Proyecto: ¿Adivina quién?

Marzo de 2014

V 6



# Historial de Cambios

Tabla 1.1 – Historial de Cambios

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha del cambio | Descripción cambio | Persona(s) | Sección(es) modificada(s) |
| 15/02/14 | Cambio de letra y formato | Sergio Bernal | Todas |
| 22/02/14 | Tablas, cambio formato general | Felipe Valderrama | Todas |
| 22/02/14 | Revisión y adición al documento | Sergio Bernal | 2, 6.1, 6.2, 6.3, 8.2. |
| 01/03/14 | Marca de agua y pie de pagina | Felipe Valderrama | Todas |
| 02/03/14 | Revisión y adición al documento | Sergio Bernal |  |
| 03/03/14 | Adición al documento, formato | Felipe Valderrama | 8.1, 8.2, 14 |
| 07/03/14 | Adición al documento y corrección según sugerencias del cliente | Felipe Valderrama | 8.1 |
| 07/03/14 | Adición de alternativas, corrección de tabla. | Sergio Bernal | 8.2, 8.2.1 |
| 08/03/14 | Corrección de: riesgos y administración de la configuración. Prefacio supuestos y restricciones | Sergio Bernal | 12.2,12.3 |
| 09/03/14 | Inserción de títulos y nombres de tablas y figuras | Felipe Valderrama | 8.4.2, 12 |
| 10/03/14 | Adición de nuevas secciones | Sergio Bernal | 6.6,8,3 8.4.1,9.2.1, 9.2.2,9.3.1,9.3.3,9.3.4,  11 |
| 11/03/14 | Revisión y corrección de formatos y tablas | Felipe Valderrama | Todas |
| 11/03/14 | Actualización de referencias. | Felipe Valderrama y Sergio Bernal | Todas |
| 11/03/14 | Inclusión de IDs para herramientas y lenguajes | Sergio Bernal | 8.2 |
| 12/03/14 | Adición de nuevas secciones | Sergio Bernal | 7,10.2,10.3,12.4,12.5 |
| 13/03/14 | Adición de nueva sección | Sergio Bernal | 9.1 |
| 13/03/14 | Revisión general | Sergio Bernal | Todas |
| 1/04/14 | Corrección secciones | Felipe Valderrama-Sergio Bernal | 6.4,8.3 |
| 2/04/14 | Corrección secciones | Sergio Bernal | Sección 6 |
| 22/04/14 | Correción Secciones | Felipe Valderrama | 8.1 |

# Prefacio

Este documento tiene la finalidad de explicar de una manera detallada y explícita el desarrollo y planeación de un proyecto (¿Adivina Quién?) de ingeniería de software, entendiéndose este último como la ejecución de un plan para cumplir con el diseño e implementación de una aplicación interactiva y multimedia, la cual en el presente caso ha sido definida como el famoso juego de mesa Guess Who? o ¿Adivina quién? [1], en su respectiva traducción al español. En este documento se han incluido y posteriormente detallado aspectos importantes dentro del proceso ingenieril para el desarrollo del software tales como:

* Una vista general del proyecto, que tiene como propósito facilitar el entendimiento de la razón de este proyecto.
* Definición de roles del equipo de trabajo que desarrolla el presente proyecto, los cuales permiten entender qué función tiene cada miembro del equipo y cuál es su principal tarea a desarrollar dentro del plan de trabajo.
* Definición de un modelo de ciclo de vida para un proyecto de desarrollo de software, el cual permite entender e identificar de qué manera y con qué fases fue desarrollada la aplicación.
* Un calendario que facilita la ejecución de las tareas de una manera organizada y coherente, así como permite tener una medición en tiempo del progreso del presente proyecto.
* Un glosario, para facilitar al lector el entendimiento de cualquier término con el que no esté familiarizado.
* Un plan administrativo de proyecto, el cual tiene como propósito facilitar a los integrantes del equipo de desarrollo los aspectos detallados del proyecto y su gestión en sí mismo.
* Un monitoreo y control del proyecto, el cual fue incluido con el fin de que el equipo de trabajo tenga conocimiento tanto del estado actual del proyecto como de sus avances y posibles retrasos.
* Una sección de entrega del producto, en el cual se incluyen todos los aspectos de implementación, gestión, control de calidad, mejoras, y soporte que sean necesarios para poder implementar la aplicación a la organización interesada en el desarrollo de este proyecto.
* Anexos los cuales incluyen información relevante en el desarrollo del proyecto que no haya sido incluida en este documento.

# Tabla de contenido

[1 Historial de Cambios 2](#_Toc384230496)

[2 Prefacio 4](#_Toc384230497)

[3 Tabla de contenido 5](#_Toc384230498)

[4 Lista de Figuras 7](#_Toc384230499)

[5 Lista de Tablas 8](#_Toc384230500)

[6 Vista General del Proyecto 9](#_Toc384230501)

[6.1 Visión del Producto 9](#_Toc384230502)

[6.2 Propósito, Alcance y Objetivos 9](#_Toc384230503)

[6.2.3 Objetivos 10](#_Toc384230504)

[6.3 Supuestos y Restricciones [2] 10](#_Toc384230505)

[6.4 Entregables 12](#_Toc384230506)

[6.5 Resumen de Calendarización y Presupuesto 13](#_Toc384230507)

[6.6 Evolución del Plan 13](#_Toc384230508)

[7 Glosario 15](#_Toc384230509)

[8 Contexto del proyecto 17](#_Toc384230510)

[8.1 Modelo de Ciclo de Vida 17](#_Toc384230511)

[8.1.1 Especificación Ciclo de Vida 21](#_Toc384230512)

[8.1.2 Análisis de Alternativas y Justificación 22](#_Toc384230513)

[8.2 Lenguajes y Herramientas 24](#_Toc384230514)

[8.2.1 Análisis de Alternativas y Justificación 26](#_Toc384230515)

[8.3 Plan de Aceptación del Producto 28](#_Toc384230516)

[8.4 Organización del Proyecto y Comunicación 30](#_Toc384230517)

[Interfaces Externas 30](#_Toc384230518)

[8.4.1 30](#_Toc384230519)

[8.4.2 Organigrama y Descripción de Roles 31](#_Toc384230535)

[9 Administración del Proyecto 36](#_Toc384230536)

[9.1 Métodos y Herramientas de Estimación 36](#_Toc384230537)

[9.2 Inicio del Proyecto 38](#_Toc384230538)

[9.2.1 Entrenamiento del Personal 38](#_Toc384230539)

[9.2.2 Infraestructura 38](#_Toc384230540)

[9.3 Planes de Trabajo del Proyecto 40](#_Toc384230541)

[9.3.1 Descomposición de Actividades 40](#_Toc384230542)

[9.3.2 Calendarización 41](#_Toc384230543)

[9.3.3 Asignación de Recursos 41](#_Toc384230544)

[9.3.4 Asignación de Presupuesto y Justificación 41](#_Toc384230545)

[10 Monitoreo y Control del Proyecto 44](#_Toc384230546)

[10.1 Administración de Requerimientos 44](#_Toc384230547)

[10.2 Monitoreo y Control de Progreso 45](#_Toc384230548)

[10.3 Cierre del Proyecto 46](#_Toc384230549)

[11 Entrega del Producto 48](#_Toc384230550)

[12 Procesos de Soporte 49](#_Toc384230551)

[12.1 Ambiente de Trabajo 49](#_Toc384230552)

[12.2 Análisis y Administración de Riesgos 49](#_Toc384230553)

[12.3 Administración de Configuración y Documentación 52](#_Toc384230554)

[Métricas y Proceso de Medición 56](#_Toc384230555)

[12.4 56](#_Toc384230556)

[12.5 Control de Calidad 58](#_Toc384230557)

[13 Anexos 60](#_Toc384230558)

[14 Referencias 61](#_Toc384230559)

# Lista de Figuras

[Figura 6.1 – Fechas de entregables 12](#_Toc382447395)

[Figura 6.2 – Artefactos de entregas 13](#_Toc382447396)

[Figura 9.1 – Modelo Cascada [6] 17](#_Toc382447397)

[Figura 9.2 – Modelo Incremental Iterativo [6] 18](#_Toc382447398)

[Figura 9.3 - Modelo de ciclo de vida para el proyecto ¿Adivina Quién? 19](#_Toc382447399)

[Figura 9.4 – Diagrama BPMN para el ciclo de vida 20](#_Toc382447400)

[Figura 9.5 – Diagrama de secuencia del proyecto 21](#_Toc382447401)

[Figura 13.1 – Proceso de identificación de riesgos 51](#_Toc382447402)

[Figura 13.2 – Proceso de cambio de documentos 53](#_Toc382447403)

# Lista de Tablas

[Tabla 1.1 – Historial de Cambios 2](#_Toc382447404)

[Tabla 6.1 - Entregables 12](#_Toc382447405)

[Tabla 8.1 – Tabla de lenguajes de programación y herramientas 24](#_Toc382447406)

[Tabla 8.2 - Plan de aceptación del producto 28](#_Toc382447407)

[Tabla 8.3 – Interfaces externas 30](#_Toc382447408)

[Tabla 8.8.4 – Actividades y metas del Jefe de desarrollo 31](#_Toc382447409)

[Tabla 8.8.5 - Actividades y metas del Programador 32](#_Toc382447410)

[Tabla 8.8.6 - Actividades y metas del Jefe de Arquitectura y planeación 34](#_Toc382447411)

[Tabla 8.8.7 - Actividades y metas del Arquitecto 34](#_Toc382447412)

[Tabla 11.1 – Encargados de entregar artefactos 48](#_Toc382447413)

# Vista General del Proyecto

## Visión del Producto

Se creará una aplicación que represente al juego de mesa mundialmente conocido ¿Adivina quién?, tal aplicación permitirá jugar mediante Internet, bien sea con un computador, un celular o una Tablet. Se espera mantener un modo de juego tradicional –entiéndase como la del juego de mesa- de modo que será un juego para dos, con las reglas normales del ¿Adivina quién? (ver [reglas](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Reglamento-AdivinaQuien.docx)).

Adicionalmente los jugadores tendrán la posibilidad de jugar con diferentes categorías(Jugadores de futbol, Los Simpson y actores o actrices de cine) para sus tableros, así como tener un registro de sus victorias, o derrotas, que estarán dentro de un perfil único para cada jugador.

Uno de los beneficios del producto será crear la necesidad de interactuar con los jugadores, así como el fomento del análisis, para lograr buenas preguntas; y también remembrar la infancia con el juego en sí.

## Propósito, Alcance y Objetivos

**6.2.1 Propósito**

Crear una aplicación web(HTML5) para el juego ¿Adivina Quién? ,mediante un proceso de desarrollo de software sustentado por el conocimiento adquirido a lo largo del curso de ingeniería de software, y la investigación, de modo que al final se obtenga un producto de calidad, dentro de los estándares solicitados por el cliente (el profesor).

**6.2.2 Alcance**

El proyecto pretende llegar a una aplicación web desarrollada en HTML5 donde se simulará el popular juego de mesa ¿Adivina Quién?, con las siguientes funcionalidades principales:

* Una pantalla de inicio que permita el manejo de perfiles de usuario con estadísticas(partidas ganadas, perdidas y partidas totales)
* Un pool de mesas para la interacción entre jugadores y selección de partidas donde se podrá elegir entre diferentes categorías.
* Jugar con otros jugadores en las diferentes categorías.

### Objetivos

Objetivo general:

Construir una aplicación que represente al juego ¿Adivina quién?, siguiendo un plan (este documento) junto con la especificación de requerimientos (SRS) y el diseño del sistema (SDD).

Objetivos específicos del proyecto:

* Obtener nuevos conocimientos en el área de ingeniería de software.
* Desarrollar la aplicación siguiendo lo establecido en la sección de calidad 12.5, tanto para integrantes del grupo como para el cliente.
* Lograr éxito en nuestra calificación, una nota definitiva superior a 4,0.

## Supuestos y Restricciones [2]

**Supuestos:**

* El cliente/profesor (Ingeniero Miguel Torres) es el único que podrá modificar las fechas de entregas.
* Los prototipos que se presenten al cliente deberán tener la capacidad de ejecutarse en las salas de computadores de la Pontificia Universidad Javeriana.
* Una vez definidos y aprobados por el cliente los requisitos del sistema, no se verán sujetos a cambios por ninguna de las partes (cliente, Grupo SoftLute).
* No se subcontratara personal que tenga que ver con el proceso de desarrollo de software, programadores, diseñadores, arquitectos; pero para áreas de diseño gráfico podrán ser empleados.
* Cada entrega del proyecto deberá estar acompañada de la documentación exigida por el cliente para cada entrega; y deberá ser entrega en un medio digital confiable, muy seguro, ya que si este se daña se evaluara como 0.0.
* El producto final deberá estar cumplir con al menos un 70% de los requerimientos.
* Los roles de trabajo deberán estar bien definidos
* Se deberán utilizar rubricas y estándares entregados por el cliente.(SPMP, SRS, SDD, IEEE 16326, IEEE 29148)

**Restricciones:**

* Desarrollo del proyecto en el tiempo que dura el curso
* No uso de más del 20% de código ajeno a la propiedad intelectual del grupo de trabajo
* Grupos de trabajo de máximo 7 personas
* Proyecto seleccionado bajo supervisión del cliente.
* Uso de persistencia
* Uso de cliente/servidor
* Manejo de interfaz gráfica fuerte
* Tiempo límite para cada entrega (ver 9.3.2)
* El proyecto está sujeto a la aceptación del cliente
* Entregas en medio digital.
* Uso de fuente bibliográficas aprobadas.
* Entrega de diseño y documentación, código y documentación, uso de estándares de programación, ejecutable y manuales.
* Uso de plantillas especificadas para la documentación ver secciones(12.5, 12.3)

## Entregables

La siguiente tabla permite ver cuáles son los principales entregables del proyecto y a quien van dirigidos.

