|  |
| --- |
| SOFTWARE PROJECT MANAGEMENT PLAN |
| SPMP para SnoutPoint Networks |
| SnoutPoint Networks  Esteban Hernández Losada  Camilo Oviedo Lizarazo  Camilo Benavides Franco  Sebastián Jiménez Nieto  Fabiana Díaz Cedeño  David Suárez Guerrero  Alejandra Rocha Sabogal |

# Historial de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Versión | Cambios efectuados | Fecha de Actualización | Área(s)  Encargada(s) |
| **v0.1.0** | * Agregado 6.1. Vista General de Proyecto * Agregado 6.2. Propósito, Alcance y Objetivo * Agregado 8.4.2 Organización del proyecto y Comunicación * Agregado 11.1 Ambiente de Trabajo. * Agregado 11.2 Administración de configuraciones. | 23 de Febrero de 2015 | Gestión de Documentación, Calidad, Riesgos y Configuraciones |
| **v0.1.1** | * Corrección de numeración de elementos del SPMP.   + Ambiente de trabajo pasa de 11.1 a 12.1   + Administración de configuración pasa de 11.2 a 12.2 | 1° de Marzo de 2015 | Gestión de Documentación, Calidad, Riesgos y Configuraciones |
| **v0.2** | * Mayor profundidad en la definición de visión * Corrección de Objetivos usando S.M.A.R.T * Corrección de Supuestos y restricciones * Modificación de Entregables * Gestión de bibliografía del contenido actual | 6° de Marzo de 2015 | Gestión de Documentación, Calidad, Riesgos y Configuraciones |
| **v0.3** | * Agregado 7. Glosario * Agregado 8.4 Interfaces externas y añadido más contenido y fuentes bibliográficas en 8.2 Organigrama y Descripción de roles | 7 de Marzo de 2015 | Gestión de Documentación, Calidad, Riesgos y Configuraciones |
|  |  |  |  |

# Prefacio

Propósito: Que el lector determine si vale la pena seguir leyendo el documento

Contenido: Descripción breve del contenido, propósito y alcance del proyecto. Descripción de la audiencia esperada del documento (para quién es importante leer este documento)

Referencias: [1], sección 5

Tamaño sugerido 1/2 hoja.

# Tabla de Contenidos

[1 Historial de Cambios 1](#_Toc413329055)

[2 Prefacio 2](#_Toc413329056)

[3 Tabla de Contenidos 3](#_Toc413329057)

[4 Lista de Figuras 5](#_Toc413329058)

[5 Lista de Tablas 6](#_Toc413329059)

[6 Vista General del Proyecto 7](#_Toc413329060)

[6.1 Visión del Producto 7](#_Toc413329061)

[6.2 Propósito, Alcance y Objetivos 8](#_Toc413329062)

[6.2.1 Propósito 8](#_Toc413329063)

[6.2.2 Alcance 8](#_Toc413329064)

[6.2.3 Objetivos 10](#_Toc413329065)

[6.3 Supuestos y Restricciones 11](#_Toc413329066)

[6.4 Entregables 13](#_Toc413329067)

[6.5 Resumen de Calendarización y Presupuesto 14](#_Toc413329068)

[6.6 Evolución del Plan 14](#_Toc413329069)

[7 Glosario 15](#_Toc413329070)

[8 Contexto del proyecto 16](#_Toc413329071)

[8.1 Modelo de Ciclo de Vida 16](#_Toc413329072)

[8.1.1 Análisis de Alternativas y Justificación 22](#_Toc413329073)

[8.2 Lenguajes y Herramientas 25](#_Toc413329074)

[Gestión del proyecto 25](#_Toc413329075)

[Equipo de desarrollo 25](#_Toc413329076)

[8.3.1 Análisis de alternativas y Justificación 29](#_Toc413329077)

[8.3 Plan de Aceptación del Producto 31](#_Toc413329078)

[8.4 Organización del Proyecto y Comunicación 31](#_Toc413329079)

[8.4.1 Interfaces Externas 31](#_Toc413329080)

[8.4.2 Organigrama y Descripción de Roles 32](#_Toc413329081)

[9. Administración del Proyecto 35](#_Toc413329082)

[9.2 Métodos y Herramientas de Estimación 35](#_Toc413329084)

[9.3 Inicio del proyecto 35](#_Toc413329085)

[9.3.1 Entrenamiento del Personal 35](#_Toc413329086)

[9.3.2 Infraestructura 35](#_Toc413329087)

[9.4 Planes de Trabajo del Proyecto 36](#_Toc413329088)

[9.4.1 Descomposición de Actividades 36](#_Toc413329089)

[9.4.2 Calendarización 36](#_Toc413329090)

[9.4.3 Asignación de Recursos 37](#_Toc413329091)

[9.4.4 Asignación de Presupuesto y Justificación 37](#_Toc413329092)

[10 . Monitoreo y Control del Proyecto 38](#_Toc413329093)

[10.2 Administración de Requerimientos 38](#_Toc413329094)

[10.3 Monitoreo y Control de Progreso 38](#_Toc413329095)

[10.4 Cierre del Proyecto 39](#_Toc413329096)

[11 Entrega del Producto 40](#_Toc413329097)

[12 Procesos de Soporte 41](#_Toc413329098)

[12.2 Ambiente de Trabajo 41](#_Toc413329099)

[12.3 Análisis y Administración de Riesgos 43](#_Toc413329100)

[12.4 Administración de Configuración y Documentación 43](#_Toc413329101)

[Objetivo del Plan 43](#_Toc413329102)

[12.5 Métricas y Proceso de Medición 45](#_Toc413329103)

[12.6 Control de Calidad 45](#_Toc413329104)

[13 Anexos 47](#_Toc413329105)

[14 Referencias 48](#_Toc413329106)

Propósito: Que el lector encuentre rápidamente una sección específica del documento

# Lista de Figuras

[Gráfico 1. Esquema de Ciclo de Vida en Espiral, Tomado de [6] 16](#_Toc413513629)

[Gráfico 2. Esquema de desarrollo en incrementos, tomado de [7] 18](#_Toc413513630)

[Gráfico 3. Esquema General de RUP tomado de [12] 20](#_Toc413513631)

# Lista de Tablas

Propósito: Que el lector encuentre rápidamente una tabla específica del documento

# Vista General del Proyecto

## Visión del Producto

Se define la visión del producto utilizando la plantilla de aspectos de la plantilla de visión de RUP [[1](#RUP15)], definiendo en alto nivel las necesidades y características de SnoutPoint.

Las mascotas han servido como un puente o una buena excusa para conocer a otras personas, así como generadores de oportunidades de negocio al ofrecer servicios como veterinarias, peluquerías, paseo de mascotas, guarderías, tiendas de mascotas, entre muchos otros servicios. Aunque hay plataformas que funcionan como comunidades de amantes de las mascotas [[2](#Abo4)] , pocas aprovechan el aspecto de utilizar estas redes para aspectos de servicios. Esta situación genera como posible solución el desarrollar una red social que contemple tanto a los dueños y fanáticos de las mascotas, así como a las personas que ofrezcan servicios que involucren a estas.

Por lo tanto, SnoutPoint se proyecta como una red social para personas que pertenecen al mundo de las mascotas domésticas, específicamente gatos y perros de cualquier raza. Se tienen en cuenta como usuarios potenciales dueños de mascotas, personas que les gustan los animales y personas que tengan negocios o trabajos relacionados con mascotas.

A diferencia de otras redes sociales y aplicaciones de mascotas, SnoutPoint buscará profundizar en los servicios que más requieren los usuarios. SnoutPoint va a contribuir a la población generando interrelaciones entre las personas a través de sus mascotas, brindar ayudas e información útil para nuestros usuarios y generar oportunidades de negocio en el ámbito de las mascotas. Al terminar este proyecto se busca que tenga una utilidad, cumpliendo con las expectativas de los clientes y potenciales usuarios.

## Propósito, Alcance y Objetivos

### Propósito

**A nivel de Producto**

El propósito de SnoutPoint es crear una red social innovadora para los dueños y prestadores de servicios de mascotas e incluso para aquellos que les gustan las mascotas pero no poseen alguna.

Para la parte de los dueños y para los fanáticos de mascotas, la idea es que puedan interactuar con otros usuarios y también con los prestadores de servicios logrando buscar información necesaria para sus mascotas. A nivel de servicios y negocios, los dueños de estos mostraran que es lo que ofrecen, ya sean veterinarias, adopciones, fundaciones o tiendas.

**A nivel de Proyecto**

El propósito del proyecto SnoutPoint Social es el de gestionar todas las actividades y prácticas como un proceso formal de ingeniería de software, definiendo y siguiendo una serie de planes para asegurar la generación de un producto de software de calidad y que cumpla con las necesidades más apremiantes de los usuarios.

### Alcance

SnoutPoint Networks implementará la red social de SnoutPoint. Entendiendo plenamente a SnoutPoint como una red social, el sistema debe tener un módulo básico para la creación y autenticación de usuarios, donde se guarden registros de la información de perfil de los usuarios, así como sus mascotas o servicios/negocios si los llegaran a tener. Por su aspecto de social, también debe permitir que los usuarios entablen amistad con otros, por lo que el agregar amigos también es una operación fundamental a tener en cuenta.

Añadido al módulo básico de la red social, se realiza una encuesta con el propósito de evaluar la factibilidad y viabilidad del proyecto, teniendo en cuenta una serie de características que puedan definir la estructura del sistema, los servicios que contendrá y los que fueron descartados.

Primero se analiza cuantas personas estarían interesadas en una red social de mascotas (ver Anexos – Encuesta de SnoutPoint). De un grupo de 137 encuestados, el 74% estarían de acuerdo con participar en una red social de mascotas, lo cual permite afirmar que el proyecto de SnoutPoint llama la atención a los usuarios y por lo tanto no hay inconveniente en cuanto a aceptación del servicio.

Luego de esto, se define la forma como los usuarios van identificar dentro del sistema, así como sus mascotas. Un 57% de los encuestados dice que prefiere una cuenta de ellos que tenga asociados a todas sus mascotas frente al 37% de las personas que quisieran tener una cuenta por cada mascota. Esto facilita entender la distribución de información de las cuentas de los usuarios.

Posteriormente, se buscan los servicios que los usuarios demandarían más, para este caso, se utilizó una calificación de 1 a 5 de cada servicio (1 siendo menos importante y 5 siendo el muy importante) y cuentan los votos de las calificaciones entre 4 y 5 para cada servicio, esto con el objetivo de descartar votos de personas que no mostraran interés en el servicio. Una vez definido esto, se priorizan los servicios con mayor cantidad de votos, por lo que se busca desarrollar los requerimientos asociados con cada servicio primero y dejar las que no son tan importantes fuera del alcance del proyecto. Cabe resaltar que los usuarios están de acuerdo en el poder calificar la calidad de los servicios, para lo cual se debe hacer un código para calificación que otorgue el prestador del servicio y así evitar que las personas que no han utilizado el servicio lo califiquen erróneamente.

