**Benemérita Universidad**

**Autónoma de Puebla**

**Facultad de Ciencias de la computación**

# **Ingeniería de software I**

Conceptos básicos Ingeniería de SW

 Juan Manuel González

**Equipo 1**

* + - Abraham Sarid Acosta Herrera.
    - Abner Elías Cid Ramírez.
    - Jaime Antonio Altamirano.
    - Manuel Alejandro Ramírez Salinas.
    - Matías Hernández Hernández.
    - Pablo García Bravo.

## ***Índice general***

[*Presentación*](#_Ingeniería_de_software)*………………………………………………………pág.- 01*

[*Índice general*](#_índice_general)*……………………………………………………pág. -02*

[*Hoja de control*](#_Hoja_de_control)*…..………………………………….…………...pág. -03*

[*Introducción*](#_Introducción)*……………………………………………………...pág.-04*

[*1.-Importancia de la Ingeniería de software*](#_1._IMPORTANCIA_DE)*………………. pág. -04*

[*1.1.-Historia de la Ingeniería de SW*](#_Historia_de_la)*……………pág.- 04*

[*1.2.-Mitos de la ingeniería de SW*](#_Mitos_de_la)*………………. pág.- 05*

[*1.3.-Crisis del software*](#_Crisis_del_software)*………… ………………pág.- 07*

[*1.4.-Conceptos básicos de la ING. SW*](#_Conceptos_básicos_de)*…… …... pág.- 08*

[*2.-Los tipos de Software*](#_2._LOS_TIPOS)*…………………. …………………pág.- 09*

[*2.1.-lasificación*](#_Tipos_de_software) *de tipos de Software…………………pág.- 09*

[*2.2.-La importancia de conocer………………………. pág.- 11*](#_La_importancia_de)

[*3.-La ética de software y el código de ética del equipo*](#_3._La_ética)*……Pág. -12*

[*3.1.-La ética de SW y el código de ética del equipo*](#_La_ética_de)*…pág. -12*

[*4.-Los roles del equipo em base el ciclo de vida del SW*](#_4.-Los_roles_del)*…pág. -14*

[*4.1.-Ciclo de vida del SW y roles del equipo……… pág. -14*](#_Ciclo_de_vida_1)

[*5.-Gestion y estimación de proyectos*](#_5._GESTIÓN_Y)*…………………………pág. -20*

[*Aprobación y definición del proyecto*](#_Aprobación_y_definición)*………………...pág. -21*

[*Planificación del Proyecto*](#_Planificación_del_proyecto)*……………………………pág. -22*

[*Factores de éxito de un proyecto*](#_FACTORES_CRITICOS_DE)*…………………… pág. -23*

[*Ejecución del proyecto*](#_Ejecución_del_Proyecto.)*………………………………... pág. -24*

[*Cierre y evaluación del proyecto*](#_Cierre_y_evaluación)*…………………… pág. -25*

[*Estimación del proyecto de SW*](#_ESTIMACIÓN_DEL_PROYECTO)*……………………… pág. -26*

[*Metodologías para la gestión del SW*](#_Metodologías_para_la)*……………… pág. -28*

[*6.-Modelos de procesos de Software*](#_1.-Modelo_v)*………………………pag. -29*

[*1.-Modelo V*](#_1.-Modelo_v)*……………………………………………pag. -30*

[*2.-Cascada*](#_2.-Cascada)*……………………………………………pag. -31*

[*3.-Prototipado*](#_3.-Prototipado)*………………………………………pag. -33*

[*4.-Especificacion Operacional*](#_4.-Especificación_Operacional)*……………………pag. -34*

[*5.-Modelo Transformacional*](#_5.-Modelo_transformacional)*………………………pag. -35*

[*6.-Modelo RUP*](#_6.-Modelo_RUP)*………………………………………pag.-36*

[*7.-Modelo Espiral*](#_7.-Modelos_Espiral)*…………………………………..pag.-37*

[*8.-Modelo ágil de software*](#_8.-Desarrollo_ágil_de)*………………………..pag.-39*

[*7.-Procesos de la ingeniería de requerimientos*](#_9.-_procesos_de)*……………………….pag.-42*

[*¿Por qué es importante la ingeniería de requerimientos?*](#_¿Por_qué_es)*.......pag.-43*

[*Requerimientos del proyectos*](#_Requerimientos_del_proyectos)*…………………………………….pag.-44*

[*Pasos que seguir para nuestro proyecto*](#_Pasos_que_seguir)*…………………………pag.-46*

[*Conclusión*](#_Conclusión)*……………………………………………………. pág.- 48*

[*Bibliografía*](#_Bibliografía)*…………………………………………………… pág.- 50*

### ***Hoja de control***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo de Copia1 | Versión del Documento | Elaboradores (NOMBRES DE QUIENES LO HACEN) | Descripción del entregable o de los cambios hechos al documento | |
| D, T | 1 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  Enedina Antonio López  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Creación del documento y edición de este. | |
| D, T | 1.1 | Manuel Alejandro Ramírez Salinas | Corrección de la conclusión. | |
| D, T | 2 | Manuel Alejandro Ramírez Salinas Pablo García Bravo  Abraham Sarid Acosta Herrera  Matías Hernández Hernández | Añadido de dos nuevos apartados, ética y ciclo de software, expansión de la conclusión. | |
| D, T | 3 | Abraham Acosta Herrera.  Enedina Antonio López  Matías Hernández Hernández  Pablo García Bravo | Corrección de errores  Asignación de roles de equipo  Investigación necesaria sobre el nuevo tema. | |
| D, T | 3.1 | Abner Elías Cid Ramírez  Manuel Alejandro Ramírez Salinas  Matías Hernández Hernández | Redacción del nuevo tema  Actualización de conclusión y bibliografía | |
| D, T | 4 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  \*\*Enedina Antonio López  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Redacción de distintos modelos de procesos de software así también como sus reglas a seguir. | |
| D, T | 5 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  \*\*Enedina Antonio López  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Escritura de la ingeniería de requerimientos así como sus diferentes formas de llevarse a cabo. |
| D,T | 6 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Actualizacion de la ingeniería de requerimientos así como sus diferentes formas de llevarse a cabo.  Primera especificación de requerimientos |
| D,T | 7 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  Jaime Antonio Altamirano  Matías Hernández Hernández Pablo García Bravo | Especificacion de requerimientos, actualización de elementos funcionales y no funcionales |
| D,T | 8 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera,  Abner Elías Cid Ramírez  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Anexo de diseño del proyecto, diagramas de clases y síntesis de este en el cuerpo del documento. |

\*\*Compañera ausente, permiso temporal.

### ***Introducción***

La ingeniería del Software es una disciplina de ingeniería que se interesa con todos los aspectos de la producción de software, desde la especificación del sistema, mantenimiento hasta que haya entrado en uso.

La economía se ve basada en el software ya que cada vez en nuestro entorno se encuentra manejado por el mismo. Forma parte de una significativa parte del Producto nacional bruto que es el indicador para mostrar la cantidad de bienes y servicios finales de un país.

# **1. Importancia de la ingeniería de software**

## ***Historia de la Ingeniería de software***

Se piensa que el término surgió durante una conferencia en 1968 de la OTAN, también que Anthony Hostinger, lingüista e informático, hizo uso del término para mencionar las diferencias entre ciencias de la computación y desarrollo de software.

Grady Booch afirma que fue Margaret Hamilton, entre 1963 y 1964, utilizó el término para destacar su trabajo en el proyecto Apoyo y Skylab.

El desarrollo del software se realizaba en la década de los cincuenta, a través de la experiencia y lo consideraban como un añadido, no había métodos sistemáticos, solo una persona se encargaba de escribir, ejecutar y si fuera necesario se realizaban cambios. Esto daba entender que la planificación no estaba presente, es decir, se trabajaba sobre un concepto inconcreto que iba tomando forma de acuerdo con su avance.

En los años sesenta se toma un nuevo concepto al introducir las máquinas para poder recoger, analizar datos de diferentes fuentes. Las casas de software, en su entonces, producían programas que eran corregidos de acuerdo con los fallos que se presentaban tomando como proceso “codifica y corrige”, Como consecuencia, provocó el incremento de gastos, entregas fuera de tiempo y que el mismo software creado no se podía adaptar a los cambios llegando así a “la crisis del software”.

Debido a este fenómeno se convoca una conferencia para llegar a una solución en el desarrollo de software, es aquí donde surge la rama de la ingeniería de software donde se definen como el conjunto de conocimientos que había en un inicio. Así como fue avanzando el tiempo, el crecimiento de sistemas, redes y un acceso inmediato generó que los desarrolladores entrarán en conflicto por la complejidad que conlleva esos tipos de sistemas dando así a la definición de fases para el software; requerimientos, codificación y pruebas, así también un concepto de diseño, validación y verificación.

## ***Mitos de la Ingeniería de software***

Desde los inicios de la computación esta se ha visto envuelta en miles de mitos y leyendas, por desgracia esto ha llevado a consecuencias para los trabajadores y administradores de los proyectos de SW, en muchos casos los administradores son a los que les recae la mayor parte de la presión, por ello tienden a cometer errores debido a ciertos mitos, por ejemplo:

*“Contratar a más programadores para terminar más rápido el proyecto”*

*“Contratar a terceros para terminar más rápido o tener más tiempo libre”*

Desgraciadamente no todo lo que brilla es oro, gracias a estos proyectos se crean conflictos internos los cuales generan más caos y en vez de servir para finalizar el proyecto lo único que logran es retrasarlo más.

No solo los generadores de SW tienden a caer en los mitos, también los clientes tienden a formar ilusiones o mitos acerca de lo fácil o no que puede ser crear un SW funcional, estos clientes se llenan de expectativas lo cual es dañino para la relación de cliente-programador

*“Para comenzar a escribir un código solo se necesita una oración y nada más”*

Este es uno de los mitos más recurrentes en personas que no tienen idea de cómo es la creación de un proyecto.

Como excluir a los profesionales en este ámbito los cuales aun con su experiencia no dejan los malos hábitos, estos hábitos fueron alimentados por más de 50 años de cultura de programación, por ello es difícil que se terminen estas actitudes.

*“Una vez entregado el programa nuestro trabajo ha terminado”*

*“El único producto que se entrega es uno exitoso ya que cumple con su objetivo”*

 Estos son los mitos más recurrentes en los desarrolladores, los cuales aun a pesar de su experiencia no se deshacen de estos malos hábitos.

Fuera del mundo del desarrollo también hay mitos y leyendas por ejemplo cuántas personas no aseguran que “el software privado es mucho mejor que el libre” argumentando que este es muy fácil de vulnerar, siendo esto una total mentira. Estos mitos reflejan la desinformación de las personas en cuanto al mundo de la computación. También se puede considerar mito la idealización de hackers de las películas, ya que el término hacker está mal orientado, muchas veces estas ideas son con las que crecimos y nos afectan de una u otra manera.