Tabla 6.1 - Entregables

|  |  |  |
| --- | --- | --- |
| Artefacto | Destinatario | Entrega Asociada |
| SPMP | Cliente y Grupo de trabajo | Entrega 1 |
| SRS | Cliente y Grupo de trabajo | Entrega 2 |
| SDD | Cliente y Grupo de trabajo | Entrega 3 |
| Task Slips | Grupo de trabajo | Entregas 1 y 2 |
| Manual de Usuario | Cliente | Entregas 2 y 3 |
| Reporte de pruebas | Grupo de trabajo y cliente | Entregas 2 y 3 |
| Prototipos | Cliente | Entregas 2 y 3 |
| Reporte Gerencial | Grupo de trabajo y cliente | Entregas 1, 2 y 3 |

En el siguiente diagrama podemos ver las fechas de las entregas y su asociación con las iteraciones y fases que se van a realizar en el proyecto según el modelo de ciclo de vida en la sección 8.1.

Figura 6.1 – Fechas de entregables

*Muestra las iteraciones asociadas a cada una de las entregas y las fechas programadas para estas en el cronograma (Ver sección 9.3.2)*

En la sección 8.3 se puede ver a que iteración y entrega está asociada cada entregable y se especifican las condiciones que de cumplir cada entregable.

## Resumen de Calendarización y Presupuesto

Durante el proyecto ¿Adivina Quién? Se tienen 3 hitos que están asociados con 3 entregas al cliente las cuales se pueden ver en la sección anterior (6.4):

Figura 6.2 – Artefactos de entregas

El presupuesto del proyecto se hizo tomando el valor de la primera entrega de acuerdo con las actividades realizadas y un estimado para las 2 entregas posteriores basado en las horas disponibles del grupo de trabajo (ver sección 9.3.4).Se estimó un valor de **$6’876.000**.

*Se muestra los documentos y productos que debe incluir cada entrega.*

## Evolución del Plan

A continuación se describe el proceso para hacer cambios a este documento (SPMP), así como los responsables de hacer estos cambios y la plantilla que se debe utilizar para asegurar un orden correcto y consistencia. Adicionalmente se describe el proceso para asegurar que los cambios a este documento (plan), se controlen y se lleven a cabo.

**Proceso para cambios al plan (documento):**

**Responsable:**

* Supervisor y analista. (Para ver detalles del rol ver 8.4.2).

**Proceso:**

* El integrante del equipo realiza una tarea asignada y sube el documento a la carpeta de versiones.
* El supervisor y analista integra el aporte del integrante del grupo de trabajo al SPMP.
* El administrador de la configuración actualiza versión del SPMP.
* Al final de cada Sprint se realiza control de calidad y si hay partes que no cumplen con calidad 12.5, se solicita al encargado (responsable de esa sección) que haga los cambios pertinentes, en caso contrario se da la tarea por concluida y se procede a una siguiente tarea.

Para ver el proceso detallado ver diagramas BPMN de la sección 12.3.

**Plantilla:**

El grupo Softlute definió una plantilla en la cual el integrante del equipo realiza su tarea y sube el documento a la carpeta de versiones de acuerdo al proceso especificado en la sección12.3. El uso de la plantilla se especifica en control de calidad (ver ). (Ver [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx) )

**Proceso de control y ejecución de cambios:**

**Responsable:**

* Jefe de arquitectura y planeación (para ver detalles del rol ver 8.4.2).

**Proceso:**

* Al final de cada Sprint, y con los cambios hechos al documento por parte del integrante, el jefe de arquitectura y planeación revisa que estos cambios sean correctos y consistentes al proyecto.
* En caso que los cambios se acordes al proyecto, el jefe de arquitectura y planeación actualiza el calendario en caso de ser necesario y planea el siguiente Sprint.
* El jefe de arquitectura y planeación informa a las áreas involucradas, el cambio que se hizo al plan para que estas modifiquen lo necesario para ajustarse al cambio en un plazo no máximo a una semana.
* Al final del sprint, el jefe de arquitectura y planeación revisa que el cambio que se hizo al inicio del sprint, haya sido aplicado correctamente.
* Se llena el documento de control de cambios en el cual se especifica:
  + Cambio realizado
  + Áreas implicadas
  + Fechas:
    - Cambio fue realizado al documento.
    - Cambio es informado a áreas implicadas.
    - Cambio aplicado correctamente al proyecto.
  + Observaciones.

# Glosario

SPMP: Software Project Management Plan por sus siglas en ingles. Es un documento donde se encuentra el plan de gestión del proyecto [3].

SRS: Software Requeriments Specification, es la especificación de requerimientos software de un sistema, la cual describe el comportamiento del sistema que va a ser desarrollado y puede incluir una serie de casos de uso que describen la interacción que va a tener usuario con el sistema. También contiene los requerimientos no funcionales. Entre otras cosas.

SDD: Software Design Document, es la descripción escrita de un producto de software, la cual es escrita por el diseñador de software, y que tiene como objetivo dar al equipo de desarrollo una visión amplia de la arquitectura del producto de software.

PSP: Personal Software Process por sus siglas en inglés, es un conjunto de prácticas enfocadas a la mejora de la productividad de un proceso a través de la mejora en el desempeño del personal. [4]

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, los cuales sirven de base para la organización y desarrollo de un software.

Ofimática: Conjunto de técnicas, aplicaciones y herramientas informáticas que se utilizan en funciones de oficina para optimizar, automatizar o mejorar los procedimientos o tareas relacionadas. Estas herramientas permiten idear, crear, manipular, transmitir, almacenar o parar la información necesaria en na oficina.

Mock-ups: Es un modelo el cual es usado para mostrar a la gente como algo se va ver. En desarrollo de software es útil para mostrar al cliente como va a quedar la interfaz gráfica del software.

GNU: Es un OS desarrollado por el Proyecto GNU. Está conformado en su totalidad por software libre. Está basado en el nucleo GNU-Hurd y su objetivo es ser un sistema de software completo y compatible con Unix.

BPMN: Notación para el Modelado de Procesos de Negocio la cual tiene estandarizada su notación gráfica, y su idea principal es formar un flujo de trabajo (workflow).

LGPL: Licencia Publica General Reducida, es una licencia de software libre creada por Free Software Fundation que intenta garantizar la libertad de compartir y modificar software cubierto por ella, asegurando que el software libre sea libre para todos los usuarios.

# Contexto del proyecto

## Modelo de Ciclo de Vida

El modelo de ciclo de vida es la guía para la realización del proyecto, en este se definen las etapas y el orden en que se van a realizar, las metodologías de trabajo que se van a usar y las estrategias a utilizar. Además, sirve como base para la identificación de las principales actividades y posteriormente el cronograma.

Para el desarrollo del proyecto ¿Adivina Quién?, se tienen predefinidos 3 hitos en los que se realizara una presentación al cliente del avance del proyecto. A partir de estos, es que se realizó la selección del modelo de ciclo de vida y filosofía de trabajo.

Se tomaron 3 modelos bases de ciclo de vida para la definición del modelo que va a utilizar Softlute en el desarrollo del proyecto.

**Modelo en cascada:**

El primero modelo es el modelo en cascada, es un ciclo de vida secuencial donde el inicio de cada etapa va a depender de la finalización de la anterior [5]. Se percibe un flujo hacia abajo, debido a que no hay retroalimentación después de finalizada cada etapa o fase. Esto implica que no se hacen modificaciones sobre las fases anteriores, limitando al desarrollo del proyecto a un modelo rígido que no da lugar a cambios [6]. Para este ciclo de vida se plantearon inicialmente 5 fases:

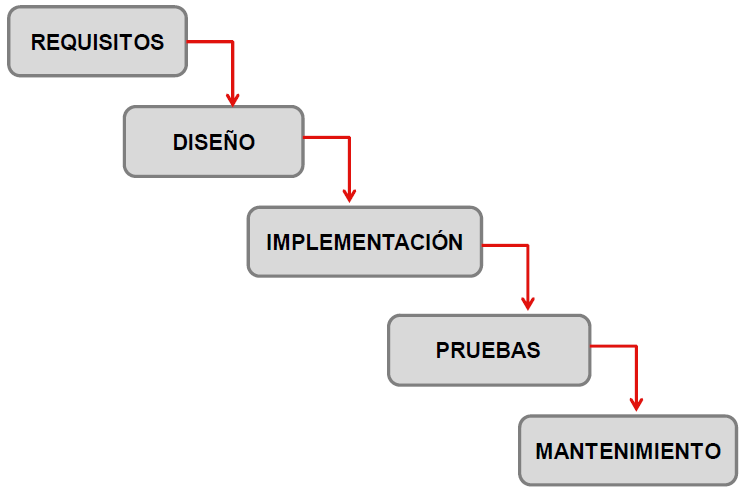


Figura 8.8.1 – Modelo Cascada [6]

*Una ilustración del modelo en cascada propuesto por Wiston W. Royce. Se ven claramente las 5 fases que definían este modelo.*

**Modelo iterativo:**

Del modelo cascada surge un modelo conocido como el modelo iterativo. Consiste en la iteración de fases repetidas veces hasta obtener un producto final. Este modelo está enfocado en el desarrollo [7]. El producto que se va mostrando al cliente va creciendo de forma incremental lo cual permite que se realicen cambios de ser necesario. Se busca estar en constante comunicación con el cliente para construir un producto más cercano a las necesidades, lo cual se logra a través de una retroalimentación realizada después de cada iteración [6].

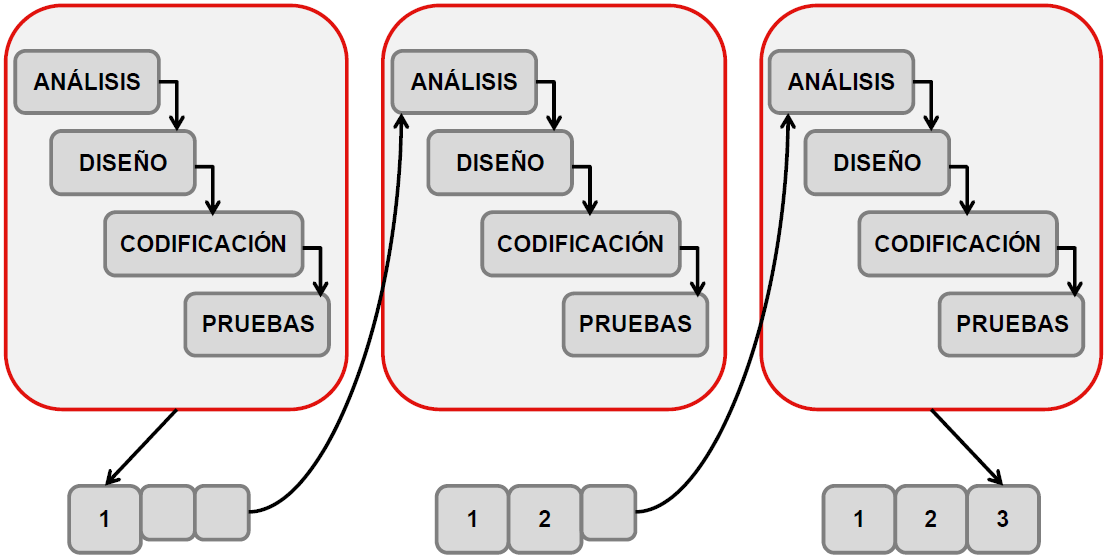


Figura 8.8.2 – Modelo Incremental Iterativo [6]

*En la figura 8.2 podemos observar un ejemplo claro de cómo es un modelo iterativo, en donde los recuadros rojos representan las iteraciones, y dentro de estos se puede apreciar un claro modelo cascada el cual es adaptado en cada incremento.*

**Modelos ágiles**

Debido al corto tiempo que se tiene para la entrega del producto se incluyeron técnicas de modelos agiles como el SCRUM y el XP.

**SCRUM:**

Este modelo toma como base el modelo iterativo y a partir de esto propone varias estrategias de trabajo. Para el proyecto *¿Adivina Quién?*, se utilizaran las siguientes.

* Product Backlog: Es una lista de responsabilidades en donde se priorizan las distintas características a desarrollar.
* Iteraciones de 1-4 semanas: Se definen las tareas a desarrollar durante el incremento
* Reuniones Semanales: En las reuniones semanales se plantea cuales tareas se desarrollaran en el siguiente incremento y también se discute sobre el avance del incremento actual y hacer la retroalimentación necesaria.