A continuación se muestran los servicios de mayor a menor prioridad:

1. **Veterinarias:** Este servicio busca localizar las veterinarias según las preferencias de los usuarios. Se muestran cuales servicios tienen, su ubicación y el horario de atención.
2. **Anuncios de Mascotas Perdidas y Encontradas:** Como un tablón de noticias y anuncios, se pueden anunciar la pérdida de una mascota de un usuario o en caso de encontrar otra. Este servicio no requiere de la creación de un usuario. Se tiene estimado que se puedan usar filtros para acelerar la búsqueda de mascotas.
3. **Adopción de Mascotas:** Como otro tablón de anuncios, se muestra un catálogo de mascotas en adopción. Se tiene estimado usar filtros para buscar mascotas. Este servicio si requiere de un registro previo.
4. **Recordatorios:** Dentro de la persona, se le pueden hacer recordatorios sobre eventos importantes relacionados con sus mascotas (cumpleaños, vacunas, revisiones médicas periódicas, etc.).
5. **Fundaciones:** La mayoría de fundaciones están sumamente comprometidas con temas de mascotas como la recuperación de mascotas, adopciones y eventos. Su servicio consistiría en mostrar fechas de eventos y mascotas que puedan dar a adopción. Está relacionado con el numeral 3.
6. **Tiendas:** Locales donde se venden mascotas y productos para estas (juguetes, comidas, accesorios, etc.).

Debido a que las otras opciones no son lo suficientemente solicitadas, se dejan fuera del alcance del proyecto de SnoutPoint; son servicios que se implementarán posteriormente si se han cumplido completamente todos los requerimientos relacionados con los anteriores seis servicios. Estas incluyen: foros de preguntas, guarderías, escuelas de entrenamiento, peluquerías y paseadores.

También se pregunta a los usuarios si conocen otras redes sociales y medios por los cuales se hace difusión de los servicios similares a los que se planean ofrecer por medio de SnoutPoint, los cuales serían nuestra principal competencia, como:

* FacePets
* Grupos de Facebook
* Cuentas de Instagram
* UniversoMascotas
* ElMundoDelPerro
* Programas de Televisión.

### Objetivos

Se tienen en cuenta tres tipos de objetivos [[3](#Rob05)] dentro del desarrollo del proyecto: individuales que van orientados a lo que el desarrollo del producto y la ejecución del proyecto debería dejar como aprendizaje y formación en cada uno de los miembros del grupo; grupales, que van orientados a como el trabajo en equipo aportará experiencias durante y después del desarrollo del proyecto; y a nivel de proyecto, que consiste en que es lo que se busca lograr al hacer desarrollando el producto, que en este caso será la red social para mascotas, SnoutPoint.

#### 6.2.3.1 Objetivos Individuales

* Desarrollar habilidades para la planeación y gestión adecuada de un proyecto de software, teniendo en cuenta todos los aspectos relacionados según el rol de cada miembro del grupo.
* Desarrollar las diversas competencias necesarias para el desarrollo del proyecto de software por medio de la investigación y la experiencia.
* Obtener un nivel aceptable de conocimientos para el desarrollo de un proyecto de software, el cual se vea reflejado en el rendimiento dentro de la clase de Ingeniería de Software.
* Fomentar el nivel de investigación para las diferentes tareas y actividades para el desarrollo del producto de software.

#### Objetivos Grupales

* Crear un ambiente de trabajo que permita el desarrollo de las diferentes actividades y tareas asignadas, basados en principios básicos como lo son el respeto y la responsabilidad.
* Desarrollar y mejorar habilidades de trabajo en equipo dentro de este proyecto y para otros en el futuro.
* Lograr el desarrollo de un producto de software como un equipo de desarrollo, buscando cumplir con todas las expectativas planeadas.

#### Objetivos de Proyecto

* Obtener el conocimiento y la experiencia del proceso formal de ingeniería de software para la creación de un producto de software.
* Comprender la importancia de la planificación, estimación, evaluación y análisis de los diferentes aspectos relacionados con el proyecto para alcanzar un producto de calidad.
* Desarrollar un producto de software que logre cumplir con el alcance del proyecto, basado en los diferentes planes, métricas y estimaciones asociadas.

## Supuestos y Restricciones

Los supuestos que se manejarán para SnoutPoint son todos aquellos detalles y hechos que se dan por sentado, hechos que deben darse para que el desarrollo del proyecto no se vea afectado. Estos son los supuestos que se deben tener en cuenta para lograr este propósito:

* Anterior a las fechas de sustentación, se definirá la sala o salón en el cual se desarrollará la muestra o entrega de prototipo.
* Los prototipos que sean presentados al cliente deberán estar en capacidad de ejecutarse en los ordenadores de la sala o salón previamente concertado.
* Los usuarios no poseen ningún tipo de discapacidad visual o cognitiva que les dificulte utilizar el sistema.
* Todos los usuarios deben estar relacionados con mascotas, sean usuarios, prestadores de servicios o simplemente que tengan gusto por las mascotas.
* Los usuarios deben tener un mínimo de conocimiento de navegación web, es decir, que puedan entender cómo realizar operaciones básicas tanto en SnoutPoint como en otros portales web.
* Los requerimientos del cliente no serán modificados durante el proceso de desarrollo del Software, una vez identificados y definidos dentro de las actividades.
* El esfuerzo de todos los miembros de SnoutPoint Networks será el máximo que puedan dar.

Las restricciones son limitantes en alguno de los aspectos relevantes para el desarrollo del proyecto, además de limitantes para los usuarios potenciales:

* Los clientes Jaime Pavlich y Miguel Torres entregarán un cronograma donde se definen los principales hitos para la entrega de versiones de lanzamiento. [[4](#Pav)]
* El grupo consta de 7 personas que deberán cumplir todas sus labores, sin ningún tipo de tercerización de estas.
* Las personas que deseen tener un usuario deben ser mayores de 13 años de edad, teniendo en cuenta ciertos lineamientos legales de la protección de información de menores. [[5](#Huf14)]
* Si un prestador de servicios desea crear un usuario, debe estar dispuesto a recibir una calificación por sus servicios prestados.
* El sistema debe manejar un paradigma orientado a objetos, poseer una arquitectura de cliente-servidor, uso riguroso de interfaces gráficas para el usuario y manejo de persistencia.

## Entregables

Se definen entregables como aquellos artefactos que se divulgarán al cliente (como una entrega, ver Administración de Configuraciones) y al equipo de trabajo (como una versión, ver Administración de configuraciones).

* **Entregables para el cliente**
  + **Primera entrega Académica – 12 de Marzo de 2015**
    - Software Project Management Plan (SPMP línea base)
    - Documentación y Diagramas de Casos de Uso
    - Reporte Gerencial
  + **Segundo Incremento – 11 de Abril de 2015**
    - Primer prototipo funcional del Software
  + **Segunda entrega Académica (Tercer Incremento) – 23 de Abril de 2015**
    - Software Requirement Specification (SRS línea base).
    - Reporte Gerencial.
    - Segundo prototipo funcional del Software.
    - Correcciones de la Primera entrega académica (si las hay).
  + **Cuarto Incremento – 12 de Mayo de 2015**
    - Tercer prototipo funcional del Software
  + **Tercera entrega académica (Quinto Incremento) – 29 de Mayo de 2015**
    - Software Design Document (SDD línea base)
    - Cuarto prototipo funcional
    - Correcciones de la segunda entrega académica (si las hay).

Los entregables al cliente también serán parte de los entregables que tendrá todo el grupo de desarrollo, pues todos estos elementos son importantes para la elaboración y ejecución del proyecto de software. Adicional a esto, se consideran estos entregables como parte del grupo pero irrelevantes para los clientes.

* + **Grabaciones de reuniones con los profesores y clientes:** para guardar un registro de todos los detalles de las reuniones, teniendo en cuenta que no todo el equipo de desarrollo no estará presente en estas reuniones.
  + **Actas de Reunión:** para almacenar a grandes rasgos todos los temas tratados en las reuniones, además de las tareas que se dejan para realizar durante esa semana.
  + **Documentos parciales:** aquellos documentos como secciones del SPMP o casos de uso individuales que serán revisados por el equipo de desarrollo para acoplarlos en un documento único y previamente revisado y aprobado.

## Resumen de Calendarización y Presupuesto

Propósito: Que el lector entienda las principales fases e hitos del proyecto, así como los ítems más importantes de presupuesto

Contenido: Una tabla que indique las principales fases e hitos del proyecto, con sus respectivas fechas. Otra tabla que indique los ítems consolidados de presupuesto y sus valores asociados, así como el costo total del proyecto.

Tamaño sugerido: 2 páginas.

Referencias: [1], sección 5.1.1.4

## Evolución del Plan

Propósito: Que el lector entienda el proceso general para hacer cambios al plan (¡este documento!) y hacerlos efectivos dentro del proyecto.

Contenido: Descripción breve del proceso de cambios al plan (este documento), referenciando las secciones correspondientes del resto del documento que entreguen más detalles.

Tamaño sugerido: 1 página

Referencias: [1], sección 5.1.2

# Glosario

*Proyecto:* Serie de procesos y actividades para desarrollar un nuevo producto o mejorar uno ya existente. [[6](#IEE10)] En este contexto, el proyecto llevará el nombre de SnoutPoint Social y su producto correspondiente será la red social de SnoutPoint.

*Producto:* Un conjunto completo de software y documentación respectiva. [[6](#IEE10)] Para el contexto de SnoutPoint Social, el producto final será la red social SnoutPoint.

*Equipo de Desarrollo:* Para el contexto de SnoutPoint Social, son todas las personas que realizan tareas y actividades para el proyecto, poseen un rol en específico y poseen un compromiso para lograr la elaboración del producto. El equipo de desarrollo lleva el nombre SnoutPoint Networks.

*Documento:* Unidad de información identificada de manera única para uso humano, como reporte, especificación, manual o libro, ya esté en forma física o digital. [[6](#IEE10)] Esta clasificación incluye los principales entregables (ver Entregables).

*Mascota:* Para el contexto del proyecto de SnoutPoint Social, se define mascota como un perro o gato domesticado, que posee un dueño y que además tiene información de su raza, edad y género.

*Dueño:* Para el contexto del proyecto de SnoutPoint Social, se define dueño como la persona que posee mínimo una mascota, la cual tiene al menos un nombre, un apellido y una fecha de nacimiento.

*Servicio:* Para el contexto del proyecto de SnoutPoint Social, se define servicio como cualquier trabajo o labor que involucre mascotas. Dentro de esta categoría se incluyen: Veterinarias, Tiendas de Mascotas, Fundaciones de adopción de mascotas, paseadores, peluquerías para mascotas, entre otros.

*Usuario:* Para el contexto del proyecto de SnoutPoint Social, se define usuario como una persona que se ha registrado en SnoutPoint y que puede tener tanto servicios asociados como mascotas asociadas.

*Perfil:* Sección donde el usuario, su mascota o su servicio muestra su información básica.

*Cuenta:* Registro interno de los usuarios para identificarse dentro del sistema. Una cuenta consiste en una combinación de un nombre de usuario (identificador único) y una contraseña para acceder.

*Registro:* Proceso mediante el cual un cliente potencial genera una cuenta para poder acceder con esta a todos los servicios de SnoutPoint.

*Equipo de Trabajo:* Subgrupo de personas del Equipo de Desarrollo, los cuales tienen tareas y actividades más específicas para ejecutar el proyecto de software.

*Área de Desarrollo y Pruebas (ADP):* Equipo de trabajo encargado de los aspectos de programación para la producción de SnoutPoint, así como el desarrollo de las pruebas de la funcionalidad del sistema.

*Área de Documentación, Calidad, Riesgos y Configuraciones (DCRC):* Equipo de trabajo encargado del control de configuraciones y versiones, evaluación y administración de riesgos, revisión de aspectos de calidad y métricas, además de la generación de documentos del proyecto y para el equipo de desarrollo y los clientes.