Uno de los mitos más grandes fuera del ámbito del desarrollo fue el llamado “*Y2K el cual decía que a partir de cierto año las computadoras dejarían de funcionar, muchas personas tiraron sus equipos teniendo miedo a perder sus datos”*, la verdad de esta leyenda era algo tan simple como un error de programación a base de estructuras de menor número de bits

Estos mitos más que fomentar una idea positiva generan caos en todo tipo de sentido, desde el punto en el que la comunicación cliente-desarrollador se ve lastimada tanto como que en los equipos de desarrollo tienen problemas internos.

El acabar con ellos generará un ambiente laboral más práctico, ya que se podrá tener una idea más precisa para los objetivos grupales.

## ***Crisis del software***

La crisis del software se dio a principios de los años 70, este término describe las dificultades para resolver problemas de la época, además tampoco existían técnicas establecidas para la creación del software como un modelo estándar.

Básicamente, la crisis del software se refiere a la dificultad en escribir programas libres de defectos, fácilmente comprensibles, y que sean verificables.

Las causas son, entre otras, la complejidad de lo que se trataba programar de manera primitiva y antes de esto los cambios a los que se tenían que ver sometidos los programas para ser continuamente utilizadas y adaptadas a las necesidades de los usuarios.

Otra causa de la crisis del software es que durante finales de los años 50 y principios de los 60, el trabajo computacional estaba dando sus primeros pasos y desde nuestro punto de vista actual, los programas eran muy sencillos y con bastantes fallas. El proceso para el desarrollo de software no tenía una metodología definida y por tanto provocaba que cada programador hiciera el trabajo a su manera dando como consecuencia poca compatibilidad con los demás programas y dispositivos.

El conjunto de problemas encontrados en el desarrollo del software de computadoras no se limita al software que no funciona correctamente. Es más, el mal abarca los problemas asociados a cómo desarrollar software, cómo mantener el volumen cada vez mayor de software existente.

El desarrollo del software estaba en crisis, pues se necesitaban nuevas técnicas y métodos para controlar la complejidad de los sistemas grandes. Sin embargo, cuanto más crezca nuestra capacidad para producir software, también lo hará la complejidad de los sistemas solicitados.

## ***Conceptos básicos de la Ingeniería de software***

**SOFTWARE:**

Los programas de cómputo cuentan por lo general con una documentación la cual explica de manera extensa la forma correcta de usar el programa, además también contienen datos como;

*“requerimientos, modelos de diseño y manuales de usuario.”*

Los componentes básicos de esta área son los siguientes

Ciencias de la Computación: el cual concierne a la teoría y fundamentos de cualquier sistema de cómputo (hardware o software)

Proceso: este es el marco de trabajo, el cual permite un desarrollo racional y oportuno de la ingeniería de software.

Método: Este indica cómo construir el software. Se incluyen técnicas de modelado.

Herramientas: las cuales proporcionan el soporte automático para el proceso y para los métodos.

CASE: es Computer-Aided Software Engineering

Programas que son usados para dar soporte automatizado a las actividades del proceso de software.

ACM: (Asociación de los Sistemas Informáticos) organización mundial que reúne profesionales interesados en las ciencias de la computación.

Actividad de requisitos: es la obtención, análisis, especificaciones y validaciones de requisitos de software.

Algoritmia: ciencia que nos permite evaluar el efecto que tienen diferentes factores externos sobre los algoritmos disponibles.

Complejidad: aquello que posee la cualidad de estar relacionado con muchos aspectos.

Metodología: es el conjunto de métodos que se siguen en una investigación científica.

Modelo: es la representación de un objeto, sistema o idea, de forma diferente al de la entidad misma.

Requerimiento: Características que se desea que posea un sistema o un software.

# **2. Los tipos de software**

## **Clasificación de tipos de software**

Existen tres principales tipos a la hora de hablar de clasificaciones de software, estas toman en cuenta el fin con el que se creó al software en cuestión, estos serían: 1) Software de sistema, 2) Software de programación y 3) Software de aplicación.

Para empezar con el más importante el software de sistema es la base de los dos siguientes pues es en sí la interfaz misma con la que se manejan los otros por lo tanto sin este no podrían existir los otros, literalmente es el sistema operativo del dispositivo por lo que se encarga de todo tipo de procesos como gestionar los recursos físicos de la máquina, administrar el uso de la memoria, coordinar tareas y también sirve para ofrecer una experiencia de uso sencilla para poder manipular e inclusive editar el sistema mismo conectando las aplicaciones con los recursos de hardware con los que cuente el dispositivo. Los softwares de sistema se dividen en diferentes tipos como: Cargador de programa, sistemas operativos, controladores, herramientas de diagnóstico y servidores. Algunos ejemplos de software de sistema de tipo sistema operativo son Windows, MacOS y Linux.

Continuamos con el software de programación el cual también tiene su correspondiente nivel de importancia siendo el puente entre el software operativo y el software de aplicación convirtiéndose en una parte vital para el correcto funcionamiento en unión de los tres softwares, principalmente esta clasificación hace referencia a los lenguajes de programación como herramientas para el diseño de utilidades digitales que ejerzan diferentes funciones, por lo tanto en este software se desarrollan nuevos sistemas ocupando como medio o base un sistema operativo. El software de programación también se divide en diferentes tipos, por ejemplo: Editores de texto, compiladores, intérpretes, enlazadores, depuradores e IDEs. Clion, CodeBlocks y Visual Studio entran en la clasificación de software de programación de tipo IDE.

Por último, pero no por eso menos importante tenemos a el software de aplicación que es el resultado del empleo en conjunto de los otros dos softwares, estos son programados dentro de sistemas operativos con extensos y diversos propósitos que se pueden encontrar en los distintos ámbitos de la vida diaria, estos softwares son ejecutados a través de toda clase de dispositivos móviles y computadoras, se puede decir que este es el software final que está destinado a un público para su consumo. Como es de esperar el software de aplicación también se divide en algunos tipos: Aplicaciones de ofimática, base de datos, videojuegos, software empresarial, software educativo y software de gestión. Street Fighter, Grand Theft Auto V y Valorant serían diferentes ejemplos de software de aplicación del tipo videojuegos.

## ***La importancia de conocer***

De acuerdo con Sommerville la Ingeniería de Software resulta especialmente importante por dos razones:1) cada vez es más frecuente que las personas y las sociedades se apoyen en sistemas de software más avanzados y complejos por lo que es necesario producir software confiable, de manera económica y rápida; y 2) generalmente resulta más barato el uso de métodos y técnicas específicas de ingeniería de software a largo plazo que solo diseñar y codificar software, como si fuese un proyecto de desarrollo personal.

Como hemos hablado, la ingeniería de software ha evolucionado a través del tiempo. Empezando con el término “Desarrollo”, un desarrollo que no llevaba un proceso, una planificación o un control mayor más que la única persona trabajando en el software hasta usar el término “Ingeniería de software” como un conjunto de conocimientos en un estado inicial.

La creación de nuevos tipos de software llevó a la clasificación de estos, ya que se volvieron parte de la economía de un país. Así como ayuda a poder diferenciar entre su función y enfoque, ayuda a poder ser comercializado a diferentes tipos de mercados, públicos para su compra/venta. Dentro del diferentes tipos de campos, entre ellos el laboral, entra la pregunta sobre qué tipo de software necesito, que necesidades tiene como persona, cliente, empresa u organización. Sea para fines educativos, de entretenimiento o una mejor accesibilidad hacia los datos que se trabajen. Todo esto, es decir, estos factores son comúnmente desconocidos para aquellas personas que no tienen un enfoque claro hacia lo que quieren, provocando costos económicos e incluso costos de recurso humano, ya que existe la posibilidad que el trabajador/persona no conozca el software y sea no tenga el rendimiento esperado.

# ***3. La ética de software y el código de ética del equipo de trabajo***

A primera vista la ingeniería de software podría parecer solo la gestión, administración, desarrollo e implementación de un código pero en la realidad estos sistemas que creamos son parte de nuestra vida en prácticamente todos los aspectos, convivimos con estos día a día y están involucrados en actividades delicadas donde un error de programación podría resultar fatal, desde pérdidas económicas hasta pérdidas humanas, en menor o mayor medida el código rige sobre nuestras vidas por ello es necesario que los ingenieros de software quienes tienen la oportunidad de hacer, permitir a otros hacer o influenciar a otros a hacer el bien o el mal, tengan y cumplan con lineamientos éticos que velen por el bienestar social, empezando por respetar la profesión siendo honestos, cumpliendo con la ley y además seguir principios que sean moralmente correctos.

La responsabilidad profesional se puede sintetizar en cuatro puntos: Confidencialidad, como ingeniero sin importar si se ha firmado o no un acuerdo de confidencialidad debes respetar la confidencialidad de trabajadores y clientes; Competencia, un ingeniero no debe tergiversar su nivel de competencia, deben negarse a trabajo del que son plenamente conscientes supera sus capacidades; Derechos de propiedad intelectual, el ingeniero debe conocer las leyes locales sobre el uso de propiedad intelectual como patentes y derechos de autor para asegurar protección de los derechos de sus empleados y clientes; Uso inadecuado de la computadora, evitar el uso inapropiado de las habilidades técnicas del ingeniero en computadoras ajenas.

Un ejemplo de lineamientos éticos es el código de ética de la ACM/IEEE, esta organización reconocida mundialmente como la asociación número uno para los profesionales de la computación practica ocho principios éticos que hablan sobre distintos temas pero que se conectan por su lineamiento moral, son reglas que todo ingeniero de software debe seguir como actuar con respecto al interés público, buscar satisfacer el interés de su empelado y cliente siendo coherente con el interés público, asegurar que sus productos sean de alta calidad cumpliendo con los estándares profesionales más altos que sean posibles, un juicio personal integro e independiente, administración ética del software, impulsar la integridad y reputación de la profesión, ayudar y ser justos con sus colegas, por ultimo nunca dejar de aprender además de promover el acercamiento ético de practicar esta profesión.

Los ocho principios mencionados son realmente la base que todo equipo de trabajo de software sigue hoy en día o que debería seguir, son internacionalmente reconocidos y según cada institución, empresa o profesional incluirá una cantidad de cláusulas analizando a fondo el fin de cada principio y como es aplicado por estos, en general los principios éticos buscan la protección de la sociedad y los individuos relacionados con profesionales de software, que las habilidades de los ingenieros no sean usadas para dañar a los individuos, un buen ingeniero debe respetar estos lineamientos además de incitar y propagar su uso pues al final el objetivo de estas son ser compartidas para generar una comunidad de profesionales comprometidos con el bienestar social y el crecimiento personal.

