# Índice

[Índice - 1 -](#_Toc397532118)

[1. Plan de proyecto - 2 -](#_Toc397532119)

[2. Creación de matriz de riesgo y contingencias - 5 -](#_Toc397532120)

[3. Medidas de mitigación - 8 -](#_Toc397532121)

[4. Evaluación de esfuerzo - 9 -](#_Toc397532122)

[5. Normas de Desarrollo. - 11 -](#_Toc397532123)

[6. Normas para la Evaluación del Desarrollo. - 16 -](#_Toc397532124)

[7. Estandarización de la documentación del proyecto - 18 -](#_Toc397532125)

# Plan de proyecto

**Anteproyecto.**

Es la forma preliminar de un proyecto para la revisión y autorización, que al ser aprobado o autorizado, adopta el carácter de proyecto. Para poder realizar un anteproyecto, es necesario tener en consideración los siguientes pasos.

* Título: Este es el nombre o denominación que se le da al proyecto

El título que le daremos a nuestro proyecto será: **MechaNetProject**.

* Antecedentes: Se describe la forma en que esta o viene funcionando el ente objeto de estudio, más específicamente en el área elegida para la aplicación del proyecto. Como su nombre lo dice se describen todos los antecedentes que ha tenido el ente objeto de estudio antes de su investigación.

La empresa sobre la cual vamos a desarrollar el sistema, se encuentra en un proceso de crecimiento, por lo cual la organización de las tareas a desarrollar no está muy bien planteadas, y no hay control sobre lo que se desarrolla en el taller mecánico.

* Definición del problema: En este punto lo que se busca es analizar y establecer la idea de manera clara, el nivel de profundidad, el periodo y centrarse en el tema objeto de estudio o investigación, para evitar desviarse al tratar un tema amplio.

El problema principal que vamos a atacar al desarrollar el sistema es la poca organización y administración del trabajo que se realiza en el taller, para así mantener un orden dentro de lo que se realiza.

* Justificación: Es la fundamentación de las razones del por qué es importante y trascendente la realización del proyecto, destacando los beneficios que se obtendrán al ser solucionado el problema.

Es muy importante la realización del sistema, ya que la empresa se encuentra preocupada al saber que no tienen forma de controlar y administrar el trabajo realizado por las personas que se encuentran en el taller mecánico.

* Objetivos: En este punto se debe dejar claramente establecido que es lo que se pretende lograr o que es lo que se va a obtener con el desarrollo del proyecto. Si el objetivo del proyecto es muy general conviene entonces dividirlo en objetivos específicos.

Se quiere lograr una mejora en la administración de los recursos a través de un adecuado control, un almacenamiento seguro y gestión de la información.

* Alcances y limitaciones: El alcance define el área o lugar en concreto donde se aplicara el proyecto, mientras que la limitación define dentro de ese lugar, la función específica de la actividad a realizar.

Taller mecánico, función a realizar, creación de sistema de apoyo de gestión.

* Procedimiento: Es la forma de cómo vamos a realizar las actividades a seguir de acuerdo al objetivo ya descrito, en otras palabras, son la serie de pasos o la secuencia lógica de actividades para el logro de los objetivos.

Levantar requerimientos, generar un prototipo, aceptación el prototipo y desarrollo del sistema, marcha blanca, capacitación, implementación, respaldo.

* Cronograma: Consiste en delimitar el tiempo que comprenderá cada una de las actividades para el desarrollo del proyecto, así como la de la fecha aproximada en que concluirá.

Ver carta Gantt adjunta.

* Presupuesto: Es el cálculo de los gastos que requieran los recursos implicados en el proyecto, haciendo referencia de quien aportara el costo del mismo.
* Bibliografía: Determinación de las fuentes que serán consultadas para el desarrollo del proyecto (libros, artículos de revistas, etc.). Siendo conveniente que estas sean actualizadas.

**Proyecto.**

Desde un punto de vista muy general puede considerarse que todo proyecto tiene tres grandes etapas.