*Área de Análisis y Diseño (AAD):* Equipo de trabajo encargado del análisis de requerimientos, modelado y arquitectura del sistema, diagramas de clases, modelos entidad-relación, diagramas de secuencia, entre otros artefactos relevantes en el análisis y diseño de software. Tareas propias de un arquitecto de software. [[6](#IEE10)] [[7](#Spa14)]

*Área de Gerencia:* Equipo de trabajo encargado de la gestión y administración de las tareas del grupo, planeación y control de actividades, toma de decisiones que afectan a todo el Equipo de desarrollo, además de efectuar revisiones finales de los entregables para los clientes.

*Caso de Uso:* Definición de interacciones entre un actor y el sistema para alcanzar un objetivo o fin. En el contexto de SnoutPoint, los casos de usos definirán los diferentes servicios que tendrán los usuarios (actores) con la red social.

*Ítem de Configuración:* Documentos y archivos que serán objeto de las políticas de control de configuración y versiones, generalmente, entregables y código fuente.

*Versión*

*Revisión*

*Entrega*

*Red Social:* Sistema que posee una serie usuarios que pueden establecer y mantener relaciones interpersonales entre sí, además de ofrecer una serie de servicios. [[8](#Mer)]

# Contexto del proyecto

## 8.1 Modelo de Ciclo de Vida

Teniendo en cuenta los diversos modelos de ciclo de vida existentes, se decide utilizar parcialmente aspectos de 3 que se consideran que se acoplan a las necesidades y habilidades del grupo de desarrollo del producto de software.

Se aplicarán aspectos del modelo ciclo de vida Rational Unified Process (RUP), el modelo en espiral y el modelo incremental-iterativo, los cuales tienen aspectos y prácticas que se ajustan en muy buena manera al modo de trabajo que se busca dentro del desarrollo de SnoutPoint. Se expresa que será parcialmente, pues no todas las prácticas podrán ser ejecutadas por el grupo de desarrollo.

**Modelo en Espiral**

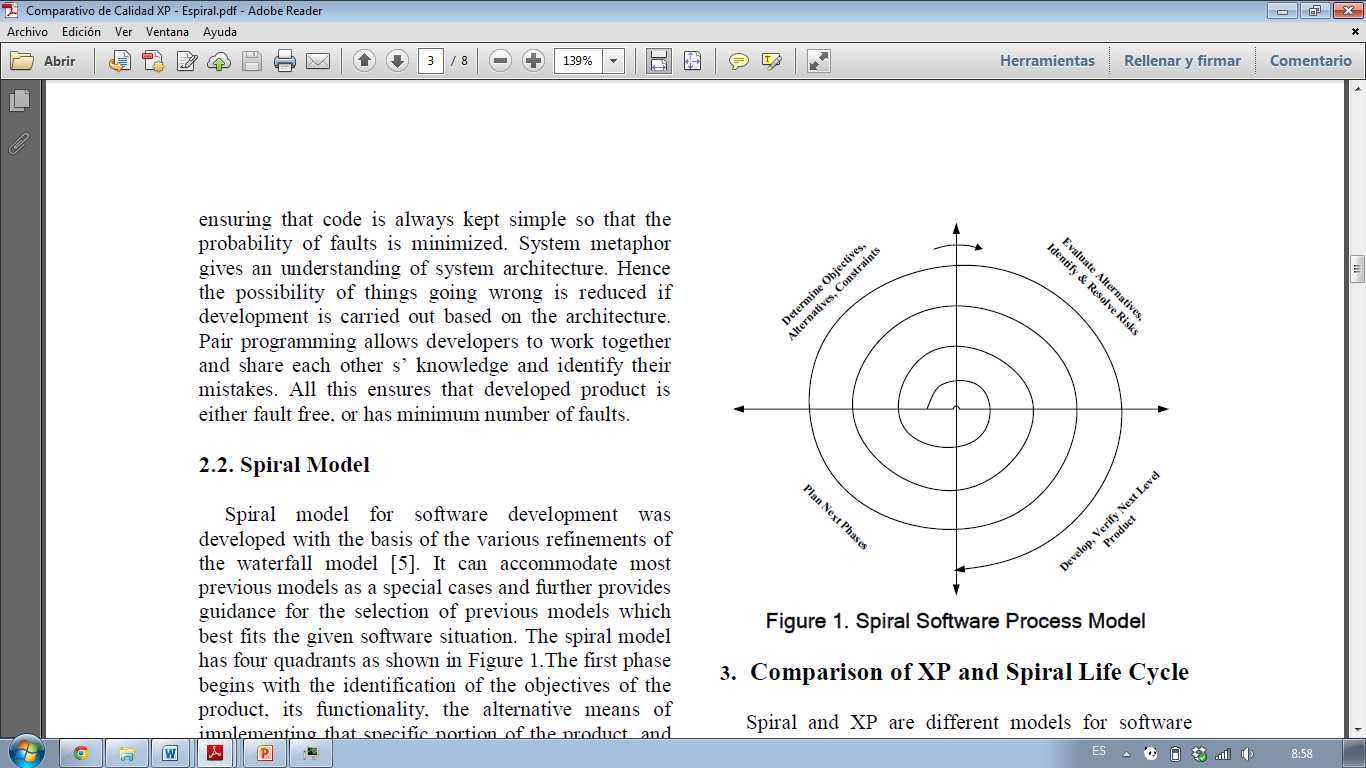
Este modelo es un modelo basado en la identificación, prevención y mitigación de riesgos. También realiza división del proyecto en pequeñas tareas o actividades repartidas en cuatro áreas o cuadrantes que se realizaran cada que se cumpla un ciclo [[9](#Has)] [[10](#Ian051)] y se ven reflejado en el Gráfico 1:

Gráfico 1. Esquema de Ciclo de Vida en Espiral, Tomado de [[9](#Has)]

1. Determinar Objetivos; identificar restricciones del proceso y del producto; identificar riesgos y planeación de alternativas.
2. Evaluar y reducir los riesgos, evaluar alternativas para mitigar los riesgos, lo cual puede incluir desarrollo de prototipos para mitigar el riesgo de requerimientos inconclusos o dudosos. [[10](#Ian051)]
3. Validación prototipos, artefactos o elementos parciales del sistema asociados a los riesgos.
4. Planeación de la siguiente fase o iteración, implicando “una revisión que involucra los desarrolladores, clientes y usuarios” [[11](#Bru02)]

La división de trabajo mediante ciclos se ajusta a los hitos principales que tiene el proyecto inicialmente, los cuales fueron dados por los clientes (ver Supuestos y Restricciones), además de realizar una evaluación de riesgos en cada una de estas etapas, lo que permite saber cuáles son los problemas potenciales que se le pueden presentar dentro de las actividades y las acciones a tomar para tratar con estos riesgos (ver Plan de Análisis y Administración de Riesgos). Esto permite que pueda haber una menor probabilidad de ocurrencias de problemas y se puede tener prevenciones o planes de contingencia para que el impacto de esos riesgos no perjudique el desarrollo del proyecto de software o puedan generar costos elevados o la finalización no deseada del proyecto [[11](#Bru02)].

Sin embargo, a medida que se avanzan las fases, la complejidad de controlar los riesgos incrementa y genera un esfuerzo mayor para las actividades de gestión, por lo que se debe tener que en las últimas iteraciones del sistema requerirán un nivel mayor de gestión y administración de riesgos.

**Modelo Incremental-Iterativo**

El modelo de desarrollo incremental-iterativo por su parte, “los clientes identifican a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuales menos.” [[10](#Ian051)] Se toman aspectos de desarrollo incremental que es el de dividir el trabajo en varias partes y programarlas en una fecha en la cual se desarrollarán y posteriormente integrar cuando se hayan completado; y elementos del desarrollo iterativo donde se examinan varios puntos importantes de lo que se ha desarrollado, realizando validaciones de lo producido y realizar una retroalimentación de lo ocurrido. [[12](#Coc08)]

Una vez que se tienen las funcionalidades generales del sistema, se definen incrementos: los cuales son un conjunto más pequeño de las funcionalidades especificadas anteriormente. Son concebidos como *mini-proyectos* donde cada resultado final será un módulo de funcionalidades parciales del sistema total, teniendo en cuenta que cada una de estas iteraciones incluye su análisis, diseño, construcción y pruebas respectivas. [[13](#Cra04)]

Durante el desarrollo del incremento, se definen en detalle los requerimientos que al inicio se identifican para la primera iteración para desarrollarlos [[10](#Ian051)]. Durante la iteración, aunque bien se pueden analizar otros requerimientos que sirvan de ayuda para los incrementos posteriores, no se pueden modificar los requerimientos del incremento actual.

Una vez completado y entregado, los clientes pueden ver resultados parciales del sistema y que satisfacen ciertos requerimientos, generalmente los más críticos. El hecho que los clientes puedan ver resultados parciales del sistema, permite obtener retroalimentaciones desde etapas tempranas, permite al equipo de desarrollo evitar las especulaciones, corregir requerimientos o diseño y también determinar el valor real de los stakeholders. [[13](#Cra04)]

Luego de generado el incremento, se revisa si ese incremento ha logrado completar todo el sistema identificado. En caso de haberlo logrado, se ejecuta el plan de cierre de proyecto al haber logrado el alcance del mismo. Caso contrario, se genera una nueva iteración siguiendo el mismo proceso y acoplando los prototipos, haciendo crecer el sistema en términos de funcionalidad, complejidad, y documentación asociada [[10](#Ian051)]. El esquema del ciclo de vida iterativo se explica con el Gráfico 2, donde lo fundamental ver cómo se desarrollan incrementos repetitivamente hasta completar el sistema, teniendo en cuenta los diferentes procesos de pruebas, validación e integración.



Gráfico 2. Esquema de desarrollo en incrementos, tomado de [[10](#Ian051)]

Las entregas sirven como prototipos y también como revisiones de si el sistema está cumpliendo con las expectativas del cliente para posteriores correcciones, lo que reduce el riesgo de generar un fallo total del sistema como provocaría un esquema de cascada donde se desarrolla totalmente la funcionalidad y no de manera parcial, además de permitir la mejora de la calidad del producto al validar y verificar el estado de los incrementos. [[12](#Coc08)]

**Modelo Rational Unified Process (RUP)**

El Rational Unified Process (RUP) es un modelo que tiene una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo en donde describe una clase de los procesos que son iterativos e incrementales. Los procesos de RUP se centran en estimar tareas y horarios del plan midiendo la velocidad de las iteraciones. La principal ventaja de RUP es que se basa en las mejores prácticas que han sido probadas con anterioridad.

Se decide usar este modelo en conjunto con los dos anteriores por tener dentro de sus actividades [[13](#Cra04)] [[14](#Rat98)]:

* Levantamiento exhaustivo de requerimientos.
* Detectar defectos en las fases iniciales.
* Reducir el número de cambios tanto como sea posible.
* Realizar el Análisis y diseño, tan completo como sea posible.
* Diseño genérico, intenta anticiparse a futuras necesidades.

Adicionalmente, se divide el proyecto en 4 grandes fases importantes para su desarrollo: Inicio donde se define el alcance del proyecto, estimaciones y una visión aproximada del proyecto; elaboración donde se efectúa el análisis y diseño del sistema con una visión más refinada; Construcción, que es cuando se realiza implementación y pruebas del sistema para su posterior despliegue y la transición que da con la finalización del proyecto al efectuar el despliegue del proyecto [[14](#Rat98)].