Nuestro equipo de trabajo se basará por ende en los siguientes 8 principios éticos para el desarrollo del software:

1. *Sociedad*: Los ingenieros de software actuarán de manera coherente con el interés social.
2. *Cliente y Empresario*: los ingenieros de software actuarán de manera que produzca el mejor resultado para cliente y empresario, y de manera coherente con el interés social.
3. *Producto*: los ingenieros de software garantizarán que sus productos y las modificaciones correspondientes cumplen los mayores estándares profesionales posibles.
4. *Valoración*: los ingenieros de software mantendrán la integridad e independencia en sus valoraciones profesionales.
5. *Gestión*: los líderes y gestores de ingeniería de software suscribirán y promoverán un enfoque ético en la gestión del desarrollo y mantenimiento del software.
6. *Profesión*: los ingenieros de software avanzarán en la integridad y reputación de la profesión, de manera consistente con el interés social.
7. *Compañeros*: los ingenieros del software serán justos y apoyarán a sus compañeros.
8. *Personal*: los ingenieros del software participarán en el aprendizaje continuo referente a la práctica de su profesión y promoverán un enfoque ético en la práctica de la profesión.

# **4.-Los roles del equipo, tomando como base el ciclo de vida de software.**

## ***Ciclo de vida del software y roles de equipo***

Se le conoce como al conjunto de fases por las que pasa el sistema que se está desarrollando, desde el surgimiento de la idea inicial hasta que el software es retirado o reemplazado.

La ingeniería de software tiene o se basa en diferentes tipos de modelos que establecen distintas etapas. El primer modelo fue establecido por Winston W. Royce conocido como “Cascada” en 1970 en cual se trabajará en este proyecto.

* *Analistas*

Son los encargados de trabajar junto con el cliente para tener la especificación correcta para la construcción de software.

Para que su trabajo no entre en confusión para los integrantes del grupo se utiliza un estándar para realizar el análisis del problema y dividirse en dos sub-fases, especificación de requisitos de usuario y especificación de requisitos de software.

Durante este proceso se realizarán reuniones con el cliente para poder atender a lo que necesita, deben de ser planificadas con una fecha de inicio y término. En cada reunión se deberá entrevistar al cliente, ayudando a identificar sus necesidades pudiendo utilizar ejemplos para de otros sistemas, verificar si los requisitos que especificó son correctos, definir una estructura básica del sistema, analizar los requisitos, la estructura definida y la definición de la arquitectura del sistema. Estas actividades principales generarán documentos para poder mostrar la evolución del mismo sistema al cliente de una manera comprensible.

Una analista deberá ser capaz de comunicarse, ser una persona socia para poder expresar sus ideas en un lenguaje compartido con el cliente. Deben conocer y manejar los métodos y tecnologías para realizar correctamente las fases de análisis. El cliente es lo más importante al momento de querer conocer lo que él quiere, entonces el escuchar y entender deberá estar dentro de sus capacidades. La creatividad y la familiarización con diferentes lenguajes de programación le permitirá establecer diferentes alternativas para escoger el modelo apropiado para la construcción del sistema.

* *Diseñadores*

Según Pressman, el diseño del software, en general, es el establecimiento de estructuras de datos, arquitectura general del software y representación de interfaz. El proceso de diseño traduce requisitos en una representación de software.

Un diseñador, como su nombre lo dice, es el encargado del diseño arquitectónico y detallado del sistema, la generación de prototipos con ayuda de analistas y programadores, la generación de un documento de diseño que se actualizará durante el proyecto y esperar que el producto final sea de acuerdo con el diseño principal.

* *Programadores y Depuradores*

Tienen como actividad esencial el transformar toda aquella especificación del sistema en un código fuente utilizando aquellos lenguajes de programación que dominen o el que se escogió en acuerdos anteriores además de reducir la complejidad del software en desarrollo.

Debe tener comunicación con los analistas y diseñadores para poder seleccionar un ambiente apropiado, con los ingenieros de testeo para que las actividades a realizar sean rápidas, eficientes y confiable para dar un código fuente con utilidad.

Aunque el programador se le designe la codificación también debe de realizar la escritura de la documentación técnica. El apoyo en diferentes tipos de herramienta de software entra los depuradores (“debugger”) que es un programa diseñado para la ayuda de detectar, localizar y corregir errores de otro programa permitiendo que se analice el programa en el que se trabaja.

Un programador debe tener conocimientos de diferentes tipos de ambientes y con ayuda de los analistas y diseñadores y descarten los no apropiados, experiencia en desarrollo de aplicaciones, el conocimiento de diferentes lenguajes de programación, así como el dominio sobre el lenguaje a trabajar y de ser posible que tenga experiencia en el tipo de proyecto a realizar.

* *Téster*

El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos, es decir, la realización de las pruebas para la detección de errores.

Los téster deben utilizar un conjunto de métodos que, en forma sistemática, organizada y estructurada, permite detectar y corregir errores en el desarrollo del sistema. Usualmente utilizan dos técnicas: test de la caja blanca y test de la caja negra.

Las pruebas de la caja blanca utilizan la estructura de control del diseño para derivar los casos de prueba. Con esta técnica puede asegurarse que todas las trayectorias independientes han sido visitadas, ejecutar las decisiones lógicas en ambos lados (verdadero o falso), así como sus ‘los’/ciclos dentro sus límites y límites operacionales, además de asegurar la validez de las estructuras de datos internas.

Las pruebas de la caja negra es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, es decir, se enfoca en las entradas y salidas del sistema, es decir, en funciones incorrectas, errores en interfaces, errores en estructura de datos o rendimiento del sistema. Los detalles de cada salida y entrada se basan en los requerimientos del mismo software.

Un téster deberá ser un buen programador en el lenguaje seleccionado, además de tener experiencia en el desarrollo de sistemas, tener una personalidad agresiva para la búsqueda de errores en el código y documentos del proyecto y una personalidad amigable, ya que se debe relacionar con la gran mayoría del equipo de desarrollo.

* *Aseguradores de calidad.*

De acuerdo con la presión que se ejerce por cumplir dentro del parámetro de tiempo estipulado, reduciendo sus costos y afectando la calidad del producto.  
La participación en las revisiones técnicas formales (RTF) es una de las actividades que destaca dentro de los aseguradores de calidad.

La RTF tiene como objetivo el descubrir errores en la función, lógica o la implementación de cualquier producto de software, verificar si cumple con los requisitos, que haya sido presentado de acuerdo con el estándar en uso, así como hacer el proyecto más manejable. Es un proceso preciso, con el afán de llegar a detectar lo antes posible los defectos o desviaciones en los productos que se van generando a lo largo del desarrollo.

Un asegurador de calidad debe ser una persona con bastante experiencia en el desarrollo de software, esto le permitirá la negociación con la calidad del producto y, en ocasiones, cambiar el enfoque de los desarrolladores.

* *Administrador de configuración.*

La administración de la configuración del software es la disciplina de administrar y controlar los cambios en la evolución de los sistemas de software. Ayuda a reducir problemas realizando modificaciones simultáneas, un código común, es decir, al existir una modificación es necesario la notificación al grupo de trabajo, sin una buena administración no habría una seguridad de poder informar a cada uno de los miembros del equipo y por último las versiones, en caso general hacia el usuario, otro en fase de pruebas y otro en desarrollo; si algún arreglo debe ser llevado a cabo, este arreglo debe ser extendido hacia todas las versiones existentes.

Un administrador de configuración tiene que poder manejar actividades administrativas, auxiliares y técnicas, contar con los recursos necesarios para que la solución a presentarse sea efectiva atribuyendo a influir el resultado de un evento de forma productiva.

* *Ingeniero de validación y verificación*

La verificación y validación es el conjunto de procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes.   
Con la verificación se comprueba que el software cumple los requisitos funcionales y no funcionales de su especificación y la validación comprueba que el software cumple las expectativas que el cliente espera, pero a como se debe considerar el software nunca estará libre de defectos.

Existen dos técnicas de verificación y validación:

1.- Inspecciones de software.

Analizan las diferentes representaciones del sistema en busca de defectos, además de que se realizan durante todo el ciclo del desarrollo.

2.-Pruebas del software.

Se evidencia las respuestas del software a series de pruebas para comprobar que su desempeño y funcionalidad sea conforme a lo requerido.

* *Documentador*

Tiene como enfoque el mantener la información que se genera durante el proceso del desarrollo del software. De manera específica la información debe ir al día, la apariencia y estructura debe ser concisa permitiendo la facilidad de almacenamiento, recuperación e intercambio de información a lo que conlleva a que todo cambio en el sistema deba ser reflejado en la documentación correspondiente, además de construir un manual para la capacitación sobre el uso del sistema, esto incluye todos los materiales que faciliten el entendimiento del producto.

En un inicio debe establecer formatos usado en documentos en proyectos, delimitando la estructura del documento.

Podrá apoyar al diseño y definición de interfaces, de los perfiles de acceso de los usuarios, así como el protocolo para el uso de documentos.

* *Ingeniero de manutención*

Dentro de sus actividades está el diagnóstico y corrección del programa encontrados en el uso del programa, la habilidad de poder adaptar el sistema a los posibles cambios que se puedan realizar en los aspectos de hardware o software y poder cumplir con la demanda por parte de los usuarios del sistema dando así mayor confiabilidad al sistema.

Los métodos que considera es la organización para definir procesos de evaluación de eventos estándar en secuencia, sistema de registro e información de actividades para el mantenimiento del sistema. Se puede llegar a entender que forma parte de la administración del software, pero se espera que, además de poder aportar soluciones pueda prevenir y tener en cuenta las nuevas actividades de manutención a futuro.

De acuerdo con la reflexión que tuvimos como equipo hemos decidido que los roles óptimos para cada persona y los que nos facilitaran la realización del software basados en las fortalezas y debilidades de cada uno serán los siguientes:

Manuel Alejandro Ramírez Salinas: Ingeniero de Manutención y Diseñador.

* *Abraham Sarid Acosta Herrera: Tester.*
* *Matías Hernández: Administrador de Configuración.*
* *Jaime Antonio Altamirano: Ingeniera de validación y verificación.*
* *Abner Elías Cid Ramírez: Programador y Asegurador de calidad.*
* *Pablo García Bravo: Analista y Documentador.*

# **5. Gestión y administración de proyectos**

**¿QUÉ ES UN PROYECTO?**

Un proyecto, en un sentido amplio, es un conjunto de actividades que se desarrolla durante un tiempo en un equipo de personas para tener un resultado. Tiene un objetivo, un inicio y un final. La temporalidad es un elemento clave y diferencial de un proyecto hacia a otra clase de proceso.