**Extreme Programming (XP):**

Un modelo ágil enfocado al trabajo en equipo y a una comunicación constante con el cliente [8] [9]. Propone técnicas que definen la modalidad de trabajo, de las cuales se han elegido 3 de 12 existentes [10] para el uso durante el proyecto:

* La escritura de código por parejas. (Esto para garantizar la calidad del código como se pude ver en la sección )
* Priorización de tareas
* Pruebas por unidad o funcionalidad.

Softlute, tomando los modelos anteriormente presentados, ha desarrollado un ciclo de vida con 4 fases principales presentadas a continuación junto a las principales actividades que se llevarán a cabo durante estas:

* Inicio del Proyecto: En esta primera fase, una vez asignado el proyecto, se definen reglas y modalidades de trabajo. Además, se preparan los documentos necesarios para poder iniciar el seguimiento del proyecto (Actas de reunión, plantilla SPMP y para definición de casos de uso). Se definen los roles que va a cumplir cada integrante de un grupo de trabajo.
* Planeación y Análisis: Se analizan los requerimientos y se definen las fechas claves y entregables del proyecto. Se redactan los documentos de planeación como el SPMP, se define el ciclo de vida que se va a tomar como base para la creación del calendario. Se realiza el Análisis de riesgos.
* Desarrollo: Se hace la implementación de los requerimientos. Es decir se realiza la programación de las soluciones planteadas en la fase de planeación.
* Fin del proyecto: Se hace la integración de artefactos, pruebas finales y la entrega final al cliente.

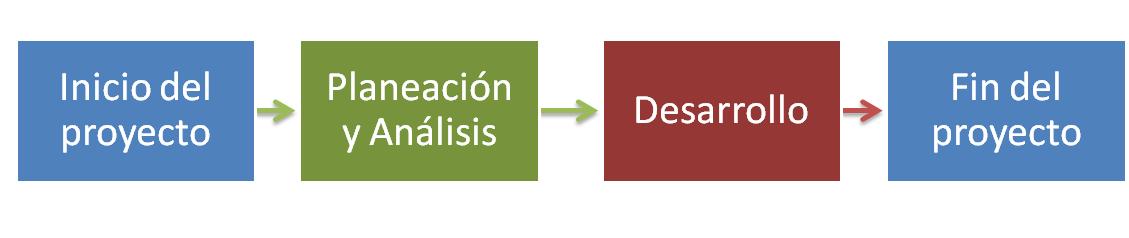


Figura 8.8.3 - Modelo de ciclo de vida para el proyecto ¿Adivina Quién?

Las fases de Planeación y análisis junto con la fase de desarrollo se van a trabajar como en el modelo de Scrum, utilizando la metodología de sprints.

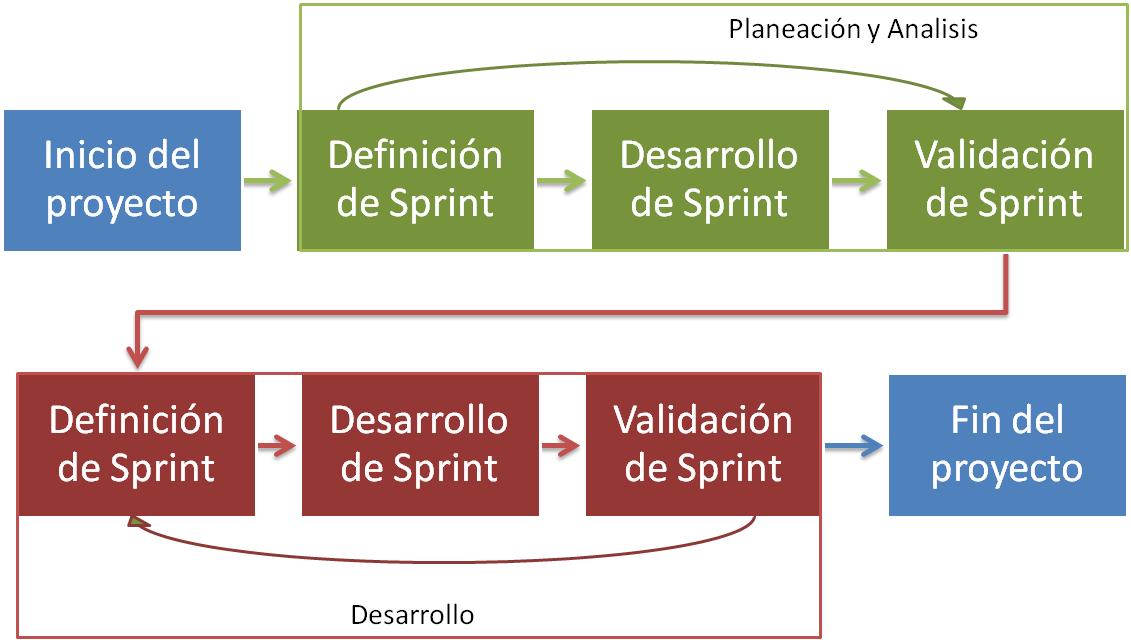


Figura 8.4 -Modelo de ciclo de vida con Iteraciones detalladas

* Definición del sprint: Se asignan las tareas que se van a realizar durante el sprint. Se consignan en el Sprint Backlog.
* Desarrollo de sprint: Se hace el desarrollo de las tareas asignadas.
* Validación de sprint: Dentro de esta fase se realizan pruebas en caso de que haya habido desarrollo o revisiones finales en caso de que solo se tengan artefactos de planeación.

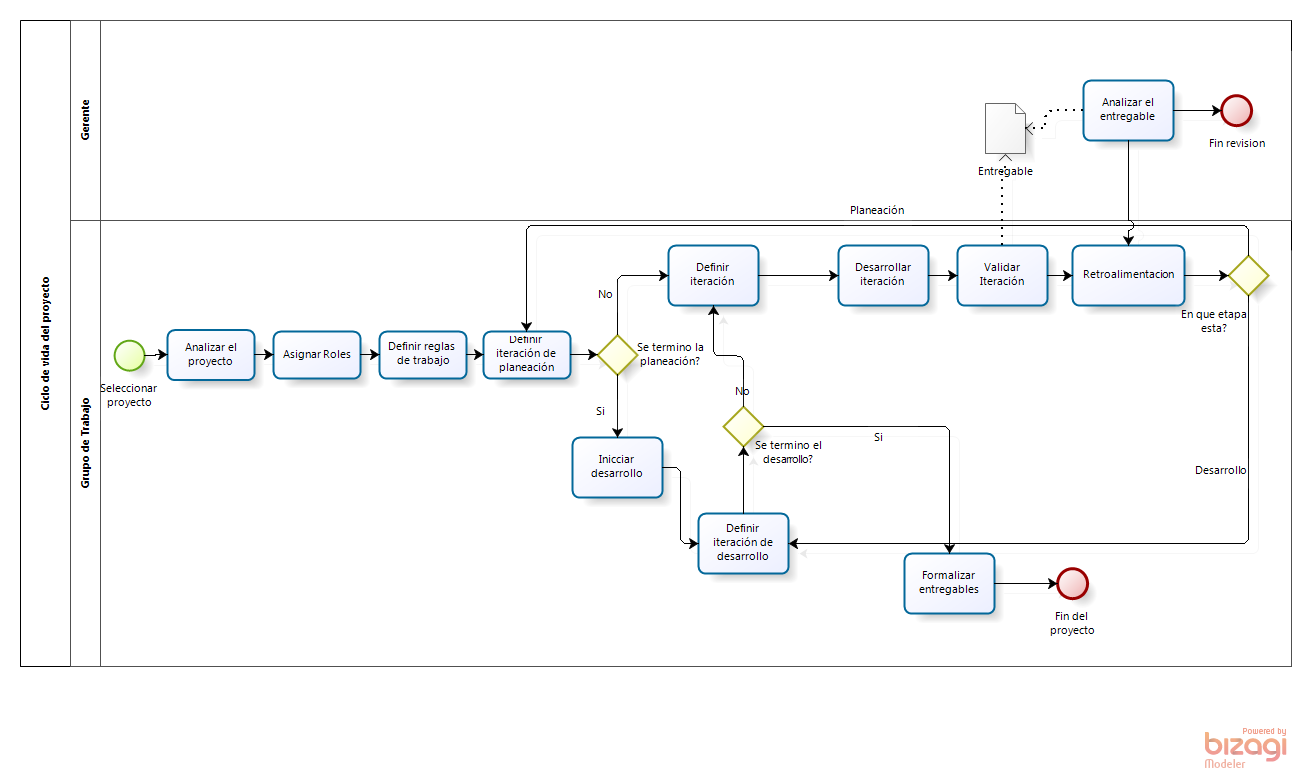
En la validación de sprint, el equipo de trabajo debe entregar todo el material que se haya modificado o creado, para que sea analizado por la gerencia posteriormente. Este diagrama está definido formalmente en él modelo BPMN que se puede apreciar en la . 

Figura 8.8.5 – Diagrama BPMN para el ciclo de vida

Para el proyecto adivina quien se consideró que son necesarias 2 sprints para la fase de planeación y 4 para la fase de desarrollo.

*Se puede apreciar claramente el proceso y la conexión entre las fases. Se presentan los documentos que son requeridos, los involucrados en la creación o revisión de los mismos.*

### Especificación Ciclo de Vida

Dentro del modelo propuesto por Softlute, los modelos de ciclo de vida base fueron el modelo cascada, iterativo, Scrum y XP

Las prácticas que aplicaron al modelo de Softlute utilizadas en Cascada son:

* La planeación de fases de forma secuencial.
* Para poder seguir a la siguiente fase se debe terminar a cabalidad las distintas tareas que representa la fase que se está desarrollando.

En el modelo iterativo:

* El desarrollo del producto se va realizando de forma incremental, esto responde también a los entregables que debe presentar Softlute (ver sección 6.4).
* Retroalimentación después de cada incremento.

**Modelos Agiles (Scrum y XP):**

* Sprints, para el trabajo de la planeación, análisis y desarrollo.
* Comunicación continua
* Reuniones Semanales
* Lista de tareas
* Planificación por entregas, esto quiere decir que se desarrollan las funcionalidades que necesita el cliente según las fechas de los entregables.
* Iteraciones de 2-3 semanas
* Programación en parejas, que en el caso de Softlute no solo será en la parte de la programación sino también en la parte de diseño y documentación.

### Análisis de Alternativas y Justificación

Previamente a la elección de los ciclos de vida utilizados como base para la creación del modelo de este proyecto, se hizo una investigación acerca de estos. El grupo de trabajo se encontró con varias alternativas que no se adaptaban a las necesidades del proyecto a realizar pero que presentaban algunas prácticas o técnicas que podían ser tenidas en cuenta.

El equipo se encontró con la categoría de Modelos Especializados de Proceso dentro del cual se encuentra el modelo de métodos formales el cual comprende un conjunto de actividades que conducen a la especificación matemática del software, esto significa que un ingeniero de software pueda especificar, desarrollar y verificar un sistema basado en una notación matemática [11] [5]. El problema con este método es que es demasiado caro y requiere de una gran inversión de tiempo, además los integrantes de un grupo que aplique este método deben tener una capacitación detallada, por esta razón este método no fue elegido.

Un caso parecido fue el del modelo en cascada, que se mencionó en el numeral anterior. El modelo sirvió de base para plantear unas fases en primera instancia, pero el modelo completo no era a fin con el proceso requerido por el proyecto de ¿Adivina Quién? Esto debido a que el modelo cascada no se enfoca en hacer retroalimentación ni en un proceso ágil.

Buscando mayor cercanía con las necesidades del proyecto, se entró a buscar modelos que tomaran como base el modelo incremental. Dentro de los modelos de proceso incremental se encuentra el DRA o desarrollo rápido de aplicaciones [11]. Un modelo llamativo debido al tiempo de trabajo que propone (60 a 90 días) [5], lo cual es cercano al tiempo disponible por el grupo de trabajo para la realización del ¿Adivina Quién? Sin embargo, su desarrollo está basado en componentes, es decir en la reutilización de componentes existentes. Vale la pena recordar que para este proyecto se exige una reutilización de código menor al 20% y también Por esta razón se rechazó este modelo DRA.

Dentro del manifestó ágil, podemos encontrar que se tiene como objetivo obtener un software funcional por encima de una documentación comprensiva. Esta afirmación lleva a pensar en la necesidad de buscar no modelos completos para aplicar al proyecto si no prácticas y técnicas de ellos y adaptarlos. El objetivo debía ser lograr un modelo de ciclo de vida que se acomodara a la necesidad de tener ciertos entregables y además permitiera trabajar de forma rápida y eficiente.

Las primeras técnicas se encontraron en el SCRUM. Se propone un trabajo por sprints, que podría considerarse como la misma iteración; Que, como se exponía en el numeral anterior, no era modificable una vez iniciado. Esto con el propósito de mantener el grupo de trabajo enfocado [12] [11]. También por temas de disponibilidad, no se pueden llevar a cabo las reuniones cortas propuestas en el SCRUM, ni entrega de versiones funcionales después de cada iteración ya que existen iteraciones que no van a generar uno.

