text{Factor}(\text{LDC}/\text{PF})$$

$$\text{LDC} = 307,98 \times 29$$

$$\text{LDC} = 8931,42$$

Las líneas de código en su totalidad son 8931,42, entonces el número estimado de líneas de código distribuidas en miles es:

$$\text{KLCD} = \text{LDC}/1000$$

$$\text{KLCD} = 8931,42/1000$$

$$\text{KLCD} = 8.93$$

Por tanto, existen 8.93 líneas de código distribuidas para el proyecto.

Ahora se aplican las fórmulas básicas de esfuerzo, tiempo calendario y personal requerido.

Las ecuaciones del COCOMO básico tienen la siguiente forma:

$$E = a_b(\text{KLDC})^{b_b}$$

$$D = c_b D_{db}$$

Donde:

E: Esfuerzo aplicado en personas por mes.

D: Tiempo de desarrollo en meses cronológicos.

KLDC: Número estimado de líneas de código distribuidas (en miles).

Tabla 24

Relación de valores del modelo COCOMO

Proyecto de software	bb	cb	db
Orgánico	1.05	2.5	0.38
Semi – acoplado	1.12	2.5	0.35
Empotrado	1.2	2.5	0.32

(Elaboración Propia)

En la tabla 24 se muestran los tipos de proyectos de software, como este es un proyecto intermedio, en tamaño y complejidad, se elige semi-acoplado.

De esta manera hallaremos el esfuerzo aplicado en personas por mes reemplazando los valores en la fórmula:

$$E = ab(KLDC)^{bb}$$

$$E = 3 \times (9.01)^{1.12}$$

$$E = 35.19$$

De la misma manera hallamos el tiempo de desarrollo en meses cronológico reemplazando los valores en la fórmula:

$$D = abD^{db}$$

$$D = 2.5 \times 35.19^{0.35}$$

$$D = 8.69$$

El personal requerido, en este caso el número de programadores (N° Prog) se obtiene con la siguiente fórmula:

$$N^{\circ} \text{ Prog} = E/D$$

$$N^{\circ} \text{ Prog} = 35.19/8.69$$

$$N^{\circ} \text{ Prog} = 4.05$$

$$N^{\circ} \text{ Prog} = 4 \text{ programadores}$$

El salario de un programador puede oscilar entre los Bs 3500, cifra que es tomada en cuenta para la estimación siguiente:

Costo del software desarrollado por persona = Numero de programadores * salario de un programador

$$\text{Costo del software desarrollado por persona} = 4 * 3500 \text{ Bs}$$

$$\text{Costo del software desarrollado por persona} = 14000 \text{ Bs}$$

De donde concluimos que para el desarrollo del “SISTEMA WEB PARA EL REGISTRO Y CONTROL DE OPERACIONES DEL LABORATORIO TECNICO DE SUELOS” CASO: SOCIEDAD DE INGENIEROS DE BOLIVIA DEPARTAMENTAL LAPAZ (SIB LA PAZ) es necesario contar con **4 programadores** durante **9 meses** para su respectivo desarrollo, el tendrá un **costo de 14000 Bs.**

5.3. Análisis de beneficios

Para evaluar los beneficios que se obtendrá al implementar el proyecto se calcula con el método VAN y el TIR.

El VAN o valor actual neto es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión.

La metodología consiste en descontar al momento actual (es decir, actualizar mediante una tasa) todos los flujos de caja futuros del proyecto. A este valor se le resta la inversión inicial, de tal modo que el valor obtenido es el valor actual neto del proyecto. La fórmula que utilizaremos para hallar el valor actual neto será:

$$VAN = \Sigma \text{Ganancias}/(1 + k)_n - \Sigma \text{costos}/(1 + k)_n$$

Dónde:

VAN: Valor Actual Neto

Ganancias: Ingreso de flujo anual

Costos: Salidas de flujo anual

n: Numero de periodo

k: Tasa de descuento o tasa de interés al préstamo

Los gastos y ganancias que se estiman en un lapso de 4 años los mostramos en la Tabla 25, para este caso en particular utilizamos una de descuento del 12% ya que es la tasa actual de interés del préstamo en las entidades financieras.