Podemos decir que un proyecto es un encargo específico que realiza una organización a un grupo interno o externo de personas que se establece para su ejecución.

Sin embargo, un proyecto de software tiene características especiales:

-Se parecen, es decir, el uso de las metodologías que se utilizan y a los productos. Muchos de estos son estándar para resolver una clase de problemas.

-Tiende a referirse a su estabilidad, volatilidad, nivel y expansión del servicio debido al cambio tecnológico tan rápido en el entorno.

Los objetivos de los proyectos son complejos y su interacción dentro de la organización es continua y cambiante entonces se ve en la necesidad en tener en cuenta todos los elementos y dar un marco general donde la manera de trabajar, técnicas y herramientas.

Con esto en mente podemos definir que un proyecto de software es una secuencia de actividades que desarrolla durante un tiempo predeterminado y con unos recursos limitados un equipo de personas, informáticos y no informáticos para poder obtener resultados. Gran parte de estas actividades requieren conocimientos y habilidades.

## **Aprobación y definición del proyecto**

El proyecto de software nace como la solución a un problema o a una necesidad de una organización, abriendo camino para el cambio, la creación del proyecto es un momento que requiere mucha creatividad e imaginación pues es donde se responden las preguntas de ¿Qué hacer?, ¿Por qué hacerlo?, ¿Es viable?, ¿Puede y debe hacerse?

En estas primeras etapas se concibe un proyecto transformando problemas y oportunidades en ideas, se realiza el estudio de viabilidad, la presentación inicial para su aprobación, su definición de necesidades y requerimientos, los riesgos del proyecto, la definición definitiva del proyecto, visualizar aspectos económicos, contractuales, comerciales, compradores, vendedores, políticos y gestionar expectativas.

Fase 1: Aprobación

1. Diagnóstico y conceptualización: El problema u oportunidad se identifica y documenta, con una buena identificación y diagnostico se pude generar un proyecto, utilizando técnicas de identificación de problemas, generación de ideas y priorización se logran mejores resultados.
2. Análisis de viabilidad: Terminando la etapa 1 queda una lista de candidatos a convertirse en proyecto, se realizan estudios de la viabilidad técnica y económica, además de predecir objetivos, resultados y costes.
3. Selección y aprobación: Se seleccionará según las mejoras en una empresa como aumentar ingresos y reducir gastos, existen métodos cuantitativos y cualitativos por la naturaleza de los aspectos a tomar en cuenta para su aprobación.

Fase 2: Definición

1. Definición de requerimientos: Se plasman como petición de propuesta, frecuentemente es de naturaleza contractual, aquí se detallan totalmente y pretenden ser definitivos, se trabaja junto con los clientes para entender y documentar a profundidad.
2. Análisis de riesgos: Todos los proyectos tienen contingencias que afectan a las especificaciones, los costes o el tiempo. Por lo que se establecen mecanismos de observación y prevención, mitigación y plan de contingencia.
3. Preparación de la propuesta: Revisar las especificaciones elaboradas por el cliente, se dirige al gerente o jefe, se presenta la calidad técnica, el presupuesto, el equipo de trabajo y la experiencia.

## **Planificación del proyecto**

La planificación es una etapa de suma importancia en el desarrollo de un proyecto ya que se establece el mapa de ruta que hay que seguir en el proyecto, es decir marca el camino que debe seguir nuestro proyecto para orientarnos y ordenar el trabajo del equipo entero para conseguir el objetivo principal.

Podemos determinar qué hay que hacer, quién lo hará, en qué tiempo y con qué recursos, para cumplir un objetivo, un plan de proyecto se puede considerar como *“un mapa de ruta estructurado que establece todas las actividades que hay que hacer para alcanzar los objetivos de negocio” ,* elplanificar proyectos es, en definitiva, estructurar y describir las actividades requeridas para alcanzar los objetivos del proyecto, teniendo en cuenta las responsabilidades y recursos requeridos en cantidad y experiencia.

El planificar es un instrumento de comunicación y dialogo, con el cliente y dentro del equipo de trabajo, por ello, la planificación de proyectos no puede ni deber considerarse un proceso estático.

El plan de un proyecto debe contemplar los siguientes elementos:

*“Los objetivos y los resultados que se esperan del proyecto, de manera que permita la evaluación del éxito o fracaso del proyecto, tal como se han descrito en los capítulos anteriores”*

*“Los mecanismos de control del alcance del proyecto, y de gestión de cambios en éste.”*

*“La involucración de los distintos agentes participantes en el proyecto, sus roles y responsabilidades.”*

*“La organización y el equipo asignado al proyecto, con la matriz de roles y responsabilidades para los diferentes hitos y actividades.”*

## **Factores críticos de éxito de un proyecto**

Es difícil encontrar éxitos y fracasos completos en cualquier clase de proyectos. En términos generales, un proyecto se considera un fracaso si no se alcanzaron los objetivos o resultados esperados, sobrepasado el tiempo asignado, aumento de costos o recursos y que los estándares de calidad deseados no fueron los esperados.

Estas son variables los que dependen uno de otro de ser exitoso o ser un fracaso.

Se denomina factores críticos de éxito a las condiciones necesarios individualmente y en un conjunto suficientes para que ocurra el éxito del proyecto.

Existen lo que se llama “Los 10 mandamientos de la gestión de proyectos informáticos”:

-Beneficio de negocio: Se establecen claramente el valor y beneficios de negocio que se obtienen al realizar el proyecto.

-Objetivos y resultados del proyecto: Se establecen los objetivos, resultados y productos que se quieren obtener.

-Alcance y limitaciones: Se establece el alcance y las limitaciones del trabajo.

-Planes detallados de hitos: Se realiza, controla y actualizan los planes detallados.

-Apoyo de la dirección: Se afirma constantemente el apoyo de dirección, en términos de autoridad, consistencia de los objetivos y provisión de recursos.

-Involucración de los usuarios: El usuario/cliente es escuchado y se interpreta las expectativas, se planifica y gestionan adecuadamente, además de asegurarse la aceptación del trabajo por parte del cliente y otras partes interesadas.

-Recursos y perfiles adecuados: Se asignan los recursos adecuados, con las habilidades necesarias, tanto técnicas como de gestión, así como otras habilidades funcionales que se requieran. Se definen los roles y responsabilidades de los miembros.

-Control y retroalimentación: Se evalúa y se obtiene retroalimentación a lo largo del proyecto.

-Tecnología estable: Uso de tecnologías para el personal formado en dicha tecnología y siempre disponible para su uso.

-Gestionar: Se debe identificar a tiempo las incidencias, crisis y desviaciones.

## **Ejecución del Proyecto.**

Lo anterior lleva a la ejecución, la cual es el trabajo del jefe del proyecto y del resto del equipo llevarlo día a día. La ejecución es como tal cuando ya se empieza a realizar dicha acción, en el mundo de la informática, donde normalmente se fabrica, se integra, se instala o se implanta un producto, una realidad física y tangible que funciona o no, que soluciona o no problemas.

La ejecución cubre básicamente dos aspectos:

-La gestión del día a día

- El control y seguimiento

La ejecución es una práctica y una disciplina, tan importante como la estrategia. Es el principal trabajo de cualquier líder, hacer que las cosas se hagan con ayuda de otros, que pase lo que prometimos, lo que tiene que pasar. La ejecución es la confrontación con la realidad. El control y el seguimiento son importantes, pero no sustituyen a la ejecución satisfactoria en tiempo, calidad y resultados.

***Componentes principales de la ejecución:***

Puede decirse que la ejecución se sostiene sobre cuatro pilares

-Las metodologías y conocimientos técnicos propios de cada tipo de proyecto.

-Las habilidades de liderazgo, organización y gestión del cambio, que hemos tratado en el capítulo anterior.

-Las necesidades de control interno y reporte, tanto de los aspectos ficticios como de los progresos en el tiempo y los aspectos económicos.

-Habilidades específicas de atención de incidencias, resolución de problemas y toma de decisiones.

***La gestión del proyecto en el día a día***

Como tal día a día el jefe de proyecto dará una meta a resolver, para así poder ir progresando entre todos como equipo esto no quiere decir que el jefe de proyecto no vaya a ser nada si no que este debe planificarlo y gestionar el tiempo que debe llevarse cada día, Además debe entrevistarse con las personas del cliente y con los miembros del equipo, formal e informalmente. Por último, debe efectivamente preparar y analizar los informes de control, reuniones de trabajo y presentaciones

***Seguimiento y control del proyecto***

“Reporting” y seguimiento son elementos instrumentales del control. Control quiere decir análisis y evaluación, previsión del futuro, comunicación y corrección.

## **Cierre y evaluación del proyecto**

La última etapa del proyecto es la de cierre. Los buenos proyectos acaban de forma controlada, resolviendo los problemas y flecos que surgen inevitablemente en la entrega, retirando ordenadamente los recursos y asegurando la satisfacción del cliente y su capacidad de usar los nuevos sistemas y recuperar los beneficios que se aspiraba a obtener.

El cierre de proyecto no es sólo un proceso administrativo. El cierre del proyecto es también un motivo de aprendizaje y de celebración. Es el momento de realizar la evaluación de resultados, desde el punto de vista del cliente y del equipo.

La documentación es uno de los aspectos más críticos de un sistema de información. Revela la historia del sistema, su diseño y construcción y sus objetivos. Sin documentación, sería muy difícil realizar cambios y modificaciones en el sistema, pues nadie conocería los archivos, informes y procedimientos que fueron realizados.

Desde la definición del proyecto, pasando por la planificación y luego la ejecución, lo más importante es entender y compartir con el cliente cuáles son los objetivos del proyecto y los productos a obtener. Es bueno establecer también las condiciones o procesos de aceptación de manera inequívoca.

## **Estimación del proyecto de software**

Es una metodología que describe una planificación del proyecto de software. Es una predicción en cuanto a tiempo y costo, este esfuerzo se verá reflejado en horas de trabajo y moneda económica.

El propósito de la estimación de software es realizar predicciones realistas basándose en la poca información que se tiene sobre el proyecto y de ahí partir a los escenarios posibles a enfrentarse considerando los mejores y peores casos.

Es importante medir el proceso de ingeniería de software y el producto que se elabora porque es la forma más objetiva de comprender y mejorar el proceso de desarrollo y el producto que se elabora. Si no se realizan mediciones, no hay forma de determinar si se está mejorando.

La estimación será utilizada principalmente para el desarrollo de planes del proyecto, para elaborar presupuestos, realizar análisis de las inversiones, fijar precios del software para el cliente y demás intereses que se quiera agregar.

