UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMATICA

PROYECTO DE GRADO

"SISTEMA WEB DE CONTROL Y ADMINISTRACIÓN DE ACTIVOS FIJOS DEL SENAPE"

SERVICIO NACIONAL DE PATRIMONIO DEL ESTADO

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMATICOS

POSTULANTE : MIGUEL ÁNGEL TICONA QUISPE
TUTOR METODOLOGICO : MSC. JORGE TERÁN POMIER
ASESOR : LIC. RAMIRO FLORES ROJAS

LA PAZ – BOLIVIA
2013

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

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

LICENCIA DE USO

El usuario está autorizado a:

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

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

DEDICATORIA

El presente proyecto lo dedico a Dios por regalarme la vida y porque nunca nos desampara.

A mi esposa Teresa Jackeline, por apoyarme en todo momento con amor y paciencia.

A mi mamá Victoria y a mi hermano José, por apoyarme en todo momento para que pueda lograr mis objetivos.

AGRADECIMIENTOS

Agradezco a Dios con toda mi alma y mi corazón porque nunca me desampara.

Agradezco a la Universidad Mayor de San Andrés – UMSA, por haberme acogido y brindado una educación profesional.

Gracias a mi Tutor Msc. Jorge Terán Pomier, por su colaboración y su tiempo, en el asesoramiento del presente proyecto.

Gracia a mi Revisor Lic. Ramiro Flores, por su apoyo en el desarrollo del presente proyecto con sus conocimientos profesionales.

Gracias a la Dra. Fabiola Consuelo Salazar Calle, Directora General Ejecutiva del Servicio Nacional de Patrimonio del Estado – SENAPE, por brindarme su apoyo permitiéndome desarrollar el presente proyecto en tan prestigiosa Institución.

Gracias a mi esposa Teresa Segales, por estar a mi lado compartiendo mis triunfos y fracasos, por entenderme, apoyarme en todo momento, porque con su paciencia y su amor supo darme las fuerzas necesarias para continuar.

Gracias a mi mamá Victoria Quispe, por apoyarme en todo momento, por tener siempre fe en mí, por su paciencia, por sus enseñanzas, por apoyarme siempre para que logre mis objetivos.

Gracias a mi hermano José Ticona, quien con sus consejos y su apoyo me lleno de confianza para seguir adelante.

Gracias a todos aquellos que me apoyaron a lo largo de mis estudios, amigos que día a día pasamos pruebas difíciles, momentos inolvidables y estuvieron ahí, muchas gracias.

RESUMEN

El proyecto de grado presenta el desarrollo del Sistema de Control y Seguimiento de Activos Fijos, elaborado para el Área de Activos Fijos del Servicio Nacional de Patrimonio del Estado – SENAPE, con el fin de mejorar los procesos de registro, asignación, transferencia, bajas, actualización de los Activos Fijos Revaluados (por peritos en la materia), depreciación de los Activos que prestan servicios a la institución. En el Área de Activos Fijos se maneja la información sobre los bienes tangibles e intangibles que se tienen en las diferentes dependencias del SENAPE, tanto en nuestra ciudad, así como también en las distritales ubicadas en el interior del país.

El sistema logró concretar los procesos principales de esta área, lo cual ayuda a generar reportes y consultas de acuerdo a los requerimientos del usuario para reducir el tiempo para la elaboración de informes. También logró integrar la información ayudando a la generación de seguimiento para cada uno de los activos fijos y de esta forma tener un mejor control y sea un apoyo para la toma de decisiones.

Para el desarrollo del sistema se aplicó la metodología Scrum. Durante la conclusión de cada Sprint (periodos cortos de trabajo) se presentó partes del sistema terminados, producto de la metodología Scrum que es iterativo e incremental y la facilidad de coordinar en todo momento con el responsable de Activos Fijos, de esta forma se cumplieron con las expectativas del usuario.

De esta manera, una vez concluido el sistema se pudo evidenciar que cumple con los objetivos planteados y los requerimientos establecidos por el Área de Activos Fijos.

ÍNDICE

CAPÍTULO I INTRODUCCIÓN

CAPÍTULO II MARCO TEÓRICO

CAPÍTULO III MARCO APLICATIVO

PÁG

CAPÍTULO IV CONCLUSIONES Y RECOMENDACIONES

PÁG

ANEXOS

ÍNDICE DE FIGURAS

FIGURA **PÁG**

ÍNDICE DE TABLAS

PÁG

CAPÍTULO 1
INTRODUCCIÓN

SISTEMA WEB DE CONTROL Y ADMINISTRACIÓN DE ACTIVOS FIJOS DEL SENAPE

1.1 INTRODUCCIÓN

Hoy en día se exige que las organizaciones e instituciones públicas mejoren el uso de tecnología a fin de lograr mejores resultados en los procesos que realizan. Los problemas de automatización en la sociedad boliviana están en constante crecimiento y desarrollo, el uso transparente y eficiente de los recursos informáticos es una actividad de máxima prioridad ya que establecen un ambiente de control.

El Servicio Nacional de Patrimonio del Estado – SENAPE, es una entidad desconcentrada del Ministerio de Economía y Finanzas Públicas, encargada de formular normas y procedimientos para el proceso de registro, preservación, y saneamiento legal de los bienes públicos, tangibles o intangibles cuya propiedad corresponde al Estado boliviano y vigilar su cumplimiento. El Responsable del Área de Activos Fijos, es el responsable ante el Máximo Ejecutivo de la Unidad Administrativa dependiente de la Dirección Administrativa y Financiera del SENAPE, de cumplir las normas, reglamentos, procedimientos/instructivos establecidos para el desarrollo de sus funciones, controlar y demandar servicios de mantenimiento y salvaguarda de bienes del SENAPE.

Sin embargo, debido a que el Área de Activos Fijos sólo cuenta con una base de datos en las hojas de cálculo de Excel, los procesos son realizados con pérdida de tiempo, bajo rendimiento en el control y seguimiento de los activos fijos, es por ello que en procura de mejorar la atención a los usuarios, de transparentar el accionar público e incrementar la eficiencia, se ha propuesto el desarrollo de un sistema informático para ofrecer sus diferentes servicios en tiempo real y a través de Internet.

Con el desarrollo e implementación del presente proyecto denominado Sistema Web de Control y Administración de Activos Fijos del SENAPE, se pretende administrar y controlar los Activos Fijos de manera eficaz y eficiente con la finalidad de tener información detallada, consolidada a cerca de movimientos físicos y económicos, generando reportes de altas, bajas, transferencias, depreciaciones en cada gestión, para la toma de decisiones acertadas.

1.2 ANTECEDENTES

1.2.1 Antecedentes de la Institución

El Servicio Nacional de Patrimonio del Estado fue creado por la Ley LOPE (Ley de Organización del Poder Ejecutivo) Nº 1788 de 16 de septiembre de 1997.

Es una entidad desconcentrada del Ministerio de Economía y Finanzas Públicas, con dependencia funcional del Viceministerio de Tesoro y Crédito Público reglamentado con el Decreto Supremo N° 25152 de 4 de septiembre de 1998.

Se consolida con la nueva estructura el 22 de diciembre de 2005 con el Decreto Supremo Nº 28565.

1.2.2 Misión

El SENAPE de acuerdo al artículo 3° del Decreto Supremo N° 28565 de 22 de Diciembre de 2005, tiene la misión de efectuar el registro de todos los bienes del Estado y promover el saneamiento y valoración de los mismos. Asimismo, la misión de disponer de todos los bienes recibidos de otras instituciones, administrar el activo exigible de las entidades disueltas o en proceso de liquidación, y concluir los procesos de liquidación de ex entidades estatales y ex entes gestores de la seguridad social, conforme a disposiciones legales vigentes.

1.2.3 Visión

Institución fiable, transparente, oportuna y eficiente, reconocida como máxima autoridad en el registro, promovedora del saneamiento legal y protectora de los bienes públicos, apoyada en modernas tecnologías de información y gestión en recursos humanos, técnicos y administrativos.

1.2.4 Antecedentes bibliográficos

Luego de una revisión bibliográfica en nuestro medio se han identificado algunos proyectos relacionados al presente trabajo.

- **“Control y Seguimiento de Activos Fijos del Servicio Exterior vía Web – Ministerio de Relaciones Exteriores y Culto”**, realizado por Mario Canchillo Zanga, en la gestión 2006, su objetivo general es, desarrollar e implementar el Sistema de Control y Seguimiento de Activos Fijos vía Web centralizado en la ciudad de La Paz, como parte integrante del SIGASE, para tener un eficiente control y seguimiento de los bienes.

- “**Sistema de control de Activos Fijos para el Gobierno Municipal de El Alto**”, realizado por Epifanio Mamani Calle en la gestión 2007, su objetivo general es, desarrollar e implementar el Sistema de Información Automatizado de Activos Fijos, que permite optimizar el control del movimiento físico y económico de los activos fijos del municipio de El Alto, coadyuvando en la buena toma de decisiones y oportuna en GMEA.

- **“Sistema de control y seguimiento de Activos Fijos – Ministerio de Gobierno”**, realizado por Ramiro Máximo Mamani Cahuna en la gestión 2007**,** su objetivo general es diseñar e implementar un Sistema de Información automatizado que permita mejorar el control y seguimiento de Activos Fijos del Ministerio de Gobierno, en base a las normas básicas del Sistema de Administración de Bienes y Servicios, para tener información actualizada y centralizada de los Bienes pertenecientes de la institución.

- **“Sistema de Información de Seguimiento y Gestión de Activos Fijos”**, realizado por Humberto Condarco Alejo, en la gestión 2008, su objetivo general es diseñar, desarrollar e implantar un Sistema de Información que permita el Seguimiento y Gestión de Activos Fijos, reduciendo el tiempo en los procesos de manejo de información de la unidad en cargada de Activos Fijos de la Superintendencia de Electricidad.

- **“Sistema de Información y Control de Activos Fijos para la Fundación La Paz (Área promoción de la mujer)”**, realizado por Edwin Quispe Quispe, en la gestión 2009, su objetivo general es, implementar un Sistema de Información y Control de Activos Fijos para la Fundación La Paz (Área Promoción de la Mujer).

1.2.5 Cómo funciona el sistema actual

Actualmente el Área de Activos Fijos cuenta con una base de datos en Excel, mismo que por la cantidad de información que es generada producto de los diferentes procesos

como ser: recepción de Activos Fijos, asignación de Activos Fijos, codificación de Activos Fijos, etc., provoca rezago en el desarrollo de todas las funciones de esta área.

El Área de Activos Fijos maneja la información sobre bienes tangibles que se tienen en las diferentes Direcciones del SENAPE, llevando a cabo la administración y control de todos ellos. Los procedimientos que se realizan son los siguientes:

- Procedimiento de recepción de Activos Fijos.
    1. Contratación menor
    2. Apoyo Nacional a la Producción y Empleo
- Procedimiento de registro de Activos Fijos.
    1. Registro de Activos Fijos adquiridos/comprados.
    2. Registro de Activos Fijos donados/transferidos.
    3. Registro de Activos Fijos asignados/prestados.
    4. Asignación de Activo Fijo.
    5. Asignación de Activos Fijos a personal de planta.
    6. Asignación de Activos Fijos por cambio a otra sección.
    7. Asignación de Activos Fijos a consultores.
    8. Solicitud y asignación de Activos Fijos personal de planta/consultores.
    9. Devolución de Activo Fijo.
    10. Devolución de Activo Fijo funcionario de planta
    11. Salida de bienes.
    12. Ingreso temporal de bienes (bienes que no son de propiedad del SENAPE).
    13. Toma de inventarios.
    14. Formularios de control de Activos Fijos.

Pese al gran esfuerzo que el Área de Activos Fijos realiza para cubrir con todos los requerimientos del SENAPE, en cuanto al registro, consulta, etc. esta área necesita un mayor control de procedimientos que se realiza al interior de la misma.

1.3 PROBLEMA

1.3.1 Situación problemática

El Área de Activos fijos presenta una serie de falencias en el trabajo que realiza, mismos que se detallan a continuación:

1. Existe demora en el procedimiento de registro de nuevos activos fijos.
2. No cuenta con la debida seguridad de los datos, siendo que personas con acceso a la computadora pueden alterar datos importantes (mismos que se encuentran en hojas de cálculo del archivo Excel).
3. El procedimiento de depreciación de Act. Fijos no es realizado oportunamente.
4. Existen falencias en la conciliación de cuentas y cierre de balance de gestión.
5. Existe demora en el registro de ingreso/salida de activos fijos.
6. El control de los activos fijos es muy complicado debido a que la información generada por los diferentes procedimientos es voluminosa.
7. Existe demora en la entrega de reportes con información detallada de los activos Fijos para apoyo en la toma de decisiones.
8. Existe rezago en los procedimientos de asignación, devolución de los Activos Fijos.
9. No cuenta con registro histórico de los activos que han sido revalorizados, transferidos o dados de baja.
10. Existe rezago en el desempeño de las funciones debido a que el manejo de la información es de forma manual y/o semi-automatizada.

1.3.2 Planteamiento del problema

¿El Sistema Web de Control y Administración de Activos Fijos del SENAPE, optimizará la administración, asignación, seguimiento y control de los Activos Fijos de esta Institución?

1.4 OBJETIVOS

1.4.1 Objetivo general

Desarrollar un sistema informático web que permita optimizar los procesos de administración, asignación, seguimiento y control de los Activos Fijos, del Área de Activos Fijos del SENAPE.

1.4.2 Objetivos específicos

Los objetivos específicos para que el sistema realice procesos confiables son:

1. Diseñar una base de datos para introducir toda la información de los Activos Fijos.

2. Realizar el modulo para autentificación de usuarios.
3. Realizar un módulo para poder introducir la información de los activos fijos que sean sujetos a revalúo por parte de la empresa consultora y que permita realizar la depreciación de los mismos de manera diaria.
4. Realizar un módulo que permita generar reportes de acuerdo a formatos establecidos para enviar al departamento de contabilidad para su respectiva contabilización en el balance general década gestión.
5. Realizar un módulo para el control de ingresos y salidas de activos al SENAPE.
6. Realizar módulos que permitan generar consultas de acuerdo a los requerimientos del Encargado del Área de Activos Fijos.
7. Realizar un módulo para la asignación y devolución de Activos Fijos a los funcionarios del SENAPE.
8. Realizar un módulo que permita introducir la información de los activos Fijos dados de baja o revaluados de acuerdo a informes técnicos elaborados por empresas consultoras contratadas para tal efecto.

1.5 OBJETO DE ESTUDIO

El presente Proyecto de Grado tiene como objeto de estudio, al Área de Activos Fijos del SENAPE, con la finalidad de dar soluciones a diferentes falencias mencionadas anteriormente.

1.6 JUSTIFICACIÓN

1.6.1 Justificación técnica

En la institución se cuenta con equipos computacionales suficientes para el desarrollo e implementación del sistema por estar de acuerdo con los requerimientos técnicos.

1.6.2 Justificación económica

El sistema incrementara los beneficios de la Institución generando un mejor seguimiento y control en los Activos Fijos, reducirá la perdida de tiempo y los posibles errores.

En la elaboración del proyecto utilizaremos “software libre” por lo que el desarrollo del sistema no tiene costo monetario

1.6.3 Justificación social

Con la implementación del sistema se automatizara los procesos manuales realizados en el Área de Activos Fijos, lo que permitirá obtener información confiable rápida y oportuna de esta manera el personal a cargo de interactuar con el sistema incrementa su nivel productivo y competitivo y ayuda de manera indirecta a los funcionario que requieren de información de esta área.

1.7 DISEÑO METODOLÓGICO

1.7.1 Herramientas de desarrollo

Debido a los requerimientos de la institución de entregas de productos usables en el menor tiempo posible, se adopto una Metodología Ágil y más interactiva con el cliente, la Metodología adoptada es SCRUM.

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Scrum es una metodología ágil, y como tal:

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

Se denominada Sprint a cada iteración o sucesiones de ciclos cortos de trabajo, cada Sprint tiene una duración preestablecida de entre 2 y 4 semanas, obteniendo como resultado una versión del software con nuevas prestaciones listas para ser usadas.

1.7.2 Requerimiento de hardware

El Área de Activos Fijos cuenta con dos equipos con las siguientes características, mismas que son suficientes para la implementación del sistema Web.

- Microprocesador core i3 de 2.6 Ghz de velocidad.
- Memoria RAM de 4 Gb.

- Impresora

1.7.3 Requerimiento de software

Para el desarrollo de las aplicaciones se utilizará el lenguaje de programación PHP, debido a que es un lenguaje de script de código abierto para servidores que puede ser descargado gratuitamente, está diseñado específicamente para el desarrollo y producción de páginas Web, utiliza amplia gama de servidores y es independiente del sistema operativo.

Como Gestor de Base de Datos, para el almacenamiento y consulta de datos, se utilizará Postgresql, debido a que su código fuente está disponible libremente, el motor de bases de datos es de código abierto, es el más potente del momento y en sus últimas versiones empieza a no tener que envidiarle nada a otras bases de datos comerciales.

Como servidor de prueba se utilizará el Xampp, debido a que es un servidor independiente de plataforma, de software libre, que consiste principalmente en la base de datos Mysql, servidor Web apache, y los intérpretes para el lenguaje de script PHP.

1.8 Alcance y límite

El presente proyecto de grado se limita a la administración automatizada de la información del área de Activos Fijos del SENAPE, dentro de las Normas Básicas del Sistema de Administración de Bienes y Servicios (NB - SABS) del capítulo III Administración de Activos Fijos Muebles y el Manual de Operaciones de Activos Fijos del SENAPE, sobre registro y control de Activos Fijos.

El presente proyecto se centrará en administrar y controlar el estado de los activos fijos mediante los siguientes procedimientos:

- Registro de nuevos activos fijos que pueden ser adquiridos, donados, transferidos.
- Adición de Activos Fijos revaluados
- Asignación y devolución de activos fijos tanto a personal de planta como a consultores.
- Asignación de activos fijos por cambio de sección.
- Procedimiento de salida de activos fijos.

- Procedimiento de ingreso temporal de activos fijos (Bienes que no son de propiedad del SENAPE).
- Generar reportes de actualización y depreciación
- Baja de Activos Fijos
- Controlar accesos al sistema
- Registrar tipo de cambio
- Realizar la depreciación de Activos Fijos.

1.9 APORTES

En términos de la utilidad, el proyecto permitirá optimizar la Administración y Gestión de Activos Fijos en los siguientes aspectos:

Automatización de los procesos manuales actuales del Área de Activos Fijos en procesamiento de información al momento, coadyuvando en el registro, actualización, inventariación, asignación, depreciación, bajas, y el suministro de información para la toma de decisiones.

A partir de las revisiones y comprobaciones de obtención de información permitirá realizar el análisis, estudios y emprender acciones correctivas o la toma de decisiones a fin de cumplir con los objetivos de la Institución, coadyuvando en la regulación de los procesos administrativos en base a la normativa que exige la Ley Safco (Ley 1178).

CAPÍTULO 2
MARCO TEÓRICO

2 .1 INTRODUCCIÓN

El capítulo describe los conceptos de activos fijos que se aplicarán en la elaboración del presente proyecto; además, se hace referencia a la metodología de desarrollo implementada "metodología Scrum", también se describe las herramientas para el desarrollo del software. Y finalmente se definen los parámetros para definir las métricas de calidad del software.

2.2 ACTIVO FIJO

Se denomina Activo Fijo o Bien en Uso a aquellos de naturaleza permanente, con vida útil superior a un año y que tiene un valor relativamente significativo. Comprende bienes materiales como bienes inmuebles, equipos de oficina, adquiridos por la entidad con recursos propios, recibidos en donación o transferencia (en el caso del SENAPE, recibidos por entidades en proceso de liquidación), que no se agotan en su primer y poco uso, ni cambian de forma y son incorporados a la institución para operaciones propias o desarrollo especifico de las actividades de las entidades públicas sin ninguna intención de ser vendidas a futuro. [MOAF del SENAPE, 2012][1]

2.2.1 Clasificación de los Activos Fijos

Los activos fijos generalmente se clasifican en dos grandes grupos, activos fijos intangibles (bienes inmateriales) y activos fijos tangibles (bienes materiales).

2.2.1.1 Activos Fijos Intangibles

Constituyen aquel conjunto de bienes que no poseen forma corpórea, no ocupan un lugar en el espacio; razón por la cual, no se pueden ver, tocar, pesar. Su valor no reside en una propiedad física, sino en los derechos que su posesión confiere a su propietario. También se puede mencionar a los costos de estudios, planos, patentes de inversión, franquicias, concesiones, etc., debiendo estos amortizarse, de acuerdo a la vigencia de las mismas y/o en cumplimiento a las normas vigentes Ejemplo: derechos de autor, patentes, licencias, etc.

---

[1]MOAF DEL SENAPE: Manual de Operaciones de Activos Fijos – Primera versión

2.2.1.2 Activos Fijos Tangibles

Un activo fijo tangible si tiene sustancia corpórea ocupa un lugar físico en el espacio, razón por la cual se puede ver, tocar, pesar, etc. Además de tener un valor monetario, los mismos para fines contables son sujetos de depreciaciones y agotamiento. Ejemplo: Equipos de computación, vehículos, muebles, etc [MOAF del SENAPE, 2012]

2.2.2 Objetivo de la administración de Activos Fijos

El objetico de la administración de Activos Fijos, es establecer los lineamientos y los controles administrativos respectivos que optimicen la disponibilidad, el uso, conservación, el control físico-valorado del activo fijo y la minimización de los costos de sus operaciones de los bienes de uso en las entidades públicas, en base a las disposiciones legales técnicas y contables que permita generar información confiable para la toma de decisiones respecto a futuras adquisiciones, mantenimientos, reparaciones, disposición o baja y así alcanzar un manejo eficiente de Activos Fijos. [MOAF del SENAPE, 2012]

2.2.3 Bienes Fungibles

Se denomina bienes fungibles, a los bienes que se consumen con el uso inmediato, pierden su valor o cambian de forma, en la gran cantidad de casos no pueden ser utilizados nuevamente.

Pertenecen a este grupo los bienes que su vida útil estimada es inferior a un período (un año). Los bienes fungibles no son sujetos a revalorización técnica ni contable.

El Área de activos fijos deberá llevar en lo posible un control de aquellos fungibles que son dependientes o forman parte de un Activo Fijo y sean necesarios para el funcionamiento del mismo. [MOAF del SENAPE, 2012]

2.2.4 Verificación del Estado Técnico del bien

Se debe efectuar en el lugar de trabajo considerando coeficientes técnicos para determinar si el bien está en condiciones técnicas de pasar a la siguiente etapa de la

inventariación o ser excluido del proceso a un formulario de bien en mal estado. Como producto del proceso técnico, practicado por el equipo de inventariadores.

Se usa una simbología aplicada como propuesta técnica de la evaluación del estado de los bienes que generalmente toma en cuenta cinco categorías a ser consideradas:

- **Excelente - nuevo:** se considera a aquel Activo que aún no ha sido objeto de uso, debiendo estar al 100% con sus características originales de fabricación.
- **Bueno:** se considera como bueno, aquel Activo que se encuentra en perfecto estado de operación, funcionamiento, conserva sus características originales y cuyo estado de conservación se ha mantenido invariable a través del tiempo.
- **Regular:** Será considerado como regular, aquel Activo cuyo grado de conservación se ve alterado en algunas de sus características, como consecuencia del tiempo que ha transcurrido o del uso en sí mismo.
- **Malo:** Será considerado como malo, aquel Activo que presenta un estado físico de deterioro significativo, no cumplen con las condiciones tecnológicas o de funcionalidad pero aún presta servicios.
- **Pésimo – dar de baja:** Será considerada Pésimo a ser dado de baja, aquel Activo que no cumple con las condiciones tecnológicas o de funcionalidad o tiene un alto nivel e obsolescencia tecnológica, además ya no presta servicio, posee una condición de inutilización, no tiene valor de uso ni económico.

2.2.5 Criterios para la Descripción de Bienes de Uso

Para la descripción de los Activos Fijos se debe utilizar los siguientes criterios.

- Rubro: Equipos de Oficina y Muebles.
- Rubro: Equipo de Comunicación.
- Rubro: Equipo Educacional y Recreativo.
- Rubro: Equipo de computación.

2.2.6 Codificación

Para controlar la distribución de los bienes. El Responsable de Activos Fijos adoptará sistemas de identificación interna mediante código, claves o símbolos que:

- Permitan la identificación, ubicación y el destino del bien.
- Que diferencien claramente un bien del otro.

- Sea compatible con el sistema de Acticos Fijos.
- Faciliten el recuento físico.

La codificación de Activos Fijos debe basarse a la descrita posteriormente:

Código de la institución	Ubicación Geográfica	Grupo contable	Auxiliar contable	Correlativo
NNN	NN	NN	NN	NNN

Tabla 2.1 Codificación de Activos Fijos.
Fuente: [MOAF del SENAPE, 2012]

2.2.6.1 Código de la Institución:

Se considera el Nº 035, que le asignó el Ministerio de Economía y Finanzas Públicas.

2.2.6.2 Ubicación Geográfica:

Se realiza de acuerdo al siguiente detalle:

CÓDIGO	DESCRIPCIÓN
01	OFICINA CENTRAL
02	DLEGSS
03	DEPÓSITO EL ALTO
04	DISTRITAL BENI
05	DISTRITAL COCHABAMBA
06	DISTRITAL CHUQUISACA
07	DISTRITAL ORURO
08	DISTRITAL SANTA CRUZ

Tabla 2.2 Codificación de Ubicación Geográfica.
Fuente: [MOAF del SENAPE, 2012]

2.2.6.3 Grupo del Activo Fijo

En base al Decreto Supremo Nº 24051 de fecha 29 de junio de 1995, Impuesto Sobre Utilidades de las Empresas, y al Clasificador de Bienes con instrucción Administrativa SNPE/DAF-001/2011.