* Fase de planificación: Se trata de establecer cómo el equipo de trabajo deberá satisfacer las restricciones de prestaciones, planificación temporal y costo. Una planificación detallada da consistencia al proyecto y evita sorpresas que nunca son bien recibidas.
* Fase de ejecución: Representa el conjunto de tareas y actividades que suponen la realización propiamente dicha del proyecto, la ejecución de la obra de que se trate. Responde, ante todo, a las características técnicas especificadas de cada tipo de proyecto y supone poner en juego y gestionar los recursos en la forma adecuada para desarrollar la obra en cuestión. Cada tipo de proyecto responde en este punto a su tecnología propia, que es generalmente bien conocida por los técnicos en la materia.
* Fase de entrega o puesta en marcha: Como ya se ha dicho, todo proyecto está destinado a finalizarse en un plazo determinado, culminando en la entrega de la obra al cliente o la puesta en marcha del sistema desarrollado, comprobando que funciona adecuadamente y responde a las especificaciones en su momento aprobadas. Esta fase es también muy importante no solo por representar la culminación de la operación sino por las dificultades que suele presentar en la práctica. Alargándose excesivamente y provocando retrasos y costos imprevistos.

A estas tres grandes etapas es conveniente añadir otras dos que, si bien pueden incluirse en las ya mencionadas, es preferible nombrarlas de forma independiente ya que definen un conjunto de actividades que resultan básicas para el desarrollo del proyecto.

* Fase de iniciación: Definición de los objetivos del proyecto y de los recursos necesarios para su ejecución. Las características del proyecto implican la necesidad de una fase o etapa previa destinada a la preparación del mismo, fases que tienen una gran trascendencia para la buena marcha del proyecto y que deberá ser especialmente cuidada. Una gran parte del éxito o fracaso del mismo se fragua principalmente en estas fases preparatorias que, junto con una buena etapa de planificación, algunas personas tienden a menospreciar, deseosas por querer ver resultados excesivamente pronto.
* Fase de control: Monitorización del trabajo realizado analizando cómo el progreso difiere de lo planificado e iniciando las acciones correctivas que sean necesarias. Incluye también el liderazgo, proporcionando directrices a los recursos humanos, subordinados para que hagan su trabajo de forma efectiva y a tiempo.

# Creación de matriz de riesgo y contingencias

Elaboran la matriz de riesgo y contingencia

**Matriz de riesgo.**

Una matriz de riesgo constituye una herramienta de control y de gestión normalmente utilizada para identificar las actividades (procesos y productos) más importantes de una empresa, el tipo y nivel de riesgos inherentes a estas actividades y los factores exógenos y endógenos relacionados con estos riesgos. Igualmente, una matriz de riesgo permite evaluar la efectividad de una adecuada gestión y administración de los riesgos financieros que pudieran impactar los resultados y por ende al logro de los objetivos de una organización.

La matriz debe ser una herramienta flexible que documente los procesos y evalúe de manera integral el riesgo de una institución, a partir de los cuales se realiza un diagnostico objetivo de la situación global de riesgo de una entidad. Exige la participación activa de las unidades de negocios, operativas y funcionales en la definición de la estrategia institucional de riesgo de la empresa. Una efectiva matriz de riesgo permite hacer comparaciones objetivas entre proyectos, áreas, productos, procesos o actividades. Todo ello constituye un soporte conceptual y funcional de un efectivo “Sistema Integral de Gestión de Riesgo”.

**Plan de contingencia.**

Podríamos separar la definición de lo que es un plan de contingencia en las siguientes:

Definición general.

Conjunto de procedimientos alternativos a la operativa normal de cada empresa, su objetivo es permitir el funcionamiento de ésta, aun cuando alguna de sus funciones deje de hacerlo por culpa de algún incidente tanto interno como ajeno a la organización.

Definición específica (sistemas informáticos).

Es la identificación de aquellos sistemas de información y/o recursos informáticos aplicados que son susceptibles de deterioro, violación o pérdida y que pueden ocasionar graves trastornos para el desempeño normal de la organización, con el propósito de estructurar y ejecutar aquellos procedimientos y asignar responsabilidades que salvaguarden la información y permitan su recuperación garantizando la confidencialidad, integridad y disponibilidad de ésta en el menor tiempo posible y a unos costos razonables.

El plan de contingencia debe cubrir todos los aspectos que se van a adoptar tras una interrupción, esto implica suministrar servicios alternativos, revisar las operaciones cotidianas además de analizar a los principales distribuidores, clientes, negocios y socios, así como la infraestructura en riesgo, esto incluye cubrir los siguientes elementos: hardware, software, documentación, talento humano y soporte logístico, también debe ser lo más detallado posible y fácil de comprender.