En este proyecto, consideramos que nuestra técnica de Estimación se basa en la técnica de Estimación mediante juicio de expertos en la que se refiera a: Un experto con experiencia directa en el software apoye a un equipo analizando los requerimientos del software, hace observaciones y define el tiempo para desarrollar el proyecto.

En nuestro caso el experto es el Profesor que nos da las observaciones y guía para los requerimientos del software. Se debe asegurar que todos los integrantes del equipo tengan un mismo entendimiento de cómo debe funcionar el software a desarrollar y que se espera de él.

Una vez que el experto o grupo de expertos ha dividido el problema en actividades, pueden asignar un estimado de cada una.

Bajo el análisis pasado se considera que la estimación del tiempo para realizar el proyecto se encuentra en la siguiente tabla.

UNIDAD 1

|  |  |
| --- | --- |
| ACTIVIDADES | TIEMPO ESTIMADO |
| * **Documento del Proyecto: Conceptos Básicos de Ingeniería de Software. Clasificación de Sistemas de Software. Historia de la Ingeniería de Software, mitos y crisis del software** | SEMANA1 |
| * **Documento del Proyecto: ciclo de vida de Desarrollo y Código de ética del equipo de trabajo** | SEMANA2 |
| * **Documento del Proyecto: Gestión de Proyecto** | SEMANA3 |

La tabla muestra la división de actividades de la primera unidad sobre el proyecto que es un tiempo estimado de 3 semanas, posteriormente las unidades siguientes también se tienen contempladas y divididas en varias actividades, el tiempo estimado para acabado del proyecto es de 17 semanas.

## **Metodologías para la gestión del software**

Dentro de todas las diferentes metodologías que existen para llevar a cabo un proyecto, se pueden clasificar en tres tipos: secuencial, iterativo y ágil.

**1.Secuencial**

**1.1Cascada**

División de un proyecto en fases sin realimentación a la fase anterior Su principio se basa en el hecho de que solo una etapa concluida y validada permite pasar a la siguiente.

Reducción de riesgos, Intervención de los usuarios al final del ciclo de vida, con frecuencia muy tarde Controla de la calidad al final de cada fase. Está destinado a un proyecto menos de 1 año y debidamente reglamentado.

**1.2 Integración**

División de un proyecto en fases (inicialización del proyecto, preparación de la implementación y la implementación).

Un mejor control del presupuesto del proyecto.

**1.3 En “V”**

Es un derivado del ciclo de vida en cascada que considera las actividades vinculadas a las pruebas.

Proyectos de tamaño mediano y poco complejos

**1.4 RAD**

Construcción de la solución con el usuario. Se basa en cinco fases (La preparación define la organización, el perímetro y el plan de comunicación; en encuadre define los objetivos, las soluciones y los medios que se implementarán; el prototipo modela la solución y valida su coherencia; el desarrollo realiza la solución y la conclusión constituye una última verificación de calidad).

Implicación fuerte de parte del usuario. Experiencia de todos los participantes.

**2.- Iterativo**

**2.1 Incremental**

División del proyecto en dominios (cada uno tiene un ciclo de vida autónomo en cascada), permite la corrección de errores de desarrollo, diseño y especificación. Se usa Proyectos grandes

**2.2 Espiral**

Método iterativo basado en la realización de prototipos

Destinado a grandes proyectos internos complejos que son difíciles de contratar.

**3.-Ágiles**

**3.1 Scrum**

Método riguroso, Control continuo, ciclos de un mes, proyectos de tamaño pequeño.

**3.2 Programación extrema**

Entregable cada interacción de lo disponible, destinado a proyectos pequeños, flexibilidad al cambio, equipos pequeños.

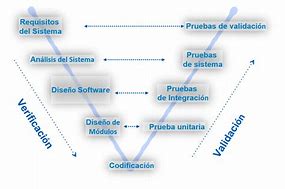
# **6.- Modelos de procesos de software**

Cuando se habla sobre procedimientos, normalmente se describen las actividades durante estos procesos como la especificación de un modelo de datos, diseño de IU (interfaz de usuario). Estas descripciones normalmente incluyen los productos, es decir, los resultados de una actividad, roles que reflejan las responsabilidades involucradas y las pre- y postcondiciones que son explicaciones que se utilizan durante el proceso de actividades.

No hay procesos correctos o incorrectos en el software, usualmente es impulsado por un plan donde todas las actividades son planeadas en el avance y progreso del mismo proceso. En la práctica, la mayoría de los grandes sistemas son desarrollados usando un proceso que incorpore todos los elementos de los modelos a querer usar.

## **1.-Modelo v**

El modelo V fue utilizado por primera vez en Alemania. Y apareció con su primera versión en agosto de 1992, El modelo fue proyectado en junio de 1997 y es prescrito en febrero del 2005 bajo la asignación de la V XT se dice que es para el desarrollo de los sistemas de información que proporciona una guía para la planificación y realización de proyectos.

Este modelo se centra en minimizar los riesgos del proyecto, Mejora la transparencia del proyecto y control del proyecto, especificando los enfoques estandarizados, describe los resultados correspondientes y funciones de responsabilidad. Permite una detección temprana de las desviaciones y los riesgos, mejora la gestión de procesos, reduciendo así los riesgos del proyecto.

La relación entre las etapas de desarrollo y los distintos tipos de pruebas facilitan la localización de fallos.

Algunas ventajas de este modelo son:

• Es un modelo sencillo y de fácil aprendizaje

• Hace explícito parte de la iteración y trabajo que hay que revisar

• Especifica bien los roles de los distintos tipos de pruebas a realizar

• Involucra al usuario en las pruebas

Sus desventajas:

• Es difícil que el cliente exponga explícitamente todos los requisitos

• El cliente debe tener paciencia pues obtendrá el producto al final del ciclo de vida

• Las pruebas pueden ser caras y, a veces, no lo suficientemente efectivas

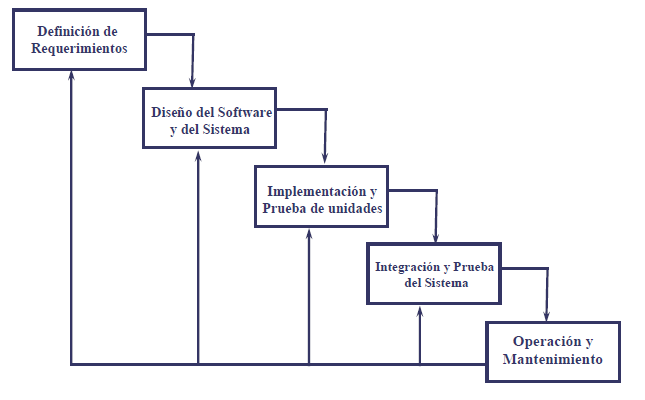
• El producto final obtenido puede que no refleje todos los requisitos del usuario

## **2.-Cascada**

El modelo de cascada es un ordenamiento riguroso de un proceso por etapas en el desarrollo de software, este enfoque metodológico también llamado “Lineal Secuencial” se basa en planes con distintas especificaciones y desarrollos para fases separadas.

Originalmente creado y propuesto por Winston W. Royce en 1970, después el modelo fue revisado por Barry Boehm en 1980 y finalmente en 1985 Ian Sommerville.

Fases del modelo:



Ventajas:

-Modelo común y muy usado.

-Orientado a documentos.

-Entendimiento e implementación sencilla.

-Pocas herramientas para hacerlo funcionar de manera óptima.

-Efectividad: Definir antes que diseñar, diseñar antes que codificar.

Desventajas:

-Los proyectos rara vez siguen una secuencia lineal.

-Tarda mucho tiempo el proceso.

-Si el software no está completo no se opera.

-Errores en una etapa significan rediseño y nueva programación.

Reglas:

* Para empezar cada etapa debe terminarse antes la anterior.
* Debe seguir la secuencia de fases.
* Al final de cada etapa se lleva a cabo una revisión final.
* Si pasa la revisión se avanza sino se vuelve a realizar la etapa.
* No se pueden saltar etapas o dejarse sin terminar o revisar.
* Solo puede haber cambios tras finalizarse la etapa actual.
* No se puede cambiar una etapa anterior se tiene que volver a iniciar.
* El software solo se opera una vez finalizado.

Por lo tanto, este software aprovecha sus bondades en grandes proyectos donde se tiene que trabajar el sistema en varios sitios por lo que su rigidez y orden ayudan a mantener un ritmo eficiente.

## **3.-Prototipado**

Un prototipo es un modelo experimental de un sistema o de un componente de un sistema que tiene los suficientes que permiten su uso.

Su estructura general es:

* Escuchar al cliente.
* Construir/revisar maqueta.
* Probar maqueta.

Objetivos:

Son un medio eficaz para aclarar los requisitos de los usuarios e identificar las características de un sistema que deben cambiarse o añadirse.

Características:

1. Es una aplicación que funciona.
2. Su finalidad es probar varias suposiciones con respecto a las características requeridas por el sistema.
3. Se crea con rapidez.
4. Evolucionan a través de un proceso iterativo.
5. Tienen un costo bajo de desarrollo.

Reglas:

* Se debe de contar con la participación del cliente.
* El modelo debe de estar actualizándose.
* Es útil cuando el cliente no identifica los requisitos detalladas.
* El responsable del desarrollo no está seguro de la eficiencia de un algoritmo, sistema operativo o de la interfaz hombre-máquina.

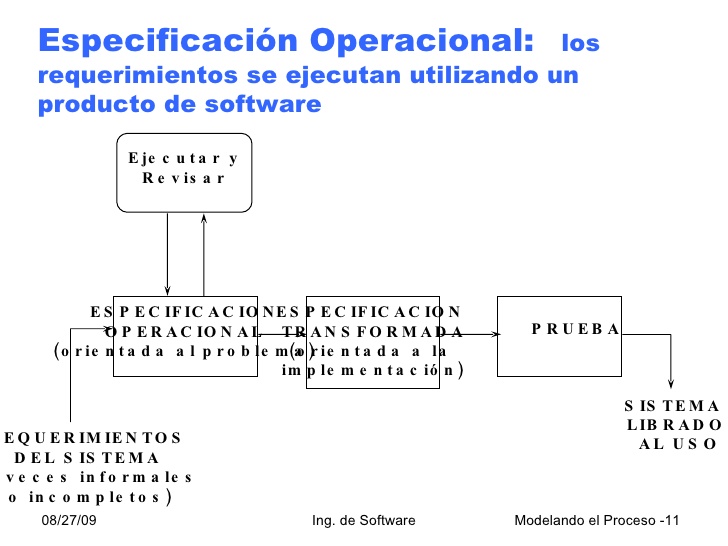
Ventajas: Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.