Por ejemplo:

CÓDIGO	DESCRIPCIÓN
01	EQUIPO DE OFICINA Y MUEBLES
02	EQUIPO DE COMPUTACIÓN
03	EQUIPO DE COMUNICACIÓN
04	EQUIPO EDUCACIONAL Y RECREATIVO

Tabla 2.3 Codificación de Grupo Contable.
Fuente: [MOAF del SENAPE, 2012]

2.2.6.4 Auxiliar del Activo Fijo

Representa la sub-agrupación de los tipos de Activos Fijos, en función al Clasificador de Bienes con instrucción Administrativa SNPE/DAF-001/2011. [MOAF del SENAPE, 2012][1]

Por ejemplo:

CÓDIGO	DESCRIPCIÓN
01	AIRE ACONDICIONADO
02	ASPIRADORA
03	BANCA
04	BANCO
05	CAFETERA

Tabla 2.4 Codificación de Auxiliar de Activos Fijos.
Fuente: [MOAF del SENAPE, 2012]

2.2.7 Vida útil

Un activo fijo tangible (bienes materiales) de una organización siempre tienden a un desgaste físico con el transcurrir del tiempo, concluyendo con una vida útil preestablecida de uso.

La vida útil de un activo se estima en base a la vida física del bien en sí y del tiempo que se ha planificado para su utilización, dependiendo de la clase de uso que se le dé. Dentro de la planificación, se puede prever el uso de un equipo aun en condiciones de trabajo por tiempo limitado, pues la política de la empresa podría establecer el cambio de ciertos equipos cada cierto tiempo, con el fin de obtener un valor más alto en su reventa, o cambiar con equipos que con nuevos adelantos técnicos sean menos costosos de operarlos, o sea más productivos, cuidando de no caer en la obsolescencia de los activos. Otros aspectos puede referirse a la posible substitución del producto fabricado por otro más conveniente.

La vida útil de un bien material de una organización (activo fijo tangible) se estima de la forma más apropiada posible, el número posible de unidades que se producirá, si el factor más importante es la producción y no precisamente el tiempo.

2.2.8 Método de depreciación

Se denomina método de depreciación a la mecánica utilizada para distribuir el costo actualizado del bien tomando en cuenta sus años de vida estimados, horas de trabajo, unidades de producción o cualquier otro parámetro.

Estableciendo un monto depreciable, este podrá distribuirse ya sea en función de la producción o en función del tiempo de vida predeterminado, dividiendo el monto depreciable entre el número posible de unidades a producirse, el mismo que podrá ser unidades del producto en sí o unidades expresadas en términos de horas. [Palenque, 2001]

Un método de depreciación nos permite obtener un costo actualizado del activo fijo en uso, tomando en cuenta los años de vida útil preestablecidos, sus horas de trabajo y unidades de producción, entre los métodos de depreciación más usados se tienen:

- Método de línea recta.
- Método de las unidades de producción.
- Método exponencial.

2.2.9 Baja de Activos Fijos

Se denomina baja de bienes de uso al retiro de estos por encontrarse en condiciones no aptas para prestar servicio a la institución. Este hecho se produce por inclemencias climatológicas, por siniestros, por obsolescencia o por haber cumplido con su vida útil. [Terán, 1997].

2.2.10 Revalorización técnica de Activos Fijos

Es un procedimiento que está reconocido contablemente, a través del cual los peritos expertos independientes asignan nuevos valores o determinan justiprecio a estos mas los correspondientes años de vida útil residual de acuerdo al estado de conservación. [Terán, 1997].

Los objetivos de la revalorización técnica de activos fijos son:

- Asignar nuevos valores a los bienes
- Asignar años de vida útil residual
- Cumplir con disposiciones legales y normas contables.

2.2.11 Costo de un Activo Fijo

El costo es el valor monetario con el cual fue adquirido uno o más activos para la organización, en un determinado tipo de cambio.

Es un gasto monetario que corre la institución al momento de adquirir un activo fijo en unidades o en cantidad, un costo es observado al momento de proporcionar mantenimiento a uno de los activos fijos para continuar con su trabajo y en óptimas condiciones. [Terán, 2001]

2.3 LAS METODOLOGÍAS ÁGILES

La historia de las Metodologías Ágiles y su apreciación como tales en la comunidad de la Ingeniería de Software tiene sus inicios en la creación de una de las metodologías utilizada como arquetipo: XP - eXtreme Programming, que nace de la mente de Kent Beck, tomando ideas recopiladas junto a Ward Cunningham.

En 1996, Beck al ser llamado por Chrysler como asesor del proyecto Chrysler Comprehensive Compensation (C3) payroll system, debido a la poca calidad de los sistemas, decide utilizar las prácticas que había definido a lo largo de su carrera. De esta forma en Mayo de 1997, el sistema, que administra la liquidación de aproximadamente 10.000 empleados, es puesto en operación con gran éxito. De esta forma, Kent Beck dio origen a la metodología XP iniciando de esta forma el movimiento de metodologías ágiles.

Inicialmente estas metodologías eran llamadas “metodologías livianas”, sin embargo, aun no contaban con una aprobación debido a que eran consideradas intuitivas por los desarrolladores. Luego, en febrero de 2001, tras una reunión celebrada en Utah-EEUU, se define formalmente el término “ágil” aplicado al desarrollo de software. El objetivo era ofrecer una alternativa a los procesos de desarrollo de software, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada actividad que se desarrolla.

Tras esta reunión se creó The Agile Alliance, sin fines de lucro y con la finalidad de promover los conceptos de desarrollo ágil de software y de esta forma ayudar a las organizaciones para que implementen estos conceptos. Todo surgió con el Manifiesto Ágil. [Amaro, Valverde 2007].

2.3.1 El Manifiesto Ágil

El Manifiesto Ágil comienza enumerando los principales valores del desarrollo ágil. Según el Manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas, debido a que es mejor crear el equipo y este desarrollara su entorno de acuerdo a sus necesidades.
- Desarrollar software que funciona más que una buena documentación. Basándose en la regla “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser concretos.
- Más vale el colaborar con el cliente que la negociación de un contrato. Para lo cual debe existir una interacción constante entre el quipo de desarrollo y el cliente, lo que garantizará el éxito del proyecto.
- La capacidad de poder responder a los cambios que pudieran más que seguir estrictamente un plan.

Los valores anteriores inspiran los doce principios del manifiesto. Los principios son:

1. La prioridad es satisfacer al cliente mediante entregas de software más rápido y de manera continua.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software funcionando desde un par de semanas a un par de meses, con intervalos de tiempo cortos entre cada entrega.
4. El cliente y el equipo de desarrolladores deben trabajar juntos hasta finalizar el proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles todo el apoyo necesario y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo entre los miembros del equipo es muy importante para que siempre exista comunicación entre ellos.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Todos los comprometidos con el proyecto deben tener la capacidad de mantener la paz.

9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. A intervalos de tiempo regulares, el equipo según sus debilidades debe mejorar para lograr ser más efectivo. [Amaro, Valverde 2007]

2.3.2 Metodologías Ágiles versus Metodologías Tradicionales

La Tabla 2.5 recoge las principales diferencias de las metodologías ágiles frente a las tradicionales ("no ágiles"). Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2.5 Diferencias entre Metodologías Ágiles y no Ágiles.
Fuente: [Amaro, Valverde 2007]

2.3.3 ¿Por qué usar Metodologías Ágiles?

Las metodologías ágiles presentan diversas ventajas, entre las que podemos destacar:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo
- Entrega continua y en plazos breves de software funcional
- Trabajo conjunto entre el cliente y el equipo de desarrollo
- Importancia de la simplicidad, eliminado el trabajo innecesario
- Atención continua a la excelencia técnica y al buen diseño
- Mejora continua de procesos y equipo de desarrollo. [Amaro, Valverde 2007]

2.3.4 Metodología SCRUM

Scrum es un modelo de desarrollo ágil, mismo que fue adoptado por los años 80 por algunas empresas del rubro de la tecnología. Scrum emplea el principio ágil "iterativo e incemental", en el cual se denomina Sprint (ciclos cortos de trabajo) a cada iteración de desarrollo.

Scrum es un proceso ágil que se puede usar para gestionar y controlar desarrollos complejos de software y productos. [INTECO, 2009]

2.3.4.1 Historia

En 1986, Hirotaka Takeuchi e Ikujiro Nonaka describieron un enfoque integral que incrementaba la velocidad y flexibilidad del desarrollo de nuevos productos comerciales. Compararon este nuevo enfoque integral al rugby, donde todo el equipo trata de ganar distancia como una unidad y pasando el balón una y otra vez.

Jeff Sutherland desarrolló un enfoque similar en Easel Corporation aplicando los principios de Nonaka y Takeuchi y fue la primera vez que se llamó Scrum. En 1995 Sutherland y Schwaber presentaron de forma conjunta un artículo describiendo Scrum en OOPSLA '95 en Austin, su primera aparición pública. Schwaber y Sutherland colaboraron durante los siguientes años para unir los

artículos, sus experiencias y las mejores prácticas de la industria en lo que ahora se conoce como Scrum. En 2001, Schwaber se asoció con Mike Beedle para poner en limpio el método en el libro Agile Software Devlopment with Scrum. [INTECO, 2009]

2.3.4.2 Introducción al modelo

Scrum como metodología de desarrollo requiere de trabajo duro, debido a que no se basa en el seguimiento de un plan sino se adapta de forma continua a las circunstancias a medida que se va desarrollando el proyecto y como metodología ágil es adaptable antes que predictiva, está orientada a las personas antes que a los procesos y herramientas, emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones. [Palacio, 2008].

Scrum denomina "sprint" a cada una de las iteraciones de desarrollo del ciclo de vida iterativo y según las características del proyecto y las circunstancias del sprint puede determinarse una duración desde dos hasta cuatro semanas, aunque lo más recomendable es que no sea más de un mes.

2.3.4.3 Elementos

Pila del producto: (Product Backlog) es una lista de requisitos del usuario que crece y evoluciona a través del desarrollo del proyecto, pueden ser mejoras, corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones de desarrollo.

Es todo aquello que los clientes, usuarios esperan. Todos los trabajos deben estar en esta pila de producto por ejemplo:

- Permitir a los usuarios la consulta de obras publicadas por un autor.
- Mejorar la escalabilidad del sistema.

La pila del producto no es un documento, sino una herramienta de referencia para el equipo, por lo que se recomienda que la lista incluya la siguiente información:

- Identificador único de la historia de usuario.

- Descripción de la funcionalidad.
- Sistema de priorización.
- Estimación

También se pueden adicionar los siguientes campos dependiendo al tipo de proyecto:

- Observación
- Criterio de validación
- Persona asignada.
- Nro. de Sprint al que pertenece.
- Modulo del sistema la que pertenece.

Pila del sprint: (Sprint Backlog) lista de los trabajos que descompone las funcionalidades de la pila del producto en las tareas necesarias para construir un incremento. Es una parte completa y operativa del producto.

Cada tarea de la pila del Sprint tiene una persona asignada y un tiempo indicado que falta para terminarla.

Descomponen el proyecto en tamaños adecuados para determinar el avance a diario, también permite identificar riesgos y problemas. También es herramienta de comunicación directa con el equipo.

La pila del Sprint cumple con las siguientes condiciones:

- Es realizada conjuntamente por todos los miembros del equipo.
- Cubre las tareas identificadas por el equipo para lograr el objetivo del Sprint.
- Cada tarea tiene una duración aproximada de 2 a 16 horas de trabajo.
- Es visible para todo el equipo.

Se recomienda que la pila del Sprint contenga el diseño del formato que sea más cómodo para todos:

- Lista de tareas, personas responsables de cada una, el estado en el que se encuentra y el tiempo que queda para completarla.

- Solo incluye la información necesaria.
- Sirve de soporte ya que se registra en cada reunión del sprint, el tiempo que le queda a cada tarea.

  Ejemplo:

SPRINT	INICIO	DURACIÓN
1	1-mar-07	12

	J
	1-mar
	23
	276

SPRINT BACKLOG			
Tarea	Estad	Responsal	
Descripción de la tarea 1	Terminada	Luis	16
Descripción de la tarea 2	Terminada	Luis	12
Descripción de la tarea 3	Terminada	Luis	4
Descripción de la tarea 4	Terminada	Elena	8
Descripción de la tarea 5	Terminada	Elena	16
Descripción de la tarea 6	Terminada	Elena	6
Descripción de la tarea 7	Terminada	Antonio	16
Descripción de la tarea 8	Terminada	Antonio	16
Descripción de la tarea 9	Terminada	Antonio	12
Descripción de la tarea 10	En curso	Luis	12
Descripción de la tarea 11	Pendiente	Luis	8

Figura 2.1 Sprint Backlog
Fuente: [Palacio, 2008]

Al mismo tiempo, con estos datos traza el grafico o “burn-down”, mismo que describe el estado de avance del trabajo del sprint que está en curso.

Incremento: Es parte de producto o subsistema realizado en un Sprint y potencialmente entregable: terminada y probada.

2.3.4.4 Roles y responsables del proyecto

Propietario del producto: (Product Owner) es el responsable de lograr el mayor valor de producto para los clientes, usuarios y resto de implicados, es quien decide como debe ser el producto final, en qué orden se construirán los sucesivos incrementos, es decir prioriza la pila del producto.

Es responsable de la financiación del proyecto, decide las fechas de entrega y las funcionalidades.

Equipo de desarrollo: (Team) grupo o grupos de trabajo que desarrollan el producto, se recomienda que sea entre 4 y 8 personas debido a que la velocidad en la comunicación se dificulta cuando son más de 10 personas. Se trata de un equipo multidisciplinario, todos trabajan de forma conjunta para realizar cada sprint. El equipo

tiene un espíritu de colaboración y el propósito común de conseguir el mayor valor para la visión del cliente. Un equipo Scrum responde en su conjunto trabajando de forma auto organizada. Nadie delimita, asigna o coordina las tareas, siendo los componentes quienes lo realizan. En el equipo:

- Todos conocen y comprenden la visión del propietario del producto.
- Aportan con el desarrollo del la pila del producto.

Scrum Manager – Tem Leader: Es el responsable del funcionamiento de la metodología Scrum en el proyecto, cubriendo los aspectos importantes según el conocimiento o aquellos que no cubra con otras personas con la formación e idoneidad adecuada.

- Asesoría y formación al equipo.
- Revisión y validación de la pila del producto.
- Gestión de la dinámica de grupo en el equipo.
- Respeto de la organización e implicados con tiempos y formas de Scrum.
- Configuración, diseño y mejora continua de las practicas de Scrum en la organización. [Palacio, 2008].

2.3.4.5 Medición uso y herramientas:

Grafico de producto: (burn-up) Este gráfico muestra el plan general de desarrollo del producto, y la evolución. Este diagrama en el plano cartesiano de las ordenadas representa el trabajo estimado para desarrollar el producto, y en las abcisas las fechas, medidas de acuerdo a lo que se estimo la duración de cada sprint.

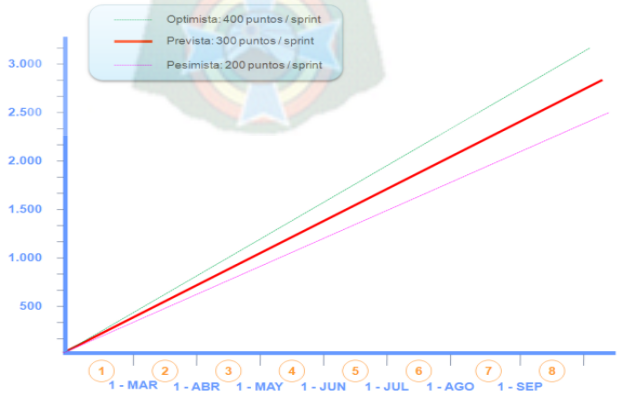


Figura 2.2 Grafico del producto(Burn up)
Fuente: [Palacio, 2008]

Monitorización del sprint: (burn-down) todos los días el equipo actualiza este gráfico en las reuniones el seguimiento del sprint, esto permite controlar el avance y verificar si se cumple lo estimado.

La estrategia ágil para hacer el seguimiento de los proyectos se basa en medir el esfuerzo faltante, no el realizado y el seguimiento diario.

En este gráfico en el eje y se muestra el trabajo que aún falta por realizar y se actualiza a diario.

Ejemplo:

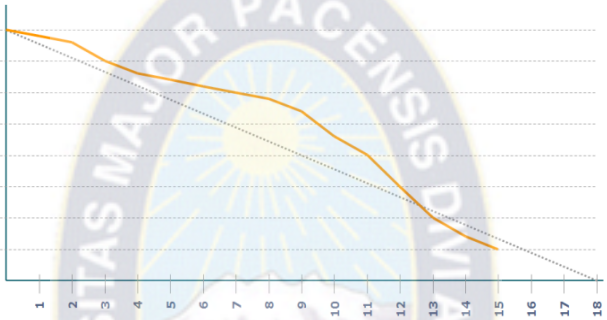


Figura 2.3 Gráfico de avance (Burn Down)
Fuente: [Palacio, 2008]

2.3.4.6 Scrum las reuniones:

Una parte muy importante de Scrum son las reuniones que se realizan durante cada una de las iteraciones. Hay distintos tipos:

Scrum diario: cada día durante la iteración, tiene lugar una reunión de estado del proyecto de cómo máximo 15 minutos donde el equipo se sincroniza. A esta reunión se le denomina Daily Sprint meeting.

Reunión de planificación de iteración: (Sprint Planning) se lleva a cabo al principio del ciclo de la iteración aquí el Product Ownwer presenta las historias del product backlog por orden de prioridad, el equipo determina la cantidad de historias que puede terminar en ese Sprint.

Reunión de revisión de iteración: al final del ciclo de la iteración el equipo presenta las historias mediante una demostración del producto.

Reunión Iteración retrospectiva: al final del ciclo de la iteración en la retrospectiva el equipo analiza que se hizo bien, que procesos serian mejorables y discute como mejorarlos. [INTECO, 2009]

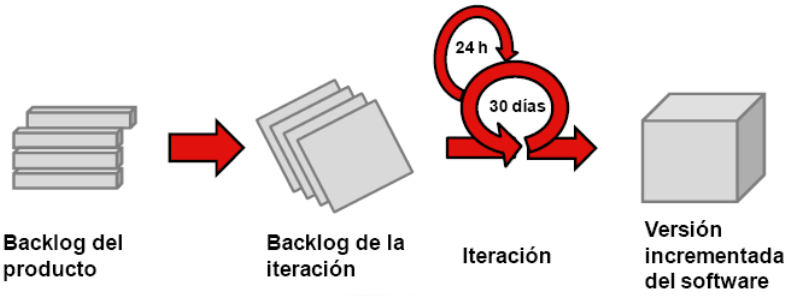


Figura 2.4 Flujo de proceso de Scrum
Fuente: [INTECO, 2009]

2.3.4.7 Cómo aplicar Scrum en el proyecto:

A continuación se muestra una plantilla de cómo se podría llegar a aplicar.

Etapa 1: Toma de requerimientos.

En esta etapa se define el "Product Backlog", el cual corresponde a todas las tareas, requerimientos o funcionalidades a realizar.

Esta información se la recopila durante las reuniones entre el equipo de trabajo y el "Product Owner".

Etapa 2: Análisis de requerimientos y diseño arquitectónico.

En esta etapa se analiza el "Product Backlog" y a través de este se genera el diseño base del sistema.

Etapa 3: Desarrollo del sistema.

1. Durante todo el proceso el equipo de trabajo realizara reuniones de 15 a 30 minutos denominadas "Scrum diario", con la finalidad de que todo el equipo se entere del estado de avance del proyecto y se resuelvan las diversas dudas que pudieran surgir.
2. Durante la reunión de planificación de iteración se definen los diversos "Sprint Backlog" con una duración de entre 2 a 4 semanas, los cuales contemplaran parte de las funcionalidades descritas en el "Product Backlog".

3. Una vez finalizado un “Sprint Backog”, se realizará una reunión denominada reunión de revisión de iteración donde se mostrara al “Product Owner” los avances realizados y el mismo podrá realizar las observaciones correspondientes.
4. Finalmente se realiza la reunión retrospectiva, donde se marcan los aspectos positivos para repetirlos y los negativos para corregirlos en el siguiente “Sprint Backlog”.

Etapa 4: Fase de medición de calidad

En esta etapa se realiza la medición de calidad y posteriormente el cierre definitivo del proyecto con la metodología Scrum.

Etapa 5: Aceptación y entrega del producto.

Contempla las siguientes tareas:

1. Revisión final del producto.
2. Entrega del sistema.
   - Código fuente.
   - Documento de diseño arquitectónico.
   - Manuales de administración y uso del sistema
3. Instalación del sistema en el ambiente de producción. [GetGeek - 2009]

2.3.5. Prácticas ágiles

Son varias las prácticas que se utilizan en el desarrollo ágil. A continuación vamos a explicar algunas de ellas. [INTECO, 2009]

2.3.5.1. Test Driven Development (TDD)

El desarrollo orientado a pruebas (TDD) es una técnica de desarrollo de software que usa iteraciones de desarrollo cortas basadas en casos de pruebas escritas previamente. En esta práctica las pruebas se deben escribir antes de que se escriba el código para que este cumpla con las pruebas.

Los beneficios de escribir las pruebas antes que el código son:

- Ayuda a asegurar que la aplicación se escribe para poder ser probada debido a que los desarrolladores deben probar la aplicación desde el principio.
- También asegura que se escriban pruebas para cada característica. [INTECO, 2009]

2.3.5.1.1 Pruebas Unitarias

Las pruebas unitarias (unit testing) sirven para verificar el correcto funcionamiento de un modulo o una librería, para lo cual se deben realizar una serie de chequeos para verificar cómo responde el código creado. Estas pruebas deben ser hechas de forma periódica a medida que se va avanzando el proyecto, de esta forma al ir modificando el código se puede tener la certeza de que no se altera la funcionalidad de los módulos ya creados.

Las pruebas unitarias son códigos de programación que tienen el objetivo de probar otro código de programación. [De Luca, 2013].

2.3.5.2. Integración continua

La integración continua es un conjunto de prácticas de ingeniería del software que aumentan la velocidad de entrega de las nuevas versiones de software disminuyendo los tiempos de integración.

El concepto de integración continua es que se debe integrar el desarrollo de una aplicación de una forma incremental y continua.

A medida que se van realizando los cambios en el repositorio, más trabajo debe hacer el desarrollador antes de subir sus propios cambios, producto de ello el repositorio se puede convertir tan diferente de la línea base del desarrollador que entran en algo llamado a veces, “infierno de integración”, donde el tiempo que usan para integrar es mayor que el tiempo que se le han llevado sus cambios originales.

Una práctica recomendada es automatizar la construcción, el sistema debería ser construible usando un comando único. La automatización de la construcción debería incluir la integración, que con frecuencia incluye despliegue en un entorno similar al de producción. En muchos casos, el script de construcción no

solo compila binarios, sino que también genera documentación, estadísticas y distribución.

Las ventajas de la integración continua son: reducción de tiempo de integración, los problemas de integración se detectan y arreglan continuamente, no hábitos del último minuto antes de la fecha de liberación, disponibilidad constante de una construcción actual para propósitos de pruebaso o liberación. [INTECO, 2009]

2.3.5.2.1. Phing

Phing es una herramienta desarrollada en PHP que permite la automatización de tareas más comunes y repetitivas. Phing es una herramienta análoga a Apache Ant para el PHP.

Phing usa archivos buil de XML que contiene una descripción de las cosas para hacer. El archivo build se estructura en targets que contengan las órdenes reales para realizar por ejemplo:

- Con la ejecución de un único comando, despliegues/actualizaciones de aplicaciones (entornos locales y remotos, tanto vía FTP como SFTP con SSH).
- Documentación de código.
- Testeo e integración continua.
- Checkouts de repositorios de código.
- Despliegue de bases de datos.

Para ejecutarse por defecto phing buscara el archivo build nombrado como build.xml [(Aderhold, Black, Holtgrewe, Lellelid, Rook), 2013].

2.4 Patrón Modelo Vista Controlador (MVC)

2.4.1 ¿Qué es el patrón MVC?

El patrón MVC es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario, de esta forma facilita la funcionalidad, mantenibilidad, y escalabilidad de las aplicaciones web, además no permite mezclar lenguajes de programación en el mismo código. MVC divide las aplicaciones en tres niveles:

- **Modelo:** representa la lógica de negocios. Es el encargado de accesar de forma directa a los datos, hace de intermediario con la base de datos.
- **Vista:** es la encargada de mostrar la información al usuario de forma gráfica y humanamente legible, una vista normalmente será una página web.
- **Controlador:** es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que pueda mostrarla al usuario.

2.4.2 ¿Cómo funciona el patrón MVC?