RUP define nueve disciplinas a practicar en cada fase del proyecto, solo que algunas tienen más incidencia en cada fase con respecto a otras, que es uno de los objetivos buscados dentro de la organización del equipo de trabajo (ver Organigrama y Descripción de Roles), donde todos los roles tienen actividades por hacer pero en ciertos hitos no será tan significativo como las actividades de otros.

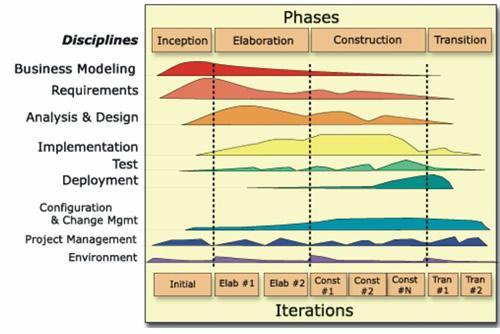


Gráfico 3. Esquema General de RUP tomado de [[15](#Mar)]

El gráfico 3 muestra un esquema de cómo se desarrolla un proyecto por medio de RUP. En el eje horizontal está la división del tiempo mediante fases e iteraciones, lo cual se puede ver como el aspecto dinámico del proceso, pues cada iteración y etapa posee un propósito distinto, aun cuando se desarrollen de manera similar. Por su parte, el eje vertical describe el proceso en términos de actividades, artefactos, trabajadores y flujos de trabajo, lo cual se puede ver como el aspecto estático del proceso, pues son tareas y roles que no van a cambiar durante todo el proyecto, lo único que cambia es la intensidad del tiempo que dedican.

Teniendo estos modelos de ciclo de vida para el proyecto de desarrollo de software, se puede concluir que se van a realizar varias iteraciones en varios hitos, adicionales a los ya planteados por los clientes, donde se van a realizar prototipos intermedios que servirán como medio para efectuar control, administración de riesgos y retroalimentación con respecto al cliente y los requerimientos definidos inicialmente.

El marco de trabajo permite que todos los equipos de trabajo tengan actividad durante los diferentes hitos definidos, solo que con una carga y profundidad que estará cambiando a medida que se avance en el proyecto, además de mantener una constante actividad de las actividades de soporte y control durante todo el tiempo de desarrollo del producto.

De esta manera se definen varios aspectos a tomar en cuenta para el desarrollo del proyecto de software, utilizando elementos de los modelos de ciclo de vida anteriormente mencionados:

* División del proyecto en fases e iteraciones donde se tienen en cuenta las fechas de los entregables e incrementos definidos (ver Entregables).
* Uso del esquema en espiral aplicado en las iteraciones, donde cada iteración desarrolla la misma secuencia de actividades:
  1. Realizar análisis del incremento actual: Eso incluye determinar los posibles riesgos que pueden ocurrir durante la iteración y generar alternativas dentro de esta iteración.
  2. Efectuar y desarrollar el incremento: Efectuar el prototipo propio de cada iteración, teniendo correspondencia a los requerimientos asignados para esa iteración. También se tienen en cuenta las estrategias para lidiar con los riesgos y las decisiones que se han tomado para cada uno.
  3. Desarrollar el incremento: Dado que cada incremento posee un conjunto determinado de requerimientos, se desarrolla ese de manera modular para no afectar el sistema total y hacer validaciones del módulo individualmente.
  4. Revisión y Corrección: En caso de encontrarse inconsistencias o incumplimiento de los requerimientos planteados, se efectúan correcciones hasta que se cumpla con lo planteado.
  5. Integración y validación: una vez corregido completamente el módulo funcional, se acopla con el sistema total y se hacen pruebas a nivel del sistema completo, generando el prototipo para mostrar a los clientes.
  6. Presentación al cliente: Ya cuando se ha asegurado que el prototipo ha cumplido con todo lo planteado por el requerimiento y ha sido completamente validado, se le muestra al cliente el prototipo para que pueda ver resultados parciales del sistema.
  7. Retroalimentación y planeación: una vez realizada la presentación con el cliente, se tienen en cuenta sus comentarios y sugerencias y a partir de esto realizar la planeación de la siguiente iteración, donde se pueden efectuar cambios de los requerimientos que vienen.
* Desarrollo de actividades de las disciplinas de RUP de manera paralela. Esto quiere decir que todas las áreas de desarrollo siempre estarán desarrollando actividades, aunque en diferente intensidad dependiendo de la fase en la que se encuentra.

Gráfico 3. Modelo de Ciclo de vida de SnoutPoint

### 8.1.1 Análisis de Alternativas y Justificación

Dado que se han decidido una serie de prácticas y modelos de ciclos de vida con los cuales se va a desarrollar el proyecto de software de SnoutPoint, fue necesario considerar otros modelos de ciclos de vida para poder evaluar su utilidad o no dentro del grupo de desarrollo. Los siguientes ciclos de vida fueron revisados y descartados pues no se ajustaban a lo requerido por el grupo de trabajo.

**Modelo en Cascada:**

El modelo en cascada consiste en desarrollar fases secuencialmente, describiendo las etapas fundamentales para el desarrollo de software [[10](#Ian051)]:

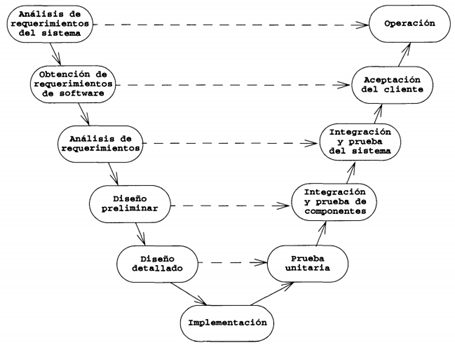
* Análisis y definición de requerimientos: Definir en detalle las funcionalidades y aspectos esenciales que debe tener el sistema.
* Diseño del sistema: Donde se describe la estructura del sistema y se hace una abstracción conceptual del mismo.
* Implementación y pruebas: Donde se realiza el software como un conjunto de unidades de programas.
* Integración: Todos los módulos se unen como un sistema único y se prueba como una sola unidad para asegurar que cumpla los requerimientos.
* Mantenimiento: lograr la permanencia del sistema a través del tiempo y corrección de errores no descubiertos anteriormente.

El esquema de cascada obliga que no se puede avanzar de una etapa sin que se haya completado en su totalidad la etapa previa. Esto implica que una vez finalizada una etapa, no se puede retroceder y es poco flexible en el momento de tratar con cambios o errores inesperados que no fueron tenidos en cuenta en etapas anteriores.

**Modelo en V**

El Modelo en V es una variante del modelo en cascada que “hace explícita la dependencia entre actividades de desarrollo y las de verificación [...] hace explícita la noción de nivel de abstracción”. [[11](#Bru02)]

Pese a tener varias etapas de validación, posee la misma falencia del esquema en cascada de no poder retroceder en el proceso, pues “asumen que después de que una actividad se termina y se revisa, el producto de trabajo asociado puede tomarse como línea base”(Bruegge), asunciones que sólo servirán si hay completa certeza que los requerimientos no se van a ver modificados ni corregidos durante el resto de etapas.

Gráfico 4. Esquema del Modelo en V

**Modelo de diente de sierra:**

“El modelo de diente de sierra muestra las percepciones del sistema por parte del usuario y el desarrollador de software en diferentes niveles de abstracción a lo largo del tiempo”. [[11](#Bru02)]

El principal problema que se pretende atacar con este ciclo de vida es que los requerimientos de software son normalmente dinámicos y podrían cambiar en forma drástica durante todo el desarrollo del proyecto y si se siguen otros ciclos de vida como cascada o modelo en V donde la primera vez que el usuario interactúa con el sistema es solo hasta el final del proyecto se corre el gran riesgo de que lo que se le entrega al usuario no es lo que él esperaba (no se satisfacen los requerimientos) ya que los usuarios y los implementadores tienen diferentes maneras de comprender los sistemas de software.

Para lograr esto el modelo dientes de sierra incluye una serie de actividades que usualmente están enfocadas a presentar prototipos del estado actual del sistema al usuario con el fin de que este último pueda evaluar cada prototipo, informar a los desarrolladores que se podría cambiar, agregar o que definitivamente se debe eliminar del prototipo ya que no es lo que se quiere del sistema final. Esto con el fin de corregir a tiempo problemas de requerimientos que cambian con el tiempo, logrando así una retroalimentación por parte del usuario en las distintas etapas del desarrollo del proyecto y haciendo que el cliente se involucre en su nivel de abstracción obteniendo un nivel de confianza mayor en cuanto a que el sistema final se parezca a lo que el usuario quiere. De aquí obtiene este ciclo de vida su nombre ya que cada demostración de prototipo al cliente da como resultado un “diente”.

Durante el desarrollo del sistema el usuario permanece en el nivel de los requerimientos, qué quiere del sistema y cómo lo quiere, mientras que los desarrolladores se enfocan en la factibilidad, si es posible lograr lo que el usuario quiere y como sería posible lograrlo. Entonces al inicio del proyecto los desarrolladores y el usuario están en el mismo nivel de abstracción, es decir, se acuerdan los requerimientos iniciales del sistema; luego los desarrolladores basándose en estos requerimientos elaboran un diseño tentativo del sistema el cual implementan en un prototipo, muy sencillo si se quiere, que podrían ser simples secuencia de las pantallas desde el punto de vista de los casos de uso, se muestra este prototipo al usuario quien evaluará si satisface los requerimientos inicialmente acordados. Esta actividad de “…desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las distintas versiones…”[5] se puede repetir varias veces durante el tiempo de vida del proyecto obteniendo cada vez un prototipo más cercano a lo que el usuario quiere.

## Lenguajes y Herramientas

### Gestión del proyecto

Para la gestión del proyecto, se hace necesario utilizar gran variedad de herramientas para controlar la organización de los documentos importantes y su correspondiente actualización, por parte de los responsables para mantener al día el calendario del proyecto.

Se desarrolló un estudio de mercado para determinar el alcance del proyecto (ver alcance), definido mediante una encuesta realizada en los formularios de Google: Encuestas y formularios fáciles de crear para todo el mundo [2], esta aplicación vinculada con la cuenta de correo en Gmail [3], nos permitió entrevistar a dueños de mascotas y dueños de compañías que prestan sus servicios a las mismas; gracias a sus modelos de análisis de encuestas mediante gráficas, se hizo más sencillo determinar las principales funcionalidades.

Para la organización de tareas, el grupo de trabajo hará uso de Asana, donde estarán consignadas tareas para los miembros del equipo y podrán ser revisados por cualquiera de los miembros. Se eligió Asana, porque además de proveernos gratuitamente un servicio de almacenamiento, nos permite crear tareas y asignarlas a alguno de los miembros del grupo de trabajo, admitiendo que otro de los integrantes pueda “seguir” el desarrollo de la misma.

Todos los documentos, tanto presentaciones como plantillas pertinentes para el grupo, serán realizados en *Microsoft Office [*9*] o Google Docs [10]* (editor online)*,* herramientas que nos permite desarrollar de manera estética y uniforme, cumpliendo con los estándares de calidad (ver plan de calidad hipervínculo), los entregables indispensables para el proyecto.