Desventajas:Su principal desventaja es que una vez que el cliente ha dado su aprobación final al prototipo y cree que está a punto de recibir el proyecto final, se encuentra con que es necesario reescribir buena parte del prototipo para hacerlo funcional.

Es posible que el prototipo sea muy lento, muy grande, no muy amigable en su uso, o incluso, que esté escrito en un lenguaje de programación inadecuado.

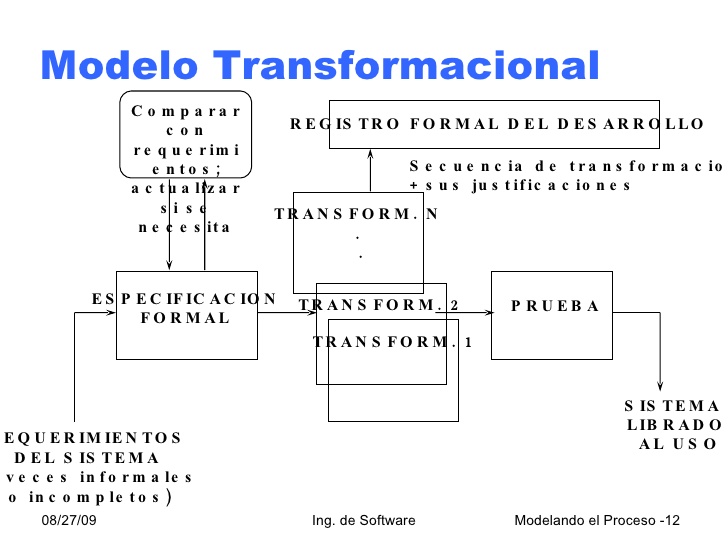
## **4.-Especificación Operacional**

El objetivo principal de la Especificación de Requisitos del Sistema (ERS) es servir como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores. En la ERS deben recogerse tanto las necesidades de clientes y usuarios (necesidades del negocio, también conocidas como requisitos de usuario, requisitos de cliente, necesidades de usuario, etc.) como los requisitos que debe cumplir el sistema software a desarrollar para satisfacer dichas necesidades.



## **5.-Modelo transformacional**

Se basa en disponer de una especificación formal del sistema y en transformar, con métodos matemáticos, está en una implementación. Si al momento de aplicarse son correctas se tiene que asegurar que el sistema satisface la especificación. Tiene la posibilidad de realizar el mantenimiento a nivel de la especificación por lo que no es necesario disponer una especificación inicial correcta y diseñadores calificados.



REGLAS DEL MODELADO

* La elección de los modelos tiene una profunda influencia sobre cómo se acomete el problema y se moldea la solución
* Todo modelo debe estar ligado a la realidad.
* Mejorar la productividad del equipo de desarrollo (por ej. los modelos se pueden usar para generar código de forma semiautomática)

* Reduce el número de defectos en el código (los modelos permiten una validación temprana del sistema)

* Facilitar la comprensión (con lo que además se facilita la integración de nuevos miembros en el equipo) y, ya de paso, documentan el sistema para el futuro.

* Mejorar la descomposición y modularización del software

* Facilitar la evolución y mantenimiento del software

* Mejorar la reusabilidad

## **6.-Modelo RUP**

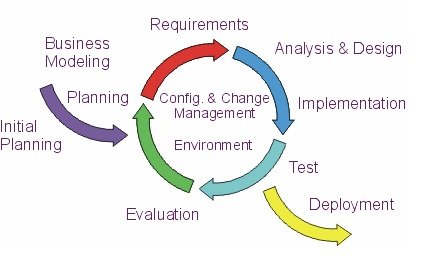
Proceso de desarrollo de software puesto a la venta en 1998 por la empresa Rational Software ahora propiedad de IBM, consiste en un conjunto de metodologías que funcionan según el contexto del proyecto, por lo tanto, el sistema RUP (Rational Unified Process o Proceso Unificado Racional) no es estricto en sus pasos.

**Inicio:**

El objetivo general de esta fase es establecer un acuerdo entre todos los interesados acerca de los objetivos del proyecto.

Es significativamente importante para el desarrollo de nuevo software, ya que se asegura de identificar los riesgos relacionados con el negocio y requerimientos.

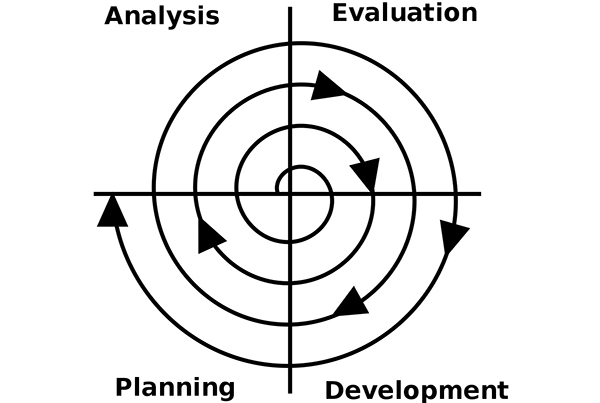
Para proyectos de mejora de software existente, esta fase es más breve y se centra en asegurar la viabilidad de desarrollar el proyecto.



**Reglas:**

* Este tiene que ser dirigido por casos de uso: Los casos de uso son los artefactos primarios para establecer el comportamiento deseado del sistema.
* Este tiene que centrarse en la Arquitectura: –La arquitectura es utilizada para conceptualizar, construir, administrar y evolucionar el sistema en desarrollo.
* Puede ser Iterativo e Incremental:  
  –Maneja una serie de entregas ejecutables.  
  –Integra continuamente la arquitectura para producir nuevas versiones mejoradas.
* Tiene que tener un enfoque orientado a objetos.

## **7.-Modelos Espiral**

Este modelo es un arquetipo del proceso de desarrollo de aplicaciones. Se basa en la hipótesis de que el desarrollo de software es un ciclo iterativo que se repite hasta alcanzar los objetivos establecidos.

Este es uno de los modelos más importantes para brindar soporte a la gestión de riesgo. Como su nombre indica, este modelo se muestra como en forma de espirar, donde las diferentes etapas del modelo se distribuyen en diferentes ciclos. El número de ciclos en este modelo no es fijo y pueden variar de un proyecto a otro.

El modelo espiral fue definido por el matemático y profesor de ingeniería de software estadounidense Barry Boehm. Parte de esta publicación de 1988 representaba gráficamente el modelo espiral, mostrando de manera íntegra cómo se ve el proceso de desarrollo de software en forma de espiral y respaldado por ciclos.

El modelo de espirar cuenta con diferentes características, por ejemplo:

Control del riesgo:

Este modelo se diferencia de otros debido a que reconoce explícitamente los riesgos.

Descripción de la espiral:

Cada giro de la espiral representa un ciclo completo, por donde siempre pasan los cuatro cuadrantes, que representan las cuatro etapas del modelo.

Flexible:

Es bastante flexible, al realizar por separado para cada fase del proyecto los procesos de definición de objetivos, análisis de riesgos, desarrollo y planificación.

Las diferentes etapas del modelo:

Determinar objetivos, alternativas y restricciones:

Los requerimientos del sistema se definen con el mayor detalle posible, incluyendo rendimiento, interfaces de hardware/software, indicadores claves de éxito, etc. y se consideran cuáles objetivos se deben asociar con el ciclo actual de desarrollo.

Evaluación de riesgos:

Se evalúan todas las alternativas propuestas. Los objetivos y restricciones sirven como referencias determinantes para seleccionar la mejor solución.

Desarrollo y prueba:

Se realiza todo el desarrollo necesario, utilizando la tecnología y solución seleccionada. Con cada iteración se va creando una mejor versión de la aplicación.

Planificación del próximo ciclo:

Se realiza todo el desarrollo necesario, utilizando la tecnología y solución seleccionada. Con cada iteración se va creando una mejor versión de la aplicación.

REGLAS:

1.- Se utiliza en proyectos donde el coste de un fallo es un gran riesgo.

2.- Es considerable aplicar este método en proyectos grandes, largos, caros y complejos.

3.- Este método debe ser utilizado para los riesgos.

4.- Se requiere la participación continua por parte del cliente.

## **8.-Desarrollo ágil de software**

El desarrollo de software ágil es un concepto usado en el desarrollo de software para describir las metodologías de desarrollo incrementales (Cohen, Lindvall & Costa, 2003). Es una alternativa en la gestión tradicional de proyectos TI, donde se hace hincapié en el empoderamiento de las personas para colaborar y tomar decisiones en equipo, además potencia la planificación continua, pruebas permanentes y la integración conjunta del código y los despliegues.

Las técnicas ágiles varían en prácticas y énfasis, pero comparten características comunes, incluyendo el desarrollo iterativo y un enfoque en la interacción, la comunicación y la reducción de artefactos intermedios que consumen muchos recursos

Para que una metodología sea considerada como ágil es necesario que cumpla con los siguientes puntos:

1.-Satisfacer al cliente con entregas tempranas y continuas de software valioso.

2.-Los requisitos cambiantes son bienvenidos, incluso en fases tardías del desarrollo.

3.-Entregar con frecuencia software funcionando, -de dos semanas a dos meses, - cuanto antes se haga mejor.

4.-El cliente y los desarrolladores deben trabajar juntos a diario a lo largo del proyecto.

5.-Individuos motivados. Darles el ambiente y el soporte que necesitan, y confiar en ellos para obtener el trabajo realizado.

6.-El método más eficiente y efectivo de transmitir información hacia y dentro del equipo es la conversación cara a cara.

7.-El software en funcionamiento es la medida principal de progreso.

8.-El desarrollo debe ser sostenible. Los participantes deben ser capaces de mantener un paso constante de manera indefinida.

9.-Atención continua a la excelencia técnica y a un buen diseño.

10.-La simplicidad es esencial, maximizando el avance del trabajo no realizado.

11.-Las mejores arquitecturas, los mejores requisitos y diseños emergen de equipos autoorganizados.

12.-A intervalos regulares el equipo refleja la forma en que se puede volver más efectivo, entonces su comportamiento se ajusta y adecua en concordancia.

Las metodologías ágiles son en realidad una familia de modelos o técnicas, todas ellas compartiendo la característica de interpretar el desarrollo de software como una actividad en la que siempre hay un cierto grado de incertidumbre. Incertidumbre que hace necesario poner el énfasis en las personas, dejar que se autoorganicen y que interactúen buscando siempre satisfacer los requisitos del cliente, y planificando iteración a iteración, adaptándose con flexibilidad a los cambios que se producirán de seguro durante la vida del proyecto.