El funcionamiento básico del patrón MVC, puede resumirse en:

- El usuario realiza una petición a través de la vista.
- El controlador captura el evento y hace la llamada al modelo.
- El modelo será el encargado de interactuar con la base de datos, ya sea en forma directa, con una capa de abstracción para ello y retornará esta información al controlador.
- El controlador recibe la información y la envía a la vista.
- La vista, procesa esta información creando una capa de abstracción para la lógica (encargada de procesar los datos) y otra para el diseño de la interfaz gráfica y los entregará al usuario de forma que pueda reconocer la información. [Eugenia Bahit, 2011]

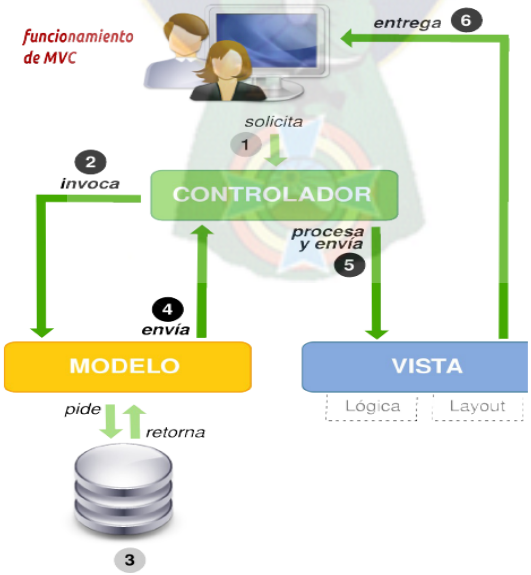


Figura 2.5 Funcionamiento del patrón modelo-vista-controlador
Fuente: [Eugenia Bahit, 2011]

2.5 HERRAMIENTAS DE DESARROLLO

2.5.1 Lenguaje de programación PHP

PHP es **"Pre-Procesador Hipertexto" (Hypertext Pre-Processor).** Como él mismo sugiere PHP es un procesador de hipertexto que se ejecuta en un servidor Web remoto para procesar páginas Web antes de que sean cargadas en el navegador. Además de sus potentes características, PHP es en sí un lenguaje simple diseñado específicamente para el desarrollo y producción de páginas Web. PHP es software de código abierto (open-saurce) y puede ser descargado gratuitamente.

2.5.1.1 Características fundamentales de PHP

Las fundamentales a considerar son:

- Es un lenguaje de script de código abierto para servidores.
- Es independiente del sistema operativo y puede ser utilizado en cualquiera de ellos, incluyendo Microsoft Windows, Mac OS, Linux, HP.UX y Solaris, etc.
- Utiliza una amplia gama de servidores Web como Apache, Microsoft Internet Information Server, Netscape e iPlanet.
- Se conecta a gran cantidad de bases de datos como ser MySQL, Ingres, Sybase, Oracle, Postresql.
- Se puede utilizar para crear imágenes y ficheros de lectura/escritura, así como para enviar mensajes de correo electrónico. Para proporcionar estos servicios, PHP se sirve de bastantes protocolos, como HTTP, POP·, SNMP, LDAP e IMAP. [Joel de la Cruz, 2006].

2.5.2 Framework Codeigniter

Codeigniter es un framawoerk que permite desarrollar aplicaciones web usando php mucho más rápido, cuenta con una amplia cantidad de bibliotecas o librerías para tareas comunes; así como, una interfaz sencilla. Permite minimizar la cantidad de código necesario para realizar una tarea.

Codeigniter se puede usar de manera libre ya que esta liberado bajo licencia open source del estilo Apache/BSD.

2.5.2.1 Un vistazo a Codeigniter

- Codeigniter es liviano, el núcleo sólo requiere de algunas bibliotecas muy pequeñas, las bibliotecas adicionales se cargan bajo pedido.
- Es rápido.
- Usa el enfoque arquitectónico modelo – vista - controlador MVC, lo que permite separar la lógica y la presentación.
- Genera URLs claras, debido a que usa el enfoque basado en segmentos por ejemplo: ejemplo.com/noticias/articulo/123.
- Trae un puñado de paquetes, viene con gran cantidad de bibliotecas que permiten facilitar tareas como acceso a la base de datos, envío de correo electrónico, validación de datos de formularios, manejo de sesiones, etc.
- Es extensible, se puede extender fácilmente a través de sus propias librerías, helpers, sistema de hooks.
- Incentiva a trabajar con programación orientada a objetos, ya que para trabajar con codeigniter debemos hacerlo con POO.

2.5.2.2 Seguridad de Codeigniter

En esta parte se describen algunas prácticas de seguridad web y características de la seguridad interna de Codeigniter.

- Seguridad URI, codeigniter minimiza la posibilidad de que datos maliciosos se puedan pasar a la aplicación por tal motivo las URIs solo pueden contener texto alfanumérico, tilde “~”, punto, dos puntos “:”, guión de subrayado “_”, guión “-”.
- Filtro XSS, Codeigniter viene con un filtro contra XSS (Cross Site Scripting). Permite filtrar Javacript malisioso en sus datos u otro tipo de código que intente hacer cosas maliciosas aquí.
- Validar datos, Codeigniter tiene una clase que permite validar los datos introducidos a través de los formularios, esta clase tiene reglas de validación que filtra y prepara los datos.

- Escapar todos los datos antes de insertarlos en la base de datos, este framework tiene métodos que permiten escapar datos y evitar que se introduzca basura en la base de datos. [EllisLab, 2011]. Traduccion Fernando “seacat” Velo, diciembre 2011.

2.5.3 Postgresql

PostgreSQL en un gestor de base de datos orientado a objetos (SGBDOO) usado en entornos de software libre, cumple con los estándares SQL92 y SQL99 y por el conjunto de funcionalidades que soporta, lo que lo sitúa en un mejor nivel.

Se distribuye bajo licencia BSD (Berkeley Software Distribution) es una licencia de software permisiva como la licencia OpenSSL, lo que permite su uso, redistribución, modificación con la única restricción de mantener el copyright del software de sus autores.

Puede funcionar en múltiples plataformas (en todas las basadas en UNIX), también en Windows d eforma nativa.

2.5.3.1 Historia

PostgreSQL tiene su origen en gestor de base de datos POSTGRES desarrollado en la Universidad de Berkeley en 1986 con un proyecto del profesor Michael Stonebraker y un equipo de desarrolladores.

A partir de 1994 Andrew Yu y Jolly Chen incluyeron SQL en Postgres y liberaron su código en la web con el nombre Postgres95, con múltiples cambios al original y mejor rendimiento y legibilidad.
En 1996 el nombre cambio a PostgreSQL, liberando la versión 6.0

2.5.3.2 Prestaciones

PostgreSQL cuenta con una amplia lista de prestaciones que lo hacen competitivo ante cualquier SGBD comercial:

- Esta desarrollado en C, con herramientas como Yacc y Lex.

- La interfaz de de la aplicación al SGBD se ecuentra disponible en C, C++, Java, Perl, PHP, Pyton y TCL, etc.
- Cuenta con un amplio conjunto de datos.
- Su administración se basa en usuarios y privilegios.
- Cuenta con opciones de conectividad TCP/IP, sockets Unix y sockets NT y soporta completamente ODBC (Open DataBase Connectivity).
- Es altamente confiable en cuanto a estabilidad.
- Puede extenderse con librerías externas para soportar encriptación, búsquedas por similitud fonética.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados, subconsultas, y casi todos los tipos de operadores.
- Implementación de algunas extensiones de orientación a objetos. Permite definir un nuevo tipo de tabla a partir de otra ya definida. [Gibert y Pérez, 2007].

2.6 SEGURIDAD DEL SOFTWARE

La seguridad del software es una actividad que garantiza la calidad del software, se centra en la identificación y evaluación de riesgos potenciales y los mismos pueden llegar a producir impactos negativos en el sistema.

Hacer consideraciones de seguridad significa ponerse a pensar cómo el agresor trata de encontrar el diagrama de vulnerabilidades y buscar mecanismos que puedan solucionar estas vulnerabilidades.

Algunas de estas vulnerabilidades son:

- Inyección SQL
- Script entre sitios
- Modificación de ingreso
- Reemplazo de identidad
- Ataques XSS

2.6.1. MEDIDADAS DE SEGURIDAD

2.6.1.1 AUTENTICACIÓN

Se refiere a verificar la identidad del usuario, para ello se lo debe someter a un proceso de validación de su identidad para afirmar que el usuario es quien dice ser. El mecanismo más utilizado para lograr este objetivo el login/password, donde el usuario provee nombre y contraseña para autentificarse. [Luis Capa, 2008]

2.6.1.2 ENCRIPTACIÓN

Es un término usado para definir el proceso de transformación de datos o texto para ser ilegible y debería ser virtualmente imposible que un intruso pueda descifrarlo, en el presente proyecto para el modulo del logeo se utilizó encriptación con MD5 (Message Digest 5) que es una función hash irreversible de PHP, es decir encripta el password tecleado por el usuario y es imposible que partiendo de esta cadena encriptada se vuelva a la contraseña origen. Encripta la contraseña con una cadena de 128 bits.

2.6.1.3 CAPTCHAS DE SEGURIDAD

Los captchas de seguridad nos ofrecen seguridad contra scripts de fuerza bruta para probar masivamente contraseñas sobre un formulario, así como para verificar que quien envía el formulario es un apersona y no una máquina o un robot con un script automático.[Ventura, 2012] .

2.6.1.4 SEGURIDAD DE ESCRITORIO

Otros factores que influyen en la seguridad del software son las amenazas causadas por virus y gusanos. Es responsabilidad de cada usuario el contrarrestare estas amenazas manteniendo los sistemas actualizados, navegar por sitios seguros, utilizar antivirus o firewalls.[Luis Capa, 2008]

2.7 CALIDAD DE SOFTWARE

2.6.1. Qué es calidad de software?

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente" [PRESSMAN, 1998].

Además de la definición anterior se puede decir que la calidad de software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Donde la calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenimiento, portabilidad, usabilidad, seguridad e integridad [PRESSMAN, 2001].

La calidad de software es medible y puede variar de un sistema a otro o de un programa a otro. Un producto de software para ser explotado durante un largo período (10 años o más), necesita ser confiable, mantenible y flexible para disminuir los costos de mantenimiento y perfeccionamiento durante el tiempo de explotación.

De acuerdo al estándar ISO 9126 identifica seis atributos clave de calidad: funcionalidad, confiabilidad, usabilidad, eficiencia, facilidad de mantenimiento, portabilidad, subatributos. Estos factores no necesariamente son utilizados como medidas directas, facilitan una valiosa base para las medidas indirectas y una excelente lista para determinar la calidad se software.

2.6.2 Funcionalidad

Las métricas orientadas a la función son medidas indirectas del software y del proceso por el cual se desarrolla. En lugar de calcular las LDC (líneas de código), las métricas orientadas a la función se centran en la funcionalidad o utilidad del software, midiendo la aplicación desde la perspectiva del usuario dejando de lado los detalles de codificación. La medida de punto de función se propuso en 1979 por Albercht quién sugirió un acercamiento a la medida de la productividad; siendo ahora un estándar de ISO: ISO/IEC 20926:2003. [PRESSMAN, 2001].

Los puntos de función se obtienen utilizando una función empírica basada en medidas cuantitativas del dominio de información del software y valoraciones subjetivos de la complejidad del software. Se determinan cinco características del ámbito de la información y los cálculos aparecen en la posición apropiada de la tabla. Los valores del ámbito de información están definidos de la siguiente manera; los puntos de función se determinan rellenando la tabla como se muestra en la Figura .

- **Número de entradas de usuario:** Se cuenta cada entrada del usuario que proporcione al software diferentes datos orientados a la aplicación. Las entradas deben ser distinguidas de las peticiones que se contabilizan por separado.
- **Número de salidas del usuario:** Se cuenta cada salida que proporciona al usuario información orientada a la aplicación. En este contexto las salidas se refieren a informes, pantalla, mensajes de error. Los elementos de datos individuales dentro de un informe se cuentan por separado.
- **Número de peticiones al usuario:** Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva. Se cuenta cada petición por separado.
- **Número de archivos:** Se cuanta cada archivo maestro lógico, o sea una agrupación lógica de datos que puede ser una parte en una gran base de datos o un archivo independiente.
- **Número de interfaces externas:** Se cuentan todas las interfaces legibles por la maquina por ejemplo: archivos de datos, en cinta o discos que son utilizados para transmitir información a otro sistema.

Parámetro de medición	**Cuenta**		**Simple**	**Medio**	**Complejo**		
Número de entradas de usuario		*	3	4	6	=	
Número de salidas de usuario		*	4	5	7	=	
Número de peticiones de usuario		*	3	4	6	=	
Número de archivos		*	7	10	15	=	
Número de interfaces externas		*	5	7	10	=	
Cuenta total	→						

Figura 2.6 Factor de ponderación [PRESSMAN, 2003]

Para calcular los puntos de función se utiliza la siguiente relación.

$$PF=\text{Cuenta Total}*[0.65+0.01*\sum Fi]$$

Donde: Cuenta total es la suma de todas las entradas de PF obtenidas de la tabla anterior.

Fi donde i puede ser de uno hasta 14 los valores de ajuste de complejidad basados en las respuestas a un listado de cuestiones, donde cada respuesta tomará un valor o factor en la escala de 0 a 5 tomando en cuenta lo siguiente:

0	1	2	3	4	5
Sin influencia	incidental	moderado	medio	significativo	Escencial

Los valores constantes de la ecuación anterior y los factores de peso aplicados en las encuestas de los ámbitos de información han sido determinados empíricamente, seguidamente se describen las preguntas:

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

    Una vez evaluada la ecuación el valor que toma el PF indica si la funcionalidad del sistema es óptimo o no.

2.6.3 Mantenibilidad

El estándar IEE982.1-1988 sugiere el índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto software basada en los cambios que ocurren con cada versión del producto. Con el IMS se determina la siguiente información:

- MT= Número de módulos en la versión actual.
- Fc= Número de módulos en la versión actual que se han cambiado.
- Fa= Número de módulos en la versión actual que se han añadido.
- Fe= Número de módulos en la versión actual que se han eliminado.

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

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

A medida que el IMS se aproxima a 1 el producto se empieza a estabilizar [PRESSMAN, 2001].

2.6.4 Confiabilidad

Se pretende que el software funcione de forma consistente y correcta durante largos períodos de tiempo que esté disponible cuando se lo necesita, y que sea reparado rápido y fácilmente si acaso falla. Formalmente se dice que la confiabilidad del software es la probabilidad de que un sistema opere sin fallas bajo determinadas condiciones para un intervalo de tiempo dado.

Los usuarios quieren que el software que vayan a utilizar tenga un funcionamiento correcto y los desarrolladores esperan que el número de defectos que existan puedan ser solucionados sin causar más problemas a medida que se lo va probando, esto hace que la confiabilidad en el software crezca u los tiempos entre fallas se hacen cada vez mas grandes [PRESSMAN, 2001].

En esta sección se examina la confiabilidad en relación a la incertidumbre, considerando una incertidumbre de tipo 1, que refleja la incertidumbre sobre la utilización del

sistema. Por lo tanto, en cualquier instante, el tiempo hasta la próxima falla es incierto y se lo puede considerar como una variable aleatoria.

Para estudiar esta variable aleatoria utilizaremos una distribución exponencial en causada en sus pruebas con cero fallas, que deriva en una función de cifras de falla, se supone que el número de fallas en el instante *t* es igual a:

$$ae^{-b(t)}$$

Donde: a,b son constantes.

Este modelo se puede utilizar para determinar por cuantas horas se puede robar un sistema a fin de satisfacer una meta de confiabilidad. Por lo tanto el modelo requiere tres entradas; el número proyectado promedio de fallas como objetivo (fallas), el número total de fallas observada en las pruebas (fallas probadas) y el número total de horas de ejecución de pruebas hasta la última falla (horas hasta la última falla). El cálculo de las horas necesarias de prueba para cero fallas es:

$$\frac{\text{Ln}[(\text{fallas}) / (0{,}5 + \text{fallas}) * (\text{horas hasta última falla})]}{\text{Ln}[(0{,}5 + \text{fallas}) / (\text{fallas probadas} + \text{fallas})]}$$

2.6.5 Portabilidad

La portabilidad es la facilidad con la que el software puede ser llevado de un entorno a otro. Este criterio se subdivide: Facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio [NUÑEZ, 2005].

Es el esfuerzo necesario de poder transferir el sistema de una maquina a otras. Esfuerzo necesario para transferir el programa de un entorno se sistema hardware y/o software a otro. [PRESSMAN, 1998].

CAPÍTULO 3
MARCO APLICATIVO

3.1 INTRODUCCIÓN

En este capítulo se desarrolla la solución basándonos en el problema planteado para obtener el producto software, describiendo la implementación de la metodología de trabajo Scrum en el Servicio Nacional de Patrimonio del Estado – SENAPE, para la gestión del desarrollo del proyecto “Sistema Web de Control y Administración de Activos Fijos del SENAPE”.

Incluye junto con la descripción de este ciclo de vida iterativo e incremental para el proyecto, los artefactos o documentos con los que se gestionan las tareas de adquisición y suministro: requisitos, monitorización y seguimiento del avance, así como las responsabilidades y compromisos de los participantes en el proyecto.

Para un mejor entendimiento se utiliza algunas herramientas UML, ya que facilitaran el desarrollo del proyecto

3.2 CAPTURA DE REQUERIMIENTOS FUNCIONALES COMO CASOS DE USO

El Área de Activos Fijos maneja la información sobre bienes tangibles que se tienen en las diferentes Direcciones del SENAPE, llevando a cabo la administración y control de todos ellos. Los actores que se identifican en el modelo del negocio y otros que son necesarios para el sistema son los siguientes.

- **Actores del Negocio**

Proveedores: Empresas o personas que proveen de activos fijos al SENAPE.

Responsable de transferencia y/o donación: Es la persona encargada de realizar las transferencias y/o donaciones de activos fijos, mismos que pueden ser provenientes de ex entidades o ex entes gestores de la seguridad social en liquidación, de las cuales el SENAPE se hace cargo.

Responsable de Activos Fijos: Persona encargada de controlar y administrar los Activos Fijos del SENAPE y de dar cumplimiento a las normas establecidas para el cuidado y preservación de activos.

Jefe Unidad Administrativa: Es la persona encargada de aprobar las solicitudes de Activos Fijos de las diferentes direcciones, autorizar la salida de Activos Fijos fuera de las instalaciones del SENAPE.

Funcionario: Son los servidores públicos del SENAPE, a quienes se les asignará Activos Fijos, durante la asignación son los responsables de dar el debido cuidado de los Activos Fijos y son sujetos a responsabilidades por el mal uso de los mismos.

Responsable de adquisiciones: Es la persona encargada de realizar las adquisiciones de nuevos Activos Fijos.

- **Casos de uso del Negocio**

Clasificar Activo Fijo: El Encargado de Activos Fijos para realizar el registro del Activo Fijo en el sistema correspondiente, previamente realiza la clasificación del mismo de acuerdo al grupo contable y su respectivo auxiliar contable.

Registro de activos fijos nuevos/donados/transferidos: El Encargado de Activos Fijos realiza el registro en el sistema correspondiente de Activos Fijos nuevos, donados o transferidos (los donados o transferidos, principalmente son de ex entidades o ex entes gestores de la seguridad social en proceso de liquidación por parte del SENAPE).

Asignación de activos fijos: El Encargado de Activos Fijos asigna el Activo Fijo a un servidor público de planta o consultor que pase a ser responsable del bien.

Devolución de activos fijos: El servidor público, responsable del bien una vez concluida su relación laboral con el SENAPE, por cambio de sección u otro motivo realiza la devolución del mismo al Encargado de Activos Fijos, en las mismas condiciones que se le fue entregado.

Adquirir activo fijo: El Responsable de Adquisiciones una vez que realiza la adquisición del activo, remite la documentación pertinente (Orden de compra, Nota de Remisión o entrega, factura de compra, garantía si corresponde), al Responsable de Activos Fijos.

3.2.1 Modelo del Negocio

A continuación se realiza la identificación de los diferentes procesos del negocio de la entidad. El siguiente gráfico muestra los casos del uso del modelo del negocio, además se desglosan estos en diagramas de flujo respectivos.

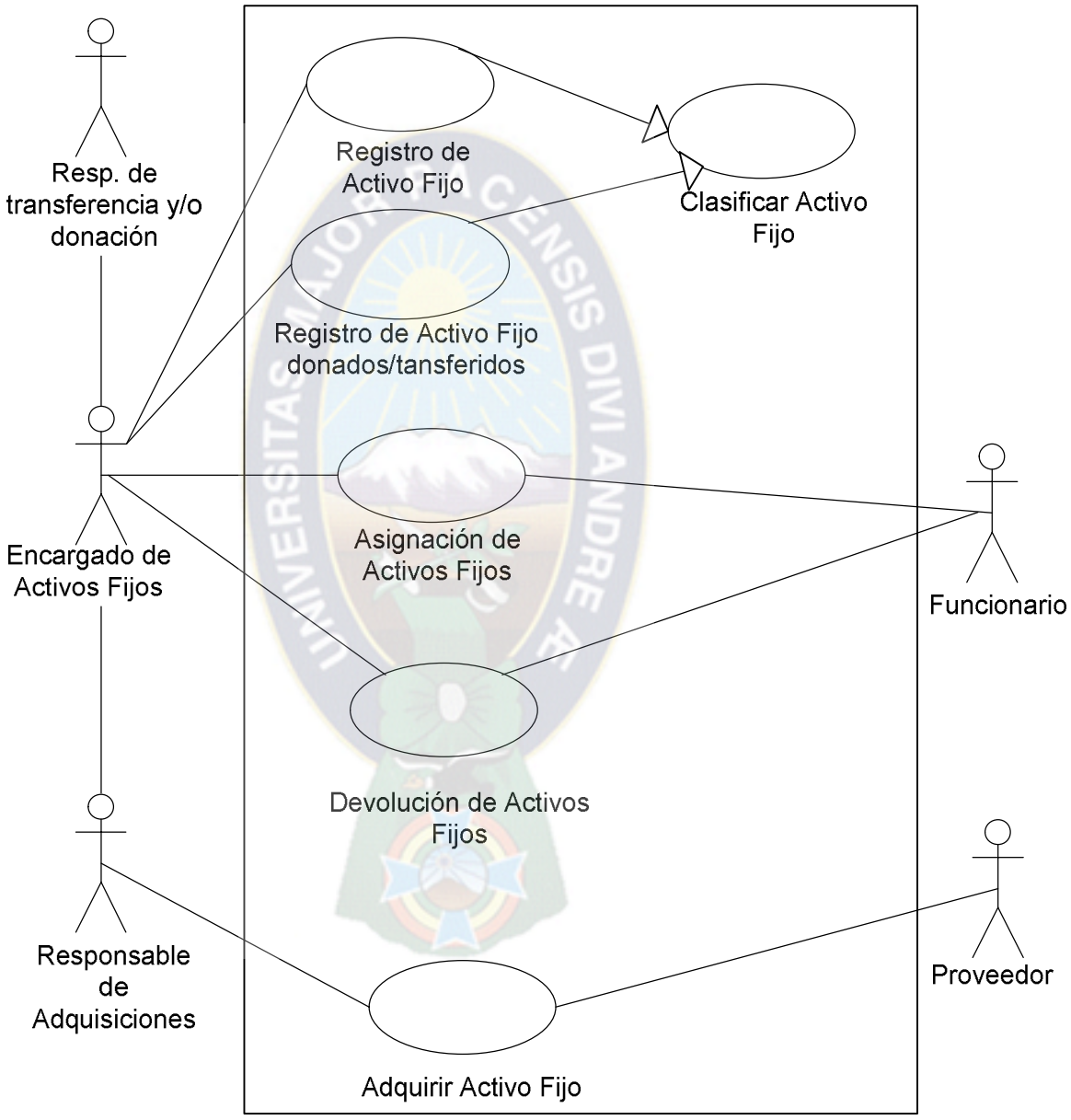


Figura 3.1: Modelo de Casos de uso del Negocio
Fuente [Elaboración propia]

- **Actores del sistema**

  Responsable de Activos Fijos: Es el encargado de controlar y administrar el bien, desde la recepción, incorporación, y asignación. Se encarga de informar de los activos que han salido del SENAPE; así como también, los que deben ser revaluados y dados de baja.

  Auxiliar de Activos Fijos: Es el Servidor público que coadyuva el trabajo del Responsable de Activos Fijos y realiza las asignaciones y devoluciones de Activos Fijos mediante el sistema.

  Funcionario: Es el servidor público de las diferentes Direcciones del SENAPE, al que se le asigna Activos Fijos, para la ejecución de sus labores y es el responsable de cuidar y preservar el bien.

- **Casos de Uso del Sistema**

  Adicionar Activo Fijo: Este caso de uso realiza el registro de ingreso del nuevo activo fijo generando el código del activo (tomando en cuenta la ubicación geográfica a la cual pertenecerá y la clasificación del activo) y su respectivo formulario de ingreso, permitirá al incorporación de un activo fijo donado o transferido, proveniente de ex Entidades o ex Fondos complementarios en proceso de liquidación por el SENAPE.

  Clasificar Activo Fijo: este caso de uso realiza la clasificación del activo fijo de acuerdo a su grupo contable y su respectivo auxiliar contable.

  Salidas u retornos de Activos Fijos: Este caso de uso permite controlar las salidas de los Activos Fijos de la institución por un determinado motivo y su respectivo retorno, asignando esta salida a un servidor público quien será el responsable del activo durante este período de salida.

  Ingreso temporal de Activos Fijos: Este caso de uso permite controlar los ingresos de activos que no pertenecen a la institución.

Asignar Activo Fijo: Este caso de uso realiza la asignación del activo fijo a los responsables.

Dar de Baja a los Activos Fijos: Este caso de uso realiza la baja de Activos Fijos, previo informe de un perito en la materia quien determina los motivos de la baja en el mencionado documento.

Actualizar datos de Activos Fijos Revaluados: Este caso de uso realiza la actualización de la información de los Activos Fijos que han sido sujetos a revalúo por parte de un perito en la materia, externo a la institución.

Depreciación de Activos Fijos: Este caso de uso realiza la depreciación de la parte contable de los Activos Fijos y posteriormente la actualización de la información de los activos.

Transferencia de Activos Fijos: Este caso de uso realiza las transferencias de Activos Fijos de oficinas y responsables.