Las funciones de un plan de contingencia son:

* Determinar acciones preventivas, reduciendo el grado de vulnerabilidad y exposición al riesgo.
* Reducir el tiempo de reacción ante la emergencia.
* Dimensionar el riesgo potencial.
* Tomar decisiones rápidas ante anormalidades o fallas.
* Generar cultura de seguridad en la organización.
* Asegurar la estabilidad de la organización.
* Hacer sistemático, ordenado y eficiente lo que, sin un plan debidamente concebido y ensayado, sería arbitrario, caótico e ineficiente.
* Cumplir con las normativas legales.

|  |  |  |
| --- | --- | --- |
| Riesgo | Descripción | Probabilidad |
| Cambios de requisitos. | Existencia de más cambios de requerimientos de los previstos inicialmente. | **Media.** |
| Planificación demasiado optimista. | Existencia de una planificación previamente realizada, que no calzara con el tiempo de desarrollo final del sistema. | **Alta.** |
| Diseño inadecuado. | Encontrar que el sistema diseñado anteriormente para resolver el problema, no se encuentra bien diseñado. | **Medio.** |
| Los recursos no se encuentran disponibles en el momento de utilizarlos. | Cortes de energía, falta de conexión a internet para poder desarrollar el sistema. | **Baja.** |
| Las herramientas de programación no dan los resultados esperados. | El lenguaje de programación elegido inicialmente no es el apropiado para desarrollar el software. | **Baja.** |

|  |  |
| --- | --- |
| Riesgos | Contingencia |
| Cambios de requisitos. | Replanteamiento de lo que requiere el cliente. |
| Planificación demasiado optimista. | Realizar una nueva planificación tomando bien en cuenta los tiempos necesarios para desarrollar el sistema. |
| Diseño inadecuado. | Realizar varios tipos de diseño para el mismo sistema para que en caso de que alguno falle tener otro de respaldo. |
| Los recursos no se encuentran disponibles en el momento de utilizarlos. | Tener precaución en cuanto a los recursos que se utilizaran, ya sean licencias o computadores, para que estos se encuentren siempre disponibles. |
| Las herramientas de programación no dan los resultados esperados. | Tener programadores familiarizados con otros lenguajes de programación, aparte del ya previamente elegido para desarrollar el sistema |

# Medidas de mitigación

**Definición.**

Se entienden como medida de mitigación, la implementación o aplicación de cualquier política, estrategia, obra y/o acción tendiente a eliminar o minimizar los impactos adversos que pueden presentarse durante las etapas de ejecución de un proyecto (construcción, operación y terminación).

Llevando esto al ámbito informático o de nuestro proyecto, significa que se deben plantear políticas de seguridad para determinar las formas en que se van a enfrentar ciertos problemas mientras se encuentre funcionando el sistema.

# Evaluación de esfuerzo

**Puntos de función**

El análisis por puntos de función es un método para cuantificar el tamaño y la complejidad de un sistema software en términos de las funciones de usuario que este desarrollará. Esto hace que la medida sea independiente del lenguaje o herramienta utilizada en el desarrollo del proyecto.

El análisis por puntos de función está diseñado para medir aplicaciones de negocios; no es apropiado para otro tipo de aplicaciones como aplicaciones técnicas o científicas.

El enfoque de puntos de función tiene características que permiten superar los principales problemas de utilizar líneas de código como métrica del tamaño del software. Primero, los puntos de función son independientes del lenguaje, herramientas o metodologías utilizadas en la implementación; por ejemplo, no tienen que considerar lenguajes de programación, sistemas de administración de bases de datos, hardware, o cualquier otra tecnología de procesamiento de datos. Segundo, los puntos de función pueden ser estimados a partir de la especificación de requisitos o especificaciones de diseño, haciendo posible de este modo la estimación del esfuerzo de desarrollo en etapas tempranas del mismo. Como los puntos de función están íntimamente relacionados con la declaración de requisitos, cualquier modificación a ésta, puede ser reflejada sin mayor dificultad en una re estimación.

Tercero, como los puntos de función están basados en una visión externa del usuario del sistema, los usuarios no técnicos del software poseen un mejor entendimiento de lo que los puntos de función están midiendo. El método resuelve muchas de las inconsistencias que aparecen cuando se utiliza líneas de código como métrica del tamaño del software.