En cuanto a la creación del logo, usamos una herramienta gratis online llamada Graphics Spring: *Create yor own logo for free* [1], que no requiere habilidades técnicas o de diseño, basta con seguir un tutorial que va mostrando los pasos que el usuario debe completar, eligiendo la imagen y el nombre, los colores y el tamaño.

### Equipo de desarrollo

Las herramientas que elegimos para el desarrollo del proyecto son:

**Enterprise Architect:** Es una herramienta para diseño y modelado basado en el lenguaje OMG ML (lenguaje unificado de modelado), fue desarrollada por la empresa sparx systems. Con un gran conjunto de características como velocidad, trazabilidad y buen rendimiento [7], permite construir modelos que son necesarios para el desarrollo de nuevos sistemas como el nuestro, adicionalmente permite en la construcción de modelos generar archivos con su respectiva documentación [7].

El manejo de esta herramienta le corresponderá al equipo de análisis y diseño; es bastante útil debido a la experiencia adquirida en asignaturas previas, lo que implica que hay un entorno familiarizado de trabajo. Además, dichas habilidades adquiridas por los miembros del equipo a lo largo de la carrera permiten explotar la herramienta para un manejo eficiente de la información involucrada en el desarrollo de este proyecto.

Otras de las ventajas que presenta son que es una herramienta bastante robusta y tiene licencia paga en algunos computadores de la universidad.

**Bizagi process modeler**: Es un modelador freeware de procesos de negocio, fue desarrollado y ha sido distribuido por la compañía Bizagi Limited. Entre las características de esta herramienta están el poder realizar diagramas de procesos de tipo BPMN, además de poder documentarlos y simular su funcionalidad, buscando mejorar la eficiencia y entendimiento  operacional de la organización que se beneficiará con el producto a desarrollar [8].

Esta herramienta fue elegida ya que cuenta con elementos útiles para ajustarse a las necesidades del proyecto y del producto que estamos desarrollando. Además, el equipo de análisis y diseño ya tiene experiencia con este programa, ya que anteriormente se vio involucrado en el modelado de procesos; visto en el curso Sistemas de Información, donde se aprendió sobre la importancia de la buena definición de los procesos de impacto en un proyecto y/o negocio, todo esto modelado a través de prácticas funcionalidades que son brindadas por esta herramienta. Adicionalmente entre sus funciones está la posibilidad de alinear y organizar los recursos de tal manera que contribuyan a un mejor conocimiento de la organización para la cual estamos creando software [8].

**Paint.net:** Es un software gratuito para PCs con Windows, editor de fotos e imágenes. Cuenta con una interfaz de usuario intuitiva e innovadora con soporte para capas, efectos especiales, y una amplia variedad de herramientas útiles y potentes. Una comunidad en línea activa y creciente proporciona ayuda amistosa, tutoriales y plugins. Comenzó el desarrollo como un proyecto de diseño de alto nivel universitario de pregrado apadrinado por Microsoft, y es mantenido actualmente por algunos de los ex alumnos que originalmente trabajaron en ella. Originalmente concebido como un reemplazo libre para el software Microsoft Paint que viene con Windows, se ha convertido en una poderosa herramienta de imagen y editor de fotos con todo simple. Se ha comparado con otros paquetes de software de edición de fotos digitales, tales como Adobe® Photoshop®, Corel® Paint Shop Pro®, Microsoft Photo Editor, y GIMP.[11] La aplicación en este Proyecto es editar imágenes que se vayan a colocar en nuestra aplicación web.

**Brakets:** “A modern, open source text editor that understands web design.” Fue creado por Adobe Systems, con licencia bajo la licencia MIT, y en la actualidad se mantiene en GitHub. Bracktes, está disponible para su descarga multiplataforma para Mac, Windows y Linux. El 4 de noviembre de 2014, Adobe ha anunciado la versión 1.0 de los soportes. La actualización introduce nuevas características tales como acceso directo personalizadas y combinaciones de teclas más preciso que JavaScript. Algunas de sus características, y por la cuales escogimos esta herramienta, son:

Edición rápida de documentos, Live Preview, JSLint, Menos apoyo, Integración Teseo, Open Source, extensibilidad [12]

**Sublime text: “***Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance***.” Sublime Text**  es un [editor de texto](http://es.wikipedia.org/wiki/Editor_de_texto) y [editor de código fuente](http://es.wikipedia.org/wiki/Editor_de_c%C3%B3digo_fuente) está escrito en [C++](http://es.wikipedia.org/wiki/C%2B%2B) y [Python](http://es.wikipedia.org/wiki/Python" \o "Python) para los plugins. Desarrollado originalmente como una extensión de [Vim](http://es.wikipedia.org/wiki/Vim" \o "Vim), con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo [vi](http://es.wikipedia.org/wiki/Vi) llamado *Vintage mode*. Se distribuye de forma gratuita, sin embargo no es [software libre](http://es.wikipedia.org/wiki/Software_libre) o de [código abierto](http://es.wikipedia.org/wiki/C%C3%B3digo_abierto), se puede obtener una licencia para su uso ilimitado, pero al no disponer de ésta no genera ninguna limitación más allá de una alerta cada cierto tiempo. [13]

**GitHub:** Aloja el repositorio de código y documentos, además de brindar herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls. Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un push al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original. [14]

**Source Tree:**  “Is a powerful Git and Mercurial desktop client for developers on Mac or Windows. Say goodbye to the command line and use the full capabilities of Git and Hg through SourceTree's beautifully simple interface.” [15] Un programa que permitirá la fácil gestión del repositorio de documentos en GitHub, ya que realiza las operaciones de comando de consola de manera transparente para el usuario, facilitando tiempo de realizar las tareas, controlar y gestionar el repositorio.

**MySQL** es un manejador de bases de datos Open Source y actualmente es el más popular del mundo gracias a sus excelentes características como velocidad, confiabilidad, escalabilidad, fácil uso, fuerte protección, flexibilidad, entre otros [16]. Es una herramienta con la cual los programadores del proyecto ya han tenido experiencia y permite trabajar con bases de datos de tablas relacionales, que es precisamente lo que se necesita para el sistema en desarrollo.

**HTML** Es el lenguaje que se emplea para el desarrollo de páginas de internet. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.

HTML nació públicamente en un documento llamado HTML Tags (Etiquetas HTML), publicado por primera vez en Internet por Tim Berners-Lee en 1991. En esta publicación se describen 22 etiquetas que mostraban un diseño inicial y relativamente simple de HTML. Varios de estos elementos se conservan en la actualidad. Otros se han dejado de usar, y muchos otros se han ido añadiendo con el paso de los años. De esta manera, podemos hablar de que han existido distintas versiones de HTML a lo largo de la historia de internet. Nosotros vamos a trabajar con el HTML estándar actual, que es el utilizado por los navegadores y páginas web de hoy en día. Sin embargo, no vamos a prestarle atención a las versiones y especificidades de cada versión, ya que el objetivo de este curso es aprender**los fundamentos de HTML** y entender cómo funciona, no conocer la sintaxis o especificidades de una versión concreta. ¿Por qué no le damos importancia a la versión? Porque una persona que cuenta con los fundamentos y comprensión básica sobre el lenguaje es capaz de adaptarse a las características particulares de una versión sin problema. En cambio, centrarse en los detalles de una versión sin conocer los fundamentos hará que no tengamos capacidad para comprender lo que hacemos ni para adaptarnos a los continuos cambios que tienen lugar en el ámbito de los desarrollos web. [17]

**Ruby on rails** Es una plataforma de trabajo para realizar desarrollos web. Podemos definirlo como un framework de software libre. Ruby hace referencia al lenguaje de programación que se utiliza: todo se escribe en Ruby. Por otro lado, on Rails (sobre raíles) indica que el framework “nos va guiando” para hacer fáciles nuestros desarrollos. Por framework entendemos un entorno o conjunto de programas o herramientas agrupadas. El número de usuarios de Ruby está yendo en aumento en los últimos años; sus usuarios lo defienden dando varios argumentos entre los que podríamos destacar que “funciona de forma eficiente” y por otro lado valoran mucho la sencillez de su sintaxis, que se aproxima a lo que podría ser el inglés escrito, un lenguaje “casi natural”. Hemos dicho que todo lo que se escribe en Ruby on Rails se escribe en Ruby, luego el framework se encarga de transformarlo en otros lenguajes en función de las necesidades: por ejemplo, en javascript para mostrar un formulario en una página web o en SQL para realizar comunicaciones con una base de datos. El hecho de manejar un solo lenguaje hace que Ruby on Rails resulte sencillo de utilizar. [18]

**Corel** Ilustración vectorial y diseño de páginas, herramientas de dibujo y vectorización versátiles, edición profesional de fotografías, diseño sencillo de sitios web. Esta es la herramienta alternativa de Draw.io, no será utilizada debido a la licencia que hay que pagar[19]

**Aptana Studio E**s un IDE de desarrollo para aplicaciones de la web 2.0, gratuito, código libre, con soporte Ajax, PHP, Ruby on Rails, Adobe Air, iPhone, etc. Con Aptana se facilita en desarrollo integrado de Ajax con las tecnologías emergentes. [27]

### 8.3.1 Análisis de alternativas y Justificación

Respecto a la organización de los archivos, actualmente existen herramientas como *Google Drive* y aunque nos permite editar los documentos, no nos ofrece ningún tipo de control de versiones; por la misma razón no usaremos las herramientas *Dropbox[5]* ni *Onedrive[6].*

**Suite de Adobe** Esta es la herramienta alternativa, no será utilizada debido a la licencia que hay que pagar, al igual que Corel. [21]

Para la parte de programación se presentaron varias alternativas que pudieran haber suplantado el uso de MySql. Algunas como:

* Oracle
* SQL Server
* DB2
* Informix

**StarUML:** Es una herramienta de modelado open source de UML, fue desarrollada por la empresa Plastic Software con los aportes y la ayuda de la comunidad mundial. Es una herramienta sencilla, con gran usabilidad, cuenta con un gran conjunto de funcionalidades como la creación de modelos y diagramas, además permite convertirlos a diferentes formatos de imagen como JPG o PNG, entre otras [22].

Sin embargo, esta herramienta no se eligió ya que no genera la documentación de los diagramas o modelos, y las opciones o características que ofrece son muy básicas, no son suficientes para cubrir las necesidades del producto y del proyecto en general que estamos desarrollando.

**Borland Together:** Es una herramienta para el modelado UML, fue desarrollada por la compañía Microfocus; es un programa que cuenta con características ideales diseñar diferentes tipos de proyectos de software [23].

Además, es una herramienta con la cual el equipo de análisis y diseño ha trabajado anteriormente, en clase de Análisis y diseño orientado a objetos, sin embargo, no se eligió este programa ya que se debe pagar una licencia cercana a los 450.000 pesos colombianos para su uso y generaría un gasto innecesario ya que contamos con herramientas con funcionalidades similares [23].

**BPMN2 modeler:** Es una herramienta para el modelado de procesos de negocio (BPMN), fue desarrollada por Eclipse Foundation, cuenta con unas características de modelado y diseño útiles para nuestro proyecto, sin embargo no se eligió ya que aunque es muy similar a la herramienta Bizagui, al ser una herramienta ajena al grupo, nos tomaría tiempo la adaptación y capacitación para su uso [24].