Generar consultas y reportes: Este caso de uso permite generar consultas y reportes de acuerdo a los requerimientos del Responsable de Activos Fijos.

Controlar accesos: Este caso de uso realiza el control de acceso de los usuarios de acuerdo a las funciones que desempeñan.

3.2.2. Modelo de Casos de Uso del Sistema

Los casos de uso nos ayudan a capturar los requisitos adecuados y se utilizan para el modelado del sistema desde el punto de vista del usuario para representar las acciones que realiza cada tipo de usuario. Cada usuario necesita que haga alguna función y los casos de uso representan los modos de utilizar el sistema

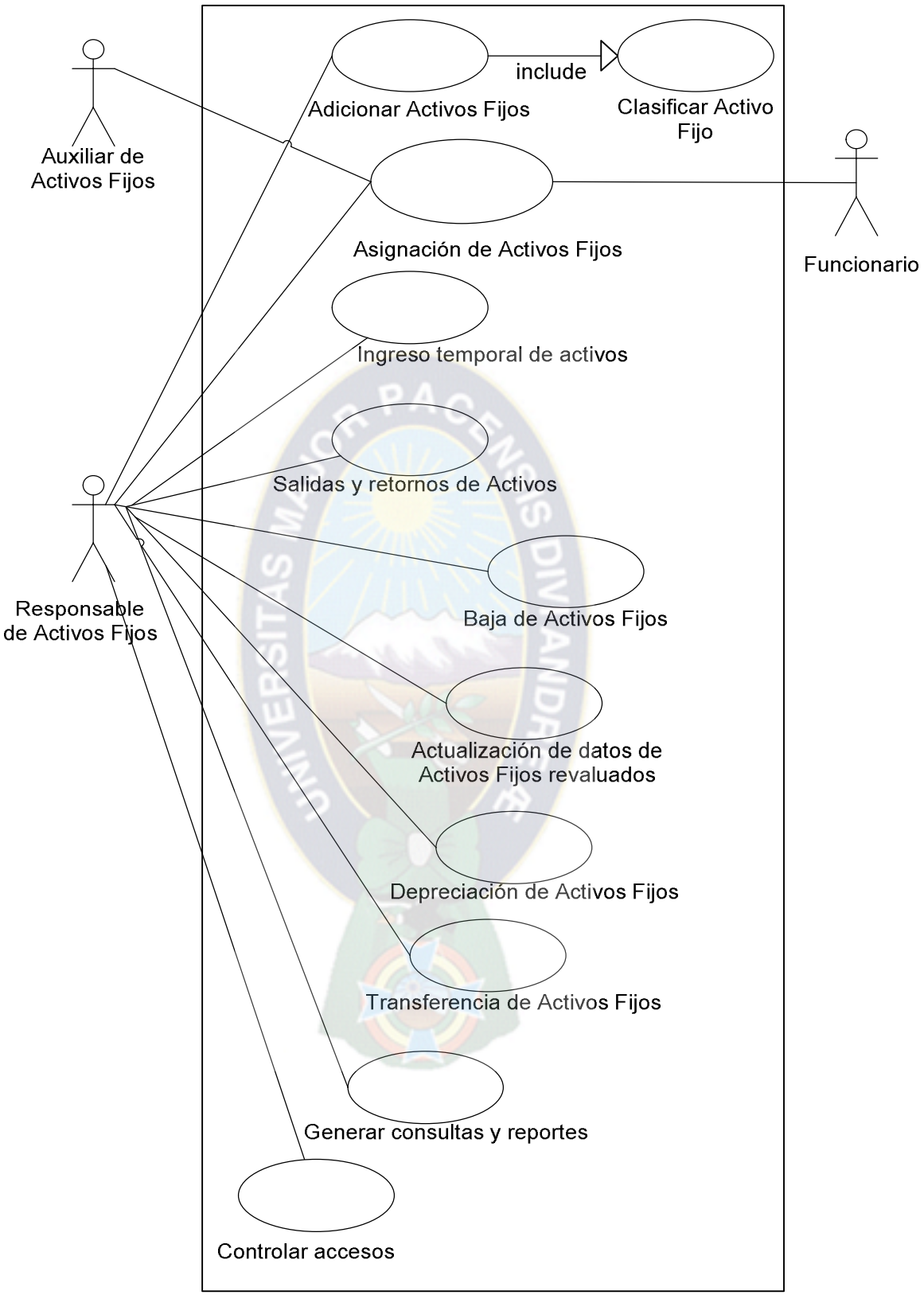


Figura 3.2: modelo de Casos de Uso del Sistema
Fuente [Elaboración propia]

- Casos de uso del sistema y su descripción

Ahora se describen de manera general los principales Casos de Uso del sistema.

PRECONDICIÓN	• El Responsable del Área de Activos Fijos debe tener los grupos contables ya definidos y registrados.
FLUJO DE EVENTOS	Camino básico • El encargado verifica a qué grupo contable pertenece el activo • El encargado verifica a que sub grupo contable pertenece el activo • El encargado verifica a que ubicación geográfica pertenece el activo. • El encargado de Activos Fijos le da una codificación (de acuerdo al Manual de Procesos y Procedimiento – MPP), tomando en cuenta la ubicación geográfica, el grupo contable y subgrupo contable al cual pertenecerá el Activo Fijo. • El encargado registra la clasificación y codificación en el sistema. Camino Alternativo • En dos puede que el Activo Fijo no tenga un subgrupo contable, entonces debe adicionar un sub grupo contable de acuerdo al grupo contable al cual pertenece.
POSCONDICIÓN	• El Caso de Uso termina cuando el responsable registra la clasificación del Activo Fijo

Tabla 3.1 Descripción del Caso de Uso Clasificar AF

PRECONDICIÓN	• El Responsable del Área de Activos Fijos debe tener la factura y toda la información del activo fijo a adicionar.
FLUJO DE EVENTOS	Camino básico • El encargado verifica toda la documentación del activo de compra. • El encargado realiza la clasificación del activo • El encargado registra todos los datos y detalles del activo. • El encargado realiza la adición del activo al sistema • El sistema genera el formulario de ingreso del activo Camino Alternativo • En dos puede que el Activo Fijo no tenga un subgrupo contable, entonces debe adicionar un sub grupo contable de acuerdo al grupo contable al cual pertenece.
POSCONDICIÓN	• El Caso de Uso termina cuando el responsable registra el activo en el sistema.

Tabla 3.2 Descripción del Caso de Uso Adicionar AF

PRECONDICIÓN	• El encargado debe tener la codificación, la clasificación, descripción del activo a asignar y debe tener los datos actualizados de los funcionarios a quienes se asignaran activos.
FLUJO DE EVENTOS	Camino básico • El encargado debe ser administrados del sistema. • El encargado de activos busca la lista actualizada de los funcionarios que estén trabajando actualmente en el SENAPE. • El encargado realiza la asignación del o los activos fijos a un determinado funcionario. • El encargado entrega al funcionario el Acta de Asignación de activos Fijos. Camino Alternativo • En dos puede que no existe el funcionario debido a que el puesto este vacante, pero se puede asignar el activo a un funcionario anterior y luego realizar una nueva asignación
POSCONDICIÓN	• El Caso de Uso termina cuando el responsable realiza la asignación de activos a un determinado funcionario que este en plena ejecución de sus funciones.

Tabla 3.3 Descripción del Caso de Uso Asignación de AF

PRECONDICIÓN	• El Responsable del Área de Activos Fijos debe observar el estado del activo para su actualización por cierre de gestión.
FLUJO DE EVENTOS	Camino básico • El encargado revisa y verifica el estado del activo. • El encargado verifica la Depreciación del activo para obtener el valor neto. • Si el valor neto es igual a uno se requiere de un informe por parte de peritos en la materia para dar de baja el activo. • De acuerdo al informe de baja, se procede a dar de baja el Activo Fijo. • El sistema genera el formulario de respaldo del proceso de baja del Activo Fijo. Camino Alternativo • En tres existen otros motivos aparte de que tengan valor neto igual a uno, como por ejemplo: que se haya extraviado y se procede directamente con el paso cuatro.
POSCONDICIÓN	• El Caso de Uso termina cuando se da de baja el Activo fijo en el sistema.

Tabla 3.4 Descripción del Caso de Uso Baja de Activos Fijos

3.3 METODOLOGÍA DE DESARROLLO

Debido a los requerimientos del cliente de entregas de productos usables en el menor tiempo posible se adopto una Metodología Ágil y más interactiva con el cliente, la Metodología adoptada como se mencionó en el capítulo anterior es SCRUM.

3.3.1. Presentación del equipo de trabajo y roles

Product Owner: Representa a las personas implicadas en el negocio. El Product Owner escribe historias de usuario, las prioriza, y las coloca en el Product Backlog.

ScrumMaster (o Facilitador): El Scrum es facilitado por un ScrumMaster, cuyo trabajo primario es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint. El ScrumMaster se asegura de que el proceso Scrum se utiliza como es debido. El ScrumMaster es el que hace que las reglas se cumplan.

Scrum team(o Equipo): El equipo tiene la responsabilidad de entregar el producto, incluye a los desarrolladores.

Estos roles están asignados para las siguientes personas:

Persona	Rol
Ing. Linder Poclava	Coordinador/Scrum master
Marcelo Antequera Eguez	Product Owner
Miguel Ángel Ticona Q.	Equipo

Esta metodología de desarrollo pide un mayor compromiso del cliente, para el desarrollo ágil, por lo que el cliente tendrá la posibilidad de contribuir más a través de su representante en las reuniones o el mismo.

3.3.2. ETAPA 1: TOMA DE REQUERIMIENTOS Y DEFINICIÓN DEL PRODUCT BACKLOG

En esta etapa se define el “Product Backlog”, el cual corresponde a todas las tareas, requerimientos o funcionalidades a realizar, priorizados por valor de negocio. Esta información se recopilo a través de reuniones entre el “Scrum Team” y el “Product Owner” y la captura de los requerimientos funcionales como los Casos de Uso.

ID	Descripción	Usuario	Prioridad	Estim	Sprint
H1	Elaboración de las tablas de la base de datos	Miguel Ticona	Muy alta	15	1
H2	Autentificación de usuario	Miguel Ticona	Muy alta	15	1
H3	Diseño de la interfaz de usuario	Miguel Ticona	Muy alta	20	1
H4	Registrar grupo contable	Miguel Ticona	Alta	8	1
H5	Registrar sub grupo contable	Miguel Ticona	Alta	7	1
H6	Registrar ubicación geográfica	Miguel Ticona	Alta	6	1
H7	Registrar piso planta de la ubicación geográfica	Miguel Ticona	Alta	6	1
H8	Registrar ambiente de trabajo	Miguel Ticona	Alta	6	2
H9	Registrar estados de activos fijos	Miguel Ticona	Alta	8	2
H10	Registrar empresa proveedora de activos fijos	Miguel Ticona	Alta	8	2
H11	Registrar UFV del día	Miguel Ticona	Alta	10	2
H12	Registrar activos fijos adquiridos por el SENAPE	Miguel Ticona	Alta	15	2
H13	Registrar entidad que asigna activos fijos al SENAPE	Miguel Ticona	Alta	15	2
H14	Registrar activos fijos asignados por entidades(en liquidación) al SENAPE	Miguel Ticona	Alta	33	3
H15	Permitir modificar datos de los activos fijos	Miguel Ticona	Alta	22	3
H16	Asignación de Activos Fijos	Miguel Ticona	Baja	40	3
H17	Devolución de Activos Fijos	Miguel Ticona	Baja	21	3
H18	Salidas y retornos de Activos Fijos	Miguel Ticona	Baja	32	3
H19	Ingreso temporal de activos fijos	Miguel Ticona	Baja	31	3
H20	Dar de baja Activo Fijo	Miguel Ticona	Baja	20	4
H21	Actualizar datos de Activos Fijos revaluados	Miguel Ticona	Baja	20	4
H22	Adicionar usuarios con respectivos privilegios en el sistema	Miguel Ticona	Baja	21	4
H23	Diseño de interfaz para cada tipo de usuario para controlar permisos y roles de usuarios a lo largo del sitio	Miguel Ticona	Baja	11	4
H24	Depreciación de Activos Fijos	Miguel Ticona	Baja	33	4
H25	Consultas y reportes	Miguel Ticona	Baja	16	4
H26	Backup de la base de datos	Miguel Ticona	Baja	15	4
H27	Cambiar password del usuario	Miguel Ticona	Baja	15	4
H28	Búsqueda de activos	Miguel Ticona	Baja	18	4
H29	Transferencia de activos	Miguel Ticona	Baja	16	4

Figura 3.3: Product Backlog
Fuente [Elaboración propia]

3.3.3 ETAPA 2: ANÁSIS DE REQUERIMIENTOS Y DISEÑO ARQUITECTÓNICO

En esta parte se hace el análisis del “Product Backlog” y a través de este el diseño base del sistema, el cual involucra modelo de base de datos y los principales módulos del sistema.

En la figura 3.4 se puede observar el modelado de la base de datos del sistema de Activos Fijos, este diagrama muestra todas las tablas y sus atributos, también las relaciones entre tablas con su respectiva cardinalidad.

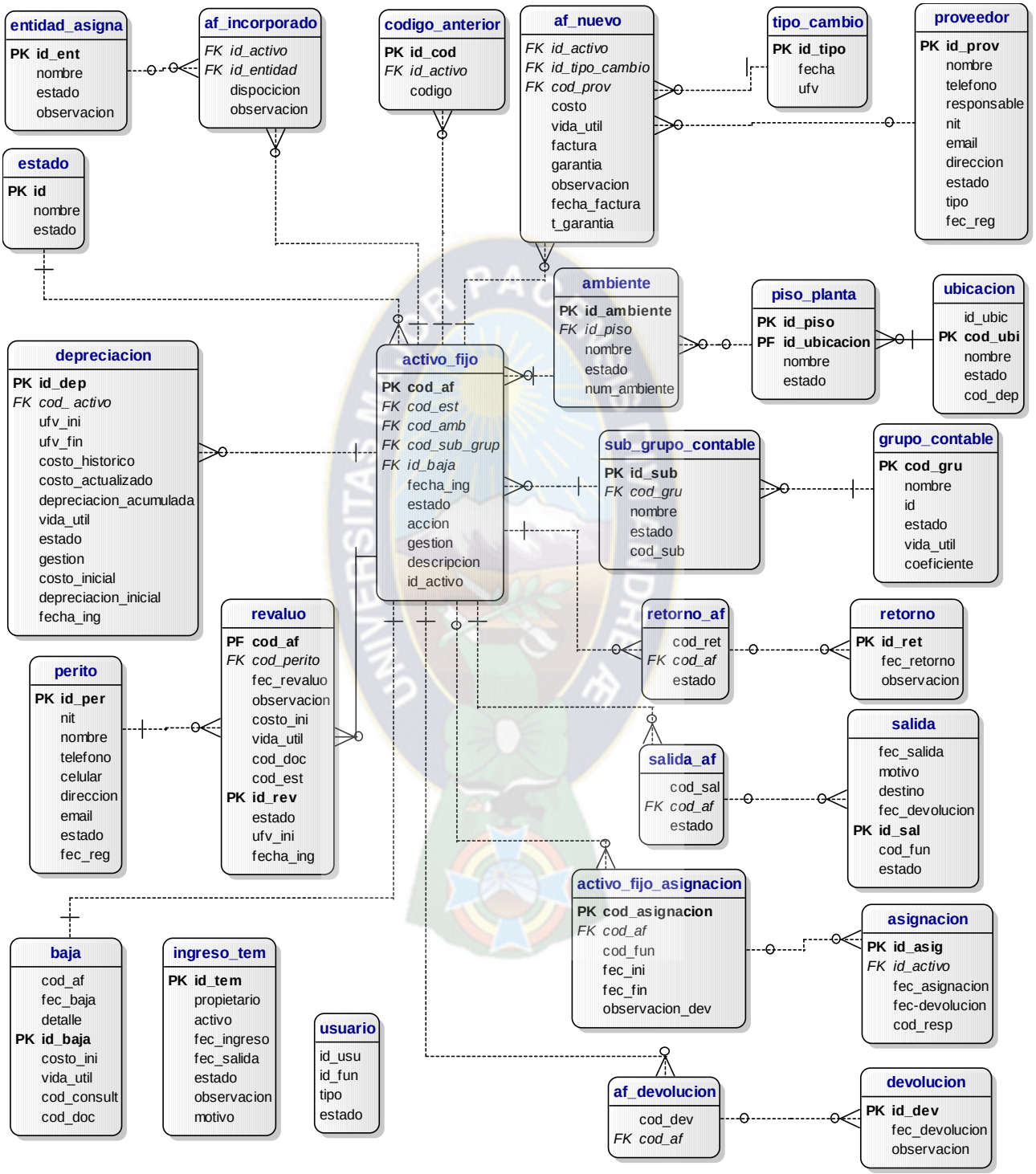


Figura 3.4: Modelado de base de datos
Fuente: [Elaboración propia]

En la figura 3.5 se muestran los módulos del sistema en el mapa navegacional

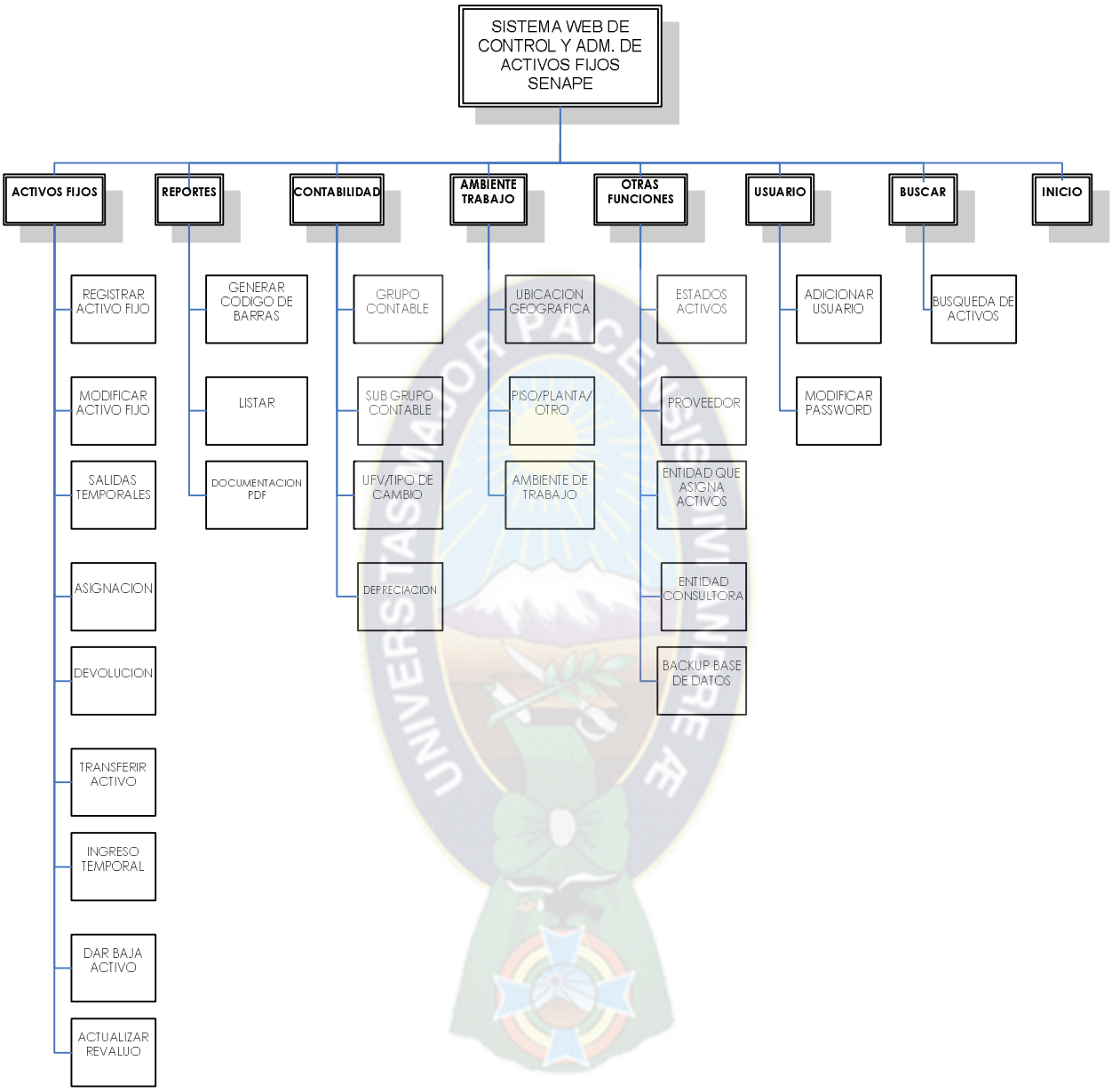


Figura 3.5: Mapa navegacional
Fuente: [Elaboración propia]

3.3.4 ETAPA 3: DESARROLLO DEL SISTEMA

Durante todo el proceso de desarrollo, el equipo de trabajo realizó reuniones diarias de 15 a 30 minutos, denominado “Daily Scrum Meeting”. El objetivo es que todo el equipo se entere del estado de las diversas tareas y se resuelvan dudas que pueden surgir en este proceso.

Mediante las “Sprint Planing Meeting” se definieron los diversos “Sprint Backlog” (Conjunto de requerimientos que equivalen a un incremento del sistema).

Una vez finalizado cada Sprint se realizaron reuniones denominadas “Sprint review” donde se mostraron al “Product Owner” las nuevas versiones terminadas y el mismo realizao las observaciones respectivas.

Por último se realizaron las reuniones denominadas “Sprint Retrospective”, donde se analizaron los aspectos positivos (para repetirlos) y lo aspectos negativos (para no repetirlos) en cada sprint.

Todo este ciclo se repitió hasta abarcar todo el contenido en el “Product Backlog”.

3.3.4.1 Primer Sprint

Se reparten las tareas que se vean convenientes, realizando la estimación de la tarea y la preparación del ejecutor de la tarea.