En resumen, los puntos de función aparecen con ventajas substanciales por sobre las líneas de código, para fines de estimación temprana del tamaño del software, y por ende, del esfuerzo de desarrollo. Además es una medida ampliamente utilizada, y con éxito, en muchas organizaciones que desarrollan software en forma masiva.

**Métrica para Estimación Temprana del Tamaño del Software.** Para poder realizar una estimación temprana, se propone la utilización de la métrica de puntos de función modificada. Este método consiste en clasificar, listar y contar a partir de la especificación de requisitos todos los componentes, desagregados en:

a. Entradas Externas.

b. Salidas Externas

c. Archivos Lógicos

d. Consultas.

En el método original además se consideraba la clasificación Archivos de Interfaces externos, los cuales son utilizados para la comunicación entre aplicaciones. Debido a que a esta altura del desarrollo difícilmente se puede contar con esta información, se ha eliminado. En caso de ser obvia la necesidad de contar con un archivo de interface, éste debe cuantificarse como un archivo lógico más.

Para resolver el problema de determinar la complejidad asociada a cada componente de la aplicación (difícil de conocer en esta etapa), se ha optado por asignar a todos los componentes una complejidad promedio. Esta es una aproximación válida, debido a que normalmente "en media" las aplicaciones son de complejidad promedio, teniendo sólo unos pocos componentes de complejidad simple y otros complejos.

De este modo, la fórmula para obtener los puntos de función para estimación temprana (PFET) es la siguiente:

http://www.inf.udec.cl/%7Emvaras/papers/arica/IMG00002.GIF

Donde

PFET : Puntos de función para estimación temprana.

IN : Número de Entradas Externas

OUT : Número de Salidas Externas

INQ : Número de Consultas

FILE : Número de Archivos Lógicos

ACP : Factor de Ajuste de Complejidad de Proceso

# Normas de Desarrollo.

5.1Metodologías

Existe más de un tipo de metodología de desarrollo de software, a continuación explicamos brevemente los grandes grupos de metodologías, para finalmente, adherir a una para nuestro proyecto:

METODOLOGÍAS ESTRUCTURADAS

Los métodos estructurados comenzaron a desarrollarse a fines de los 70’s con la Programación Estructurada, luego a mediados de los 70’s aparecieron técnicas para el Diseño (por ejemplo: el diagrama de Estructura) primero y posteriormente para el Análisis (por ejemplo: Diagramas de Flujo de Datos). Estas metodologías son particularmente apropiadas en proyectos que utilizan para la implementación lenguajes de 3ra y 4ta generación.

METODOLOGÍAS ORIENTADAS A OBJETOS

Su historia va unida a la evolución de los lenguajes de programación orientada a objeto, los más representativos: a fines de los 60’s SIMULA, a fines de los 70’s Smalltalk-80, la primera versión de C++ por Bjarne Stroustrup en 1981 y actualmente Java o C# de Microsoft. A fines de los 80’s comenzaron a consolidarse algunos métodos Orientadas a Objeto.

En 1995 Booch y Rumbaugh proponen el Método Unificado con la ambiciosa idea de conseguir una unificación de sus métodos y notaciones, que posteriormente se reorienta a un objetivo más modesto, para dar lugar al Unified Modeling Language (UML), la notación Orientada a Objetos más popular en la actualidad.

METODOLOGÍAS TRADICIONALES

Las metodologías no ágiles son aquellas que están guiadas por una fuerte planificación durante todo el proceso de desarrollo; llamadas también metodologías tradicionales o clásicas, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema.

Todas las propuestas metodológicas antes indicadas pueden considerarse como metodologías tradicionales. Aunque en el caso particular de RUP, por el especial énfasis que presenta en cuanto a su adaptación a las condiciones del proyecto (mediante su configuración previa a aplicarse), realizando una configuración adecuada, podría considerarse Ágil.

METODOLOGÍAS ÁGILES

Un proceso es ágil cuando el desarrollo de software es incremental (entregas pequeñas de software, con ciclos rápidos), cooperativo (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), sencillo (el método en sí mismo es fácil de aprender y modificar, bien documentado), y adaptable (permite realizar cambios de último momento).