**Toad Data Modeler:** Es una herramienta para el modelamiento de bases de datos que es actualmente distribuida por DELL, cuenta con las características básicas para el manejo de entidades relacionadas con las que debe contar cualquier modelador de bases de datos. Sin embargo, el grupo de análisis y diseño no posee conocimiento para la utilización de esta herramienta, la adaptación y capacitación consumiría tiempo que puede ser aprovechado para otros fines, para poder acceder a todas sus funcionalidades existe una versión de prueba gratuita pero es por tiempo limitado, la licencia permanente tiene un valor cercano a los 656,00 dólares, lo que generaría un gasto innecesario [25].

**BitBucket** Es una herramienta alternativa para GitHub, aunque esta herramienta nos ofrece un repositorio on-line, no nos ofrece un permiso para más de 5 personas sin pagar, para un grupo grande como SnoutPoint, es mejor contar con herramientas que nos permitan editar código y documentos, sin tener que pagar por licencias [21].

**StarOffice** Herramienta Open Source para Linux [26], al igual que Microsoft Office, nos permite crear y editar todo tipo de documentos, no utilizaremos esta herramienta ya que los formatos con los que quedan guardados los archivos son incompatibles para los integrantes del grupo que utilicen el sistema operativo Microsoft.

**PHP** es un lenguaje que aunque es plenamente utilizado para el desarrollo de páginas web, presenta una seria inestabilidad en cuanto al aprendizaje y capacitación del lenguaje, pues hay gran diversidad de frameworks de desarrollo en PHP como cakePHP, CodeIgniter, Zend, FuelPHP, entre otros. Hecho por el cual el entrenamiento y capacitación en este lenguaje puede resultar sumamente compleja en cuanto a que cada tutorial o documentos de aprendizaje utilizan distintos frameworks, los cuales poseen APIs distintos, unos de otros, haciendo que el aprendizaje del lenguaje no sea del todo uniforme como lo es con Ruby on Rails que es un framework único.

## Plan de Aceptación del Producto

Propósito: Acordar entre cliente y equipo de desarrollo, cuáles son los criterios para que cada entregable sea aceptado a satisfacción por el cliente.

Contenido: Para cada entregable, indicar qué criterios debe cumplir para ser aceptado a satisfacción por el cliente. Dichos criterios deben ser objetivos y medibles de manera formal. Indicar, también, las técnicas y herramientas requeridas para medir el cumplimiento del criterio, así como referenciar las secciones que apliquen del presente documento.

Si el modelo de ciclo de vida es iterativo o ágil, hay que tener en cuenta que entregables, tales como el código, deben satisfacer los criterios de aceptación para cada iteración.

Tamaño sugerido: 2 páginas

Referencias: [1], sección 5.4.5

## Organización del Proyecto y Comunicación

### Interfaces Externas

SnoutPoint Networks posee dos grupos de clientes a tener en cuenta: los clientes académicos y los clientes potenciales que utilizarán los servicios de SnoutPoint, sean aficionados a las mascotas, dueños o prestadores de servicios.

**Clientes Académicos**

Los clientes académicos serán los profesores encargados del curso de Ingeniería de Software: Miguel Eduardo Torres y Jaime Andrés Pavlich Mariscal [[4](#Pav)], profesores del departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana. Estos clientes serán con los que el grupo de desarrollo tendrá una mayor interacción y con quienes se les mostrarán los resultados de cada incremento, además de las entregas principales (ver Entregables).

Por ser quienes tienen una relación más directa con el grupo serán quienes efectuarán la retroalimentación y evaluación respectiva de los prototipos y entregables, lo cual será un elemento clave para el desarrollo y planeación del proyecto de software, además de la corrección oportuna de los posibles problemas encontrados en los entregables y prototipos.

**Potenciales Clientes de SnoutPoint**

Los usuarios finales de SnoutPoint serán todas las personas interesadas en el mundo de las mascotas, ya sean dueños de mascotas, aficionados a estas que no las tienen pero les gustan estas o prestadores de servicios de mascotas. Ellos serán quienes sacaran el mayor provecho de todos los servicios que ofrecerá la red social y quienes van a entablar relaciones interpersonales por medio de esta red social y sus mascotas, además de proveer oportunidades de negocio a quienes prestan servicios, logrando alcanzar un mercado mucho más amplio al poder promocionarse a una gran cantidad de dueños de mascotas que probablemente no los conocerían antes de ver su perfil.

A diferencia de los usuarios académicos, estos clientes no se les mostrará el producto hasta que haya sido completado el sistema, es decir, hasta el último incremento y cuando se haya cumplido el alcance definido (ver Alcance).

### Organigrama y Descripción de Roles

Para la división de responsabilidades dentro de SnoutPoint Networks, se decide manejar la organización mediante equipos de trabajo en vez de utilizar un rol específico, dado que la exigencia de algunos roles puede ser menor durante algunos hitos en el proyecto, por lo cual una persona puede estar asignada en varios equipos de trabajo. Adicionalmente, el trabajar en varios sectores del proyecto le dará a los miembros diferentes perspectivas del proyecto.

Teniendo en cuenta el esquema del modelo de RUP [[14](#Rat98)], la etapa de concepción e inicio del proyecto las tareas de programación se limitan únicamente a tareas de capacitación de los lenguajes de programación requeridos y ejercicios básicos para manejar las herramientas, además de no requerirse un prototipo en el primer incremento (ver Entregables).

Gráfico 4. Organigrama de Equipos de Trabajo

Todas las áreas de trabajo estarán bajo la misma jerarquía de trabajo y vigilados constantemente por la labor de la gerencia. Sin embargo, los equipos de trabajo no estarán aislados entre sí, como se puede ver en el Gráfico 4, todos los grupos tienen líneas de conexión entre sí, lo que quiere mostrar que pueden comunicarse entre sí y hacer solicitudes sin necesidad de tener intermediación de la gerencia.

También se decide usar el esquema de equipo de trabajo ya que si hay dos personas o más trabajando en una misma tarea o actividad, se pueden tener distintas perspectivas para abordar el desarrollo de la tarea, además de poder detectar y solucionar problemas de manera acelerada y oportuna: "Dada una base suficiente de desarrolladores asistentes y beta-testers, casi cualquier problema puede ser caracterizado rápidamente, y su solución ser obvia al menos para alguien. O, dicho de manera menos formal, con muchas miradas, todos los errores saltarán a la vista.” [[16](#Eri10)]

Los equipos de trabajo, así como los miembros que pertenecen a estos equipos y las tareas correspondientes en términos generales se describen en la Tabla 1.

|  |  |  |
| --- | --- | --- |
| **Equipo de Trabajo** | **Miembros Pertenecientes al área de trabajo** | **Tareas Correspondientes** |
| **Gerencia** [[6](#IEE10)][[17](#Pro)] | * Camilo Oviedo * Esteban Hernández | * Gestionar el talento humano y relaciones del grupo de desarrollo. * Planeación de reuniones y actividades del desarrollo del proyecto. * Integración de comunicación entre las áreas del equipo de desarrollo. * Generación de reporte gerencial. * Hacer cumplir los compromisos pactados dentro del reglamento. * Registro y gestión de los fondos recolectados por multas. * Planificar las actividades a realizar por cada área de trabajo y hacer seguimiento del cumplimiento de este plan. * Revisiones finales de los entregables para los clientes |
| **Desarrollo y Pruebas** [[7](#Spa14)] | * David Suarez * Sebastián Jiménez * Camilo Benavides | * Capacitación de los lenguajes y técnicas de programación requeridas para el desarrollo del proyecto. * Pruebas del sistema para verificar el cumplimiento de funcionalidades y requerimiento. * Documentación de código. * Uso de buenas prácticas de programación para el desarrollo de un código entendible y limpio. * Estar involucrado e informado con el desarrollo del diseño del sistema. |
| **Análisis y Diseño** [[7](#Spa14)] | * Fabiana Díaz * Sebastián Jiménez * Camilo Benavides * Camilo Oviedo | * Definición del sistema y la oportunidad de negocio. * Definición de los requerimientos y funcionalidades del software. * Desarrollo de modelos de dominio, diagramas de clase, casos de uso, secuencias, modelos relacionales y cualquier otro esquema que apoye a la abstracción y comprensión del software. * Aplicación de patrones y buenas prácticas de modelado. * Análisis Especificación de casos de usos. |
| **Gestión de Documentación, Calidad, Riesgos y Configuraciones** | * Esteban Hernández * Fabiana Díaz * Alejandra Rocha | * Generación de documentos de apoyo que den explicación del funcionamiento del sistema, sus componentes, organización, etc. Adicionalmente, actas, métricas del desarrollo, planificaciones, manuales y otros documentos para alcanzar la categoría de software. * Revisión detallada y verificación de la calidad de cualquier entregable, ya sean diseños, códigos fuente, documentación, etc. * Plantear posibles riesgos que se pueden presentar en cualquier actividad del desarrollo del proyecto. * Desarrollo de planes de contingencia en caso de presentarse problemas críticos dentro de los procesos fundamentales del proyecto. * Definir y verificar el estilo, presentación, redacción, ortografía y coherencia de los documentos y entregables. * Revisión de la aplicación de estándares y buenas prácticas. * Revisión de control y administración de configuraciones de archivos, documentos y código fuente, así como revisión del versionamiento aplicado del mismo. |

Tabla 1. Equipos de Trabajo y Actividades Generales

En cuanto a los canales de comunicación se definen diferentes formas de efectuar la comunicación dentro del grupo, cada uno determinado propósito para el desarrollo de las diferentes actividades dentro del grupo de trabajo:

* **Reuniones presenciales cada lunes y viernes:** donde todos los miembros deben estar presentes para planear actividades durante la semana o revisar el resultado de las anteriormente planeadas, así como discutir propuestas, dificultades y soluciones encontradas para el grupo. Todo elemento que es relevante para el desarrollo del proyecto se consigna dentro del acta de la reunión (ver Anexos, Plantillas).
* **Chat Grupal en Facebook:** Dado que no todos los miembros del grupo tienen acceso a Smartphone y por lo tanto tampoco para Whatsapp (que resultaría una herramienta de mensajería instantánea mucho más constante para la comunicación) se decide que las comunicaciones dentro del grupo se realicen mediante un chat creado en Facebook donde participan todos los miembros del grupo de desarrollo de SnoutPoint. En este canal se comunican avisos como lugares de reunión, recordatorios de actividades y dudas que se hayan encontrado después de las reuniones.
* **Notificaciones de tareas en Asana:** Como medio para constatar y dejar un registro adicional de las tareas y actividades a realizar de cada miembro del grupo, se utiliza Asana (ver Lenguajes y Herramientas) para generar recordatorios y notificaciones a cada persona, pues cada que se asigna una actividad, la persona a quien se le asigna recibe un mensaje por correo electrónico dando aviso de esta.

También cabe aclarar que el esquema organización de los equipos de trabajo no hace que cada equipo este aislado uno de otro. Teniendo en cuenta el modelo de trabajo de Bazar desarrollada por Torvalds y luego descrita por Raymond [[16](#Eri10)], donde el grupo de desarrollo esta “colmado de individuos con propósitos y enfoques dispares […] de donde surgiría un sistema estable y coherente únicamente a partir de una serie de artilugios” [[16](#Eri10)], el equipo de trabajo funcionará mediante un esquema colaborativo, donde todos puedan intervenir en las actividades de las otras áreas cuando lo requieran y todo el contenido producido por cada equipo de trabajo estará a disposición para los demás equipos de trabajo y la gerencia.