3.3.4.1.1. Pila del primer Sprint

SPRINT	INICIO	DURACIÓN
1	07-01-2013	24 días

PILA DEL SPRINT 1 O SPRINT BACKLOG

Backlog ID	Tarea	Tipo	Estado	Responsable	Estimado/ horas
H1	Elaboración de las tablas en la base de datos postgresql				**20**
H1A1	Instalación de manejador de base de datos postgresql		terminado	Miguel Ticona	1

H1A2	Análisis y diseño de la base de datos	Análisis y diseño	Terminado	Miguel Ticona	10
H1A3	Elaboración de las tablas en la base de datos		terminado	Miguel Ticona	9
H2	Autentificación de usuario				**36**
H2A1	Diseño de formulario de login	código	Terminado	Miguel Ticona	15
H2A2	Validación de datos del formulario de login con mensajes emergentes de error de inicio de sesión	código	Terminado	Migue Ticona	4
H2A3	Creación de las consultas a la bd para verificar datos del login	código	Terminado	Miguel Ticona	3
H2A4	Creación del hook (gancho de codeigniter), para verificar que el usuario este logeado correctamente en toda la sesion	código	Terminado	Miguel Ticona	6
H2A5	Creación de imágenes captcha con el helper captcha de codeigniter para la seguridad en el acceso al sistema.	código	Terminado	Miguel Ticona	4
H2A6	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H3	Diseño de la interfaz de usuario				**29**
H3A1	Configuración de la librería template (plantilla) de Codeigniter.	Código	Terminado	Miguel Ticona	5
H3A2	Diseño del menú, header, leter, content y footer de la plantilla.	código	Terminado	Miguel Ticona	8
H3A3	Creación de las hojas de estilo CSS para mejorar el aspecto de la interfaz de usuario.	código	Terminado	Miguel Ticona	12
H3A4	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H4	Registrar grupo contable				**32**
H4A1	Diseño del formulario para adición de grupo contable	código	Terminado	Miguel Ticona	5
H4A2	Validación de los datos de la adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	5
H4A3	guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	3
H4A4	Diseño del Formulario modificación del grupo contable.	código	Terminado	Miguel Ticona	3
H4A5	Validación de datos del formulario de modificación y mensajes emergentes de	código	Terminado	Miguel Ticona	3

	introducción de datos erróneos.				
H4A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H4A7	Creación de las funciones para eliminar y restaurar un grupo contable	código	Terminado	Miguel Ticona	6
H4A8	Guardar en la base de datos la información de eliminar o restaurar grupo contable	código	Terminado	Miguel Ticona	1
H4A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H5	Registrar sub grupo contable				**24**
H5A1	Diseño de formulario adición de sub grupo contable	código	Terminado	Miguel Ticona	4
H5A2	Validación de los datos del formulario del adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H5A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H5A4	Diseño del formulario modificación del sub grupo contable.	código	Terminado	Miguel Ticona	3
H5A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	2
H5A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H5A7	funciones para eliminar y restaurar un sub grupo contable	código	Terminado	Miguel Ticona	3
H5A8	Guardar en la bd la información modificada de eliminar o restaurar sub grupo contable	código	Terminado	Miguel Ticona	1
H5A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H6	Registrar ubicación geográfica				**25**
H6A1	Diseño del formulario para adición de ubicación geográfica	código	Terminado	Miguel Ticona	4
H6A2	Validación de los datos del formulario del adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H6A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H6A4	Diseño del formulario modificación de la ubicación geográfica.	código	Terminado	Miguel Ticona	3

H6A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	3
H6A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H6A7	Creación de las funciones para eliminar y restaurar una ubicación geográfica.	código	Terminado	Miguel Ticona	3
H6A8	Guardar en la bd la información de eliminar o restaurar ubicación geográfica.	código	Terminado	Miguel Ticona	1
H3A4	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H7	Registrar piso planta u otra división de la ubicación geográfica.				24
H7A1	Diseño del formulario adición del piso o planta	código	Terminado	Miguel Ticona	4
H7A2	Validación de los datos del formulario de adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H7A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H7A4	Diseño del formulario modificación del piso, planta u otra división de la ubicación geográfica.	código	Terminado	Miguel Ticona	3
H7A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	2
H7A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H7A7	Creación de las funciones para eliminar y restaurar información del piso o planta.	código	Terminado	Miguel Ticona	3
H7A8	Guardar en la bd la información de eliminar o restaurar el piso o planta.	código	Terminado	Miguel Ticona	1
H7A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
				Horas	**193**

Figura 3.6: Spring Backlog 1
Fuente: [Elaboración propia]

3.3.4.1.2. Diagrama de horas de esfuerzo para el primer Sprint

SPRINT	INICIO	DURACIÓN
1	7-ene-13	24

	L 7-ene	M 8-ene	X 9-ene	J 10-ene	V 11-ene	L 14-ene	M 15-ene	X 16-ene	J 17-ene	V 18-ene	L 21-ene	M 22-ene	X 23-ene	J 24-ene	V 25-ene	L 28-ene	M 29-ene	X 30-ene	J 31-ene	V 1-feb	L 4-feb	M 5-feb	X 6-feb	J 7-feb
Tareas pendientes	49	48	47	46	46	44	42	40	39	38	38	37	36	35	33	30	28	25	23	19	15	10	7	3
Horas de trabajo pendientes	190	182	172	165	157	149	139	131	127	121	114	106	105	96	90	84	77	66	60	50	40	28	17	8

PILA DEL SPRINT						ESFUERZO																			
Backlog		Tarea	Tipo	Estado	Responsa	L 7-ene	M 8-ene	X 9-ene	J 10-ene	V 11-ene	L 14-ene	M 15-ene	X 16-ene	J 17-ene	V 18-ene	L 21-ene	M 22-ene	X 23-ene	J 24-ene	V 25-ene	L 28-ene	M 29-ene	X 30-ene	J 31-ene	V 1-feb	L 4-feb	M 5-feb	X 6-feb	J 7-feb
H1	A1	Instalación de manejador de base		Terminada	Miguel Ticona	1																					
	A2	Análisis y diseño de la base de	Análisis	Terminada	Miguel Ticona	10	3																				
	A3	Elaboración de las tablas en la	Codificación	Terminada	Miguel Ticona	9	9	2																			
H2	A1	Diseño del formulario de login	Codificación	Terminada	Miguel Ticona	15	15	15	10	2																	
	A2	Validación de datos del formulario	Codificación	Terminada	Miguel Ticona	4	4	4	4	4																	
	A3	query de login lado servidor de BD	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	1																
	A4	Creación del hook (gancho) del	Codificación	Terminada	Miguel Ticona	6	6	6	6	6	6																
	A5	creacion de imagen captcha para	Codificación	Terminada	Miguel Ticona	4	4	4	4	4	4	1															
	A6	creacion de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4	4	4	4	4	4	4															
H3	A1	Configuracion de la plantilla	Codificación	Terminada	Miguel Ticona	5	5	5	5	5	5	5	2														
	A2	diseño del menu, header, leter,	Codificación	Terminada	Miguel Ticona	8	8	8	8	8	8	8	6														
	A2	Creacion de las hojas de estilo	Codificación	Terminada	Miguel Ticona	12	12	12	12	12	12	12	12	12	5												
	A4	creacion de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	1											
H4	A1	diseño del form grupo contable	Codificación	Terminada	Miguel Ticona	5	5	5	5	5	5	5	5	5	5	5	5	5									
	A2	Validación de datos de adicion	Codificación	Terminada	Miguel Ticona	5	5	5	5	5	5	5	5	5	5	5	5	5	1								
	A3	Guardar en bd los datos grupo cont	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3	3	3	3	3								
	A4	diseño del form modif grupo	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1							
	A5	validacion de datos modificados	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3							
	A6	Guardar datos modificados	Codificación	Terminada	Miguel Ticona	2	2	2	2	2	2	2	2	2	2	2	2	2	2								
	A7	elim, restaurar grupo contable	Codificación	Terminada	Miguel Ticona	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6						
	A8	guardar en bd if de elim y restaur	Codificación	Terminada	Miguel Ticona	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						
	A9	creacion de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4						
H5	A1	diseño formulario adicion sub grupo	Codificación	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4					
	A2	validacion de los datos del sub	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3					
	A3	guardar en la bd datos del sub	Codificación	Terminada	Miguel Ticona	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	A4	diseño del form modific sub grupo	Codificación	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3				

Figura 3.7: Diagrama horas de esfuerzo del primer sprint
Fuente: [Elaboración propia]

3.3.4.1.3. Gráfica (Burn Down)

Gráfico que permite determinar el tiempo que queda para terminar el sprint

Proyecto
Sistema Web de Control y Adm de Activos Fijos - SENAPE

SPRINT	INICIO	DÍAS
1	7-ene-13	24

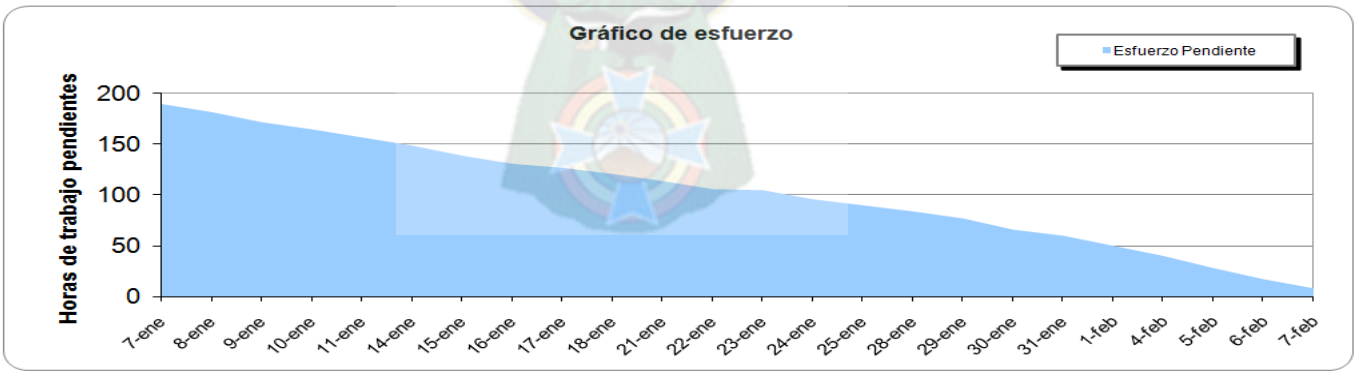


Figura 3.8: Gráfico burn down del primer Sprint
Fuente: [Elaboración propia]

Gráfico de tareas pendientes

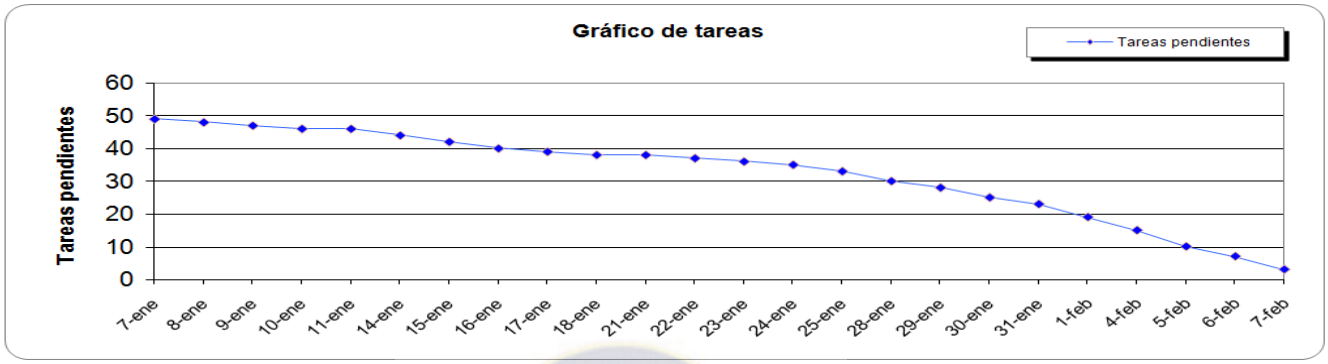


Figura 3.9: Gráfico horas de tareas pendientes 1er sprint
Fuente: [Elaboración propia]

Gráfico de horas pendientes individual

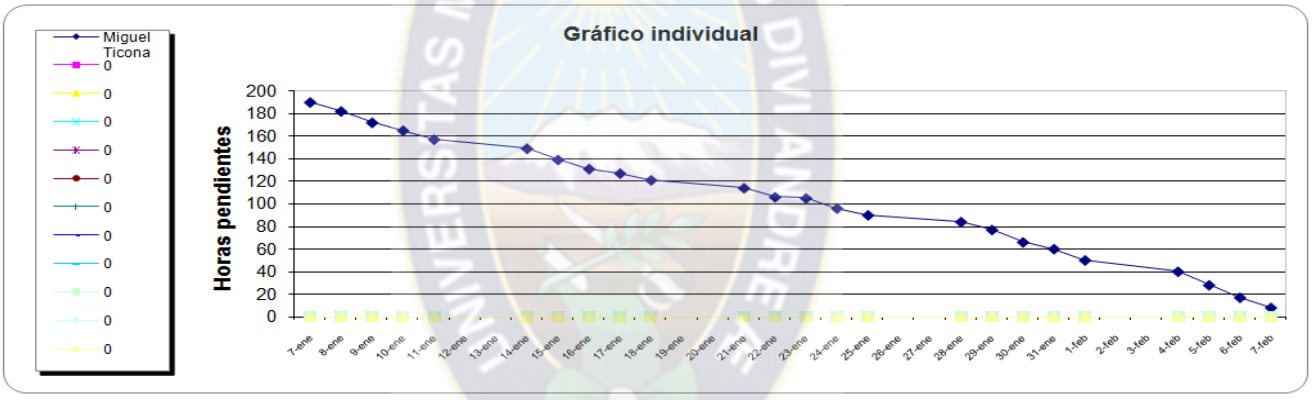


Figura 3.10: Gráfico horas pendiente individual 1er sprint
Fuente: [Elaboración propia]

Tenemos que la estimación del esfuerzo para el primer Sprint es de:

190 horas, el cual podemos convertir a días con la siguiente relación.

8 HORAS ESFUERZO = 1 DIA DE ESFUERZO (Ecuación 3.1)

190 horas = 24 días, la metodología Scrum sugiere tener para cada Sprint una duración de un mes; por tal motivo es aceptado por el Product Owner.

3.3.4.2. Segundo Sprint

Se reparten las tareas que se vean convenientes, realizando la estimación de la tarea y la preparación del ejecutor de la tarea para el segundo sprint.

3.3.4.2.1 Pila del segundo Sprint

SPRINT	INICIO	DURACIÓN
2	11 – 02 - 2013	24 días

PILA DEL SPRINT 2 O SPRINT BACKLOG

Backlog ID	Tarea	Tipo	Estado	Responsable	Estimado
H8	Registrar ambiente de trabajo				**27**
H8A1	Diseño del formulario adición del ambiente de trabajo.	código	Terminado	Miguel Ticona	4
H8A2	Validación de los datos del formulario de adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H8A3	Guardar los datos del formulario en la base de datos.	código	Terminado	Miguel Ticona	2
H8A4	Diseño del formulario modificación de información del ambiente de trabajo.	código	Terminado	Miguel Ticona	3
H8A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	3
H8A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H8A7	Creación de las funciones para eliminar y restaurar un ambiente de trabajo.	código	Terminado	Miguel Ticona	4
H8A8	Guardar en al bd la información de eliminar o restaurar el ambiente de trabajo.	código	Terminado	Miguel Ticona	2
H8A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H9	Registrar estados de activos fijos				**26**
H9A1	Diseño del formulario, adición de estados de activos fijos.	código	Terminado	Miguel Ticona	4
H9A2	Validación de los datos del formulario de adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3

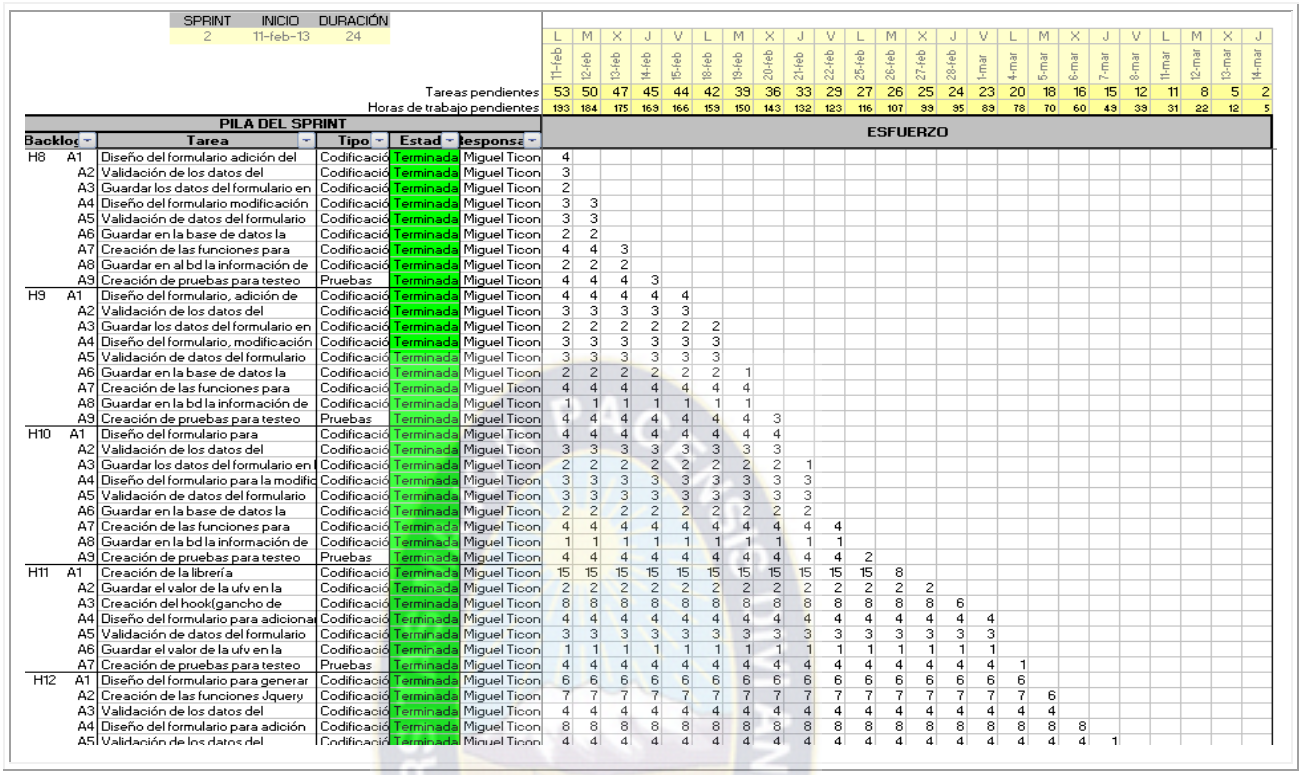
H9A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H9A4	Diseño del formulario, modificación de estados de los activos fijos.	código	Terminado	Miguel Ticona	3
H9A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	3
H9A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H9A7	Creación de las funciones para eliminar y restaurar un estado de activos fijos.	código	Terminado	Miguel Ticona	4
H9A8	Guardar en la bd la información de eliminar o restaurar el estado.	código	Terminado	Miguel Ticona	1
H9A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H10	Registrar empresa proveedora de activos fijos				**26**
H10A1	Diseño del formulario para adicionar empresas proveedoras de activos.	código	Terminado	Miguel Ticona	4
H10A2	Validación de los datos del formulario de adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H10A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H10A4	Diseño del formulario para la modificación de los datos de las empresas proveedoras.	código	Terminado	Miguel Ticona	3
H10A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	3
H10A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H10A7	Creación de las funciones para eliminar y restaurar una empresa proveedora.	código	Terminado	Miguel Ticona	4
H10A8	Guardar en la bd la información de eliminar o restaurar una empresa proveedora.	código	Terminado	Miguel Ticona	1
H10A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H11	Registrar UFV del día				**37**

H11A1	Creación de la librería load_ufv.php para descargar el valor de la ufv del día, de la pagina del Banco Central de Bolivia	código	Terminado	Miguel Ticona	15
H11A2	Guardar el valor de la ufv en la base de datos.	código	Terminado	Miguel Ticona	2
H11A3	Creación del hook(gancho de codeigniter) para verificar que se haya descargado el valor de la ufv del día, caso contrario para que llame a la librería load_ufv.php para que descargue dicho valor.	código	Terminado	Miguel Ticona	8
H11A4	Diseño del formulario para adicionar la ufv del día de forma manual (esta opción servirá en caso de que no se pueda bajar de forma automática la ufv del BCB)	código	Terminado	Miguel Ticona	4
H11A5	Validación de datos del formulario de adición de ufv y mensajes emergentes de la introducción de datos erróneos.	Código	Terminado	Miguel Ticona	3
H11A6	Guardar el valor de la ufv en la base de datos.	Código	Terminado	Miguel Ticona	1
H11A7	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H12	Registrar activos fijos adquiridos por el SENAPE				**51**
H12A1	Diseño del formulario para generar el código del activo fijo.	código	Terminado	Miguel Ticona	6
H12A2	Creación de las funciones Jquery Ajax para generar peticiones y respuestas al servidor de manera asíncrona para generar el código del activo fijo de acuerdo a requerimiento del SENAPE.	código	Terminado	Miguel Ticona	7
H12A3	Validación de los datos del formulario para generar el código del activo y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	4
H12A4	Diseño del formulario para adición de datos del nuevo activo fijo	código	Terminado	Miguel Ticona	8
H12A5	Validación de los datos del formulario de adición de activo fijo y mensajes emergentes de introducir datos erróneos	código	Terminado	Miguel Ticona	4

H12A6	Guardar en la base de datos la información del nuevo activo fijo	código	Terminado	Miguel Ticona	3
H12A7	Diseño del pdf del formulario de alta del nuevo activo fijo	código	Terminado	Miguel Ticona	6
H12A8	Diseño del pdf del código de barras del nuevo activo fijo	código	Terminado	Miguel Ticona	5
H12A9	implementación de la clase cart (carrito de compras de Codeigniter) para generar un pdf de formulario de alta por varios activos fijos adicionados	código	Terminado	Miguel Ticona	4
H12A10	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
H13	Registrar entidad que asigna activos fijos al SENAPE				**26**
H13A1	Diseño del formulario para adicionar entidad.	código	Terminado	Miguel Ticona	4
H13A2	Validación de los datos del formulario de adición y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H13A3	Guardar los datos del formulario en la base de datos	código	Terminado	Miguel Ticona	2
H13A4	Diseño del formulario para la modificación de los datos de las entidades.	código	Terminado	Miguel Ticona	3
H13A5	Validación de datos del formulario de modificación y mensajes emergentes de introducción de datos erróneos.	código	Terminado	Miguel Ticona	3
H13A6	Guardar en la base de datos la información modificada.	código	Terminado	Miguel Ticona	2
H13A7	Creación de las funciones para eliminar y restaurar una entidad.	código	Terminado	Miguel Ticona	4
H13A8	Guardar en la bd la información de eliminar o restaurar una entidad.	código	Terminado	Miguel Ticona	1
H13A9	Creación de pruebas para testeo	pruebas	Terminado	Miguel Ticona	4
				Horas	**193**

Figura 3.11: Spring Backlog 2
Fuente: [Elaboración propia]

3.3.4.2.2. Diagrama de horas de esfuerzo para el segundo Sprint



SPRINT	INICIO	DURACIÓN
2	11-feb-13	24

	L 11-feb	M 12-feb	X 13-feb	J 14-feb	V 15-feb	L 18-feb	M 19-feb	X 20-feb	J 21-feb	V 22-feb	L 25-feb	M 26-feb	X 27-feb	J 28-feb	V 1-mar	L 4-mar	M 5-mar	X 6-mar	J 7-mar	V 8-mar	L 11-mar	M 12-mar	X 13-mar	J 14-mar
Tareas pendientes	53	50	47	45	44	42	39	36	33	29	27	26	25	24	23	20	18	16	15	12	11	8	5	2
Horas de trabajo pendientes	193	184	175	169	166	159	150	143	132	123	116	107	99	95	89	78	70	60	49	39	31	22	12	5

PILA DEL SPRINT						ESFUERZO
Backlog	Tarea	Tipo	Estado	Responsable		
H8	A1	Diseño del formulario adición del	Codificación	Terminada	Miguel Ticona	4
	A2	Validación de los datos del	Codificación	Terminada	Miguel Ticona	3
	A3	Guardar los datos del formulario en	Codificación	Terminada	Miguel Ticona	2
	A4	Diseño del formulario modificación	Codificación	Terminada	Miguel Ticona	3 3
	A5	Validación de datos del formulario	Codificación	Terminada	Miguel Ticona	3 3
	A6	Guardar en la base de datos la	Codificación	Terminada	Miguel Ticona	2 2
	A7	Creación de las funciones para	Codificación	Terminada	Miguel Ticona	4 4 3
	A8	Guardar en al bd la información de	Codificación	Terminada	Miguel Ticona	2 2 2
	A9	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4 4 4 3
H9	A1	Diseño del formulario, adición de	Codificación	Terminada	Miguel Ticona	4 4 4 4 4
	A2	Validación de los datos del	Codificación	Terminada	Miguel Ticona	3 3 3 3 3
	A3	Guardar los datos del formulario en	Codificación	Terminada	Miguel Ticona	2 2 2 2 2 2
	A4	Diseño del formulario, modificación	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3
	A5	Validación de datos del formulario	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3
	A6	Guardar en la base de datos la	Codificación	Terminada	Miguel Ticona	2 2 2 2 2 2 1
	A7	Creación de las funciones para	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4
	A8	Guardar en la bd la información de	Codificación	Terminada	Miguel Ticona	1 1 1 1 1 1 1
	A9	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4 4 4 4 4 4 4 3
H10	A1	Diseño del formulario para	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4
	A2	Validación de los datos del	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3 3 3
	A3	Guardar los datos del formulario en	Codificación	Terminada	Miguel Ticona	2 2 2 2 2 2 2 2 1
	A4	Diseño del formulario para la modific	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3 3 3 3
	A5	Validación de datos del formulario	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3 3 3 3
	A6	Guardar en la base de datos la	Codificación	Terminada	Miguel Ticona	2 2 2 2 2 2 2 2
	A7	Creación de las funciones para	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4
	A8	Guardar en la bd la información de	Codificación	Terminada	Miguel Ticona	1 1 1 1 1 1 1 1 1
	A9	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4 2
H11	A1	Creación de la librería	Codificación	Terminada	Miguel Ticona	15 15 15 15 15 15 15 15 15 15 15 8
	A2	Guardar el valor de la ufv en la	Codificación	Terminada	Miguel Ticona	2 2 2 2 2 2 2 2 2 2 2 2 2
	A3	Creación del hook(gancho de	Codificación	Terminada	Miguel Ticona	8 8 8 8 8 8 8 8 8 8 8 8 8 6
	A4	Diseño del formulario para adicionar	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
	A5	Validación de datos del formulario	Codificación	Terminada	Miguel Ticona	3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
	A6	Guardar el valor de la ufv en la	Codificación	Terminada	Miguel Ticona	1 1 1 1 1 1 1 1 1 1 1 1 1 1
	A7	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1
H12	A1	Diseño del formulario para generar	Codificación	Terminada	Miguel Ticona	6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
	A2	Creación de las funciones Jquery	Codificación	Terminada	Miguel Ticona	7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6
	A3	Validación de los datos del	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
	A4	Diseño del formulario para adición	Codificación	Terminada	Miguel Ticona	8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
	A5	Validación de los datos del	Codificación	Terminada	Miguel Ticona	4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1

Figura 3.12: Diagrama horas de esfuerzo del 2do Sprint
Fuente: [Elaboración propia]

3.3.4.2.3. Gráfica (Burn Down)

Proyecto
Sistema Web de Control y Adm de Activos Fijos - SENAPE

SPRINT	INICIO	DÍAS
2	11-feb-13	24

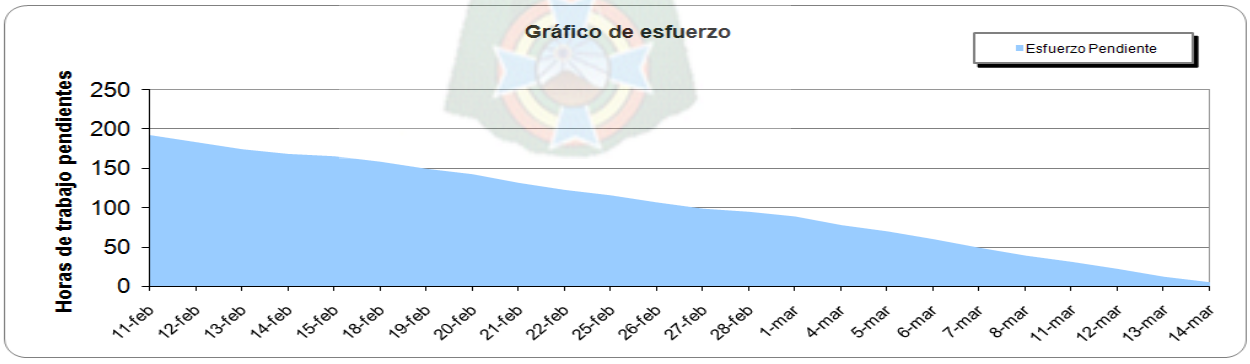


Figura 3.13: Gráfico burn down del 2do Sprint
Fuente: [Elaboración propia]