Esta es la metodología que más nos acomoda de forma que permite ir adaptando el trabajo a necesidades que aparecen en el camino, permitiendo pequeños avances que sirven de muestra y se modifican rápidamente. Dentro de las metodologías agiles, se encuentran SCRUM Y XP (extreme programing)

5.2 CVS (Control de versiones de Software).

Se llama control de versiones de software a la gestión de los diversos cambios que se realizan sobre los elementos o una configuración de un software. Estos sistemas facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (por ejemplo, para algún cliente específico). Ejemplos de este tipo de herramientas son entre otros: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, Mercurial, Perforce, Fossil SCM.

El control de versiones se realiza para controlar las distintas versiones del código fuente dando lugar a los sistemas de control de código fuente o SCM (siglas del inglés Source Code Management). Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web, etc.

En nuestro proyecto usaremos el CVS llamado “GIT HUB”

* 1. Nomenclatura

|  |  |  |
| --- | --- | --- |
| Identificador | Reglas de Nomenclatura | Ejemplo |
| Clases | Los nombres de clases deben ser palabras completas, en mayúsculas y minúsculas, con la primera letra de cada palabra en mayúscula. Los nombres de clases deben ser simples y descriptivos, utilizando palabras completas y acrónimos o abreviaturas (a no ser que la abreviatura sea ampliamente conocida, como URL o HTML). | class Raster;  class ImageSprite; |
| Interfaces | Los nombres de interfaces deberías seguir las mismas reglas indicadas para las clases. | interface RasterDelegate;  interface Storing; |
| Métodos | Los métodos deberían ser verbos, en mayúsculas y minúsculas, con la primera letra en minúscula, y la primera letra de cada una de las palabras internas en mayúscula. | run();  runFast();  getBackground(); |
| Variables | Todos los nombres de variables de instancia o de clase deben estar constituidos por palabras con la primera letra de la primera palabra en minúscula y la primera letra de las palabras internas en mayúscula.  Los nombres de variables deben ser cortos y significativos. La elección del nombre de las variables debe ser mnemotécnica, es decir, pensado para que un lector casual al verla comprenda su uso. Se deben evitar las variables de una sola letra, excepto en variables temporales de corto uso. Nombres comunes para este tipo de variables son: i, j, k, m y n para enteros; c, d, y e para caracteres. | int i;  char \*cp;  float myWidth; |
| Constantes | Los nombres de variables declaradas como constantes de clase y constantes ANSI, deberías escribirse siempre en mayúsculas, con las palabras internas separadas por el signo de subrayado (“\_”). Las constantes ANSI se deben evitar en lo posible, para facilitar la depuración del código. | int MIN\_WIDTH = 4;  int MAX\_WIDTH = 999;  int GET\_THE\_CPU = 1; |

En la Bases de datos:

* Utilizar caracteres alfanuméricos
* Limitar los nombres a menos de 64 caracteres (es una restricción de MySQL).
* Utilizar el guión bajo (\_) para separar palabras.
* Utilizar palabras en minúsculas.
* Los nombres de las tablas deberían ir en plural y los nombres de las columnas en singular.
* En una tabla, colocar primero la clave primaria seguida de las claves foráneas.

5.4 Modelos de seguridad de la información

Un Modelo de Seguridad de la Información es un diseño formal que promueve consistentes y efectivos mecanismos para la definición e implementación de controles a través de políticas.

A nivel de desarrollo de una aplicación, la seguridad debe implementarse desde ya, de forma de robustecer el sistema prototipo. Así los mecanismos de seguridad deben estar presentes dentro de los requerimientos del sistema y deben definir que controles son necesarios cuando y porque.

Por esto fijaremos dentro de los requisitos funcionales:

* Identificación y autenticación: uso de técnicas de validación de identidad.
* Gestión de seguridad: definición de perfiles de usuario y niveles de acceso asociados.
* Privacidad: soporte del anonimato de los usuarios.

5.5 Modelos de madurez (CMM)

Los cinco niveles de maduración de proceso de software

El CMM brinda a las organizaciones un marco para organizar los pasos de la mejora en cinco niveles de madurez, estableciendo fundamentos para la mejora continua del proceso. Estos cinco niveles, definen una escala ordenada para evaluar la madurez del proceso de software.

Además, estos niveles sirven para priorizar sus esfuerzos para la mejora.

Un nivel de maduración es una plataforma definida con objetivos claros de manera tal, que cuando son cumplidos, se estabiliza una componente importante del proceso de software. Cada uno de los niveles de madurez provee fundamentos para la mejora del siguiente nivel, resultando en un crecimiento en la capacidad de la organización.