# 9. Administración del Proyecto



## Métodos y Herramientas de Estimación

Propósito: Fundamentar adecuadamente todas las estimaciones realizadas en el presente documento

Contenido: Para la estimación realizada en el presente documento, explicar los métodos y herramientas utilizadas para elaborar dicha estimación (debe incluir la estimación del Software y la del proyecto), así como la sección donde se muestran dichas estimaciones. Ejemplos de estimaciones pueden ser: tiempos requeridos para realizar una tarea, tamaño o complejidad del software, presupuestos asignados, etc.

Tamaño sugerido: 3 páginas.

Referencias: [1], sección 5.5.1.1

## Inicio del proyecto

Propósito: Que los integrantes del proyecto sepan qué deben hacer para poner en marcha el proyecto.

Contenido: ver sub-secciones

### Entrenamiento del Personal

Propósito: Que los integrantes del proyecto sepan cómo van a obtener los conocimientos necesarios para realizar el proyecto y cómo transferir su propio conocimiento a los demás participantes, tanto al inicio, como durante el proyecto.

Contenido: Para cada tipo de conocimiento o habilidad requerida para el proyecto que sea insuficientemente satisfecha por el equipo de proyecto, indicar quiénes se entrenarán en dichos conocimientos o habilidades, quién les enseñará y cómo se realizará el proceso de enseñanza. Importante esto es un plan, no en sí mismo lo que hay que aprender.

Tamaño sugerido: 1 página

Referencias: [1], sección 5.5.1.4

### Infraestructura

Propósito: Que los integrantes del proyecto entiendan cómo se van a obtener, poner en marcha y mantener todas las herramientas software e infraestructura hardware necesarias para el proyecto.

Contenido: Indicar tareas a realizar para obtener las herramientas, ponerlas en marcha para el proyecto y mantenerlas durante el mismo, así como qué roles son los responsables de cada tarea, cuándo se realizará cada tarea y con qué frecuencia.

Tamaño sugerido: 3 páginas

Referencias: [1], sección 5.4.3

## Planes de Trabajo del Proyecto

Propósito: Que los integrantes del proyecto sepan qué es lo que deben hacer durante la ejecución del proyecto.

Recuerde un plan incluye: quien, cómo, cuándo y con qué herramientas se ejecutara el plan (además de las actividades del plan!).

Contenido: ver sub-secciones

### Descomposición de Actividades

Para la estructura de descomposición de tareas o WBS, se decide dividir el proyecto en Procesos que serán 5 incrementos correspondientes 3 entregas principales definidas por los clientes (ver Entregables y ver Supuestos y Restricciones) y 2 entregas intermedias de prototipos. Cada Proceso tendrá una serie de actividades principales para realizarse y cada actividad una serie de tareas asignadas puntualmente a un miembro de SnoutPoint Networks. En el Gráfico 5, se puede visualizar a grandes rasgos los principales procesos y sus actividades respectivas.

Gráfico . Procesos y Actividades por Incremento

Propósito: Que los integrantes del proyecto sepan cuáles son las principales actividades del proyecto y su descomposición

Contenido: Una estructura de descomposición de tareas (WBS) que describa las principales actividades a realizar en el proyecto y las relaciones de agregación entre ellas.

Para ciclos de vida iterativos o ágiles, no es necesario hacer el WBS para todas las iteraciones. Basta con descomponer una sola.

Tamaño sugerido: 1 página

Referencias: [1], sección 5.5.2.1

### Calendarización

Propósito: Que los integrantes del proyecto sepan cuáles son las fechas de inicio y término de cada actividad y las relaciones de precedencia entre actividades (si las hubiere).

Contenido: Una carta Gantt que describa las principales actividades del proyecto y sus relaciones de precedencia (si las hubiere).

Para ciclos de vida iterativos o ágiles, no es necesario indicar aquellas actividades que no tengan fechas ni precedencias claras o que sean muy repetitivas, por ejemplo: diseño, implementación, pruebas, reuniones diarias. Sin embargo, es necesario detallar las actividades que se realicen en momentos específicos, por ejemplo: reunión de planeación de iteración, demostración al cliente, etc. Dado que son las mismas actividades por iteración, basta con mostrar el detalle para una sola iteración. En ese caso, la estructura de la iteración se puede explicar con un diagrama BPMN, en adición a la carta Gantt.

**SE DEBE ACTUALIZAR EN TODAS LAS ENTREGAS**

Tamaño sugerido: 2 páginas

Referencias: [1], sección 5.5.2.2

### Asignación de Recursos

Propósito: Que los integrantes del proyecto sepan qué actividades deben ejecutar y los recursos que tienen disponibles para ejecutarlas.

Contenido: Para cada actividad principal identificada en el WBS (por ejemplo, fases del proyecto), indicar los recursos necesarios y los roles requeridos para ejecutarla.

Tamaño sugerido: 1 página

Referencias: [1], sección 5.5.2.3

### Asignación de Presupuesto y Justificación

Propósito: Que los integrantes del proyecto y las personas responsables de la parte financiera del proyecto sepan cuáles son los flujos de dinero durante el proyecto y cuándo ocurren.

Contenido: Una tabla de flujo de caja detallando las entradas y salidas de dinero para las principales actividades identificadas en el WBS, así como los períodos en los cuales ocurren.

Si bien no se maneja dinero en el proyecto como tal, debe incluirse la estimación de costo del software. Esto es complementario al costo del proyecto calculado previamente.

Tamaño sugerido: 1 página

Referencias: [1], sección 5.5.2.4

# . Monitoreo y Control del Proyecto

Propósito: Que los administradores del proyecto sepan cómo enterarse del "estado de salud" del proyecto, así como las acciones de control a ejecutar en caso de ocurrir algún problema en el proyecto.

Contenido: ver sub-secciones.

## Administración de Requerimientos

Propósito: Que los ingenieros de requerimientos sepan cómo detectar, reportar y controlar cambios en los requerimientos del producto. Que los clientes y desarrolladores sepan cuáles son sus responsabilidades en este proceso

Contenido

Artefactos a generar producto de la administración de requerimientos. Por ejemplo: Descripción de casos de uso, matrices de trazabilidad, etc.

Diagrama BPMN y un párrafo que describa el proceso de administración de requerimientos (Tamaño sugerido: 1 página). El diagrama debe también incluir los momentos en los cuales se crean o refinan los diferentes artefactos.

Tamaño sugerido: 2 páginas

Referencias: [1], sección 5.6.1

## Monitoreo y Control de Progreso

Propósito: Que los administradores sepan cómo medir el progreso del proyecto. Que los demás integrantes sepan cómo reportar dicha información

Contenido:

*CxOne Requierements*

*División de monitoreo por áreas. Cada área posee sus propios procedimientos, plantillas asociadas.*

*Reuniones como actividades de seguimiento*

Descripción de las unidades (**METRICAS**) utilizadas para medir progreso. Por ejemplo: número de requerimientos implementados, horas faltantes para completar una tarea, etc. Si esto ya fue descrito en la sección 9.1, referenciar dicha sección.

Actividades a realizar para reportar progreso: responsables, momentos en que se realizan, descripción. Por ejemplo, cómo medir que un requerimiento fue implementado exitosamente. Esto debe ser consistente con las unidades de progreso definidas.

Acciones correctivas. Qué hacer en caso que el progreso del proyecto no vaya según lo esperado: responsables, momentos en que se realizan, descripción.

Tamaño sugerido: 5 páginas

Referencias: [1], sección 5.6.3

## Cierre del Proyecto

Propósito: Que los integrantes del proyecto y las entidades externas a éste, sepan qué hay que hacer para cerrar el proyecto

Contenido: En un proyecto real, esta sección describiría las acciones a seguir al finalizar el proyecto y sus responsables. Para efectos del curso, se deben describir las acciones e insumos necesarios, para las actividades a realizar previas a la cada entrega del proyecto: post-mortem de la fase, elaboración del reporte gerencial, etc.

Tamaño sugerido: 1/2 página

Referencias: [1], sección 5.6.7

# Entrega del Producto

Propósito: Describir el proceso de transición, es decir todo lo requerido para implantar el sistema en la organización del cliente

Contenido: Un proyecto real requeriría describir muchas actividades, tales como: capacitaciones, transición de sistemas antiguos al sistema nuevo, entrega de manuales, instalación del sistema, etc. En el contexto del proyecto, basta con explicar brevemente las acciones a realizar para entregar el producto final al cliente al final del semestre, incluyendo: artefactos a entregar, momentos en que se entregarán y responsables de realizar dichas entregas. Debe relacionarse con el Plan de Aceptación.

Tamaño sugerido: 1/2 página

Referencias: [1], sección 5.7

# Procesos de Soporte

## Ambiente de Trabajo

**Compromisos**

1. Toda relación entre los miembros del grupo debe ser respetuosa, solidaria y profesional.
2. La comunicación debe ser abierta y sin ningún tipo de restricción.
3. Seguir conducto regular para la resolución de conflictos. Se entiende como conducto regular el camino para la resolución de problemas. Si se tiene un inconveniente puntual con una persona del grupo, se discutirá con esta directamente. De no solucionarse, se discutirá a nivel de todo el grupo de desarrollo con la gestión oportuna de la gerencia. Si ya no se solucionara, se procede a discutirlo con alguno de los profesores encargados del curso de Ingeniería de Software, Jaime Pavlich y Miguel Torres.
4. Todos los miembros del grupo deben estar plenamente comprometidos con cualquier actividad relacionada con la materia, sea dentro o fuera de la clase.
5. Todo documento entregado debe estar completo, hecho con calidad y que corresponda a la realidad y que refleje un nivel aceptable de investigación mediante bibliografía y referencias.
6. Todo lo que se discuta o desarrolle dentro del grupo será tratado de manera confidencial, evitando la divulgación de documentos, plantilla o código fuente desarrollado dentro las labores del grupo.
7. Queda prohibido cualquier tipo de plagio o copia de información ya sea dentro de documentos o código fuente.
8. Cualquier contenido que pertenezca a otros autores, debe tener su correspondiente referencia.
9. Cualquier entregable debe ser presentado en la fecha establecida, sin falta. En alguna eventualidad, debe ser avisada con anterioridad de mínimo 2 días. Se tendrán en cuenta casos excepcionales.
10. Todas las decisiones de carácter gerencial, es decir, decisiones que afecten a todo el grupo de desarrollo, deben ser tomadas con todo el grupo de trabajo.
11. Se efectuarán sanciones y multas en caso de cualquier acto que afecte el pleno desarrollo del proyecto. Se llevará un control de las sanciones de cada integrante del grupo.
12. Todos los miembros deben ser sinceros en su capacidades de trabajo y subestimar o sobreestimar estas.

**Multas**

1. Se definirá multa como cualquier acto que vaya en contra de los compromisos pactados anteriormente.
2. La reincidencia en 3 multas hará que la persona tenga una sanción.
3. Cualquier llegada tardía a una reunión previamente concertada que no sea avisada anteriormente o justificada incurrirá en una multa de 4.000 (Cuatro Mil) COP. Se considerará una llegada tardía al asistir a una reunión pasados los primeros 15 minutos de la reunión.
4. Cualquier distracción o llamado de atención en clase o durante las reuniones incurrirá en una multa de 2.000 (Dos Mil) COP.
5. La inasistencia a reuniones sin justificación incurrirá en una multa de 3.500 (Tres mil quinientos) COP.
6. En caso de ser reincidente de estos actos, la multa será incrementada en un 50% con respecto al valor inicial. Es decir, si la multa es de 5.000 COP y se reincide, la siguiente multa tendrá un valor 7.500 COP.
7. El dinero recolectado por las multas será utilizado por una actividad de incentivo para cada corte. Esto será gestionado por el Gerente, Camilo Oviedo.
8. La entrega tardía de artefactos, documentos o elementos intermedios para el desarrollo de las entregas tendrá una multa de 5.000 (Cinco Mil) COP.