Gráfico de tareas pendientes

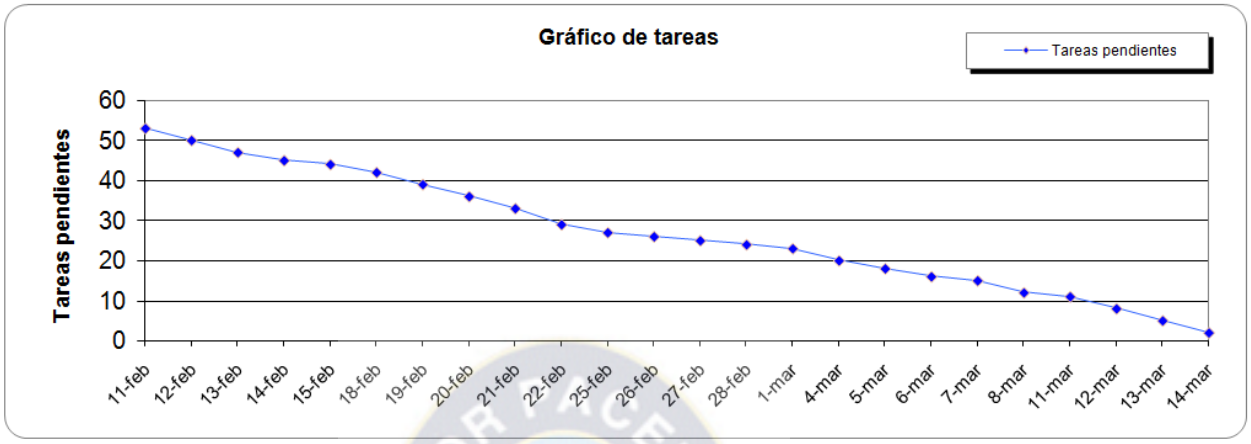


Figura 3.14: Gráfico tareas pendientes 2do Sprint
Fuente: [Elaboración propia]

Gráfico de horas pendiente individual

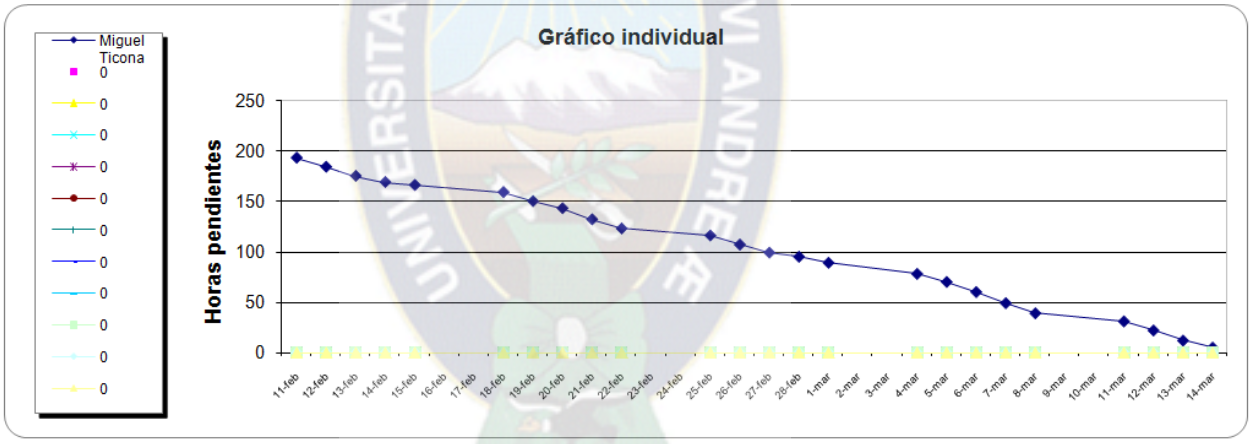


Figura 3.15: Gráfico horas pendiente individual 2do Sprint
Fuente: [Elaboración propia]

Tenemos que la estimación del esfuerzo para el segundo Sprint es de:

193 horas, el cual podemos convertir a días con la siguiente relación.

8 HORAS ESFUERZO = 1 DIA DE ESFUERZO (Ecuación 3.1)

193 horas = 24 días, la metodología Scrum sugiere tener para cada Sprint una duración de un mes; por tal motivo es aceptado por el Product Owner.

3.3.4.3. Tercer Sprint

Se reparten las tareas que se vean convenientes, realizando la estimación de la tarea y la preparación del ejecutor de la tarea.

3.3.4.3.1 Pila del tercer Sprint

SPRINT	INICIO	DURACIÓN
3	18 – 03 - 2013	22 días

PILA DEL SPRINT 3 O SPRINT BACKLOG

Backlog ID	Tarea	Tipo	Estado	Responsable	Estimado
H14	Registrar activos fijos asignados por entidades(en liquidación) al SENAPE				33
H14A1	Diseño del formulario para generar el código del activo fijo.	código	Terminado	Miguel Ticona	4
H14A2	Creación de las funciones Jquery Ajax para generar peticiones y respuestas al servidor de manera asíncrona para generar el código del activo fijo de acuerdo a requerimiento del SENAPE.	código	Terminado	Miguel Ticona	2
H14A3	Validación de los datos del formulario para generar el código del activo y mensajes emergentes de introducción de datos erróneos	código	Terminado	Miguel Ticona	3
H14A4	Diseño del formulario para adición de datos del activo fijo donado	código	Terminado	Miguel Ticona	4
H14A5	Validación de los datos del formulario de adición de activo fijo y mensajes emergentes de introducir datos erróneos	código	Terminado	Miguel Ticona	3
H14A6	Guardar en la base de datos la información del activo fijo donado	código	Terminado	Miguel Ticona	2
H14A7	Diseño del pdf del formulario de alta del activo fijo donado	código	Terminado	Miguel Ticona	4
H14A8	Diseño del pdf del código de barras del activo fijo donado	código	Terminado	Miguel Ticona	3
H14A9	implementación de la clase cart (carrito de compras de Codeigniter) para generar un pdf de formulario de alta por varios activos fijos adicionados	código	Terminado	Miguel Ticona	4
H14A10	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H15	Permitir modificar datos de los activos fijos				22
H15 A1	Diseño del formulario para ingresar código del activo a modificar.	Código	Terminado	Miguel Ticona	1

H15 A2	Diseño del formulario modificación de activos nuevos	Código	Terminado	Miguel Ticona	4
H15 A3	Validación de los datos del formulario de modificación de activos nuevos y mensajes emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H15 A4	Guardar en la bd la información del formulario de modificación.	Código	Terminado	Miguel Ticona	1
H15 A5	Diseño del formulario modificación de activos donados	Código	Terminado	Miguel Ticona	4
H15 A6	Validación de los datos del formulario de modificación de activos donados y mensajes emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H15 A7	Guardar en la bd la información del formulario de modificación	Código	Terminado	Miguel Ticona	2
H15 A8	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H16	Asignación de Activos Fijos				40
H16 A1	Diseño del formulario asignación de activos por ubicación de ambiente de trabajo	Código	Terminado	Miguel Ticona	4
H16 A2	Creación de las funciones Jquery Ajax para generar peticiones y respuestas al servidor de manera asíncrona para listar activos de determinado ambiente de trabajo.	Código	Terminado	Miguel Ticona	5
H16 A3	Validación de los datos del formulario de asignación de activos y mensajes emergentes por introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H16 A4	Guardar en la bd la información de la asignación de activos	Código	Terminado	Miguel Ticona	2
H16 A5	Diseño del pdf de asignación de activos	Código	Terminado	Miguel Ticona	4
H16 A6	Diseño del formulario de asignación de activos fijos por código del activo	Código	Terminado	Miguel Ticona	4
H16 A7	Creación del las funciones jquery ajax para generar peticiones y respuestas asíncronas para cargar activos a asignar	Código	Terminado	Miguel Ticona	5
H16 A8	Validación de los datos del formulario y mensajes emergentes por introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H16 A9	Guardar en la base de datos la información de la asignación de activos	Código	Terminado	Miguel Ticona	2
H16 A10	Diseño del pdf de asignación de activos	Código	Terminado	Miguel Ticona	4
H16 A11	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H17	Devolución de Activos Fijos				21
H17A1	Diseño del formulario devolución de activos fijos	Código	Terminado	Miguel Ticona	4

H17A2	Creación de las funciones jquery ajax para listar activos asignados a un determinado funcionario	Código	Terminado	Miguel Ticona	4
H17A3	Validación de datos del formulario de devolución de activos y mensajes emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H17A4	Guardar en la bd información de activos fijos devueltos	Código	Terminado	Miguel Ticona	2
H17A5	Diseño del pdf del formulario de devolución de activos fijos	Código	Terminado	Miguel Ticona	4
H17A6	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H18	Salidas y retornos de Activos Fijos	Miguel Ticona	Baja		32
H18A1	Diseño del formulario salida de activos	Código	Terminado	Miguel Ticona	4
H18A2	Validación de datos del formulario de salida de activos y mensajes emergentes de introducir datos erróneos.	Código	Terminado	Miguel Ticona	3
H18A3	Creación de las funciones ajax jquery y la clase cart(carro de compras de codeigniter) para listar y cargar activos para la salida	Código	Terminado	Miguel Ticona	5
H18A4	Guardar en la base de datos la información de activos con salida de la institución	Código	Terminado	Miguel Ticona	2
H18A5	Diseño del formulario de retorno de activos	Código	Terminado	Miguel Ticona	4
H18A6	Creación de las funciones jquery ajax para listar activos con salida de un determinado funcionario.	Código	Terminado	Miguel Ticona	5
H18A7	Validación de datos del formulario de retorno y mensajes emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H18A8	Guardar en la bd la información modificada	Código	Terminado	Miguel Ticona	2
H18A9	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H19	Ingreso temporal de activos fijos	Miguel Ticona	Baja		31
H19A	Diseño del formulario de ingreso temporal de activos	Código	Terminado	Miguel Ticona	5
H19A2	Validación de datos del formulario de ingreso temporal de activos	Código	Terminado	Miguel Ticona	3
H19A3	Guardar en la bd la información de los activos con ingreso temporal ajenos a la institución	Código	Terminado	Miguel Ticona	2
H19A4	Diseño del pdf del formulario de ingreso temporal de activos	Código	Terminado	Miguel Ticona	5

H19A5	Diseño del formulario modificación de datos de activos con ingreso temporal	Código	Terminado	Miguel Ticona	4
H19A6	Validación de los datos introducidos al formulario y mensajes emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	3
H19A7	Guardar en la bd la información modificada	Código	Terminado	Miguel Ticona	2
H19A8	Realizar la función para la devolución del activo con ingreso temporal.	Código	Terminado	Miguel Ticona	3
H19A9	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
				Horas	179

Figura 3.16: Spring Backlog 3
Fuente: [Elaboración propia]

3.3.4.3.2 Diagrama de horas de esfuerzo para el tercer Sprint

SPRINT	INICIO	DURACIÓN
3	18-mar-13	22

	L 18-mar	M 19-mar	X 20-mar	J 21-mar	V 22-mar	L 25-mar	M 26-mar	X 27-mar	J 28-mar	V 29-mar	L 1-abr	M 2-abr	X 3-abr	J 4-abr	V 5-abr	L 8-abr	M 9-abr	X 10-abr	J 11-abr	V 12-abr	L 15-abr	M 16-abr
Tareas pendientes	53	50	48	45	44	42	39	36	35	34	32	30	30	29	26	24	22	18	14	11	8	4
Horas de trabajo pendientes	179	170	162	154	150	143	137	128	124	117	112	106	106	98	90	84	74	61	49	37	26	12

PILA DEL SPRINT						ESFUERZO																
Backlog		Tarea	Tipo	Estado	Responsable																	
H14	A1	Diseño del formulario para generar	Codificació	Terminada	Miguel Ticona	4																
	A2	Creación de las funciones Jquery	Codificació	Terminada	Miguel Ticona	2																
	A3	Validación de los datos del	Codificació	Terminada	Miguel Ticona	3																
	A4	Diseño del formulario para adición	Codificació	Terminada	Miguel Ticona	4	4															
	A5	Validación de los datos del	Codificació	Terminada	Miguel Ticona	3	3															
	A6	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	2	2	1														
	A7	Diseño del pdf del formulario de	Codificació	Terminada	Miguel Ticona	4	4	4														
	A8	Diseño del pdf del código de	Codificació	Terminada	Miguel Ticona	3	3	3														
	A9	implementación de la clase cart (car	Codificació	Terminada	Miguel Ticona	4	4	4	4													
	A10	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4	4	4	4	4												
H15	A1	Diseño del formulario para ingresar	Codificació	Terminada	Miguel Ticona	1	1	1	1	1												
	A2	Diseño del formulario modificación	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	2											
	A3	Validación de los datos del formular	Codificació	Terminada	Miguel Ticona	3	3	3	3	3	3											
	A4	Guardar en la bd la información del	Codificació	Terminada	Miguel Ticona	1	1	1	1	1	1											
	A5	Diseño del formulario modificación	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4										
	A6	Validación de los datos del	Codificació	Terminada	Miguel Ticona	3	3	3	3	3	3	3										
	A7	Guardar en la bd la información del	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2	2										
	A8	Creación de pruebas para testeo	Pruebas	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4									
H16	A1	Diseño del formulario asignación	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4								
	A2	Creación de las funciones Jquery	Codificació	Terminada	Miguel Ticona	5	5	5	5	5	5	5	5	5	2							
	A3	Validación de los datos del formular	Codificació	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3	3	3							
	A4	Guardar en la bd la información de l	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2	2	2	2	2	2						

Figura 3.17: Diagrama horas de esfuerzo 3er Sprint
Fuente: [Elaboración propia]

3.3.4.3.3. Gráfica (Burn Down)

Proyecto	SPRINT	INICIO	DÍAS
Sistema Web de Control y Adm de Activos Fijos - SENAPE	3	18-mar-13	22

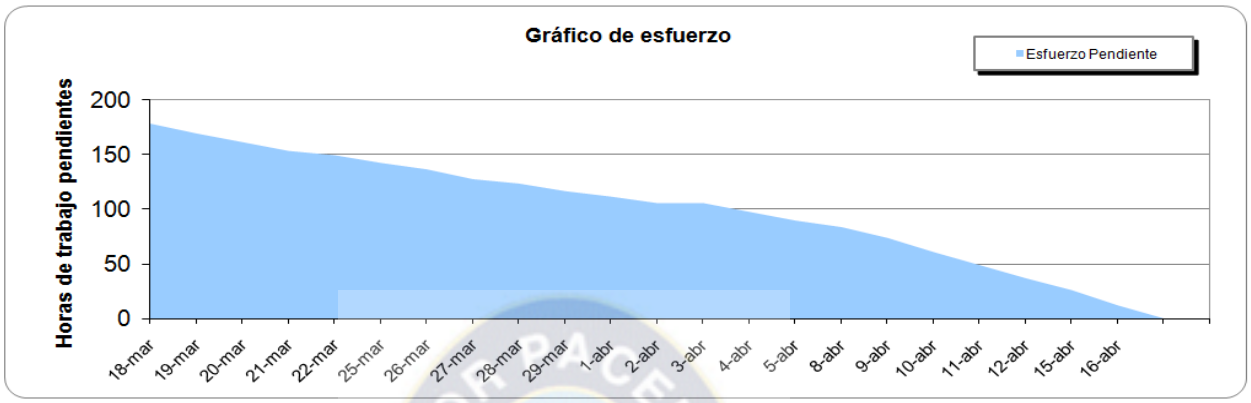


Figura 3.18: Gráfico Burn Down 3er Sprint
Fuente: [Elaboración propia]

Gráfico de tareas pendientes

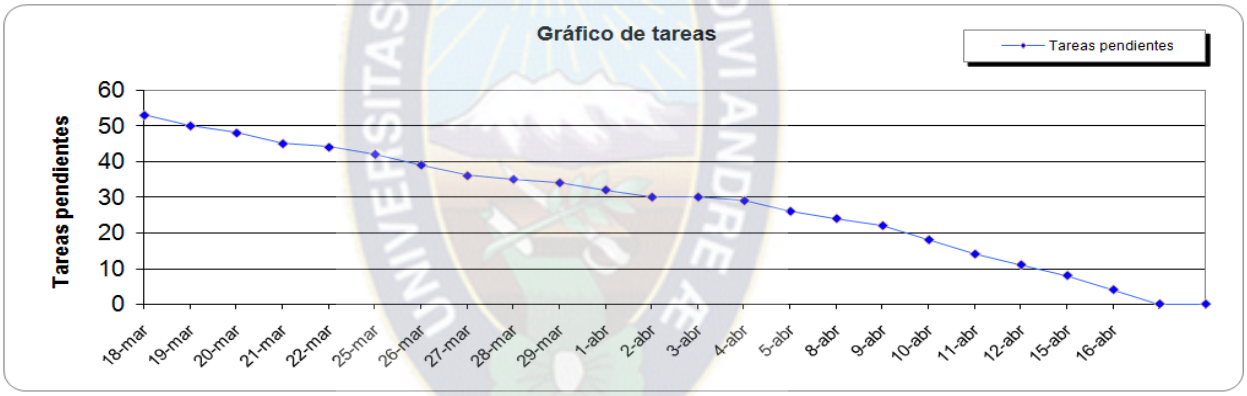


Figura 3.19: Gráfico tareas pendientes 3er Sprint
Fuente: [Elaboración propia]

Gráfico de horas pendiente individual

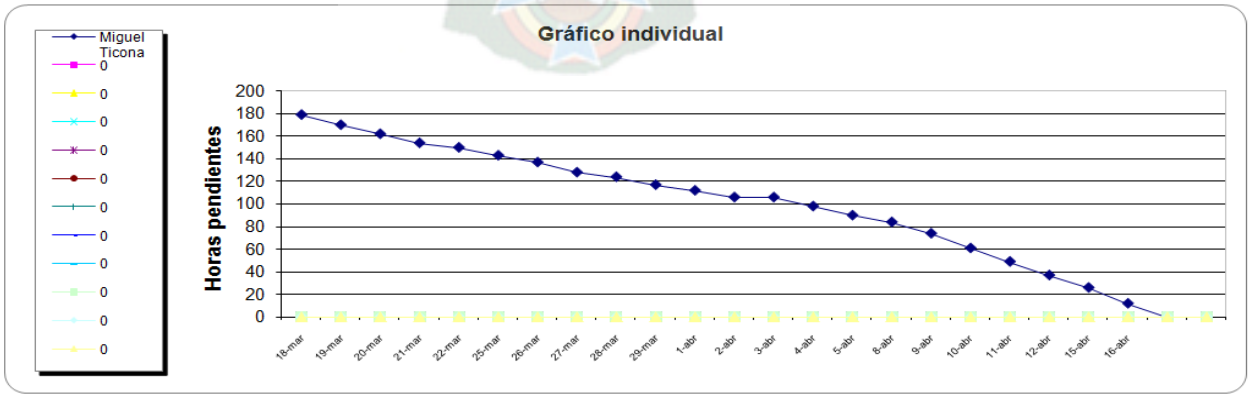


Figura 3.20: Gráfico horas pendientes individual 3er Sprint
Fuente: [Elaboración propia]

Tenemos que la estimación del esfuerzo para el tercer Sprint es de:

179 horas, el cual podemos convertir a días con la siguiente relación.

8 HORAS ESFUERZO = 1 DIA DE ESFUERZO (Ecuación 3.1)

179 horas = 22 días, la metodología Scrum sugiere tener para cada Sprint una duración de un mes; por tal motivo es aceptado por el Product Owner.

3.3.4.4. Cuarto Sprint

Se reparten las tareas que se vean convenientes, realizando la estimación de la tarea y la preparación del ejecutor de la tarea.

3.3.4.4.1 Pila del cuarto Sprint

SPRINT	INICIO	DURACIÓN
4	22 – 04 - 2013	23 días

PILA DEL SPRINT 4 O SPRINT BACKLOG

Backlog ID	Tarea	Tipo	Estado	Responsable	Estimado
H20	Dar de baja Activo Fijo				20
H20A1	Diseño del formulario baja de activos	código	Terminado	Miguel Ticona	4
H20A2	Creación de la función jquery ajax para mostrar datos del activo a ser dado de baja	código	Terminado	Miguel Ticona	4
H20A3	Validación de datos del formulario de baja y mensajes de error emergentes de introducir datos no validos	código	Terminado	Miguel Ticona	3
H20A4	Guardar en la base de datos la información del formulario de baja	código	Miguel Ticona	Baja	2
H20A5	Diseño del pdf formulario de baja de activos fijos	código	Miguel Ticona	Baja	3
H20A6	Creación de pruebas para testeo	Pruebas	Miguel Ticona	Baja	4
H21	Actualizar datos de Activos Fijos revaluados				20
H21 A1	Diseño del formulario de actualización de datos por revalúo	Código	Terminado	Miguel Ticona	4
H21 A2	Creación de la función jquery ajax para mostrar datos del activo a modificar Datos por revalúo	Código	Terminado	Miguel Ticona	4
H21 A3	Validación de datos del formulario de revalúo y mensajes de error	Código	Terminado	Miguel Ticona	3

	emergentes de introducir datos no validos				
H21 A4	Guardar en la base de datos la información modificada por el revalúo	Código	Terminado	Miguel Ticona	2
H21 A5	Diseño del pdf formulario de revaúo de activos fijos.	Código	Terminado	Miguel Ticona	3
H21 A6	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H22	Adicionar usuarios con respectivos privilegios en el sistema				21
H22 A1	Creación del formulario para adicionar usuarios y sus respectivos privilegios	Código	Terminado	Miguel Ticona	3
H22 A2	Validación de datos del formulario de adición y mensajes de error emergentes de introducir datos no válidos	Código	Terminado	Miguel Ticona	3
H22 A3	Guardar en la base de datos la información de los usuarios	Código	Terminado	Miguel Ticona	2
H22 A4	Diseño del formulario para modificar privilegios de usuarios	Código	Terminado	Miguel Ticona	2
H22 A5	Guardar en la base de datos la información modificada	Código	Terminado	Miguel Ticona	2
H22 A6	Creación de las funciones para eliminar y restaurar usuarios	Código	Terminado	Miguel Ticona	3
H22 A7	Guardar en la base de datos la información de eliminar y restaurar usuarios	Código	Terminado	Miguel Ticona	2
H22 A8	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H23	Diseño de interfaz para cada tipo de usuario para controlar permisos y roles de usuarios a lo largo del sitio				11
H23A1	Creación de la librería que permita seleccionar el menú de acuerdo al tipo de usuario	Código	Terminado	Miguel Ticona	3
H23A2	Diseño del menú para cada tipo de usuario	Código	Terminado	Miguel Ticona	4
H23A3	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H24	Depreciación de Activos Fijos				33
H24A1	Diseño de la interfaz para procesar la depreciación de los activos fijos	Código	Terminado	Miguel Ticona	2
H24A2	Validación de datos del formulario y mensajes de error emergentes de introducir datos erróneos	Código	Terminado	Miguel Ticona	1
H24A3	Procesar la depreciación de la presente gestión	Código	Terminado	Miguel Ticona	8
H24A4	Guardar en la base de datos la información de la depreciación	Código	Terminado	Miguel Ticona	4
H24A5	Diseño de la vista para mostrar el resultado de la depreciación	Código	Terminado	Miguel Ticona	8

H24A6	Diseño de la vista para exportar las depreciaciones a Microsoft Word para su posterior impresión.	Código	Terminado	Miguel Ticona	6
H24A7	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H25	Consultas y reportes	Miguel Ticona	Baja		16
H25A1	Diseño de los formularios para mostrar los resultados de las consultas	Código	Terminado	Miguel Ticona	6
H25A2	Consultas a la base de datos de acuerdo a los requerimientos del usuario	Código	Terminado	Miguel Ticona	6
H25A3	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H26	Backup de la base de datos	Miguel Ticona	Baja		15
H26A1	Diseño del formulario para realizar el backup de la base de datos	Código	Terminado	Miguel Ticona	1
H26A2	Creación de la librería que permita importar y exportar la base de datos	Código	Terminado	Miguel Ticona	10
H26A3	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H27	Cambiar password del usuario	Miguel Ticona	Baja		15
H27A1	Diseño del formulario modificar password del usuario	Código	Terminado	Miguel Ticona	3
H27A2	Validación de datos del formulario y mensajes emergentes de introducir contraseña errónea	Código	Terminado	Miguel Ticona	3
H27A3	Guardar en la base de datos la información del nuevo password	Código	Terminado	Miguel Ticona	5
H27A4	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H28	Búsqueda de activos fijos	Miguel Ticona	Baja		18
H28A1	Diseño del formulario para introducir parámetro de búsqueda de activo	Código	Terminado	Miguel Ticona	2
H28A2	Consultas a la base de datos para listar activos que coincidan con el parámetro de búsqueda.	Código	Terminado	Miguel Ticona	5
H28A3	Diseño de la ventana para mostrar resultado de la consulta	Código	Terminado	Miguel Ticona	4
H28A4	Diseño de la ventana para mostrar datos de un determinado activo de la lista anterior.	Código	Terminado	Miguel Ticona	3
H28A5	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
H29	Transferencia de activos	Miguel Ticona	Baja		16
H29A1	Diseño del formulario de transferencia de activos	Código	Terminado	Miguel Ticona	3

H29A2	Creación de las funciones jquery ajax para y la clase cart (carro de compras de codeigniter) para cargar activos a ser transferidos.	Código	Terminado	Miguel Ticona	4
H29A3	Validación de datos del formulario y mensajes emergentes de introducir datos erróneos.	Código	Terminado	Miguel Ticona	3
H29A4	Guardar en la base de datos la información de la transferencia de activos.	Código	Terminado	Miguel Ticona	2
H29A5	Creación de pruebas para testeo	Pruebas	Terminado	Miguel Ticona	4
				Horas	185