Se puede evidenciar entonces, que el modelo elegido por Softlute para el desarrollo del proyecto ¿Adivina Quién?, es una combinación de diferentes bases, fases, prácticas, enfoques y de más características adaptadas para cumplir activamente las necesidades.

## Lenguajes y Herramientas

Tabla 8.1 – Tabla de lenguajes de programación y herramientas

*Los detalles acerca de los programas, entornos de programación, repositorios, controlador de versiones y demás herramientas o lenguaje de programación que va a ser utilizado para el desarrollo del proyecto.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Nombre | Versión | Licencia | Descripción | Usuario(s) | Link |
| H1 | Adobe Photoshop | CS6 | Paga | Es un software de procesamiento digital de imágenes. [13] | Diseñador | http://www.photoshop.com/ |
| H2 | Dropbox | 2.4.11 | Gratuita | Es un servicio de alojamiento de archivos multiplataforma en la nube. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros. [14] | Todos los roles. | https://www.dropbox.com/install |
| H3 | GitHub | 1.2.11 | Gratuita | Es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Una forja es una plataforma de desarrollo cooperativo, donde múltiples desarrolladores pueden manipular una pieza de código al mismo tiempo. [15] | Programador. | https://github-windows.s3.amazonaws.com/GitHubSetup.exe |
| L1 | HTML | 5 | Gratuita | Es un lenguaje sobre el cual se escriben las páginas Web, y se caracteriza por ser un lenguaje de etiquetas. HTML se compone también de CSS(Estilo) y JavaScript(Funcionalidad) que van embebidos dentro del archivo HTML. [16] | Programador. | http://www.w3.org/TR/html5/ |
| H4 | Microsoft Office | 2014 | Paga | Una suite de oficina que abarca e interrelaciona aplicaciones de escritorio, servidores y servicios para sistemas operativos.  Microsoft Word es el procesador de texto de la suite, Microsoft Excel es el programa de hoja o planilla de cálculo y Microsoft Project es el software de administración de proyectos. [17] | Microsoft Word: Todos los roles; Microsoft Excel: todos los roles; Microsoft Project: Jefes del proyecto. | http://office.microsoft.com/es-es/ |
| H5 | NetBeans | 7.3.1 | Gratuita | Es un IDE el cual facilita el desarrollo de aplicaciones Java, web, HTML5 y PHP. [18] | Programador. | https://netbeans.org/downloads/ |
| H6 | Pencil | 2.0.5 | Gratuita | Es un software de creación de mock-ups. [19] | Diseñador | http://evoluspencil.googlecode.com/files/Pencil-2.0.5.win32.installer.exe |
| L2 | PHP | 5.5.9 | Gratuita | Es un lenguaje de programación el cual está especialmente pensado el desarrollo web y puede ser embebido dentro de páginas HTML, este lenguaje entre otras características tiene la posibilidad de tener una conexión directa con servidores. [20] | Programador. |  |
| H7 | Skype | 6.11.0.102 | Gratuita | Es un software que permite conversaciones entre 2 o más personas. Se utiliza Skype para hacer llamadas de vídeo y voz gratuitas, enviar mensajes instantáneos y compartir archivos con otros usuarios de Skype. [21] | Todos los roles. | http://www.skype.com/en/download-skype/skype-for-computer/ |
| H8 | Sprintometer | 6.50 | Gratuita | Es una herramienta para la gestión y seguimiento de proyectos Scrum y XP. [22] | Jefes, programador, analista. | http://sprintometer.com/downloads/sprintometer.zip |
| H9 | XAMPP | 1.8.3 | Gratuita | Es una aplicación la cual facilita la instalación de un servidor web Apache, esta posee un manejo de base de datos MYSQL y el uso de scripts PHP y Perl. [23] | Programador. | http://downloads.sourceforge.net/project/xampp/XAMPP%20Windows/1.8.3/xampp-win32-1.8.3-3-VC11-installer.exe |
| H10 | Bizagi process modeler | 2.6 | Gratuita | Es un software de modelado de procesos. Utiliza la notación BPMN para diseñar los distintos procesos de una organización. [24] | Jefes, analistas. | http://www.bizagi.com/index.php/productos/bizagi-process-modeler#freeware |

### Análisis de Alternativas y Justificación

De las herramientas mencionadas anteriormente, se realizó una investigación de las distintas opciones con respecto a las herramientas a implementar. Esta investigación arrojó como resultados los siguientes programas:

**Desarrollo:**

* Django: Es un framework de desarrollo web basado en Python, este programa tiene un alto grado de abstracción de patrones comunes de desarrollo y provee distintas plantillas sobre la cual se desarrollará la página web sin necesidad de empezar desde 0 [25] [26].
* ActionScript 3(Adobe Flash): Es una plataforma orientada a objetos, a diferencia de otras versiones pasadas, ActionScript 3 no es tan orientado a eventos y de esta manera se puede tener una manipulación mayor del código. En la parte de entorno de desarrollo ActionScript soporta: FlashIDE, FlashDevelop, Eclipse y Flex Builder [13].
* Dreamweaver CC: Es un editor profesional orientado al desarrollo de páginas Web donde se puede manejar distintas plataformas como HTLM, CSS y JavaScript y tiene una compatibilidad con JQUERY para crear diseños en plataformas móviles.

**Organización:**

* TortoiseCVS: Es una herramienta la cual está basada en el sistema concurrente de versiones (CVS) y en donde se puede manejar distintos módulos, actualizar y hacer cambios a los distintos documentos que se manejan en un proyecto, este programa permite ver distintas versiones de un archivo y de esta manera tener pleno control de los archivos y ver el cambio que han tenido. [27]

**Ofimática:**

* LibreOffice: Es una herramienta gratuita de ofimática, la cual presenta toda una suite de herramientas de producción de texto, creación de hojas de cálculo, gestor de bases de datos, creación de diagramas y gráficos, editor de ecuaciones y producción de presentaciones; posee compatibilidad con varios formatos de otras herramientas, licenciamiento público LGPL donde se puede usar, personalizar y modificar el software. [28]

**Diseño visual:**

* Mockingbird: Es una herramienta online enfocada al desarrollo de mock-ups, posee un módulo de colaboración en tiempo real. [29]
* GIMP: es una herramienta enfocada a procesamiento de imágenes, con soporte de hardware, licencia publica general GNU. [30]

**Gestión de procesos:**

* Bonita BPM: Es una herramienta para modelado de procesos en formato BPMN, de código abierto completo bajo los términos de la licencia GPL. [31]

Habiendo analizado estas distintas herramientas, en la parte de desarrollo se decidió optar por HTML5, PHP y XAMPP (Véase Tabla 8.1) ya que los integrantes del grupo tienen conocimiento previo en distintas áreas y lenguajes que involucran estos programas; en la parte de organización, GitHub es un programa el cual presenta mayor afinidad en la parte de código, contando también que el grupo de trabajo tiene experiencia previa usando GitHub; en la parte de ofimática se decidió el uso de la suite de Microsoft debido a la familiarización que se tiene al ambiente de Office y a la poderosa herramienta que es Project para la planeación y calendarización, en diseño visual se medirá el desempeño de Pencil; debido a que Mockingbird presenta un entorno colaborativo, resolviendo consigo algunos contratiempos de las correcciones posteriores a las entregas al jefe; Photoshop presenta al diseñador la posibilidad de manejar una herramienta conocida, con un rango de herramientas y tutoriales mucho mayor a GIMP; finalmente el uso de Bizagi se consideró también por la experiencia de los miembros del equipo en esta plataforma de modelado de procesos.

## Plan de Aceptación del Producto

Softlute ha acordado con el cliente una serie de condiciones que debe cumplir cada entregable para que sea aceptado a satisfacción por el cliente. En la siguiente tabla podemos ver claramente estas condiciones con su respectiva entrega.

Tabla 8.2 - Plan de aceptación del producto

*Se especifican los artefactos requeridos para cada entrega y las condiciones que deben cumplir para ser aceptados por el cliente*

|  |  |  |
| --- | --- | --- |
|  | Entregables | Condiciones de entrega |
| Entrega #1  (Iteración 1 y 2) | SPMP | Cumple los parámetros de calidad especificados en la rúbrica [32]. |
| Task Slips | Se definen 40% de los Task slips. Cada Task Slip contiene todos los campos requeridos en la [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Task slips.xlsx) definida para la especificación de estos. |
| Reporte Gerencial | Resumen del trabajo realizado para la entrega. Aspectos destacables del trabajo (Positivos y Negativos). Burndown chart. |
| Entrega #2  (Iteración 3 y 4) | SPMP Corregido | Incluye el contenido corregido de acuerdo a las sugerencias del cliente. |
| SRS | Cumple con el estándar de calidad. |
| Primer prototipo | Implementación del Task Slip con la mayor prioridad. Debe incluir diseño y código ambos documentados, ejecutables y manuales. |
| Reporte Gerencial | Resume de manera completa y clara el trabajo realizado para la entrega correspondiente. |
| Entrega #3  (Iteración 3 y 5) | SPMP Corregido | Modificado según la revisión hecha por el cliente. Mantiene el estándar de calidad. |
| SRS Corregido | Modificado según la revisión hecha por el cliente. Mantiene el estándar de calidad. |
| SDD | Cumple con el estándar de calidad. |
| Segundo prototipo | Implementa 70% de los Task Slips. Debe incluir diseño y código ambos documentados, ejecutables y manuales. |
| Reporte de pruebas | Resume de manera clara los resultados de las pruebas. |
| Manual de usuario | Explica de manera clara y concisa el funcionamiento de la aplicación. |
| Reporte gerencial | Resume de manera completa y clara el trabajo realizado para el proyecto. |

El plan de control de calidad donde se explica cómo se va a garantizar el cumplimiento de las condiciones de entrega y que herramientas se van a utilizar se encuentra en la sección 12.5.

## Organización del Proyecto y Comunicación

### Interfaces Externas

A continuación se muestra la tabla de entidades externas al equipo de desarrollo Softlute, la cual describe a la entidad, su relación con el proyecto y el canal de comunicación con ella.

Tabla 8.3 – Interfaces externas

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Descripción | Responsabilidad | Canal de comunicación |
| Ingeniero Miguel Torres | Profesor de planta PUJ, Ingeniería de sistemas. Materia de ingeniería de software 2014-1 | Cliente-Auditor-Asesor | -Correo electrónico institucional.  -Reuniones programadas con actas de reunión.  -En clase(sesiones de retroalimentación) |
| Biblioteca Alfonso Borrero C. PUJ | Lugar oficial y principal de reunión (salas de estudio) de Softlute. Lugar de consulta de información (libros) y préstamo de equipos (PCs). | N.A. | -Carné de la PUJ validado con el sticker amarillo.  -Por medio de los funcionarios de la biblioteca. |
| Salas de computo Ingeniería de Sistemas PUJ | En estas se evaluará el producto por lo tanto será lugar de desarrollo y de pruebas del producto. | N.A. | -Carné de la PUJ validado con el sticker amarillo.  -En caso de necesitar utilizar la sala en horarios que no esté libre se necesitará hacer una reservación al encargado de la sección de ofimática [34] de Ingeniería de sistemas |



### Organigrama y Descripción de Roles

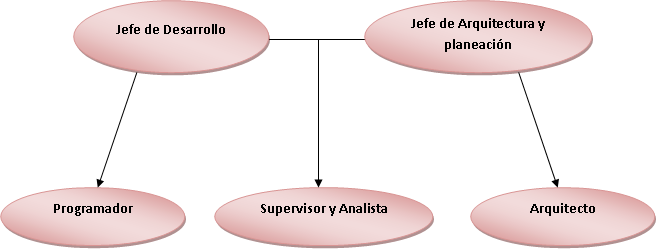


Figura 8.8.6 - Organigrama Softlute

A continuación se describe cada rol de Softlute, los cuales fueron basados en los conceptos del capítulo 4 del documento Apuntes de Taller de Ingeniería de Software [35].

**Jefe de Desarrollo**

Tabla 8.4 - Actividades y metas del Jefe de desarrollo

|  |  |
| --- | --- |
| Actividad | Meta |
| Coordinar el equipo de desarrollo para garantizar el cumplimiento de las actividades y objetivos del calendario | Cumplir efectivamente y de manera correcta las actividades propuestas en el calendario |
| Mantener constante comunicación con los supervisores para estar al tanto del desarrollo general del proyecto | Lograr un buen flujo de información entre los diferentes miembros del equipo de trabajo |
| Guiar y ayudar a los programadores en su proceso de desarrollo | Garantizar el buen funcionamiento y correcto desarrollo del software |

Se desempeñara como el líder en el desarrollo del software del proyecto, buscando lograr la coherencia entre lo que se diseña y lo que se implementa-realiza. Para esto definirá las herramientas apropiadas para el desarrollo. Debe también comunicar dudas del equipo de desarrollo a los analistas y arquitectos, así como aportar su visión de que puede realizarse dentro del calendario y recursos.