**Sanciones**

1. Se definirá la sanción como cualquier falta o acto que perjudique el desarrollo del grupo o del proyecto.
2. Tres sanciones acumuladas serán justificante para una reunión donde se decidirá su situación en el grupo de desarrollo. En caso de continuar en el grupo y reincidir en cualquier sanción.
3. Las siguientes faltas serán consideradas como sancionables:
   1. La falta de respeto ante cualquier otro miembro del grupo o los clientes.
   2. El uso indebido de propiedad intelectual o contenidos sin su respectiva referencia bibliográfica.
   3. Entrega de documentos o código fuente no corregido cuando se ha notificado anteriormente de tales correcciones.

## Análisis y Administración de Riesgos

## Administración de Configuración y Documentación

**Objetivo del Plan**

El plan de administración de configuración y documentación busca llevar seguimiento del estado de los documentos, así como el control de versiones del mismo y aspectos como ortografía, cohesión, presentación y estilo, entre otros. Se tendrán en cuenta procedimientos para efectuar el control de versiones, definición de ítems de configuración, formatos y plantillas asociadas para el desarrollo de documentos.

**Encargado del Plan**

El control de esta administración dependerá del equipo de Documentación, Calidad, Riesgos y Configuraciones, lo que permitirá que se efectúe una revisión más profunda y con una menor probabilidad de no corregir un posible error por tratarse de todo un equipo y no solo integrante del grupo.

Se define que la política de versionamiento se hará añadiendo a los documentos el sufijo vA.B.C., donde A es el número de *entregas* (“distribuciones que ha tenido el ítem de configuración fuera de la organización, por ejemplo, al cliente” [[18](#IEE87)]) que ha tenido ese documento; B el número de *versiones* (“identificador que se le da a un ítem de configuración cuando capacidades funcionales le son añadidas, modificadas o borradas” [[18](#IEE87)]) que ha tenido ese ítem de configuración y C el número de *revisiones* (“asociado con la noción de arreglo de bugs, que consiste en la corrección de errores en el diseño lógico pero que no tienen repercusión en la funcionalidad documentada pues los requerimientos no han sido modificados” [[18](#IEE87)]).

Los ítems de configuración, a los cuales se les aplicará las anteriores políticas de versionamiento, son aquellos que sean entregables, pues permite hacer trazabilidad de los cambios que han sufrido los documentos, así como cuantas veces han sido entregados al cliente:

* Software Project Management Plan (SPMP)
* Software Requirement Specification (SRS)
* Software Design Document (SDD)
* Diagramas y Documentación de especificación de casos de uso.
* Código Fuente.

El control de la numeración de versiones será apoyada mediante etiquetado de SourceTree en GitHub (Revisar Métodos y Herramientas), actualizando los documentos en el repositorio asignado y permitiendo a los demás miembros del equipo poder visualizar los cambios que han hecho otros en los documentos.

**Entregables de Documentación**

Para controlar los cambios que puedan sufrir el SPMP, SDD y SRS durante su desarrollo, se decide utilizar una rama de desarrollo para generar adelantos y cambios del documento, evitando llegar a modificar o sobrescribir la línea base de este documento. Se desarrollaran varias subsecciones de forma concurrente dentro del SPMP, los cuales serán revisados por el equipo de Documentación, Calidad, Riesgos y Configuraciones; los cuales buscarán errores o inquietudes dentro del documento y se las señalaran al autor del documento hasta que se encuentre correcta siguiendo una lista de chequeo (ver Lista de Chequeo de subsecciones) El documento aumentará su número de revisión en uno hasta que sea aprobada la sección por el equipo de Documentación, Calidad, Riesgos y Configuraciones.

La gerencia realizara una última revisión antes de agregarlas al SPMP y subirlas al repositorio, incrementando su número de versión y el número de revisión se reiniciará a cero. Una vez decidido que cierta versión del SPMP será entregado a los clientes, se le juntará la rama de desarrollo a la principal y se tratará el documento como línea base. Posterior a la retroalimentación con el cliente, el número de entrega se incrementará y los números de versión y revisión reiniciarán en cero.

**Diagramas y Documentación de especificación de casos de uso.**

Dentro de cada plantilla de caso de uso estará el indicativo de su respectiva versión, que coincidirá con el número de etiqueta correspondiente. Cada cambio que se realice en el caso de uso pasará por el equipo de DCRC y por medio de la lista de chequeo (ver Lista de Chequeo de Casos de Uso) lo aprobarán o lo devolverán para corregirlo al diseñador. Si esto ocurre, el número de revisión se incrementará en uno. Una vez que es aprobado por el equipo de DCRC, la gerencia efectuará una última revisión y finalmente lo subirá al repositorio, incrementando su número de versión. Al igual que el SPMP, luego de la retroalimentación del cliente, el número de entrega se incrementará y los números de versión y revisión reiniciarán en cero.

## Métricas y Proceso de Medición

Propósito: Que los integrantes del proyecto sepan qué se va a medir, cómo se va a medir y cómo se consolidarán las mediciones en métricas útiles para el proyecto.

Contenido:

Enumerar las métricas a utilizar (referenciando las secciones que las detallen) y relación con los planes en los que se usarán (tamaño sugerido: 1/2 página).

Descripción de los procesos para recolectar métricas. Para cada métrica: quién la recolecta, cuándo lo hace, qué herramientas utiliza; quién procesa la información recolectada y la consolida, cómo lo hace, qué herramientas utiliza; cómo se utiliza la información consolidada en el proyecto. (Tamaño sugerido: 1 página)

Referencias: [1], sección 5.8.7

## Control de Calidad

Propósito: Que los integrantes conozcan las actividades relacionadas a todos los procesos de control de calidad del proyecto.

Contenido

Explicación de alto nivel de todos los procesos asociados a control de calidad, lo cual incluye verificación, validación, revisiones y auditorías al proyecto. Por ejemplo: revisiones de calidad de documentos, pruebas unitarias, de sistema, revisiones externas, etc. Para cada proceso indicar: nombre del proceso, momentos del proyecto en que ocurren y quiénes son los responsables. Se recomienda resumir esta información en una tabla.

Describir el detalle de los procesos de control de calidad, usando BPMN y párrafos explicativos.

Tamaño sugerido: 4 páginas

Referencias: [1], secciones 5.8.8 y 5.8.9.

# Anexos

Agregue aquí cualquier información adicional relevante para el proyecto que no quepa en las secciones previas.

Es importante tener en cuenta que hasta antes de esta sección, el documento debe ser **autocontenido**, es decir, el lector no debe necesitar leer ningún documento adicional o archivo externo para entender el proyecto. La información que se coloque aquí es simplemente un complemento, en caso que el lector requiriera más detalles sobre algún tema.

**Cartas Gantt, diagramas y otros elementos similares no deben ir en esta sección, sino en el lugar que se indique dentro del documento**

# Referencias

x

|  |  |
| --- | --- |
| [1] | University of Houston. RUP - Vision Artifact (Small Projects). [Online]. <http://sce.uhcl.edu/helm/rationalunifiedprocess/webtmpl/templates/req/rup_vision_sp.htm> |
| [2] | About Tech. A List of Social Networks for Pet Lovers. [Online]. <http://webtrends.about.com/od/socialnetworks/tp/pet-social-networks.htm> |
| [3] | Robert Bogue. (2005, Abril) Use S.M.A.R.T. goals to launch management by objectives plan. [Online]. <http://www.techrepublic.com/article/use-smart-goals-to-launch-management-by-objectives-plan/> |
| [4] | Jaime Pavlich and Miguel Torres. (2015, Enero) Página de Miguel Torres -Ingenieria de Software - Programa del Curso. [Online]. <http://sophia.javeriana.edu.co/~metorres/> |
| [5] | Huffington Post. (2014, Agosto) 3 Reasons Why Social Media Age Restrictions Matter. [Online]. <http://www.huffingtonpost.com/diana-graber/3-reasons-why-social-media-age-restrictions-matter_b_5935924.html> |
| [6] | IEEE Standards Board, "Systems and software engineering — Vocabulary," IEEE Standards Board, ISO/IEC/IEEE 24765:2010, 2010. |
| [7] | Sparx Systems. Typical Project Roles. [Online]. <http://www.sparxsystems.com/enterprise_architect_user_guide/9.2/getting_started/professionalroles.html> |
| [8] | Merriam Webster. An Encyclopædia Britannica Company : Social Network. [Online]. <http://www.merriam-webster.com/dictionary/social%20network> |
| [9] | Sajid Ibrahim Hashmi and Jongmoon Baik, "Software Quality Assurance in XP and Spiral - A Comparative Study," in *Fifth International Conference on Computational Science and Applications*, Daejeon, pp. 3-5. |
| [10] | Ian Sommervile, "Ingenieria de Software," in *Ingenieria de Software*.: Pearson, 2005, ch. 4, pp. 61-69. |
| [11] | Bernd Bruegge and Allen H. Dutoit , "Ingeniería de Software orientado a objetos," in *Ingenieria de Software Orientado a Objetos*.: Prentice Hall, 2002, pp. 458-490. |
| [12] | Alistar Cockburn, "Using Both Incremental and Iterative Development," in *Using Both Incremental and Iterative Development*.: The Journal of Defense Software Engineering, Mayo 2008, pp. 28-30. |
| [13] | Craig Larman, "UML and Patterns," in *UML and Patterns*., 2004, ch. 2, pp. 13-21. |
| [14] | Rational - The Software Development Company. (1998) Rational Unified Process: Best Practices for Software Development Teams. [Online]. <https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf> |
| [15] | María de Lourdes Santiago Zaragoza. Desarrollando aplicaciones informáticas. [Online]. <http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm> |
| [16] | Eric S. Raymond. (2010, Febrero) The Cathedral and The Bazaar. [Online]. <http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/index.html#catbmain> |
| [17] | Project Management Institute. Roles, Responsibilities, And Skills In Program Management. [Online]. <http://www.pmi.org/learning/roles-responsibilities-skills-program-management-6799> |
| [18] | IEEE Standards Board, "IEEE Guide to Software Configuration Management," in *IEEE Guide to Software Configuration Management*., 1987, pp. 9-10. |
| [19] | Pete Deemer, Gabrielle Benefield, and Craig Larman, "Información Básica de SCRUM (The SCRUM Primer)," in *Información Básica de SCRUM (The SCRUM Primer)*.: Scrum Training Institute, pp. 3-6. |
| [20] | Craig Larman and Victor Basili, "Iterative and Incremental Development: A Brief Story," *IEEE Computer Society*, pp. 1-10, 2003. |

x

Indique aquí todas las referencias bibliográficas utilizadas en el documento. Utilice formato IEEE o APA para definirlas. Para administrar automáticamente las referencias, se recomienda el uso de la herramienta Zotero (www.zotero.org).

----

Referencias usadas en el formato

[1]ISO/IEC/IEEE 1074-2006 – Estándar para desarrollar un Proyecto de Software

[1]ISO/IEC/IEEE 12207-2008

[1] ISO/IEC/IEEE 16326-2009