Figura 3.21: Spring Backlog 4
Fuente: [Elaboración propia]

3.3.4.4.2. Diagrama de horas de esfuerzo para el cuarto Sprint

SPRINT	INICIO	DURACIÓN
4	22-abr-13	23

	L 22-abr	M 23-abr	X 24-abr	J 25-abr	V 26-abr	L 29-abr	M 30-abr	X 1-may	J 2-may	V 3-may	L 6-may	M 7-may	X 8-may	J 9-may	V 10-may	L 13-may	M 14-may	X 15-may	J 16-may	V 17-may	L 20-may	M 21-may	X 22-may
Tareas pendientes	50	48	45	42	39	37	33	31	28	25	24	22	20	18	16	16	15	13	11	8	6	4	1
Horas de trabajo pendientes	185	177	169	157	149	141	133	125	117	110	96	84	74	66	62	58	51	44	35	27	17	11	4

PILA DEL SPRINT						ESFUERZO																		
Backlog	Tarea	Tipo	Estad	Responsa																				
H20	A1	Diseño del formulario baja de	Codificació	Terminada	Miguel Ticona	4																		
	A2	Creación de la función jquery ajax	Codificació	Terminada	Miguel Ticona	4																		
	A3	Validación de datos del formulario	Codificació	Terminada	Miguel Ticona	3	3																	
	A4	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	2	2																	
	A5	Diseño del pdf formulario de baja	Codificació	Terminada	Miguel Ticona	3	3																	
	A6	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4																
H21	A1	Diseño del formulario de actualiz	Codificació	Terminada	Miguel Ticona	4	4	4																
	A2	Creación de la función jquery ajax	Codificació	Terminada	Miguel Ticona	4	4	4																
	A3	Validación de datos del formulario	Codificació	Terminada	Miguel Ticona	3	3	3	3															
	A4	Guardar en la base de datos la	Pruebas	Terminada	Miguel Ticona	2	2	2	2															
	A5	Diseño del pdf formulario de	Codificació	Terminada	Miguel Ticona	3	3	3	3															
	A6	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4														
H22	A1	Creación del formulario para adicio	Codificació	Terminada	Miguel Ticona	3	3	3	3	3														
	A2	Validación de datos del formulario	Codificació	Terminada	Miguel Ticona	3	3	3	3	3	2													
	A3	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2													
	A4	Diseño del formulario para	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2													
	A5	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2													
	A6	Creación de las funciones para	Pruebas	Terminada	Miguel Ticona	3	3	3	3	3	3	3												
	A7	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2	2												
	A8	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	1											
H23	A1	Creación librería para selec menú c	Codificació	Terminada	Miguel Ticona	3	3	3	3	3	3	3	3											
	A2	Diseño del menú para cada tipo de	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4											
	A3	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4										
H24	A1	Diseño interfaz procesar deprec	Codificació	Terminada	Miguel Ticona	2	2	2	2	2	2	2	2	2										
	A2	Validación de datos del formulario	Codificació	Terminada	Miguel Ticona	1	1	1	1	1	1	1	1	1										
	A3	Procesar la depreciación de la	Codificació	Terminada	Miguel Ticona	8	8	8	8	8	8	8	8	8	8									
	A4	Guardar en la base de datos la	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	4								
	A5	Diseño de vista para mostr deprec	Codificació	Terminada	Miguel Ticona	8	8	8	8	8	8	8	8	8	8	2								
	A6	Diseño vista para exportar deprec	Pruebas	Terminada	Miguel Ticona	6	6	6	6	6	6	6	6	6	6	4								
	A7	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	2								
H25	A1	Diseño de vistas para mostrar cons	Codificació	Terminada	Miguel Ticona	6	6	6	6	6	6	6	6	6	6	4	2							
	A2	Consultas a la base de datos de	Codificació	Terminada	Miguel Ticona	6	6	6	6	6	6	6	6	6	6	4								
	A3	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	4	2							
H26	A1	Diseño form para backup de la	Codificació	Terminada	Miguel Ticona	1	1	1	1	1	1	1	1	1	1	1	1	1						
	A2	Creac librería para import y export	Pruebas	Terminada	Miguel Ticona	10	10	10	10	10	10	10	10	10	10	10	10	9	5					
	A3	Creación de pruebas para testeo	Codificació	Terminada	Miguel Ticona	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4				

Figura 3.22: Diagrama horas de esfuerzo 4to Sprint
Fuente: [Elaboración propia]

3.3.4.4.3 Gráfica (Burn Down)

Proyecto	SPRINT	INICIO	DÍAS
Sistema Web de Control y Adm de Activos Fijos - SENAPE	4	22-abr-13	23

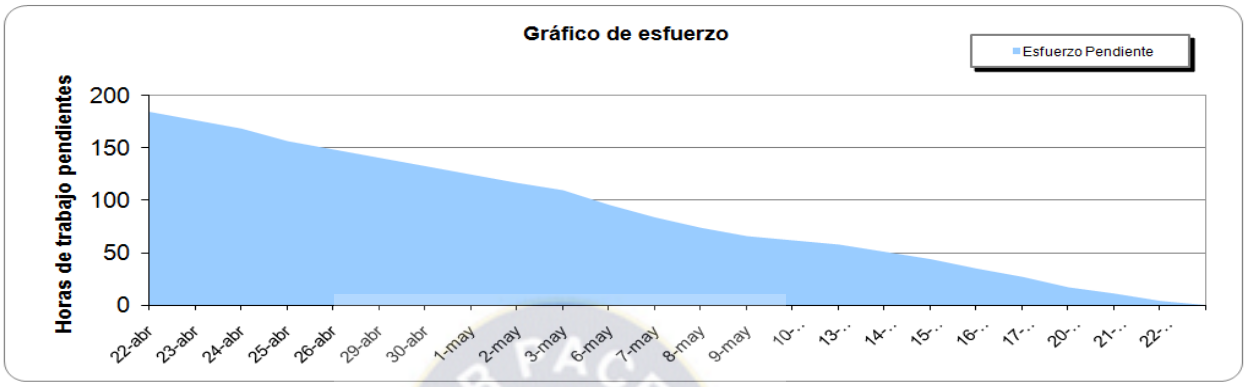


Figura 3.23: Gráfico Burn Down 4to Sprint
Fuente: [Elaboración propia]

Gráfico de tareas pendientes

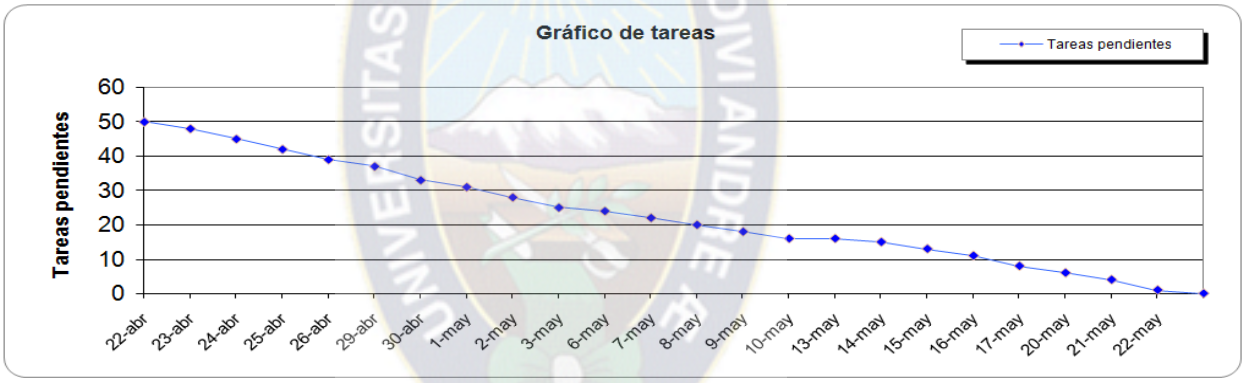


Figura 3.24: Gráfico tareas pendientes 4to Sprint
Fuente: [Elaboración propia]

Gráfico de horas pendiente individual

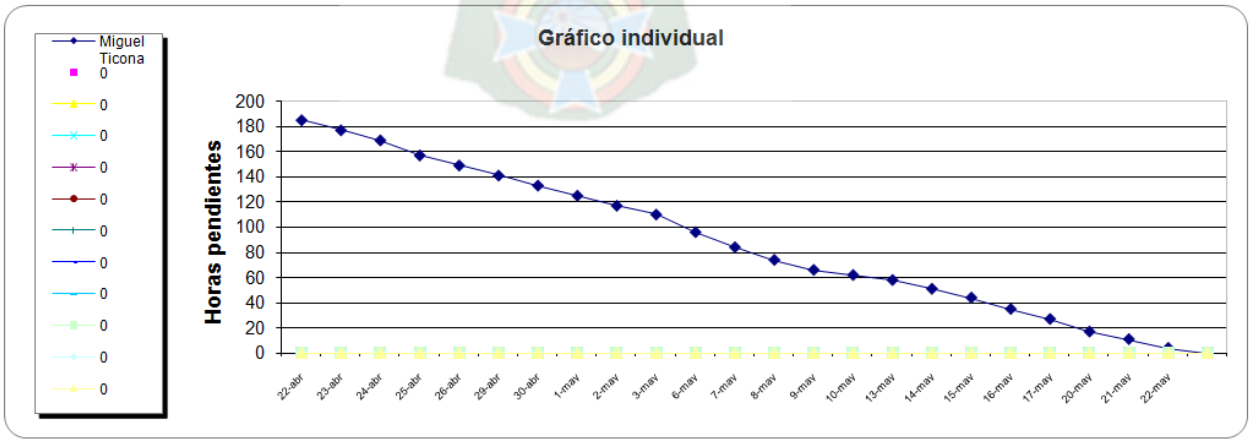


Figura 3.25: Gráfico horas pendientes individual 4to Sprint
Fuente: [Elaboración propia]

Tenemos que la estimación del esfuerzo para el cuarto Sprint es de 185 horas, el cual podemos convertir a días con la ecuación 3.1.

185 horas = 23 días, la metodología Scrum sugiere tener para cada Sprint una duración de un mes; por tal motivo es aceptado por el Product Owner.

3.3.4.5. Desarrollo del sistema en cada iteración o sprint

A continuación se muestran partes del sistema ya funcionales añadidos dentro de cada sprint.

Figura 3.26: Funcionalidad dentro del primer sprint – autentificación de usuario
Fuente: [Elaboración propia]

Figura 3.27: Funcionalidad dentro del primer sprint – interfaz de usuario – generación de código
Fuente: [Elaboración propia]

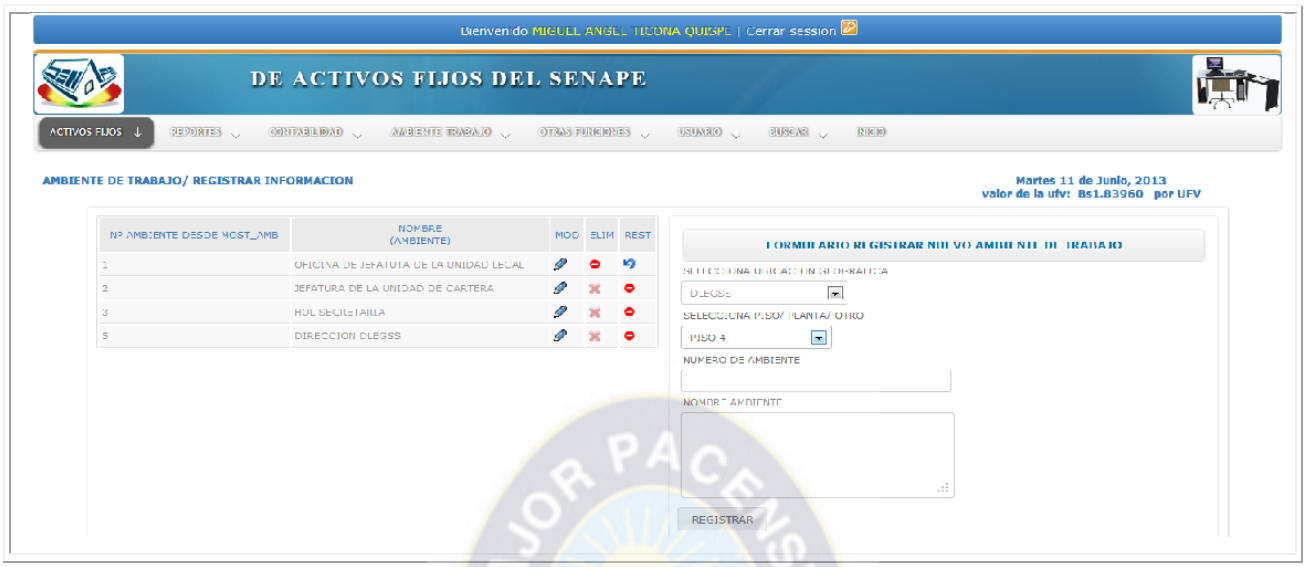
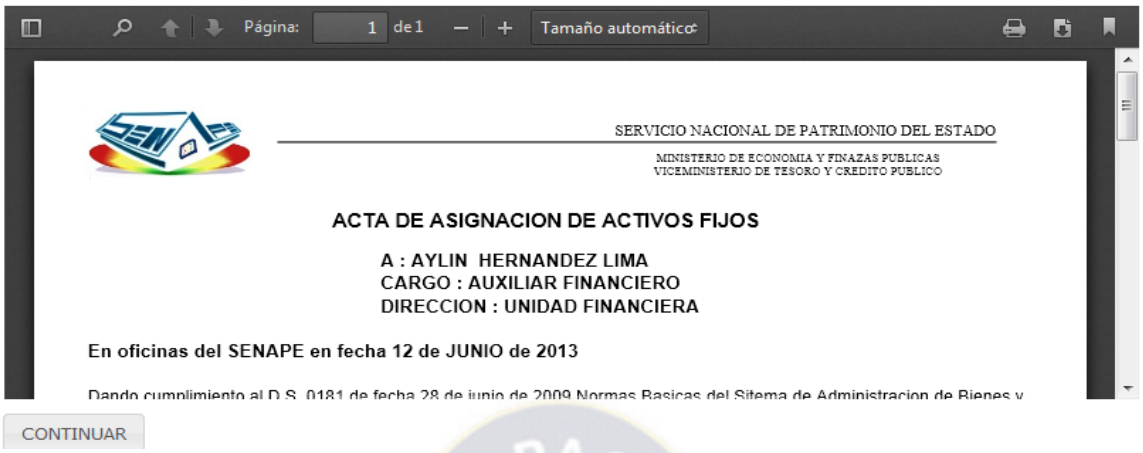


Figura 3.28: Funcionalidad dentro del segundo sprint – registrar ambientes de trabajo
Fuente: [Elaboración propia]

Figura 3.29: Funcionalidad dentro del tercer sprint – asignación de activos fijos
Fuente: [Elaboración propia]

MOSTRAMOS EL PDF EMBIBEIDO DEL DETALLE DE ASIGNACION DE ACTIVOS



CONTINUAR

Figura 3.30: Funcionalidad dentro del tercer sprint – acta de asignación de act. fijos en formato pdf.
Fuente: [Elaboración propia]

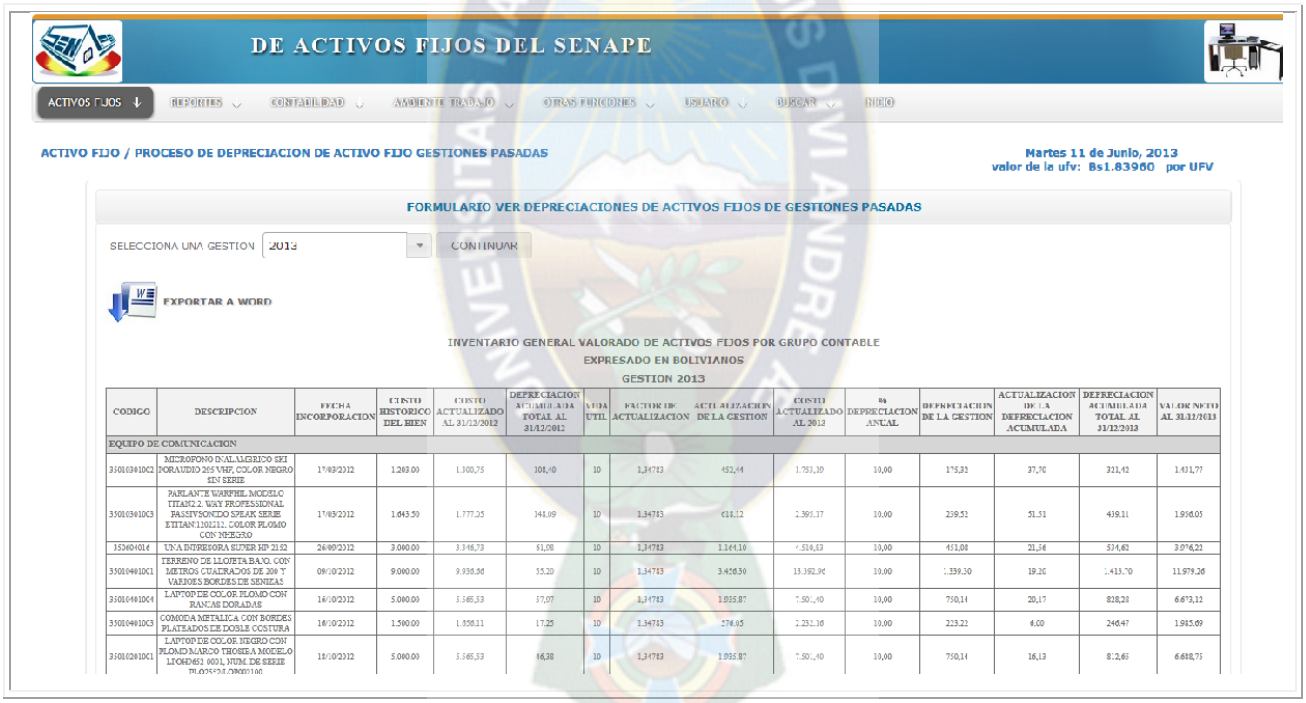


Figura 3.31: Funcionalidad dentro del cuarto sprint – listado de depreciación de activos fijos
Fuente: [Elaboración propia]

3.3.4.6 Pruebas unitarias y clasificación de errores

Las pruebas unitarias se realizaron en todo momento dentro del desarrollo del sistema, permitieron automatizar las pruebas del software.

En esta práctica las pruebas se deben escribir antes de que se escriba el código para que este cumpla con las pruebas; por tal motivo, al ejecutar las pruebas

unitarias inicialmente mostrará mensaje de error o falla ya que la función a evaluar aun no ha sido creada.

El framework Codeigniter cuenta con la clase Unit_test, que permitió realizar estas pruebas unitarias, consistiendo en una función de evaluación y dos funciones de resultado. De esta forma se logró determinar si se está produciendo el tipo de datos y el resultado correcto.

Este es un ejemplo de prueba unitaria para verificar que el resultado de una función que realiza una consulta a la base de datos devuelve el valor booleno “true”:

```
function testmodel(){
        $this->load->model('privadomodel');
        $this->load->library('unit_test');
        $this->unit->use_strict(TRUE);
        $dato = $this->privadomodel->af_asignado(10);
        $esperado = true;
        $this->unit->run($dato,$esperado,'test de pruebas unitarias');
        echo $this->unit->report();
}
```

Resultado en el navegador:

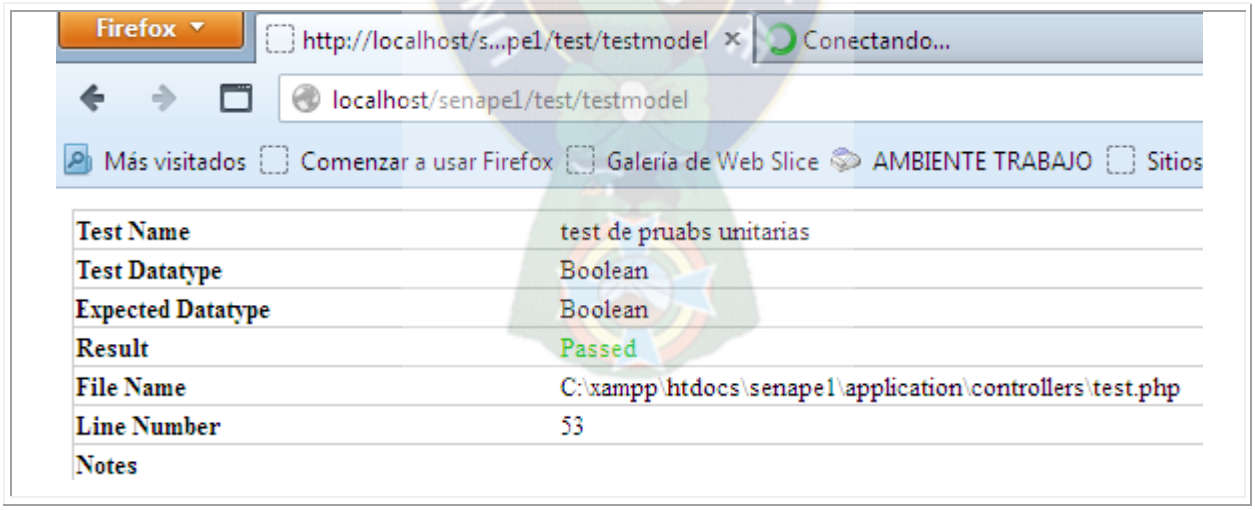


Figura 3.32: Resultado de los test en el navegador.

Con la ayuda de las pruebas unitarias se logró producir software de calidad; sin embargo, también se deben tomar en cuenta la siguiente clasificación de errores que se produjeron:

- **Errores de sintaxis o de compilación:** Los errores de escritura en el código fuente fueron corregidos oportunamente, debido a que fueron detectados por el compilador.
- **Errores de ejecución:** Estos errores se deben a operaciones no permitidas como realizar divisiones entre cero, falta de validación de datos de entrada en los formularios. También fueron detectados y corregidos en las pruebas realizadas antes de dar por terminado cada sprint.
- **Errores de lógica:** Corresponden a la obtención de resultados que no son correctos; sin embargo, con la ayuda de las pruebas unitarias fueron detectadas y corregidas a tiempo.
- **Errores en la especificación:** este tipo de errores se refieren al mal diseño del programa debido a la mala comunicación con el usuario y generalmente son detectados cuando ya se ha concluido el diseño e instalación del programa; sin embargo, debido a la utilización de la metodología Scrum que involucra al usuario en todo momento facilitando la comunicación, se logró eliminar este tipo de errores ya que el usuario presenta la lista de requerimientos (Product Backlog o pila de requerimientos) con un orden de prioridad.

3.3.4.7. Integración continua con Phing.

Una vez concluido cada sprint, con la finalidad de aumentar la velocidad de entrega de la nueva versión del sistema y disminuir los tiempos de integración, se utilizo Phing que es una herramienta análoga a Ant de Apache, para el PHP.

Phing es una herramienta desarrollada en PHP, permite mediante la generación de archivos XML, facilitar las tareas de automatización. Se ejecuta a través de la línea de comandos de Windows escribiendo el comando “phing”

Una vez ejecutado el comando “phing”, el script realiza las siguientes acciones:

- Verifica las versiones de PHP del servidor remoto y la maquina local de donde se copiaran al servidor remoto los archivos de actualización.
- Verifica las versiones del servidor web Apache del servidor remoto y la maquina local de donde se copiaran al servidor remoto los archivos de actualización.

- Ejecuta los archivos con las instrucciones en lenguaje SQL para realizar las actualizaciones a la base de datos PosgreSql del servidor remoto (creación de nuevas tablas, modificación de las tablas ya creadas, etc), utilizando como parámetros de conexión a la base de datos el host, el password y el nombre de la base de datos del servidor remoto.
- Crea el directorio de control de versiones de cada actualización.
- Copia al servidor remoto (mediante el protocolo de transferencia de archivos SFTP) todos los archivos con las actualizaciones de la nueva versión del sistema. Por razones de seguridad se utiliza el protocolo SFTP que utiliza SSH debido a que los comandos y los datos transferidos entre el usuario y el servidor están cifrados, lo que evita que usuarios no autorizados tengan acceso a esta información.

A continuación se muestra la ejecución de la herramienta phing en la línea de comandos.