El manifiesto ágil tiene **cuatro valoraciones importantes:**

1. El foco se debe poner más sobre las personas y las interacciones que sobre los procesos y herramientas.
2. El software que funciona es más importante que la documentación extensiva.
3. La colaboración con el cliente es más importante que la negociación contractual.
4. El proceso debería responder ante el cambio, en lugar de seguir un plan.

# **7.- procesos de la Ingeniería de requerimientos**

La Ingeniería de requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas.

Esta es la clave para desarrollar un proceso por el cual transformar los requerimientos dados por los clientes a especificaciones precisas, no ambiguas, consistentes y completas del comportamiento del sistema.

Para esto la Ingeniería de requerimientos se define como un “*Conjunto*” de actividades en las cuales, utilizando técnicas y herramientas, se analiza un problema y se concluye con las especificaciones de una solución por ello se dice que esta Ingeniería se utiliza para definir todas las actividades involucradas en el descubrimiento , documentación y mantenimiento de los requerimientos para un producto determinado

Para la ingeniería de requerimientos es necesario uno de estos dos puntos

*“Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo”*

*“Una condición o capacidad que debe estar presente en un sistema o componente de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.”*

Los requerimientos pueden dividirse en dos formas *“Funcionales”*

Y *“no funcionales”.*

* Los funcionales son capaces de realizar funciones. Además, describen las transformaciones que el sistema realiza sobre las entradas
* Los no funcionales se relacionan con las características de una u otra forma que puedan limitar el sistema, como, por ejemplo, el rendimiento (tiempo y espacio), interfaces de usuario, fiabilidad, (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

Estos requerimientos son muy específicos ya que cuentan con características necesarias, estas son las más importantes:

* **Necesario**: este requerimiento es necesario ya que sin él se provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso
* **Conciso**: Un requerimiento es conciso si es fácil de leer y de entender. Su redacción debe ser simple y clara para aquellos que vallan a consultarlos en un futuro
* **Completo**: un requerimiento esta completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.
* **Consistente**: Un requerimiento es consistente si no es contradictorio con otro requerimiento
* **No ambiguo**: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.
* **Verificable**: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas.

## **¿Por qué es importante la** Ingeniería de requerimientos?

Los principales beneficios que se obtienen de la Ingeniería de requerimientos son:

* + Permite gestionar las necesidades del proyecto en forma estructurada.
  + Mejora la capacidad de predecir cronogramas de proyectos
  + Disminuye los costos y retrasos del proyecto: muchos estudios han demostrado que reparar errores por un mal desarrollo no descubierto a tiempo es sumamente caro; especialmente aquellas decisiones tomadas durante la RE.
  + Mejora la calidad del software: la calidad en el software tiene que cumplir con un conjunto de requerimientos (funcionalidad, facilidad de uso, confiabilidad, desempeño, etc.)
  + Mejora la comunicación entre equipos: la especificación de requerimientos representa una forma de consenso entre clientes y desarrolladores. Si este consenso no ocurre el proyecto no será exitoso.
  + Evita rechazos de usuarios finales: la Ingeniería de requerimientos obliga al cliente a considerar sus requerimientos cuidadosamente y revisarlos dentro del marco del problema por lo que se involucra durante todo el desarrollo del proyecto.

# **8.-Requerimientos del proyecto**

Debido a que los cambios que introduce un sistema nuevo tienden a afectar a más de un tipo de usuario, los analistas de requisitos han de tomar en consideración a todos los implicados para que se obtengan y depuren sus requerimientos de la forma más fidedigna posible. Realmente, son muchas las personas involucradas en el desarrollo de los requerimientos de un sistema.

Es importante saber que cada una de esas personas tienen diversos intereses y juegan roles específicos dentro de la planificación del proyecto; el conocimiento de cada papel desempeñado asegura que se involucren a las personas correctas en las diferentes fases del ciclo de vida, y en las diferentes actividades de la IR.

Los roles más importantes pueden clasificarse como sigue:

• Usuario final: Son las personas que usarán el sistema desarrollado. Ellos están relacionados con la usabilidad, la disponibilidad y la fiabilidad del sistema; están familiarizados con los procesos específicos que debe realizar el software, dentro de los parámetros de su ambiente laboral. Serán quienes utilicen las interfaces y los manuales de usuario.

• Usuario Líder: Son los individuos que comprenden el ambiente del sistema o el dominio del problema en donde será empleado el software desarrollado. Ellos proporcionan al equipo técnico los detalles y requerimientos de las interfaces del sistema.

• Personal de Mantenimiento: Para proyectos que requieran un mantenimiento eventual, estas personas son las responsables de la administración de cambios, de la implementación y resolución de anomalías. Su trabajo consiste en revisar y mejorar los procesos del producto ya finalizado.

• Analistas y programadores: Son los responsables del desarrollo del producto en sí; ellos interactúan directamente con el cliente.

• Personal de pruebas: Se encargan de elaborar y ejecutar el plan de pruebas para asegurar que las condiciones presentadas por el sistema son las adecuadas. Son quienes van a validar si los requerimientos satisfacen las necesidades del cliente.

## **Pasos que seguir para nuestro proyecto**

1.-Definicion:

Para empezar con la creación de nuestro SW deberemos definir de manera explicita nuestro proyecto, para ello contaremos con los objetivos esperados de nuestro proyecto, con estos delimitaremos nuestros requisitos, es decir marcaremos los datos necesarios para la creación del SW de manera exitosa

2.-Analisis

Una vez teniendo nuestros limites procederemos al análisis de nuestro proyecto, en este momento tendremos que analizar nuestros objetivos iniciales para después crear una planeación acorde con nuestras metas de esto se encargara (*Pablo García Bravo*)

3.-Planeacion

Para este punto ya tendremos bien establecidos nuestros objetivos, con este avance procederemos asignar puestos específicos a nuestros integrantes, a partir de aquí nos encargaremos de planificar cada punto de nuestro proyecto, desde el diseño hasta la ejecución de esto se encargará (Manuel Alejandro Ramírez Salinas)

4.-Diseño

Los integrantes del equipo aportaran ideas con las cuales el encargado de diseño (*Manuel Alejandro Ramírez Salinas*) las tomara y se encargara en conjunto del planificador (*Pablo García Bravo*) de crear un diseño que tome en cuenta cada ida de los participantes y de misma manera satisfacer las necesidades del proyecto.

5.-Codificacion

Aquí necesitaremos a los encargados de programar el proyecto (*Abner Elías Cid Ramírez*) para codificar cada objetivo del mismo, nuestro planificador se encargará de especificar cada objetivo y el modelo que necesitará para estos.

7.-Evaluacion

Con la ayuda del tester(Abraham Sarid Acosta Herrera) y del analista(Manuel Alejandro Ramírez Salinas) se ejecutara el proyecto de SW para evaluar cada punto del mismo, es decir encargarse de encontrar vulnerabilidades y evaluar que el código cumpla con el objetivo esperado.

8.-Documentacion

En este punto se creará la documentación del proyecto especificando cada acción y posibilidad de este, el documento será muy importante ya que valdrá como un manual de uso, este será entregado en conjunto del proyecto ya que explica cada punto y el objetivo del mismo, de esto se encargará (*Pablo García Bravo*).

9.-Mantenimiento

Es importante hacer mantenimiento de código ya que esto hará que el SW este actualizado y en la mejor versión de este

Estos pasos explicados de forma general serán los que usaremos para la creación exitosa de nuestro proyecto a continuación especificare las metodologías que usaremos:

1.-Pre-requisitos

2.-Mision y visón

3.-ADDIE

4.-El proyecto será de tipo Aplicación

5.-Modelo de proceso de espiral

Estas metodologías nos ayudaran a crear un software completo que cumpla con los objetivos y metas del equipo.

**Técnicas de recolección de requerimientos de software.**

Tiene como componentes bases el dominio de aplicación, el problema a resolver, las necesidades y limitaciones de los interesados y el contexto de la organización o negocio en cuestión.

Trabaja en manera de espiral comenzando desde la elicitación (traspaso) de información, borradores de declaraciones de requerimientos, el análisis de requerimientos, los problemas de los requerimientos, la negociación y documentación de los requerimientos.

**Etapas de elicitación.**

*Establecimiento de objetivos.*

Los objetivos organizacionales deben de ser establecidos incluyendo metas generales del negocio, una descripción general del problema a resolver y el porque y limitantes del sistema.

*Adquisición de conocimientos previos.*

Conocimiento previo sobre el sistema, esto incluye la información sobre la organización donde el sistema debe de ser instalado además de información de sistemas existentes.

*Conocimiento de la organización y recolección de requerimientos de los interesados.*

Las partes interesadas del sistema son consultadas para conocer sus requerimientos además de que toda aquella información que fue siendo recolectada durante previas etapas debe ser organizada y depurada.

***TÉCNICAS***

*Análisis de documentación*

Consiste en obtener la información sobre los requerimientos funcionales y requerimientos no funcionales de software a partir de documentos que ya están elaborados.

Es útil cuando los expertos en la materia no están disponibles para ser entrevistados o ya no forman parte de la organización.

*Observación*

Consiste en estudiar el entorno de trabajo de los usuarios, clientes e interesados en el proyecto (Stakeholders). Es una técnica útil cuando se esta documentando la situación actual de procesos de negocios.

Existen dos tipos de observación, pasiva o activa.  
En observación pasiva el observador no hace preguntas, limitándose solo a tomar notas y a no interferir en el desempeño normal de las operaciones.

En observación activa el observador puede conversar con el usuario.

*Entrevistas*

El éxito de las entrevistas depende del grado de conocimiento del entrevistador y entrevistado, disposición del entrevistado de suministrar información, buena documentación de la discusión y en definitiva de una buena relación entre las partes.

Estas se realizan con los usuarios o interesados claves. Se direccionan al usuario hacia aspectos específicos del requerimiento a tratar además de ser útiles para obtener y documentar información detallada sobre los requerimientos y sus niveles de granularidad, lo detallado o lo general que será la función del software.

*Encuestas o cuestionarios*

La clave para el éxito es que tengan un propósito y audiencia claramente definida, establecer fechas torpes para llenar la encuesta, con preguntas claras y concisas.

*Mesas de trabajo (Workshop)*

Se refiere al dialogo entre las diferentes partes involucradas. Es decir, se define también como una auditoría y evolución, para buscar errores y soluciones. Por tanto, dar lugar a la configuración de diagnósticos y análisis para la toma de decisiones entre dos cosas que tienen conexiones entre ellas y sobre lo que puede pasar o no.

Es recomendable tener una agenda predefinida y preseleccionar a los participantes, siguiendo buenas prácticas para reuniones efectivas.