**Planes en los que participa:**

**-**Análisis y administración de riegos

-Métricas y procesos de medición

-Administración de riegos

-Administración de la configuración y documentación

-Planes de trabajo

-Evolución del plan

***Relación con otros roles:***

Supervisor y Analista: El jefe de desarrollo es quien deberá comunicarle las dudas al supervisor y analista respecto al diseño y la arquitectura del software, así como posibles sugerencias o limitaciones en el desarrollo de este.

Programador: El jefe de desarrollo deberá informar, asignar, asistir y verificar el cumplimiento de las actividades y tareas de desarrollo del software en cuanto a código, poniendo especial énfasis en el cumplimiento del calendario y de la coherencia entre código y diseño del software. Esta revisión se hace por medio de un checklist y se complementa la especificación de requerimientos en la pestaña de trazabilidad.

**Programador**

Tabla 8.5 - Actividades y metas del Programador

|  |  |
| --- | --- |
| Actividad | Meta |
| Implementar efectivamente en un lenguaje de programación las funcionalidades del juego | Desarrollar con éxito un prototipo para el cliente |
| Documentar correctamente las funcionalidades de los comandos implementados | Garantizar la coherencia y la cohesión entre el diseño de software y su implementación |

El (los) programador(es) se encargaran de desarrollar el software, transformando la visión del área de arquitectura en un programa ejecutable. Deberá tener buenas prácticas de programación, documentación del código hecho además de contar con calidad (ver calidad 12.5)

Las modificaciones al código serán solicitadas por el jefe de desarrollo o por el jefe de arquitectura y planeación mediante una solicitud formal en un documento. De ninguna manera el cliente debería solicitar cambios al software.

El desarrollo se realizará en parejas, donde un programador revisara el código del otro, para garantizar su calidad y consistencia con el diseño; buscando errores de programación o inconsistencia con el diseño los cuales serán reportados al jefe de programación.

***Relación con otros roles:***

Jefe de desarrollo: El programador deberá mantener comunicación constante con el jefe de desarrollo, para conocer actividades por realizar, estado del proyecto, dudas acerca del proyecto en cuanto diseño y la arquitectura. También podrá solicitar cambios al diseño o arquitectura del proyecto por medio del Jefe de desarrollo.

Supervisor y Analista: El programador debe entregar y recibir la última versión del diseño, así como del código, y en algunos casos de ser necesario versiones antiguas.

**Supervisor y analista**

Tabla 8.6 – Actividades y metas del supervisor y analista

|  |  |
| --- | --- |
| Actividad | Meta |
| Revisar que los documentos y el software cumpla correctamente con los estándares de calidad | Garantizar la calidad de la documentación según lo definido en la sección 12.5. Esto con el objetivo de que los artefactos cumplan el plan de aceptación (ver sección 8.3). |
| Revisar que la arquitectura sea coherente y cohesiva con el respectivo desarrollo | Lograr un proceso de diseño correcto con el respectivo software |
| Definir correctamente la configuración general de todos los artefactos necesarios para el proyecto | Facilitar el desarrollo y diseño del software |
| Verificar si los requerimientos son adecuados | Evitar errores en el diseño/construcción del sistema de manera temprana |

El supervisor y analista es el encargado de buscar las necesidades del cliente, así como lo que espera del proyecto, mediante reuniones con este, de modo que defina posteriormente los requisitos del proyecto, primero los de usuario y luego los de software.

Además tendrá(n) la responsabilidad de revisar si la documentación y el software son de la calidad esperada por el cliente.

También actuará como puente entre el área de diseño y desarrollo – por lo cual no solo será una persona en este proyecto en particular- de modo que deberá mantener constante comunicación con las dos áreas para informar errores, dudas o sugerencias de un área hacia otra.

***Relación con otros roles:***

Jefe de desarrollo: Recibirá dudas, problemas, sugerencias surgidas en el proceso de desarrollo, en cuanto a diseño se trata, buscando además que el código desarrollado sea coherente a el diseño generado por el área de planeación y arquitectura.

Jefe de Arquitectura y planeación: Informara inquietudes en cuanto al diseño por parte de los programadores al área de arquitectura, así como deberá comunicar y ayudar a los arquitectos en la comprensión de los requerimientos.

**Jefe de arquitectura y planeación**

Tabla 8.7 - Actividades y metas del Jefe de Arquitectura y planeación

|  |  |
| --- | --- |
| Actividad | Meta |
| Liderar correctamente al equipo de arquitectura y planeación | Lograr un flujo correcto de información manteniendo la armonía entre los diferentes miembros del equipo |
| Definir y realizar correctamente el proceso de definición y seguimiento de requerimientos | Entender de manera adecuada lo que el cliente está solicitando y de qué manera lo requiere |

El jefe de arquitectura deberá generar el diseño del sistema, arquitectónico y detallado, basándose en los requisitos, con la meta de lograr que este cumpla con las limitaciones del proyecto. Deberá elegir además una metodología adecuado para tal diseño ya que este definirá las formas de trabajo de las demás áreas; y en gran medida el tiempo y uso de recursos, así como la calendarización.

**Planes en los que participa:**

**-**Análisis y administración de riegos

-Métricas y procesos de medición

-Administración de riegos

-Control de calidad

-Planes de trabajo(9.3)

-Evolución del plan

***Relación con otros roles:***

Arquitectos: Deberá resolver las inquietudes generadas en el equipo de planeación y de arquitectos en cuanto a la definición de los requerimientos, así como su modelado en la arquitectura del software;

Supervisor y Analista: Este le informara dudas que vengan del área de desarrollo, normalmente serán acerca de la arquitectura, así como también revisará que los requerimientos entregados se ajusten a las necesidades del proyecto.

**Arquitecto**

Tabla 8.8 - Actividades y metas del Arquitecto

|  |  |
| --- | --- |
| Actividad | Meta |
| Definir correctamente los casos de uso del software | Lograr cohesión y coherencia entre el diseño y la implementación |
| Redactar y documentar correctamente los requerimientos de software | Lograr cohesión y coherencia entre el diseño y la implementación |

El arquitecto deberá generar los requerimientos adecuados a los recursos con los que se cuenta, en especial el tiempo, una vez aprobados en la especificación de requerimientos y validación de requerimientos (SRS) estos deberán diseñar la arquitectura del software, acorde con los requerimientos y las restricciones del proyecto.

Jefe Arquitectura y planeación: El arquitecto, deberá comunicar sus dudas y/o sugerencias a el jefe de arquitectura y planeación, quien es el responsable directo de resolverlas; además en conjunto con el deberán elegir la arquitectura que más se ajuste a las necesidades del proyecto, en especial con el tiempo y la calidad.

# Administración del Proyecto

## Métodos y Herramientas de Estimación

Para la estimación del costo del producto se utilizará el Método de estimación por casos de usos con el ajuste de Frohnhoff y Engel, el cual tiene algunas variaciones que se explican más adelante de acuerdo al tamaño y a las necesidades de este proyecto.

Este método estima la complejidad, y extensión final del proyecto basado en las transacciones que realiza cada caso de uso. Donde se entiende como una transacción a un conjunto de actividades atómicas que puede realizar un caso de uso. Estas actividades se ejecutan todas, o no se ejecuta ninguna, eso las hace atómicas.

“Formalmente un escenario representa una forma de uso del sistema, no es exactamente una función, pero sí representa la funcionalidad del sistema.” [36]

El cuadro de dialogo se comprende como la acción de solicitar o de suministrar información a la aplicación (al sistema) bien sea seleccionando datos o configurando variables.

Las métricas que se van a utilizar son [37]:

* Simple: <= 3 escenarios principales, transacciones y cuadros de dialogo (5puntos)
* Medio: <= 7 escenarios principales, transacciones y cuadros de dialogo (10puntos)
* Complejo: >=8 escenarios principales, transacciones y cuadros de dialogo(15 puntos)

Esta estimación se realiza basándonos en métricas de horas hombre y pagos realizados en Pesos Colombianos (COP).

|  |  |  |
| --- | --- | --- |
| Métrica | Ecuación para calcular estimación | Valor |
| A-Factor | * 1. *A-factor = (Suma de 1 a n del peso de los casos de uso) + UAW (valor original del*   2. *método UCP)* | **109** |
| T-Factor | * 1. *TCF= 0,6 + (0,01 \* Σ (T1...T13)* | **1,27** |
| M-Factor | * 1. *M-Factor = 0,8875 - Σi = 1...7 (Mi \* Gi \* 0,025)* | **0,69625** |
| Esfuerzo del proyecto (en horas/persona) | *PE = (PF \* M-factor) \* (T-factor \* A-factor)* | **771,3541** |
| Costo hora (COP) |  | **$12.000** |
| Estimación costo Total del proyecto: | | **$9´256.249** |

PF: Factor de productividad, constante, ajustado 35 hh/UCP según Frohnhoff y Engels [36]; Para el presente proyecto tomaremos un PF de 8 hh/UCP

El desglose completo de cómo se calcularon estos valores, está en el siguiente vínculo Estimacion[SoftLute].docx

## Inicio del Proyecto

### Entrenamiento del Personal

Con respecto al manejo y conocimiento de los distintos programas (ver 8.2) se realizarán tutoriales por parte del Jefe de desarrollo (ver Sección 8.4.2) dentro de las reuniones semanales para que los integrantes del grupo puedan aprender la manera correcta para utilizar las distintas herramientas.

Con respecto a la parte de los distintos lenguajes a tratar en la parte directa de la programación, se realizará también unas capacitaciones dentro de las reuniones semanales con el soporte de material académico para poder avanzar de manera eficaz en el entendimiento de los nuevos lenguajes a manejar.

En caso alguno de seguir con alguna duda con respecto al lenguaje de desarrollo, se puede realizar una pequeña consultoría con los profesores de planta especialistas en el tema de programación gráfica para aclarar dudas.

### Infraestructura

Para evitar la utilización de licencias pagadas, la mayoría de programas a utilizar son programas Open Source [38] lo que significa que son programas completamente gratuitos y en el caso de los programas que necesiten licencias pagas, se habla con el Ingeniero Hernán Hurtado del departamento de Sistemas para consultar la posibilidad de obtener una licencia de los programas requeridos para el proyecto.

Para la parte de redes y comunicación, la arquitectura cliente-servidor se desarrollará y probará dentro de la Pontificia Universidad Javeriana en la sala de Redes, la arquitectura de red dentro de la universidad ya se encuentra montada y se utilizará un servidor público de la universidad para poder montar toda la arquitectura cliente-servidor. Esta solicitud se hará a la Facultad de Ingeniería de Sistemas para poder utilizar el servidor de la Universidad Javeriana.

## Planes de Trabajo del Proyecto

### Descomposición de Actividades

Estas son las actividades principales que se realizarán en cada una de las entregas, la información detallada puede ser encontrada en los anexos adjuntos:

**Primera entrega**

* Definición de reglas de trabajo.
* Definición de roles de trabajo.
* Selección de aplicación a desarrollar.
* Planeación.
* Desarrollo plantilla SPMP.
* Corrección plantilla SPMP.
* Identificación de riesgos.
* Task Slips
* Presentación.
* Cierre primera entrega.

**Segunda Entrega**

* Análisis de la primera entrega.
* Corrección de la primera entrega.
* Estudio plantilla SRS.
* Realización plantilla SRS.
* Corrección SRS.
* Prototipo.
* Presentación.
* Cierre segunda entrega.

**Tercera Entrega**

* Análisis de la segunda entrega.
* Corrección segunda entrega.
* Desarrollo del prototipo guess who?
* Pruebas.
* Redacción de manuales.
* Reporte de métricas.
* Reporte gerencial.
* Presentación.
* Finalización de proyecto.

#### Actividades primeras entrega

Las actividades y tareas de la primera entrega se definen en el siguiente documento (ver [vínculo](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Actividades primera entrega.docx)).

### Calendarización

Para ver el cronograma que proyecta a seguir Softlute para el proyecto ¿Adivina Quién?, se puede seguir el siguiente [vinculo](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Cronograma\\Cronograma Inicial.mpp). Se realizó tomando como base el modelo y ciclo de vida y las fechas de las entregas programadas. Es por esta razón que aunque no entre en detalle, si están definidas grosso modo las actividades que se van a realizar en la totalidad del proyecto.

### Asignación de Recursos

La asignación de los recursos materiales, intelectuales, y de tiempo pueden ser encontradas en la sección de descomposición de actividades 0.

### Asignación de Presupuesto y Justificación

El costo por hora de cada ingeniero se calculó tomando como referencia el promedio entre el costo por hora de un instructor junior (11.000 pesos colombianos) y el costo por hora de un ingeniero practicante (13.000 pesos colombianos). Lo que generó un total de 12.000 pesos colombianos por hora.

De acuerdo a las horas estimadas y asignadas en cada una de las actividades y tareas, el presupuesto requerido en la primera etapa del proyecto sería:

El costo por hora de cada ingeniero se calculó tomando como referencia el promedio entre el costo por hora de un instructor junior (11.000 pesos colombianos) y el costo por hora de un ingeniero practicante (13.000 pesos colombianos). Lo que genero un total de 12.000 pesos colombianos por hora.