Tabla 25

Cálculo del VAN

Año	Costos	Ganancias	Costos/ $(1+k)_n$	Ganancia s/ $(1+k)_n$
1	6700	0	5982	0
2	1100	2000	876.09	1594.4
3	620	3500	441.3	2491.2
4	300	5000	190.7	3177.6
Σ	7670	10500	6553.5	7263.2

(Elaboración Propia)

reemplazando en la formula tenemos que:

$$VAN = \Sigma(Ganancias/(1 + 0.12)^n) - \Sigma(Costos/(1 + 0.12)^n)$$

$$VAN = 7263.2 - 6553.5$$

$$VAN = 709.7$$

Para ver si el proyecto es rentable nos fijamos en la siguiente tabla:

Tabla 26

Criterio de interpretación del VAN

Valor del VAN	Interpretación
VAN>0	El proyecto es rentable
VAN=0	El proyecto también es rentable, ya que se incorpora la ganancia de la tasa de interés.
VAN<0	El proyecto no es rentable.

(Elaboración Propia)

Entonces podemos ver que el proyecto si es rentable según la tabla 26 Puesto que el valor de VAN es mayor a cero.

5.4. Costo /Beneficio

Para hallar el costo/beneficio de un proyecto se aplica la siguiente ecuación:

$$\text{Costo/Beneficio} = \Sigma \text{ Ganancias} / \Sigma \text{ Costos}$$

Reemplazando en la ecuación anterior los valores conocidos de la tabla anterior.

$$\text{Costo/Beneficio} = 10500 / 7670$$

$$\text{Costo/Beneficio} = 1.4\$us$$

Con este resultado interpretamos de la siguiente manera: por cada dólar invertido en el proyecto de software la institución genera una ganancia de 0.4 \$us.

5.5. Tasa interna de retorno

Cuando en la fórmula del VAN el valor de “k” es igual a “0” pasa a llamarse T.I.R. (Tasa Interna de Retorno). La T.I.R. es la rentabilidad que nos proporciona al proyecto.

La ecuación que utilizaremos es la siguiente:

$$TIR = \Sigma((\text{Ganancias} - \text{Costos}) / (1 - i)^n)$$

Dónde:

TIR: Tasa Interna de Retorno

Ganancias: Flujo de entrada de un periodo

Costos: Flujo de salida de un periodo

i: Tasa de interés al ahorro

n: Numero de periodo

La tabla 27 muestra una mejor comprensión de ecuación T.I.R. y expresando los resultados encontrados en las misma.

Tabla 27

Determinación del TIR

Año	Costo	Ganancias	(Ganancias – Costos)/ (1 – i)ⁿ
1	7047	0	6420.5
2	1100	2000	1162.2
3	620	3500	4226.1
4	300	5000	7837.3
TIR			6805.1

(Elaboración Propia)

Entonces vemos que el proyecto nos dará una rentabilidad de 6805.1 \$us.

6. Seguridad del Sistema

6.1. Introducción

La seguridad de la información es el conjunto de medidas preventivas y reactivas de las organizaciones y de los sistemas tecnológicos que permiten resguardar y proteger la información buscando mantener la confidencialidad, la disponibilidad e integridad de la misma.

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

Para el hombre como individuo, la seguridad de la información tiene un efecto significativo respecto a su privacidad, la que puede cobrar distintas dimensiones dependiendo de la cultura del mismo.

La seguridad de la información es un aspecto muy importante que debemos tomar en cuenta para que todos los datos que se almacenen en el sistema estén seguros de esta forma nos enfocaremos en la seguridad física y lógica.

6.2. Seguridad física

Es inevitable que el personal tenga acceso físico a las máquinas sobre las que deben trabajar, cuando el usuario debe usar el hardware directamente, como usando el lector de CDROMs o similares la máquina que alberga estos dispositivos debe estar cercana al usuario. Revisando este y muchos factores más se tiene lo siguiente para evitar que haya un daño físico:

- Se cuenta con estabilizadores de energía en caso de que haya una sobre tensión en los equipos.
- También se cuenta con un extintor contra incendios
- Se tiene acceso restringido a personas que no sean de confianza

6.3. Seguridad lógica

- Seguridad a nivel Sistema Operativo se tiene una computadora de uso personal con el S.O. Windows 7 y sus configuraciones necesarias de red segura, como cortafuego activado, ssh y seguridad en IP.
- Seguridad a nivel Base de Datos a nivel de base de datos se aplica la seguridad de copias de seguridad, acceso y permiso a cuentas de usuario.
- Seguridad a nivel Software al ser un sistema web privado, se tiene un acceso de usuario (Login). Y un código de seguridad para evitar acceso a personas no autorizadas al sistema.

7. Conclusiones y Recomendaciones

7.1. Conclusiones

A partir del cumplimiento de los requerimientos establecidos por el laboratorio, se ha cumplido con los objetivos planteados por medio de la implementación del Sistema de información para la administración de órdenes a través de los diferentes módulos desarrollados para este proyecto.