Los cinco niveles pueden ser descritos brevemente como:

1. Inicial: el proceso de software esta caracterizado como ad hoc y ocasionalmente caótico. Algunos procesos están definidos, y el éxito depende del esfuerzo individual y no de la organización.

2. Repetible: procesos administrativos básicos en los proyectos para el seguimiento de costo, planeamiento y funcionalidad. La disciplina necesaria en los procesos es acorde para repetir éxitos anteriores de proyectos con aplicaciones similares.

3. Definido: el proceso para la administración y la ingeniería esta documentado, estandarizado e integrado a un proceso estándar para la organización. Todos los proyectos usan una versión del proceso estándar de la organización aprobada y ajustada para el desarrollo y manutención del software.

4. Administrado: Se detalla medidas para que el proceso de software y la calidad del producto sean recolectados. Ambos son entendidos y controlados cuantitativamente.

5. Optimizado: existe un feedback cuantitativo del proceso, lo que permite una mejora continua del mismo. También se manejan ideas y tecnologías innovadoras.

# Normas para la Evaluación del Desarrollo.

6.1 Control de avances: El control del proyecto se define como toda actividad que asegure que el trabajo real va de acuerdo al plan, compara lo realizado con las metas y planes, revela cuando y donde existen desviaciones, y pone en marcha acciones correctoras; ayudando a la realización de los planes.

Básicamente permite:

* Emprender acciones correctivas.
* Vigilar las desviaciones del plan.
* Recibir y evaluar cambios en los proyectos solicitados.
* Cambiar los calendarios del proyecto.
* Regresar a la etapa de planeación para hacer ajustes.

**Definir los puntos de control**: son los momentos donde se decide realizar el control (al finalizar una fase relevante del proyecto, al final de una actividad clave, etc.)

Para nuestro caso serian los siguientes: **Análisis y Modelo Conceptual de la aplicación, Diseño Lógico y Físico de la aplicación y finalmente Construcción, Pruebas e Implementación de la aplicación.**

**Controlar las tareas**: en esta actividad se establece la comparación entre los resultados obtenidos del seguimiento y los previstos con el plan, se calculan las diferencias y se realiza el análisis de las posibles causas de esas diferencias, especialmente si son significativas

**Proponer acciones correctivas**: se proponen acciones correctivas para corregir o compensar los problemas detectados, se revisan los planes originales de ser necesario Los puntos críticos no están siendo chequeados, por ejemplo no existe fijado un día para entregar el SW a una prueba.

**Controlar los cambios**: se realizarán las solicitudes y registro de los cambios, evaluados por el equipo del proyecto e información a todos de los cambios efectuados.

PMI (Project Management Institute)**:** Es una organización Internacional sin fines de lucro fundada en 1969 en Estados Unidos, cuyo objetivo es la **profesionalización del gerenciamiento de proyectos**.

La guia PMBOOK, un estándar en la gestión de proyectos desarrollado por el PMI para gestionar proyectos, donde se explica que existen cinco fases en el Ciclo de Vida de la Administración de Proyectos a las que se denomina Grupos de Procesos:

**Inicio**: Definen y autorizan un proyecto o fase.

**Planificación**: Definen y refinan los objetivos y preparan el plan de Gestión del Proyecto con la mejor alternativa de acción para lograr los objetivos y el alcance que el proyecto o fase del proyecto debe realizar

**Ejecución**: Integran a la gente y otros recursos para llevar a cabo el plan de gerencia del proyecto para un proyecto o una fase.

**Control**: Miden y supervisan regularmente el avance a fin de identificar las variaciones respecto del plan de gestión del proyecto, de tal forma que se tomen medidas correctivas cuando sea necesario.

**Cierre**: Formalizan la aceptación del producto, servicio o resultado y lleva al proyecto, o a una fase, a un final ordenado.

# Estandarización de la documentación del proyecto

Todo documento presentado para nuestro proyecto deberá seguir las siguientes directrices:

* Portada con el nombre y número del informe.
* Datos de los integrantes del grupo, profesor y fecha de entrega
* Hoja tamaño carta
* Uso del membrete institucional de INACAP en la portada.
* Índice
* Las hojas de contenido deberán llevar numero de pagina
* El pie de página deberá decir el nombre de la asignatura y la pagina
* El encabezado deberá llevar el nombre del proyecto
* Margen Normal
* Entrelineado de 1,5
* Sangría francesa invertida