De acuerdo a las horas estimadas y asignadas en cada una de las actividades y tareas, el presupuesto requerido en la primera etapa del proyecto sería:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Primera entrega | Actividad | Tarea | Tiempo en horas de la tarea | Total (pesos) |
| Definición Reglas De Trabajo | Recopilación de ideas | 1 | $ 12.000 |
| Definición de reglas | 1 | $ 12.000 |
| Refinamiento del reglamento | 0,5 | $ 6.000 |
| **TOTAL** | | **$ 30.000** |
| Selección De Aplicación A Desarrollar | Proponer aplicaciones a desarrollar | 2 | $ 24.000 |
| Exposición de propuestas | 1 | $ 12.000 |
| **TOTAL** | | **$ 36.000** |
| Definición de roles de trabajo | Investigación sobre roles de trabajo | 2 | $ 24.000 |
| Elección estructura organizacional | 1 | $ 12.000 |
| Asignación de roles | 0,5 | $ 6.000 |
| **TOTAL** | | **$ 42.000** |
| Planeación | Investigación y selección modelo de ciclo de vida | 12 | $ 144.000 |
| Definición del calendario | 4 | $ 48.000 |
| **TOTAL** | | **$ 192.000** |
| Realización SPMP | División secciones SPMP | 2 | $ 24.000 |
| Revisión secciones SPMP | 2 | $ 24.000 |
| Redacción de secciones del SPMP | 28 | $336.000 |
| Corrección secciones SPMP | 16 | $ 192.000 |
| Integración secciones SPMP | 4 | $ 96.000 |
| **TOTAL** | | **$ 624.000** |
| Identificación de riesgos | Identificación general de riesgos | 1 | $ 12.000 |
| Identificación de riesgos SPMP | 2 | $ 24.000 |
| Desarrollo de plan de riesgos | 4 | $ 48.000 |
| Priorización de riesgos | 1 | $ 12.000 |
| **TOTAL** | | **$ 84.000** |
| Task Slips | Aprender y entender las reglas del juego Guess Who? | 0,5 | $ 6.000 |
| Jugar Guess Who? | 1 | $ 12.000 |
| Identificación Task Slips | 4 | $ 48.000 |
| Creación Task Slips | 4 | $ 48.000 |
| Primera revisión Task Slips | 1 | $ 12.000 |
| Corrección Task Slips | 2 | $ 24.000 |
|  | **TOTAL** | | **$ 150.000** |
|  | **TOTAL PRIMERA ENTREGA:** $1’116.000 | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Segunda entrega | Horas disponibles semanales por integrante | Semanas | Horas totales | Total (pesos) |
| 8 | 4 | 240 | $ 12.000 |
| **TOTAL** | | $ 2´880.000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tercera entrega | Horas disponibles semanales por integrante | Semanas | Horas totales | Total (pesos) |
| 8 | 4 | 240 | $ 12.000 |
| **TOTAL** | | $ 2´880.000 |

**Total Estimado =** Primera entrega + segunda entrega + tercera entrega

1116000 + 2880000 + 2880000 = **6’876.000**

# Monitoreo y Control del Proyecto

## Administración de Requerimientos

## Monitoreo y Control de Progreso

El progreso del proyecto será medido por la metodología Goal-Question Methodology [39], El paradigma GQM (Como sus siglas lo indican), definen y evalúan un grupo de metas operacionales (Véase como el resultado de un proceso) a través de métricas [39].

A continuación, se mostraran las metas operacionales las cuales serán puestas a juicio a través del proceso GQM, si se tienen dudas acerca de proceso de control de calidad y como se enlazan con el monitoreo de las actividades, ver la sección 12.5.

**Documentación**:

* Esfuerzo: ¿qué tiempo que se invirtió para realizar una tarea?
* Manejo de plantilla: ¿se aseguró la estandarización de los documentos al utilizar plantillas predeterminadas?
* **Redacción**: ¿hay coherencia en los párrafos?
  + Mala: En varios párrafos se encuentran inconsistencias
  + Aceptable: En general se detecta una buena coherencia y consistencia del texto. Inconsistencias mínimas.
  + Excelente: Coherencia y consistencia perfecta del documento, se entiende en su totalidad el documento.

**Código**:

* Total de líneas de código no comentadas inspeccionadas (KLOC) [40]: ¿Cuál es el total de líneas de código inspeccionadas en una sola inspección?
* Líneas de código no comentadas promedio inspeccionadas (Average LOC inspected) [40]: ¿Cuantas líneas de código no comentadas se revisan en una inspección?
* Tasa promedio de fallas detectadas por KLOC ) [40]: ¿Cuál es la calidad del software inspeccionado?
* Tiempo de desarrollo: ¿qué tiempo se empleó para la implementación de un artefacto de código?
* Espacio de desarrollo [41]: ¿qué tamaño tiene el artefacto de código?

**Progreso**:

Para la evaluación del progreso sea aplicará una plantilla de evaluación de productividad (Productivity Index Evaluation Worksheet) [41], esta plantilla muestra los distintos objetivos del proyecto final en relación con los objetivos desarrollados y los actores que participan en el desarrollo; de esta manera se puede determinar el grado de productividad con la cual se está realizando el proyecto según la siguiente ecuación:

En donde la productividad alcanzada es el valor real de todas las tareas y la productividad completa es la mayor puntuación de todas las tareas, si el índice es menor a 1.0 significa desfavorable, en 1.0 significa “promedio” y superior a 1.0 es favorable. (Ver [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla Indice de Productividad.xlsx)).

Por ejemplo, si el índice de productividad es inferior a 1.0, se empieza a ver individualmente las tareas que tuvieron baja calificación y basados en la el manejo de riesgos 12.2, se pueden tomar acciones de mitigación los cuales se ven en la tabla Riesgos (ver [vínculo](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Riesgos.docx)) y así poder hacer un seguimiento de la productividad de cada integrante del proyecto al igual que su progreso en la calidad en sus tareas.

## Cierre del Proyecto

El cierre del proyecto se divide en dos etapas:

1. Una revisión previa a la entrega al cliente.
2. Una sesión de retroalimentación después de la entrega al cliente (Post-mortem). [42]

**Primera etapa**

Antes de la entrega al cliente, se revisa las iteraciones según el modelo de ciclo de vida desarrollado 8.1.1.

Para el final de cada iteración, se hace una retroalimentación de las distintas tareas que se trabajaron [43], al igual que se hace una discusión con respecto a las dudas o dificultades surgidas. 8.1.1.

Cada documento realizado contiene una tabla de control con el fin de regular las horas invertidas en un documento (ver [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx)) y también para expresar las dificultades presentadas, esta tabla de control la complementa el revisor el cual da una valoración al documento con respecto a la rúbrica y se realizan ciertos comentarios.

Antes de realizar una entrega, se le brinda al cliente una serie de reportes que muestran el avance con respecto al proyecto:

* Reporte Gerencial (ver vínculo).
* Reporte Control de Calidad (Véase sección [12.5](#_Control_de_Calidad))
* Actas de Reunión (ver [vínculo](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Actas)).

Esto en conjunto con los requerimientos solicitados, se entrega al cliente para que analice y valore la calidad del trabajo.

**Segunda etapa**

Después de la valoración dada por el cliente se realiza una reunión en donde el grupo de trabajo lee las valoraciones hechas por el cliente y se realizan las respectivas correcciones (si son necesarias).

# Entrega del Producto

La entrega del producto al cliente, corresponde a la entrega #3 presentada en la sección 8.3. Ahí se especifican que artefactos se van a entregar y que condiciones deben cumplir para que sean aceptados por el cliente.

Se debe haber cumplido las siguientes actividades incluidas en la calendarización, 9.3.2.

* Pruebas del prototipo de la iteración.
* Integración y revisión de artefactos

En la siguiente tabla se puede ver los encargados de presentar cada entregable.

Tabla 11.1 – Encargados de entregar artefactos

|  |  |
| --- | --- |
| Artefacto | Encargado |
| SPMP Corregido | Supervisor y Analista, Jefe de desarrollo y Jefe de Arquitectura |
| SRS Corregido | Supervisor y Analista, Programador |
| SDD | Arquitecto y Jefe de Arquitectura |
| Segundo Prototipo | Programador y Jefe de desarrollo |
| Reporte de pruebas | Programador |
| Manual de Usuario | Programador, Supervisor y Analista |
| Reporte Gerencial | Jefe de desarrollo y Jefe de Arquitectura |

# Procesos de Soporte

## Ambiente de Trabajo

Reglas de trabajo

* Ningún integrante puede faltar a clase sin ninguna justificación alguna con un máximo de 4 fallas, si no se evaluara la estadía del integrante.
* En las reuniones grupales, se tendrá un margen de puntualidad de hasta máximo 10 minutos de lo contrario habrá multa, si el integrante no asiste a la reunión se contara como falla y solo se podrá tener hasta máximo 4 fallas de lo contrario se evaluará si el integrante continua en el grupo o no, tomando en cuenta su desempeño (Se mide según las métricas especificadas en la sección 12.4.
* Se realizan evaluaciones de calidad donde se mirara el desempeño de cada integrante.
* Cumplir con las fechas de avance establecidas por el grupo.
* Respetar a los integrantes del grupo de manera física y verbal.
* Los integrantes deben llegar en óptimas condiciones a las reuniones.
* El pago de la multa debe ser a cada uno de los integrantes que llegaron puntual.
* El integrante que no pague la multa correspondiente tiene un plazo máximo de 2 semanas para pagar, de lo contrario será expulsado del grupo.
* Si por reunión o clase el integrante que tiene multa no paga, se le incrementara el valor de la multa en 500 pesos.
* Todos los integrantes del grupo tienen que asistir a las distintas retroalimentaciones con el ingeniero Miguel Torres de lo contrario se le cobrara una multa.
* Los integrantes que asistan sin falta alguna a las distintas reuniones grupales o que no reciban ninguna multa en una entrega, el grupo le invita a una cerveza por su puntualidad y compromiso.

Valor multa: 1000 pesos

## Análisis y Administración de Riesgos

El plan de riesgos está basado en los siguientes elementos [44]:

**Autoridad de aprobación**: Es quien se encarga de aprobar (si es relevante o no) los riesgos que han sido identificados en el proceso de identificación de riegos. Esta persona también tiene las labores de definir la probabilidad, determinar el impacto, asignar la prioridad, asignar responsables y establecer recursos para cada riesgo aprobado. En este proyecto esta autoridad es el Analista y Supervisor.

**Persona responsable**: Es el encargado de solucionar el riesgo al cual fue asignado y es especificado en la tabla de riesgos ordenados por prioridad y tabla de riegos más importantes.

**Recursos requeridos**: Es lo que se necesita para poder solucionar un riesgo en el caso de que se presente, comúnmente el recurso es tiempo extra de trabajo.

**Fecha de inicio y fin**: Están especificadas en la sección calendarización 9.3.2.

**Impacto y prioridad**: Son definidos de acuerdo a la experiencia del analista y supervisor además de incluir la opinión de los implicados directamente con el riego.

**Actividades:** Son aquellas que transforman la lista de riesgos priorizados en acciones de prevención y mitigación para cada riesgo.

Para lograr esto se utilizan las siguientes actividades [44]:

* Desarrollar un plan de acción al riesgo.
  + Este plan incluye responsables, recursos, impacto y prioridad. Estos aspectos son detallados en la tabla de riesgos ordenados por prioridad (ver tabla).
* Desarrollar un plan de posibles soluciones.
  + Es específicamente la acción de mitigación. Está es detallada en la tabla de riegos más importantes (ver tabla).
* Establecer un umbral de advertencia temprana.
  + Una semana antes al final de cada Sprint se hace un recuento por parte del supervisor y analista y el jefe de arquitectura y planeación quienes revisan el estado de las tareas asignadas a cada integrante, dependiendo del estado en el que esté la tarea y de los problemas presentados para ejecutar la tarea, establecen un umbral de advertencia temprana, generalmente es una semana después de la revisión (final del sprint).

**Acciones tomadas:** Acción para prevenir que un riesgo con prioridad 5 ocurra y acción de mitigación para que una vez el riesgo se presentó, se pueda minimizar su impacto. Estas están definidas en la tabla de riesgos más importantes (ver tabla).

**Resultados esperados:** Riesgos bien definidos en cuanto a su etapa de ocurrencia y los recursos consumidos junto con sus respectivas acciones de prevención y mitigación relacionando áreas afectadas y personas implicadas.

Las tablas de riesgos se pueden ver en el siguiente [vínculo](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Riesgos.docx).

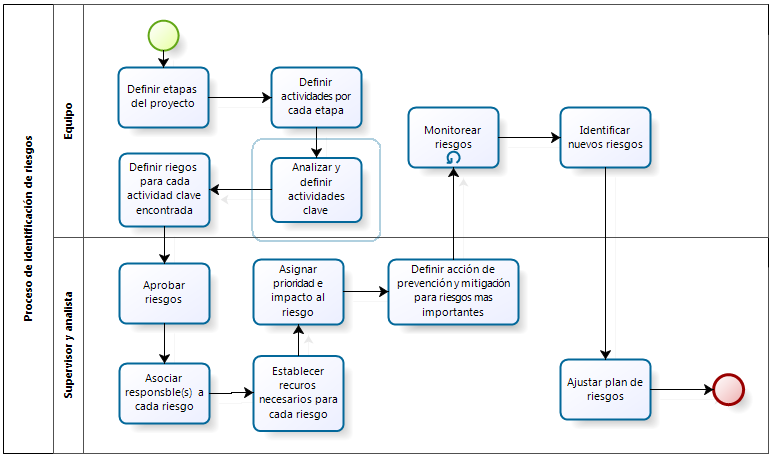


Figura 12.1 – Proceso de identificación de riesgos

El diagrama muestra el proceso que debe seguir Softlute para identificar los riesgos al inicio del proyecto.