Tomando en cuenta los objetivos planteados en el capítulo 1, se llegó a las siguientes conclusiones:

- Se estudió las diferentes falencias para mejorar el control de la información de el laboratorio.
- Se ha desarrollado con la metodología RUP e implementado satisfactoriamente de acuerdo a los requerimientos técnicos establecidos, además de las pruebas de calidad.
- Se ha diseñado una base de datos que permite el almacenamiento de la información de forma centralizada mediante el SGBD MySQL y mediante los modelos de seguridad que nos proporciona el framework CakePhp , además de establecer las políticas de la misma para su uso adecuado.
- Se desarrolló los módulos de: registro, búsqueda y reportes de las diversas ordenes que tiene el laboratorio, evitando así la verificación manual de errores en documentos.

- Con el sistema implementado se proporcionará información rápida a la comunidad.

Además, se ha podido completar a los anteriores puntos lo siguiente:

- Se cuenta con la información oportuna y actualizada para la obtención de reportes requeridos.
- Con la implementación del sistema se logró reducir el tiempo que se empleaba en la realización de reportes.
- Información precisa y confiable.
- Interfaz amigable para el usuario con pantallas compresibles y de fácil manejo.

7.2. Recomendaciones

El sistema fue desarrollado cumpliendo con requisitos solicitados por los usuarios pero aún quedan otras funcionalidades que se tiene que complementar para tener un control absoluto de la información que genera la institución.

Se recomienda realizar lo siguiente:

- Desarrollar el subsistema contable para que así la integración con el sistema esté completa y de esta manera se pueda controlar mejor.

- Implementar un módulo para el control y generación de horarios de salidas a visitas.
- Desarrollar un módulo de interacción directo con el cliente una vez se tenga un hosting.

Bibliografía

Abud, F. (2012). *Calidad en la Industria del Software La Norma ISO-9126*.

Obtenido de

<http://www.nacionmulticultural.unam.mx/empresasindigenas/docs/2094.pdf>

Arevalo, J. (2007). *Gestion de la Informacion, Gestion de Contenidos y Conocimientos*. Salamanca, España: Universidad de Salamanca.

Aula Mentor Gobierno de España Ministerio de Educación, Cultura y Deporte.

(2012). *Apache*. Obtenido de

<http://descargas.pntic.mec.es/mentor/visitas/Apache.pdf>

Cake Software Foundation. (2020). *CakePHP Cookbook Documentation*. Obtenido de https://book.cakephp.org/2/_downloads/es/CakePHPCookbook.pdf

Campderrich, B. (2003). *Ingeniería del Software*. Barcelona, España: Uoc.

Davis, G. (1985). *Sistemas de Información Gerencial: Fundamentos Conceptuales, Estructura y Desarrollo*. New York, EEUU: Mc Graw Hill.

Dominguez, L. (2012). *Analisis de Sistemas de Informacion*. Estado de Mexico, Mexico: Red Tercer Milenio.

Fowler, M., & Kendall, S. (1999). *UML gota a gota*. Mexico: Pearson Educación S.A.

Gallego, A. (2018). *Introducción al diseño Responsive*. España: University of Alicante.

Gilfillan, I. (2003). *La Biblia de MySQL*. España: Anaya Multimedia.

Kendall, K., & Kendall, J. (2011). *Análisis y Diseño de Sistemas*. Mexico: Pearson Educación S.A.

Kimmel, P. (2008). *Manual de UML*. Ciudad de México, México: Miembro de la Cámara Nacional de la Industria Editorial Mexicana.

Kruchten, P. (2004). *The Rational Unified Process An Introduction*. EEUU: Pearson Educacion S.A.

Larman, C. (1999). *UML Y Patrones, Introduccion al analisis y diseño orientado a objetos*. Mexico: Pearson.

Laudon, K., & Laudon, J. (2016). *Sistema de Información Gerencial*. Mexico: Pearson Educación S.A.

Pressman, R. (s.f.).

Pressman, R. (2010). *Ingeniería del Software Un Enfoque Practico*. Ciudad de México, México: Mc Graw Hill Educación.

Sommerville, I. (2005). *Ingeniería del Software*. Madrid, España: Pearson Educación S.A.

Universidad de Allcante. (2002). *Productos para Desarrollar Web: ASP, CFM, JSP y PHP*. Obtenido de <https://rua.ua.es/dspace/bitstream/10045/14015/4/2-asp-cfm-jsp-php.pdf>

Universidad de Asuay. (2006). *Comparacion entre MySQL vs. PostgreSQL*.

Obtenido de <http://dspace.uazuay.edu.ec/bitstream/datos/2169/1/05291.pdf>

Vaswani, V. (2010). *Fundamentos de Php*. Ciudad de Mexico, Mexico: Mc Graw Hill Educación.