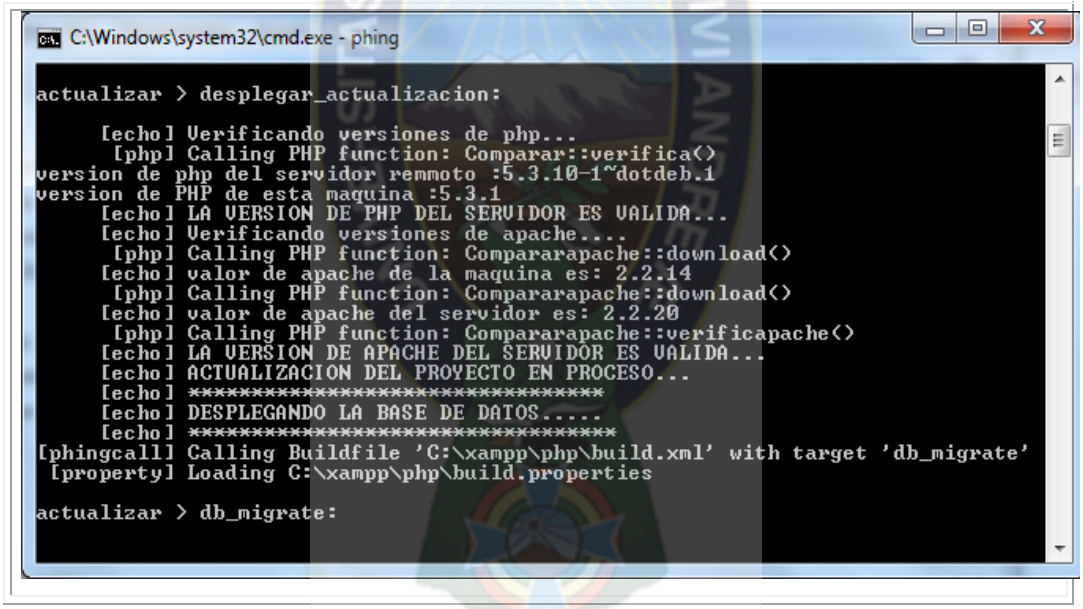


Figura 3.33: Resultado de la ejecución de Phing en la línea de comandos - Verificación de versiones

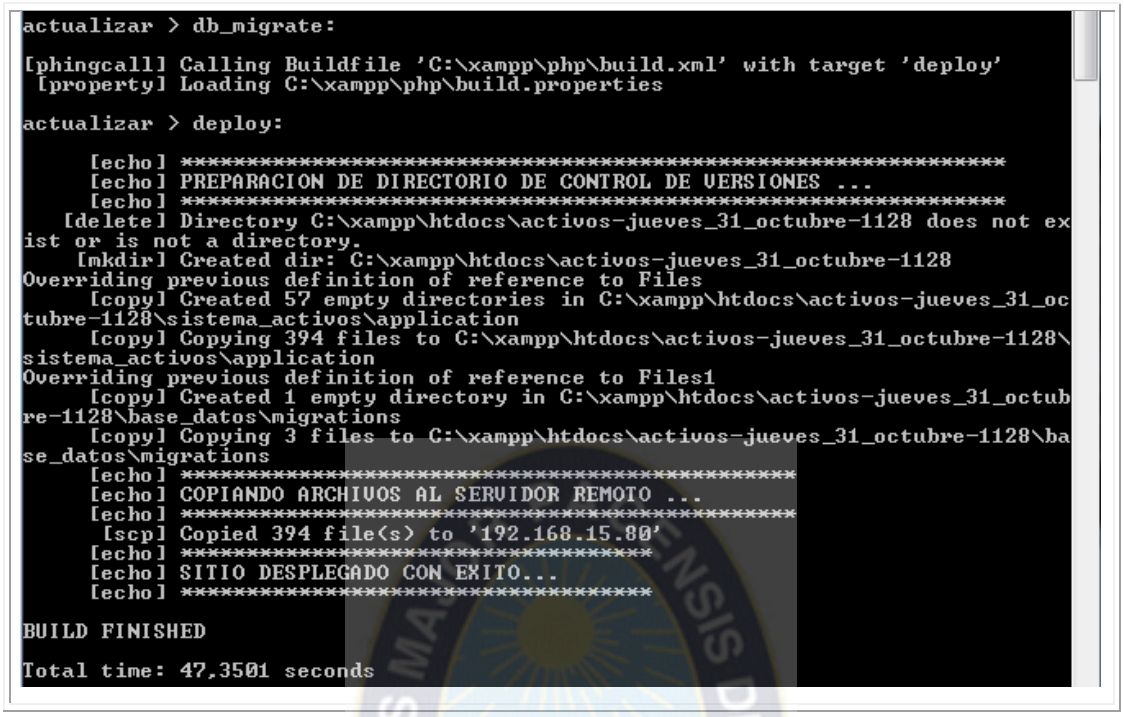


Figura 3.34: Resultado de la ejecución de Phing en la línea de comandos - Despliegue de la base de datos y copia de archivos al servidor remoto

3.3.5. ETAPA 4: FASE DE MEDICIÓN DE CALIDAD BASADA EN EL ESTANDAR ISO 9126.

3.3.5.1. Funcionalidad

El grado en que un sistema satisface las necesidades. La funcionalidad es medida a través del Punto Función (PF) misma que ofrece una medida cuantitativa y auditable del tamaño de la aplicación, basada en la visión del usuario final.

Para medir la funcionalidad del sistema se deben determinar las siguientes 5 características:

	Factor de ponderación					
Parámetro de medición	**Cuenta**		**Simple**	**Medio**	**Complejo**	**Total**
Número de entradas de usuario	54	*	3	4	6	216
Número de salidas de usuario	19	*	4	5	7	95
Número de peticiones de usuario	14	*	3	4	6	56
Número de archivos	16	*	7	10	15	160
Número de interfaces externas	4	*	5	7	10	28
Cuenta					**total**	**555**

Tabla 3.5: Hoja de trabajo para el cálculo del punto función
Fuente: Elaborado de acuerdo [PRES, 2002]

Para el cálculo del punto función se utiliza la relación:

$$PF = \text{Cuenta total} * (0{,}65 + 0{,}01 * \sum F_i)$$

Donde:

PF: Medida de funcionalidad

Cuenta total: es la suma de: número de entradas, número de salidas, número de peticiones, número de archivos y número de interfaces externas.

$\sum F_i$: Son los valores de ajuste de complejidad, donde ($1 \leq i \leq 14$).

Ahora obtenemos los "valores de ajuste de complejidad", según a las respuestas a las preguntas:

Factor	No importante 0	Incidental 1	Moderado 2	Medio 3	Significativo 4	esencial 5	F_i
1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?						X	5
2. ¿Se requieren comunicaciones de datos?					X		4
3. ¿Existen funciones de procedimiento distribuido?	X						0
4. ¿es crítico el rendimiento?			X				2
5. ¿será ejecutado el sistema en un entorno operativo existente y fuertemente utilizado?					X		4
6. ¿requiere el sistema entrada de datos interactiva?					X		4
7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples?			X				2
8. ¿se actualizan los archivos maestros de forma interactiva?					X		4
9. ¿son complejas las entradas, las salidas, los archivos o las peticiones?			X				2
10. ¿es complejo el procesamiento interno?			X				2
11. ¿se ha diseñado el código para ser reutilizable?					X		4
12. ¿están incluidas en el diseño la conversión y la instalación?					X		4

13. ¿se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?						X	5
14. ¿se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario?						X	5
						Factor de complejidad:	**47**

Tabla 3.6: Tabla de valores de ajuste de complejidad
Fuente: Elaborado de acuerdo [PRES, 2002]

Reemplazando en la relación se tiene:

PF = Cuenta total * (0,65 + 0,01*$\sum F_i$) = 555 * (0,65 + 0,01*47) = 621,6 valor en el punto función

Si consideramos el máximo valor de ajuste de complejidad como $\sum F_i = 70$, se tiene:

PF = 555 * (0,65 + 0,01*70) = 749,25

Entonces si $\sum F_i$ es considerada como el 100%, la relación obtenida entre los puntos será:

PF = (621,6 *100) **/** 749,25 = 82.96

Por lo tanto, la funcionalidad que tiene el sistema es de 82.96% tomando en cuenta el punto de función máximo.

3.3.5.2. Portabilidad

De acuerdo a los factores de calidad “la portabilidad es el esfuerzo necesario para transferir una aplicación de un entorno Sistema Hardware y/o Software a otro”.

El presente proyecto por estar diseñado para un entorno de acceso a red, mide la portabilidad en el lado del servidor y en el lado del cliente:

- Portabilidad en el lado del servidor

  Software: se necesita como sistema operativo Windows 7 de 32 bits, gestor de Base de Datos se necesita Postgresql, como servidor se utiliza apache y además deberá estar instalado correctamente el intérprete de páginas PHP5.

  Hardware: Se necesita un ordenador con procesador Core i3, con procesador de 2.13 Ghz
- Portabilidad en el lado del cliente

  El ordenador del cliente debe tener un navegador de Internet (Mozilla FireFox 16.0.1, excepto Internet Explorer debido a que presenta dificultades con las hojas de estilo CSS), con soporte para JavaScript, Acrobat Reader, conexión a Internet, y un hardware no

necesariamente potente, como ejemplo procesador Core i3, microprocesador de 2.13 Ghz, Windows 7 de 32 bits, una impresora.

Tomando en cuenta los aspectos anteriores el factor de calidad será calculado con la métrica de facilidad de instalación que calcula el porcentaje de quien mantiene el software para instalarla en el ambiente correspondiente, la misma está dada por la siguiente fórmula:

$$X = A/B$$

Donde:

A = 7 casos en que el usuario tiene éxito en la operación de instalación.

B = 7 casos

Luego se tiene:

$X = 7/7 \Rightarrow X = 1$

Por lo tanto existe un 100% de probabilidad de que el usuario instale correctamente el sistema, esto se debe fundamentalmente a que se está utilizando el framework Codeigniter que permite facilitar la instalación y configuración de manera muy sencilla con solo copiar la carpeta con el nombre del proyecto en el servidor remoto y la ejecución de la herramienta Phing (descrita anteriormente).

3.3.5.3. Usabilidad

La medición de la facilidad de uso se entiende como la facilidad que el usuario tiene para entender y conocer al sistema, tanto como para comprenderlo, aprenderlo y operarlo. A continuación en la siguiente tabla se observan los resultados obtenidos en la enseñanza de la manipulación del software a los tres tipos de usuarios autorizados para utilizar el sistema.

USUARIOS	FACILIDAD DE COMPRENSIÓN	FACILIDAD DE APRENDIZAJE	FACILIDAD DE OPERACIÓN
Usuario 1	90 %	91 %	95 %
Usuario 2	87 %	91 %	90 %
Usuario 3	95 %	90 %	93 %
Promedio	**90.7 %**	**90.7 %**	**92.7 %**

Tabla 3.7: Resultado para el cálculo de facilidad de uso
Fuente: Elaboración propia

Por lo tanto, según los resultados obtenidos mediante la tabla se obtuvo que la facilidad de uso del sistema es de un 91.4 %

3.3.6. ETAPA 5: ACEPTACION Y ENTREGA DEL PRODUCTO

En esta etapa se realizo la revisión final del producto, la entrega del sistema en el cual contempla el código fuente, script de la Base de Datos y el manual de usuario.

Finalmente, el sistema se encuentra instalado en el ambiente de producción perteneciente al Área de Sistemas del SENAPE.

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 INTRODUCCION

El capítulo describe las conclusiones en base a los resultados obtenidos en la elaboración del proyecto, de acuerdo a los objetivos planteados en el capítulo 1 y finalmente plantea recomendaciones en base a observaciones para incrementar la funcionalidad del sistema a futuro.

4.2 CONCLUSIONES

Con la implementación del "Sistema Web de Control y Administración de Activos Fijos del SENAPE", en la oficina central del Servicio Nacional de Patrimonio del Estado – SENAPE, se logró cumplir el objetivo general que es "Desarrollar un sistema informático web que permita optimizar los procesos de administración, asignación, seguimiento y control de los Activos Fijos, del Área de Activos Fijos del SENAPE".

La implementación del sistema web permitió de manera general un control adecuado de los Activos Fijos tanto adquiridos por el SENAPE como los que han sido donados a la Institución por entidades en proceso de liquidación (por ejemplo: el ex Banco del Estado, el ex Banco Minero, el ex Fondo Complementario de Seguridad Social del Magisterio Fisca, etc.), facilitando su ubicación exacta con respecto a la oficina y con qué responsable se encuentra cada activo. También se logró la automatización de procesos manuales tales como ser: registro, asignación, devolución, cálculo de las depreciaciones, generación de reportes en formato pdf para todos los procedimientos, asignación de códigos a los activos dados de alta y su respectiva generación de código de barras.

El sistema con el empleo del framework Codeigniter, para la elaboración del código fuente y para el diseño de la interfaz de usuario, presenta una aplicación amigable, comprensible y fácil de usar para los usuarios autorizados, además este framework debido a que utiliza la arquitectura MVC (modelo vista controlador) facilita la Mantenibilidad del sistema y provee seguridad web.

El sistema a su vez contribuye con el complimiento del Decreto Supremo Nº 081 de fecha 28 de junio de 2009, Normas Básicas del Sistema de Administración de Bienes y Servicios, NB-SABS, de la Ley 1178 de fecha 20 de julio de 1990 de la Administración y Control

Gubernamental y Normativas referentes al manejo de Bienes de las entidades públicas, logrando una administración eficaz y eficiente de los bienes antes descritos.

4.3 RECOMENDACIONES

Con el presente proyecto los requisitos de la Entidad con respecto al área de Activos Fijos fueron satisfechos; sin embargo, con la finalidad de lograr mejoras y proporcionar mayores funcionalidades a los usuarios del sistema, se plantean las siguientes recomendaciones:

- Implementar un modulo que permita almacenar fotos de los diferentes activos fijos.
- Elaborar instrucciones de cuidado de activos fijos dirigido a los servidores públicos del SENAPE considerando las normas vigentes.
- Implementar un módulo de depreciación para equipos de computación por partes como ser: tarjeta madre, disco duro, tarjeta de video, etc.
- Para realizar los procedimientos de asignación de activos, salidas temporales de activos, etc, el sistema maneja información del personal de la Institución de la base de datos del Sistema Centralizado de Administración – SICENAD del SENAPE; por tal motivo, es necesario mantener siempre actualizada la información de esta base de datos (información del personal como ser cargo, dirección en la que trabajan, si concluyeron su relación laboral con el SENAPE, etc.).

En cuanto al sistema se recomienda realizar la capacitación al usuario encargado del manejo de activos fijos, para el buen manejo de la información y del propio sistema.

REFERENCIA BIBLIOGRÁFICA

[CANCHILLO, 2006] Mario Canchillo Zanga: “Control y Seguimiento de Activos Fijos del Servicio Exterior vía Web – Ministerio de Relaciones Exteriores y Culto”.

[MAMANI, 2007] Epifanio Mamani Calle: “Sistema de control de Activos Fijos para el Gobierno Municipal de El Alto.”

[MAMANI C, 2007] Ramiro Máximo Mamani Cahuna: **“**Sistema de control y seguimiento de Activos Fijos – Ministerio de Gobierno.**”**

[CONDARCO, 2008] Condarco Alejo: **“**Sistema de Información de Seguimiento y Gestión de Activos Fijos.**”**

[QUISPE, 2009] Edwin Quispe Quispe: “Sistema de Información y Control de Activos Fijos para la Fundación La Paz (Área promoción de la mujer).”

[AMARO, VALVERDE, 2007] Amaro Calderón Sarah Dámaris, Valverde Rebaza. Jorge Carlos: “Metodologías Ágiles.”

[INTECO, 2009] Laboratorio Nacional de Calidad del Software: “Ingeniería de Software: metodologías y ciclos de vida.”

[YAZYI, 2011] Sergio Adrián Yazyi: “Una experiencia práctica de Scrum a través del aprendizaje basado en proyectos mediado por TIC en un equipo distribuido”

[GARABITO, 2007] Julio Garavito, Escuela Colombiana de Ingeniería: “Manual básico de Postgresql”.

[LOCKHART, 1996-9] Thomas Lockhart, El equipo de desarrollo de PostgreSQL: “Tutorial de Postgresql”.

[DORANTES, 2001] Víctor Hugo Dorantes Gonzales, Fernando Magariños Lamas, José Neif Jury Fabre: “Curso de Bases de Datos y Postgresql”.

[PALACIO,2008] Juan Palacio, “ScrumManager: Gestión de proyecto”

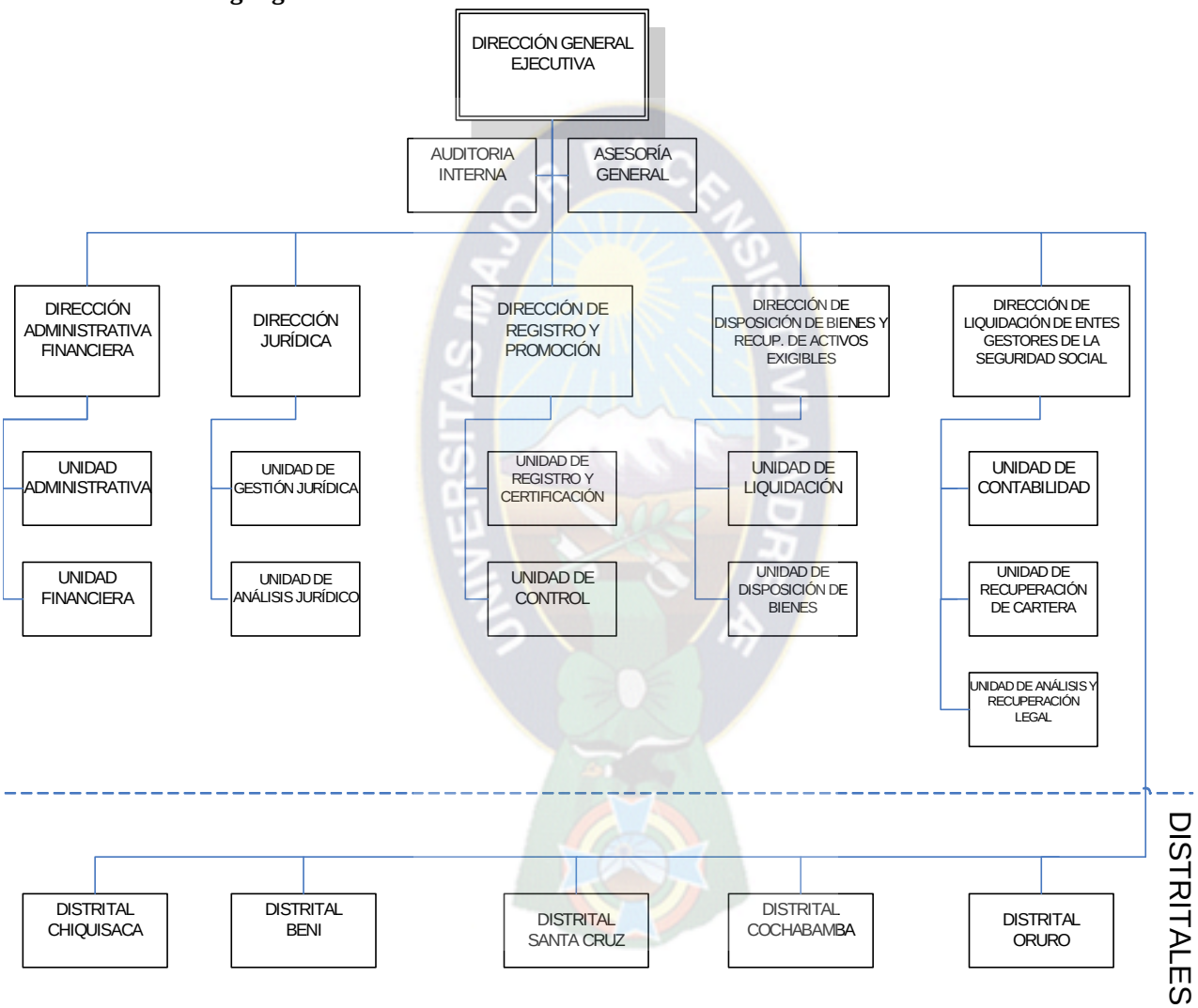
[DE LA CRUZ, 2006] Joel de la Cruz Villar, grupo editorial Megabyte: “Php 5 & Mysql”.

[GISBERT, PÉREZ , 2009] Marc Gibert Ginestá, Oscar Pérez Mora: “Base de Datos en PostgreSQL”.

[ADERHOLD, BLACK, 2011] Andreas Aderhold, Alex Black, Manuel Holtgrewe, Hans Lellelid, Michiel Rook, Johan Persson: “Phing 2.4 - User Guide”

ANEXO A:

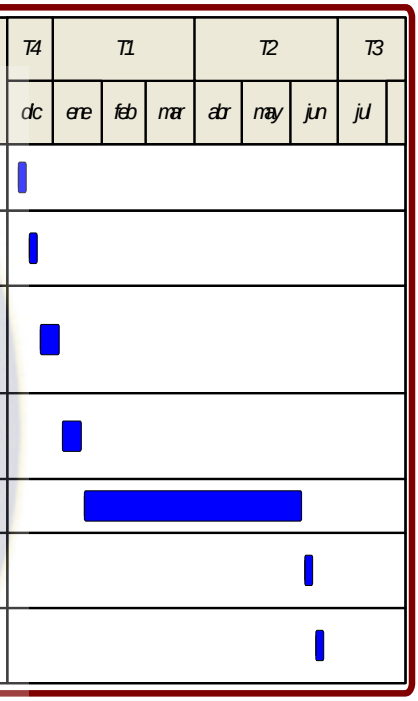
Organigrama del Servicio Nacional de Patrimonio del Estado - SENAPE



ANEXO B:

CRONOGRAMA DE TRABAJO

Id	Nombre de tarea	Comienzo	Fin	Duración
1	Elección y aceptación de la Institución	12/10/2012	12/14/2012	5d
2	Obtener información sobre las actividades (entrevistas y observación)	12/17/2012	12/21/2012	5d
3	Etapa 1: toma de requerimientos y definición del Product Backlog	12/24/2012	1/4/2013	10d
4	Etapa 2: Análisis de requerimientos y diseño arquitectónico	1/7/2013	1/18/2013	10d
5	Etapa 3: Desarrollo del sistema	1/21/2013	6/7/2013	100d
6	Etapa 4: Fase de cierre de Scrum y Medición de calidad	6/10/2013	6/14/2013	5d
7	Etapa 5: Aceptación y entrega del producto	6/17/2013	6/21/2013	5d



ANEXO C: MÉTODOS PARA CONTABILIZAR LA DEPRECIACIÓN

Existen varios métodos para determinar el gasto que sufren los Activos Fijos por depreciación; sin embargo, los cuatro procedimientos más utilizados para contabilizar la depreciación son:

METODO	CARGO DE DEPRECIACIÓN
Línea recta	Igual todos los años de vida útil
Suma de los dígitos de los años	Mayor los primeros años
Doble saldo declinante	Mayor los primeros años
Sistema Unidades de producción	De acuerdo a la producción

La depreciación de un año varía de acuerdo a método de depreciación que se utilice. Algunos métodos dan como resultado un gasto mayor en los primeros años de vida del activo, lo cual afecta en las utilidades netas del período.

1. **Sistema de Línea Recta:** Es el más utilizado y se asume que el desgaste es lineal a lo largo de la vida útil del bien.

$$\text{Depreciación} = \frac{\text{valor Libro – valor Residual}}{\text{Total de vida Útil}}$$

Señalemos que el **Valor Libro = Costo Histórico – Depreciación Acumulada**

2. **Suma de los dígitos de los años:** Es un método que conlleva mayores cargos de depreciación en los primeros años.

$$\text{Depreciación} = (\text{Costo Histórico} - \text{Valor Residdual}) * \frac{2n(n-i)}{n*(n+1)}$$

En que:

n = Vida útil total

i = año.

3. **Doble Saldo Declinante:** Se aplica el doble del porcentaje anual de la depreciación lineal sobre el valor libro del año correspondiente.

$$\text{Depreciación} = \frac{100}{2} * n * (Valor\ Libro)$$

Nota: El año n se debe determinar por la diferencia que falta para dejar el valor residual.

4. **Sistema de Unidades de Producción:** Se posee una alta correlación entre las unidades de producción, ejemplo, un camión puede utilizar una depreciación por los kilómetros recorridos.

$$\text{Depreciación} = \frac{(Costo\ Historico - Valor\ Residual)}{Unidades\ de\ producción} * Cantidad\ de\ Producción$$

ANEXO D: ANEXO DEL ARTÍCULO 22 DEL DS. 24051, DEPRECIACIONES DEL ACTIVO FIJO

Conforme a la disposición contenida en el primer párrafo del Artículo 22° de este Decreto Supremo, las depreciaciones del activo fijo se computarán sobre el costo depreciable, según el Artículo 21° de este reglamento, y de acuerdo a su vida útil en los siguientes porcentajes:

BIENES	AÑOS DE VIDA UTIL	COEFICIENTE
Edificaciones	40 años	2,5%
Muebles y enseres de oficina	10 años	10,0%
Maquinaria en general	8 años	12,5%
Equipos e instalaciones	8 años	12,5%
Barcos y lanchas en general	10 años	10,0%
Vehículos automotores	5 años	20,0%
Aviones	5 años	20,0%
Maquinaria para la construcción	5 años	20,0%
Maquinaria agrícola	4 años	25,0%
Animales de trabajo	4 años	25,0%
Herramientas en general	4 años	25,0%
Reproductores y hembras de pedigree o puros por cruza	8 años	12,5%
Equipos de computación	4 años	25,0%
Canales de regadío y pozos	20 años	5,0%
Estanques, bañaderos y abrevaderos	10 años	10,0%
Alambrados, tranqueras y vallas	10 años	10,0%
Viviendas para el personal	20 años	5,0%
Muebles y enseres en las viviendas para el personal	10 años	10,0%
Silos, almacenes y galpones	20 años	5,0%
Tinglados y cobertizos de madera	5 años	20,0%
Tinglados y cobertizos de metal	10 años	10,0%
Instalaciones de electrificación y telefonía rurales	10 años	10,0%
Caminos interiores	10 anos	10,0%
Caña de azúcar	5 años	20,0%
Vides	8 años	12,5%
Frutales	10 años	10,0%
Otras plantaciones	Según experiencia del contribuyente	
Pozos Petroleros (ver inciso II del Art. 18° de este reglamento)	5 años	20,0%
Líneas de Recolección de la industria petrolera	5 años	20,0%
Equipos de campo cié la industria petrolera	8 años	12,5%
Plantas de Procesamiento de la industria petrolera	8 años	12,5%
Ductos de la industria petrolera	10 años	10,0%