El proceso inicia cuando el equipo define las etapas de todo el proyecto, para estas etapas se definen las actividades claves (más importantes). Para estas etapas claves el equipo define posibles riegos. Todos estos riegos encontrados pasan al analista y supervisor quien es el que define y aprueba cuales riegos serán tratados. El siguiente paso es asociar responsables, establecer recursos necesarios, asignar prioridad e impacto a cada riesgo y luego se especifican las acciones de prevención y mitigación. Luego con el equipo se revisa el estado de los riegos previamente establecidos y se identifican nuevos posibles riegos. Finalmente se ajusta el plan de riegos dependiendo de los últimos cambios realizados o a los nuevos riegos encontrados.

## Administración de Configuración y Documentación

**Documentación**: Para la administración de documentos se utilizará Dropbox [14], herramienta que proporciona un historial de versiones anteriores permitiendo restaurar una versión anterior [45]. El encargado de administrar el documento (restaurar versiones, añadir, modificar o eliminar secciones y numeración) es el supervisor y analista 8.4.2. Cada modificación sólo se hace cuando hay cambios importantes al documento. Los cambios quedan en documentos en la carpeta de Dropbox “versiones” indicando en el nombre del documento la sección modificada y finalizando con una P (pendiente). Estos documentos son revisados por el supervisor y analista con una R (revisado)

**Numeración del documento**: V. <A>.<B>; A, es el número de la semana de trabajo en el documento, B, Es el número de cambios realizados en la semana Cada fila del historial de cambios representa un cambio, por lo tanto cada cambio aumenta la revisión en 1. El inicio de la semana enumera la revisión en 0.

**Código:** Para controlar el código se utilizará GIT [46] apoyados en la versión Web GitHub [15], esta herramienta le permite al supervisor y analista 8.4.2, saber quién modifica el código, que parte del código se añadió o eliminó así como agregar notas y comentarios tanto a líneas específicas de código como a los “commits” realizados por los programadores. El supervisor y analista es el encargado de la numeración del código y funciona de la siguiente manera:

* El jefe de desarrollo asigna requerimientos al programador.
* El programador comienza a desarrollar un requerimiento creando una rama.
* El supervisor y analista asigna una revisión a la rama.
* El supervisor y analista valida requerimiento contra desarrollo (rama), en caso de aprobar requerimiento con calidad, se aumenta la revisión de la rama, al haber 3 revisiones de rama, se inicia el proceso de unión ramas-tronco y se aumenta la revisión general.
* Las etiquetas son utilizadas por el jefe de desarrollo en conjunto con el supervisor y analista y el jefe de arquitectura para probar el sistema en un momento determinado y así definir si la versión es inestable, candidata a ser lanzada o un lanzamiento.

Las 3 partes del repositorio que se manejan son: tronco (versión oficial), rama; versión paralela a la del tronco, copia del tronco en donde se agregan funciones, se corrigen errores o se hacen cambios y etiquetas; copia del tronco tomada en cierto momento en donde se realizan pruebas a funciones.

La estructura para manejar el repositorio es la misma que se utiliza en el servidor. Por esto el servidor es una copia de la estructura del repositorio. Deben tener los mismos archivos en ambas partes y el encargado de controlar esto es el jefe de programación 8.4.2.

**Numeración del código:** V. <A>.<B>.<C>; A, revisión general (tronco), B, revisión (rama), C, etiqueta.

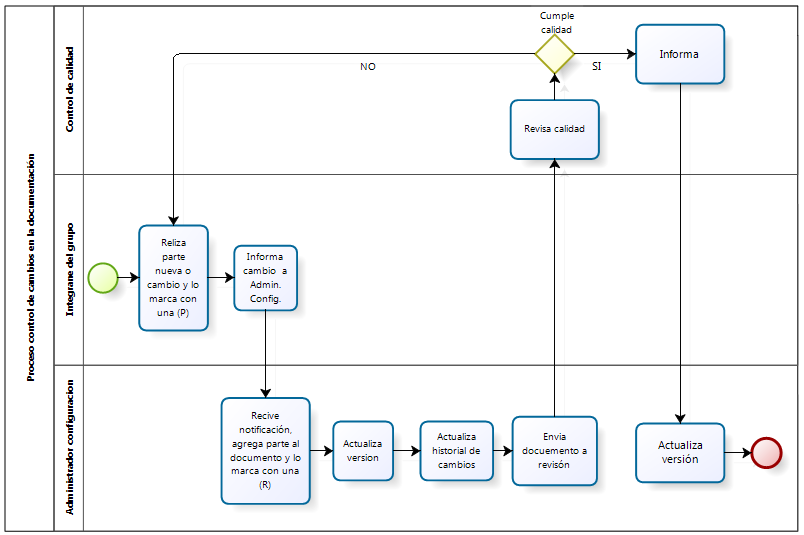


Figura 12.2 – Proceso de cambio de documentos

El diagrama BPMN muestra el proceso que se debe seguir cuando se va a realizar un cambio a algún documento. Este proceso evita perdida de información que se haya adicionado en algún cambio.

El proceso inicia cuando un integrante del grupo hace un cambio o agrega una parte nueva, esto lo hace en un plantilla, nombrándola con el nombre de la sección y finalizado con la letra P (pendiente), esta plantilla es guardada en una carpeta la cual es revisada por el supervisor y analista después de ser notificado por el integrante. El supervisor y analista une esta parte al documento central, marca el documento con una R (revisado), actualiza la versión del documento al igual que el historial de cambios. Al final de cada sprint y después de cada retroalimentación se hacen cambios y se envía el documento a control de calidad para revisión. En caso de aprobar calidad se actualiza la versión y finaliza el proceso, caso contrario el proceso inicia de nuevo.

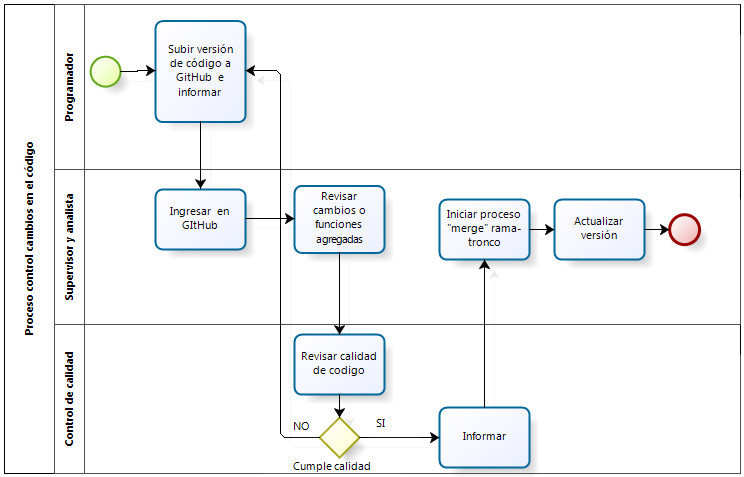


Figura 12.3 – Proceso de cambio de código

El diagrama BPMN muestra el proceso que se debe seguir cuando se va a realizar un cambio a algún pedazo de código. Evita que correcciones o adiciones hechas al código afecten la lógica o funcionamiento del mismo.

El proceso inicia cuando un programador tiene una versión del código, este la sube al repositorio GitHub y realiza un “commit”, luego informa al supervisor y analista para que este revise (requerimientos vs desarrollo) que cambios se realizaron, posteriormente se envía el código a control de calidad para revisión, en caso de cumplir con la calidad requerida, se informa al supervisor y analista, este realiza el proceso de unir la rama con el tronco para crear una nueva versión y así actualizarla, en caso contrario el proceso inicia de nuevo.

Tabla 12.1 - Artefactos de documentación y código.

|  |  |  |  |
| --- | --- | --- | --- |
| Artefacto | Descripción | Momento creación | Momento refinación |
| SPMP | Plan de gestión de proyectos de software | Primera entrega | Segunda entrega |
| Task Slips | Describen funciones del sistema asociadas a las pantallas | Primera entrega | Segunda entrega |
| SRS | Especificación de requerimientos de software | Segunda Entrega | - |
| SDD | Descripción diseño de software | Tercera Entrega | - |
| Código | HTML5, Php y Java Es el sistema en general. | Segunda entrega | Tercera entrega |
| Reporte gerencial | Incluye métricas de tiempos y tareas y conclusiones. | Primera, segunda y tercera entrega | - |
| Manual de usuario | Manual de usuario para los prototipos | Segunda entrega | Tercera entrega |

## Métricas y Proceso de Medición

**Métricas para documentos**

En el desarrollo de documentos se utilizarán dos métricas, los objetivos de estas son principalmente el mejoramiento del integrante a través de un análisis crítico de su desempeño, además, de dar un mayor entendimiento de la calidad de un artefacto producido por parte de un integrante del equipo; las métricas son:

* Métrica de esfuerzo (Ver en [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx) de documentos la tabla de control): una métrica basada en el tiempo invertido en un artefacto.
* Métricas de plantilla GQM: la plantilla GQM (ver [rúbrica](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx) GQM), brinda métricas de calidad; los componentes medidos por esta métrica son:
  + Redacción.
  + Tablas y figuras.
  + Bibliografía.
  + Ortografía y gramática.

**Métricas para código**

En el proceso de desarrollo de código se utilizaran cinco métricas, estas son más relacionadas con el desempeño del programador. Para el proceso de calidad se harán inspecciones de código en parejas 12.5; las métricas son:

* Total de líneas de código no comentadas inspeccionadas (KLOC) [40]: una métrica la cual mide la cantidad de líneas de código inspeccionadas para un artefacto (no se tiene en cuenta los comentarios).
* Líneas de código no comentadas promedio inspeccionadas (Average LOC inspected): métrica de desempeño dentro de las inspecciones, mide el desempeño dentro de la inspección por parejas.
* Tasa promedio de fallas detectadas por KLOC [40]: una métrica de calidad, determina la cantidad de error en mil líneas de código.
* Tiempo de desarrollo: métrica de desempeño que mide el tiempo empleado en el desarrollo del artefacto.
* Espacio de desarrollo [41]: métrica que muestra las líneas de código del artefacto en su totalidad (incluye comentarios).
* Páginas de documentación: cuantas páginas de documentación se generaron por artefacto.

**Plan de gestión de métricas**

Cada programador/documentador estará en la obligación de tomar sus tiempos de trabajo (el método de obtención queda al criterio de cada miembro), estos tiempos estarán almacenados a través de la tabla de control (Véase plantilla de documentos).

**Plan de gestión para documentos**

Para documentación:

* Se almacenarán las métricas de esfuerzo y la plantilla GQM (ver [rúbrica](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx) GQM) en el repositorio por artefacto (ver en [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx)).
* El equipo de gestión de calidad estará encargado de producir la plantilla GQM (ver [rúbrica](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx) GQM) para enviarlo a revisión gerencial.
* Los jefes son responsables del uso de las métricas para gestión de calidad 12.5 .

El almacenamiento de la métrica de esfuerzo se puede encontrar en cada versión del artefacto finalizado (Es almacenado en la carpeta interna de trabajo, en la subcarpeta Versiones).

Cada plantilla GQM (ver [rúbrica](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx) GQM) debe acompañar al artefacto que corresponde, el nombre y manutención de estos artefactos son responsabilidad del administrador de la configuración 12.3.

**Plan de gestión para código**

Para código y su respectiva documentación:

* Se almacenarán las métricas de esfuerzo y su plantilla de inspección en el repositorio por artefacto, en este caso, GitHub 8.2.
* El equipo de desarrollo estará encargado de producir la plantilla de inspección para enviarla a revisión gerencial.
* Los jefes son responsables del uso de las métricas para gestión de calidad 12.5.

El almacenamiento de la métrica de esfuerzo se puede encontrar en cada versión del artefacto finalizado, dentro de sus secciones de comentarios.

Cada plantilla GQM (ver [rúbrica](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx) GQM) debe acompañar al artefacto que corresponde, el nombre y manutención de estos artefactos son responsabilidad del administrador de la configuración 12.3.

## Control de Calidad

En Softlute, se han definido procesos para garantizar la calidad del producto tanto a nivel de código como de documentación.

**Control de calidad de documentos**

Cada documento redactado por Softlute para cualquier proyecto debe cumplir un estándar de calidad dada por el Ingeniero Miguel Torres en la rúbrica de control de calidad de documentos [32]

Se ha identificado que el cumplimiento de la rúbrica se ve afectado por algunos errores que se presentan comúnmente los cuales son:

* Pertinencia de la información
* Ortografía
* Coherencia

En el anexo CalidadDocumento, se puede ver el diagrama BPMN donde se explica el proceso que utiliza Softlute para evitar, identificar y corregir estos errores.

Algunos detalles a tener en cuenta para el anexo de proceso de gestión de calidad son:

* Las rubricas de control de calidad [32] están presentes tanto en el proceso de desarrollo del artefacto, como en el proceso de gestión de calidad, para asegurar que se hayan cumplido con los lineamientos.
* La tabla de control está en la plantilla de documentos (ver [plantilla](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx)).
* La métrica de esfuerzo personal se refiere al tiempo empleado en un artefacto 10.2.
* 10.20

**Control de calidad del código**

Cada artefacto de código debe ser elaborado en su totalidad antes de iniciar con el proceso de calidad; se debe cumplir con un estándar de estructuración de código como parte de una mejora a su calidad [47].

Se realiza al finalizar la implementación una inspección de código por parejas, esto asegura una visión distinta sobre el código y sobre distintos problemas que pueden surgir, la inspección maneja una serie de métricas 0, para determinar la eficacia de esta.

En el anexo CalidadCodigo, se puede ver el diagrama BPMN donde se explica el proceso que utiliza Softlute para evitar, identificar y corregir estos errores.

Algunos detalles a tener en cuenta para el anexo de proceso de gestión de calidad son:

* Se debe cumplir un [estándar](http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml) [48] para asegurar una buena calidad con respecto a la forma en que otros desarrolladores puedan entender el código.
* Las inspecciones son solamente hechas cuando se termine la escritura del código en su totalidad.
* Las métricas asociadas a inspecciones se pueden encontrar en0.
* Dentro de la documentación debe ir la plantilla de inspección para cada artefacto.
* Los actores participantes del proceso están representados por los lanes del diagrama BPMN.

# Anexos

[Actividades primera entrega.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Actividades primera entrega.docx)

[Arbol de navegación.pdf](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Arbol de navegacion.pdf)

[Descripcion funcional del proyecto.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Descripcion funcional del proyecto.xlsx)

[Enunciado del proyecto.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Enunciado del proyecto.docx)

[Estimacion[SoftLute].docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Estimacion[SoftLute].docx)

[Propuestas de Proyecto.pptx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Propuestas de Proyecto.pptx)

[Reglamento-AdivinaQuien.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Reglamento-AdivinaQuien.docx)

[Reglas de trabajo.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Reglas de trabajo.docx)

[Riesgos.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Riesgos.docx)

[Cronograma Inicial.mpp](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Cronograma\\Cronograma Inicial.mpp)

[Cronograma Planeado.mpp](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Cronograma\\Cronograma Planeado.mpp)

[Cronograma real.mpp](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Anexos\\Cronograma\\Cronograma Real.mpp)

[Acta de Retroalimentacion.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Acta de Retroalimentación.docx)

[Acta de reunión.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Acta de Reunión .docx)

[Plantilla de documentos.docx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla de documentos.docx)

[Plantilla Indice de productividad.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla Indice de Productividad.xlsx)

[Plantilla Video Juegos Diseño-Documentos Desarrollada.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla Video Juegos Diseño-Documentos Desarrollada.xlsx)

[Plantilla Video Juegos Diseño-Documentos.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Plantilla Video Juegos Diseño-Documentos.xlsx)

[Plantilla de presentaciones.pptx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Platilla de presentaciones.pptx)

[Reporte Postmortem Desarrollado.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Reporte Postmortem Desarrollado.xlsx)

[Reporte Postmortem.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Reporte Postmortem.xlsx)

[Rubrica GQM.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Rubrica GQM.xlsx)

[Task slips.xlsx](C:\\Users\\Estudiante\\AppData\\Local\\Temp\\Plantillas\\Task slips.xlsx)

# Referencias

|  |  |
| --- | --- |
| [1] | I. Hasbro, «Guess Who?». Estados Unidos Patente 76370660, 3 12 2002. |
| [2] | M. Torres, «Uvirtual,» 21 01 2014. [En línea]. Available: http://uvirtual.javeriana.edu.co/webapps/portal/frameset.jsp?tab\_tab\_group\_id=\_2\_1&url=%2Fwebapps%2Fblackboard%2Fexecute%2Flauncher%3Ftype%3DCourse%26id%3D\_3122\_1%26url%3D. [Último acceso: 08 03 2014]. |
| [3] | IEEE, Estandar Internacional IEEE 16326, 2009. |
| [4] | «Wikipedia,» 8 3 2013. [En línea]. Available: http://es.wikipedia.org/wiki/Personal\_Software\_Process. [Último acceso: 7 3 2014]. |
| [5] | R. S. Pressman, Ingeniería Del Software un enfoque práctico, sexta ed., P. E. R. Vázquez, Ed., McGraw-Hill Interamericana, 2005. |
| [6] | INTECO, Ingeniería del Software: Metodologias y ciclos de vida, Primera ed., 2009. |
| [7] | O. A. C, «UGRM - Diplomado 2012 - Gerencia deproyectos de desarrollo de Software,» 23 Octubre 2013. [En línea]. Available: http://oscararuquipacolque.blogspot.com/. |
| [8] | D. Wells, «Extreme Programming,» [En línea]. Available: http://www.extremeprogramming.org/. [Último acceso: 28 Febrero 2014]. |
| [9] | W. Hutagalung, «Extreme Programing,» [En línea]. Available: http://www.umsl.edu/~sauterv/analysis/f06Papers/Hutagalung/#xp. |
| [10] | F. M. y. S. Martel, «Citeexerx,» [En línea]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=0910357507A1BA4323528541A72989E3?doi=10.1.1.19.4281&rep=rep1&type=pdf. |
| [11] | C. Faulkner, «Software design & Modeling languages,» de *Software Engineering*, Global Media, 2009, pp. 91-94. |
| [12] | J. Turgeon, «Carnegie Mellon Univerisity,» [En línea]. Available: https://resources.sei.cmu.edu/asset\_files/Presentation/2011\_017\_001\_24111.pdf. [Último acceso: 3 3 2014]. |
| [13] | Adobe, «Adobe,» [En línea]. Available: http://www.adobe.com/la/products/photoshop.html. [Último acceso: 2014]. |
| [14] | «Dropbox,» [En línea]. Available: https://www.dropbox.com/tour. |
| [15] | GitHub, «GitHub,» [En línea]. Available: https://github.com/. [Último acceso: 17 Febrero 2014]. |
| [16] | W3C, «W3C HTML,» [En línea]. Available: http://www.w3.org/community/webed/wiki/HTML. [Último acceso: 17 Febrero 2014]. |
| [17] | Brandeis University, «Library & Technology Services,» [En línea]. Available: http://lts.brandeis.edu/techhelp/content/msoffice.html. [Último acceso: 17 Febrero 2014]. |
| [18] | Oracle, «NetBeans,» [En línea]. Available: https://netbeans.org/. [Último acceso: 17 Febrero 2014]. |
| [19] | N. Pascal, «Pencil,» [En línea]. Available: http://www.pencil-animation.org/. [Último acceso: 17 Febrero 2014]. |
| [20] | «Php,» [En línea]. Available: http://www.php.net/manual/es/preface.php. [Último acceso: 17 Febrero 2014]. |
| [21] | Microsoft, «Skype,» [En línea]. Available: http://www.skype.com/es/what-is-skype/. [Último acceso: 17 Febrero 2014]. |
| [22] | Sprintometer, [En línea]. Available: http://sprintometer.com/. [Último acceso: 17 Febrero 2014]. |
| [23] | Apache Friends, «XAMPP Apache + MySQL + PHP + Perl,» [En línea]. Available: http://www.apachefriends.org/es/index.html. [Último acceso: 17 Febrero 2014]. |
| [24] | Bizagi, «Bizagi,» [En línea]. Available: http://www.bizagi.com/?lang=es. [Último acceso: 17 Febrero 2014]. |
| [25] | «Django-Books,» [En línea]. Available: http://django-book.mkaufmann.com.ar/chapter01.html. [Último acceso: 1 3 2014]. |
| [26] | «Django Project,» [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 1 3 2014]. |
| [27] | «TortoiseCVS,» [En línea]. Available: http://www.tortoisecvs.org/. [Último acceso: 1 3 2014]. |
| [28] | «Caracteristicas,» [En línea]. Available: http://es.libreoffice.org/caracteristicas/. [Último acceso: 07 March 2014]. |
| [29] | «Mockingbird,» [En línea]. Available: https://gomockingbird.com/. [Último acceso: 7 March 2014]. |
| [30] | «FEATURE OVERVIEW: GIMP,» [En línea]. Available: http://www.gimp.org/develop/. [Último acceso: 7 March 2014]. |
| [31] | «BonitaBPM,» [En línea]. Available: http://es.bonitasoft.com/productos-servicios#how-we-do-it\_bonita-bpm. [Último acceso: 7 March 2014]. |
| [32] | M. Torres, *Rubrica2014 - SPMP,* 2014. |
| [33] | Softlute, *Plantilla Video Juegos Diseño-Documentos.xlsx,* 2014. |
| [34] | PUJ, «Departamento de Ingenieria de Sitemas,» [En línea]. Available: http://puj-portal.javeriana.edu.co/portal/page/portal/Facultad%20de%20Ingenieria/plt\_dpto\_sistemas/Profesores. [Último acceso: 10 marzo 2014]. |
| [35] | D. FullerPadilla, «Universiad católica del Maule,» [En línea]. Available: http://www.ganimides.ucm.cl/ygomez/descargas/Sist\_inf2/apuntes/2009/Roles\_desarrollo\_software.pdf. [Último acceso: 15 febrero 2014]. |
| [36] | M. I. J. I. Cao, «ITBA,» [En línea]. Available: http://www.iidia.com.ar/rgm/tesistas/cao-trabajofinaldeespecialidad.pdf. [Último acceso: 10 marzo 2014]. |
| [37] | P. T. Cristian A. Remón, «Repositorio institucional de la UNLP,» [En línea]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/19290/Documento\_completo.pdf?sequence=1. [Último acceso: 10 marzo 2014]. |
| [38] | o. s. initiative, «The open source initiative,» [En línea]. Available: http://opensource.org/. [Último acceso: 12 marzo 2014]. |
| [39] | V. Basili, «Software Modeling and Measurement: The Goal/Question/Metric Paradigm,» [En línea]. Available: http://drum.lib.umd.edu/bitstream/1903/7538/1/Goal\_Question\_Metric.pdf. [Último acceso: 10 Marzo 2014]. |
| [40] | J. Barnard, «Managing Code Inspection Information,» [En línea]. Available: https://www.unf.edu/~ncoulter/cen6070/handouts/minorreport/Barnard.pdf. [Último acceso: 8 3 2014]. |
| [41] | E. A. International, «Chapter 17 Performance and Productivity Management,» de *Skills & Knowledge of Cost Engineering*, pp. 230-240. |
| [42] | U. A. d. C. Juárez, «Proceso de Desarrollo de Videojuegos,» [En línea]. Available: http://www2.uacj.mx/iit/culcyt/enero-abril2010/7%20Art%204.pdf. [Último acceso: 9 3 2014]. |
| [43] | R. S. Pressman, Ingenieria de Software Un enfoque practico, McGraw Hill. |
| [44] | E. M. Hall, Managing Risk, Massachusetts : Addison Wesley, 2003. |
| [45] | D. help, «help,» 2014. [En línea]. Available: https://www.dropbox.com/help/11/en. [Último acceso: 8 marzo 2014]. |
| [46] | gitscm, «aboutGit,» [En línea]. Available: http://git-scm.com/about. [Último acceso: 8 marzo 2014]. |
| [47] | X. Fang, «Using a coding standard to improve program quality,» [En línea]. Available: http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=990004&tag=1. [Último acceso: 12 Marzo 2014]. |
| [48] | I. Google, «Google HTML/CSS Style Guide,» [En línea]. Available: http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml. [Último acceso: 12 Marzo 2014]. |
| [49] | University of Chicago, «Git and GitHub,» [En línea]. Available: http://www.classes.cs.uchicago.edu/archive/2013/spring/12300-1/labs/lab1. [Último acceso: 17 Febrero 2014]. |
| [50] | U. P. Fabra. [En línea]. Available: http://creaciodigital.upf.edu/~smiguel/02actionscript.html. [Último acceso: 1 3 2014]. |
| [51] | D. help, «dropbox help,» 2014. [En línea]. Available: https://www.dropbox.com/help/11/en. [Último acceso: 9 marzo 2014]. |
| [52] | T. P. S. University, «Experience with Fagan’s Inspection Method,» [En línea]. Available: http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=C79BFB9F415B7F0E8FAC5B10C2D69D6D?doi=10.1.1.14.2351&rep=rep1&type=pdf. [Último acceso: 8 3 2014]. |
| [53] | M. G. Robiolo, «Transacciones, Objetos de Entidad y Caminos:,» Gabriela Robiolo, Mayo 2009. [En línea]. Available: http://wicc2000.info.unlp.edu.ar/Carreras/Doctorado/Tesis/Robiolo\_Gabriela.pdf. [Último acceso: 10 Marzo 2014]. |