*Historia del usuario*

Las historias de usuario son una aproximación simple levantamiento de requerimientos del software en la cual la conversación pasa a ser más importante que la formalización de requerimientos escritos.   
Al redactar una historia de usuario deben tener en cuenta describir el rol, la funcionalidad y el resultado esperado de la aplicación en una frase corta.

Es recomendable que sean escritos por el mismo cliente o interesado, con el apoyo del facilitador si es necesario, con énfasis en las funcionalidades del sistema deberá realizar,

Toda la información obtenida durante la recolección de requerimientos puede ser incluido en una matriz de trazabilidad de requerimientos y en una especificación de requerimientos de software.

**Requerimientos del Proyecto.**

***1.Introducción***

***1.1 Propósito.***

Dirigido hacia la Dra. Miriam Pérez Ledezma y hacia las comunidades BUAP, diseñar una plataforma para integrar comunidades BUAP, en la que alumnos puedan construir comunidades de aprendizaje con sus docentes para transformar su realidad educativa. Del mismo modo, los docentes puedan colaborar de mejor forma con otros docentes y compartir experiencias y buenas prácticas en tiempos de la pandemia, compartiendo recursos educativos que ayuden de forma efectiva a que los docentes pueden entender mejor la tecnología educativa y sacar mejor partido de ella.

* 1. ***Ámbito del Sistema***
* *Nombre de sistema*: Tlamáchtiloyan.

Significado y explicación: La palabra del Náhuatl Tlamachtiloyan se compone de dos palabras, “Tlamachtis” que es sinónimo de enseñanza o aprendizaje, y la terminación “an” significa lugar, por lo tanto el nombre “Tlamachtiloyan” significa lugar de aprendizaje, decidimos como equipo que queríamos representar nuestros orígenes y raíces haciendo homenaje a nuestros antepasados nombrando nuestra aplicación en la lengua indígena que hablaban.

* *Explicación de que hará y lo que no hará.*

Espacio de trabajo diseñado para mejorar la comunicación y colaboración de los equipos de trabajo para instituciones educacionales. La herramienta reúne en un espacio común, las aplicaciones de colaboración necesarias para trabajar en equipo. Se podrá trabajar con notas, foros y acceso a contenido que se subirá de acuerdo con cada profesor con sus materias designadas, así también grupos donde pertenecerán a una materia en específico. No utilizará VC, sincronización con calendario personal hacia el horario de materias y será de uso exclusivo en S.O Windows.

* Beneficios, objetivos y metas

Se espera lograr una interacción más cercana entre los docentes y alumnos gracias a la creación de comunidades/foros. El objetivo principal es adaptarse a la nueva normalidad que trajo en consecuencia estos tiempos de pandemia de COVID-19 volviendo más accesible la educación evitando riesgos a la sociedad. La meta es generar una transición más sencilla entre las clases presenciales y las clases en línea previniendo una mala enseñanza o un mal aprendizaje como resultado de la nueva modalidad.

* *Referencias de documentación*

Sommerville, I. (2015). Ingeniería de Software, 10th ed. Prentice Hall.

Pressman, R. S. (2014). Software engineering: a practitioner's approach, 8th ed. McGraw-Hill Education.

Laudon, K.C. & Laudon, J.P. (2012).Sistemas de Información Gerencial. México: Pearson Educación.

Jalote, P. (2005). An integrated approach to software engineering. New York: Springer.

* *Definiciones, acrónimos y abreviaturas.*

-USUARIO: Persona que manipulará el sistema.

-ADMINISTRADOR DE SISTEMA: Persona que controlará y supervisará el sistema.

-S. O: Sistema operativo.

-ERS: Especificación de Requerimientos Software.

-CLIENTE: Persona que requiere del buen funcionamiento del sistema.

-COVID-19: Nombre completo SARS-CoV-2 o síndrome respiratorio agudo severo coronavirus 2.

* *Referencia*

*1998. “830-1998 - IEEE Guide For Software Requirements Specifications.” United State of America: IEEE.*

* 1. ***Personal involucrado***

|  |  |
| --- | --- |
| *Nombre* | Pablo García Bravo |
| *Rol* | Analista y Documentador |
| *Responsabilidad* | 1. Se encargar de trabajar junto con el cliente para tener la especificación correcta para la construcción de software.  2. Tiene como enfoque el mantener la información que se genera durante el proceso del desarrollo del software. |
| *Información de contacto* | *pablo.garciabr@alumno.buap.mx* |

|  |  |
| --- | --- |
| *Nombre* | Manuel Alejandro Ramírez Salinas |
| *Rol* | *Ing. en Manutención y Diseñador* |
| *Responsabilidad* | 1. El diagnóstico y corrección del programa encontrados en el uso del programa.  2.Diseñar lo arquitectónico y detallado del sistema, la generación de prototipos, documento de diseño. |
| *Información de contacto* | *manuel.ramirez@alumno.buap.mx* |

|  |  |
| --- | --- |
| *Nombre* | Abner Elías Cid Ramírez |
| *Rol* | Programador y asegurador de calidad |
| *Responsabilidad* | 1.Actividad esencial el transformar toda aquella especificación del sistema en un código fuente utilizando aquellos lenguajes de programación que dominen o el que se escogió en acuerdos anteriores además de reducir la complejidad del software en desarrollo.  2.Analizar detalladamente el software para que antes de que se lance este sea lo mejor para el usuario y no presente problema alguno |
| *Información de contacto* | *abner.cid@alumno.buap.mx* |

|  |  |
| --- | --- |
| *Nombre* | Matías Hernández Hernández |
| *Rol* | *Administrador de Configuración.* |
| *Responsabilidad* | Instalar, configurar y realizar un mantenimiento de la red para permitir que los equipos se comuniquen correctamente. |
| *Información de contacto* | *matias.hernandezh@alumno.buap.mx* |

|  |  |
| --- | --- |
| *Nombre* | Abraham Sarid Acosta Herrera |
| *Rol* | Tester |
| *Responsabilidad* | 1. Utilizar un conjunto de métodos que, en forma sistemática, organizada y estructurada, permite detectar y corregir errores en el desarrollo del sistema.  . |
| *Información de contacto* | *abraham.acosta@alumno.buap.mx* |

|  |  |
| --- | --- |
| *Nombre* | Jaime Antonio Altamirano Arévalo |
| *Rol* | Ingeniero de validación y verificación. |
| *Responsabilidad* | Comprueba que el software cumple los requisitos funcionales y no funcionales de su especificación y la validación comprueba que el software cumple las expectativas que el cliente espera |
| *Información de contacto* | *Jaime.altamirano@alumno.buap.mx* |

1. ***Descripción general***

La plataforma Tlamachtiloyan se está desarrollando como respuesta a un año de cambios en todos los ámbitos sociales y culturales, en el 2020 nos enfrentamos a retos que no nos habíamos planteado nunca, por ejemplo, clases 100% en línea en todos los rincones del mundo, enfocándonos en México donde hay muchos limitantes por la infraestructura el cambio ha esta nueva modalidad escolar ha sido más lento pero la situación de la pandemia no ha dejado otra opción más que la digitalización, nuestro equipo de 6 personas se ha propuesto la creación de Tlamachtiloyan como una alternativa más sencilla para la comunicación a través de mensajes de texto y archivos entre alumnos y maestros quitando la necesidad de conexiones y equipos potentes haciendo más accesible la educación.

***2.1. Perspectiva del producto.***

Al ser una plataforma independiente a otras plataformas externas se utilizará una interfaz sencilla, comprensible hacia el usuario que lo utilice. Además, el usuario deberá utilizar los datos que se utilizaron en el registro de su institución educativa que será su correo institucional y contraseña para poder acceder a la plataforma e interactuar con las materias correspondiente al alumno(usuario) así como el maestro(usuario) que administra cada uno de sus grupos dando permisos para el intercambio de ideas, trabajos y/o trabajos. De misma manera, el profesor deberá ingresar con su correo de trabajo y contraseña para poder hacer uso de la plataforma. Esto, en conjunto, permitirá el uso correcto de la plataforma entre los usuarios. (Alumno, profesor)

Así mismo, la plataforma registrará las calificaciones que se otorguen mediante el profesor hacia los alumnos, guardando cada una de ellas y dando una calificación a final de curso de acuerdo con los trabajos registrados en la plataforma.

***2.2. Funciones del producto***

Una ERS para un programa de comunicación y colaboración, esta mostrará que el sistema podrá sobrellevar diferentes grupos realizados por el profesor a cargo de la materia, mostrará para los usuarios el contenido agregado por el profesor y el uso de notas para una facilitación de acceso y formato. También mostrará la manera de poder subir archivos hacia un grupo independiente por el profesor a cargo.

**2.3. Características de los Usuarios**

Este software está destinado a estudiantes y profesores exclusivos de la Buap que tengan la necesidad de entrar a una clase virtual, se requerirá del conocimiento en el manejo de computadoras.

|  |  |
| --- | --- |
| Tipo de usuario | Estudiante |
| Formación | Estudiante |
| Actividades | Poder tomar la clase virtual |

|  |  |
| --- | --- |
| Tipo de usuario | Profesor |
| Formación | Maestro o Doctor |
| Actividades | Dar la clase el día y hora adecuado a sus estudiantes. |

**2.4. Restricciones**

-FUNCIONES DE CONTROL

Cada usuario tendrá ciertos privilegios para poder tomar notas, subir archivos dado un tiempo dado, así como el acceso a diversos contenidos de las materias que le corresponda.

-*FUNCIONES DE AUDITORÍA*

Debe de existir registros de quien ingresa, quien modifica, ve el contenido y/o sube contenido al grupo o foro en el que el usuario se encuentra y poder ser visualizado únicamente por el profesor en cuestión.

**2.5. Suposiciones y Dependencias**

El sistema solo es soportado por el S.O Windows y con la obligación de estar conectados a una red de Internet, de no llevar a cabo alguna de estas advertencias el sistema no arrancará.

1. **Requisitos específicos.**

Son factores específicos los cuales cumplirán con los requerimientos para un usuario.

Estos deben ser descritos de manera sencilla y entendible para cualquier persona. Se dividen en dos aspectos funcionales y no funcionales

* 1. **Requisitos funcionales.**

**1.-Ingreso a nuestra plataforma (Alumno)**

El alumno deberá usar su contraseña y usuario para iniciar sesión en el portal.

|  |  |
| --- | --- |
| Identificación del requerimiento | RF01 |
| Nombre del requerimiento | Ingreso a nuestra plataforma. |
| Características | Los alumnos deberán ingresar a nuestra plataforma para tener acceso a secciones específicas. |
| Descripción | El alumno gozara de privilegios según su nivel de importancia. |
| Prioridad | Alta. |
| Actor principal | Alumno. |
| Co-Desarrolladores | //Interfaz, Guía. |
| Pre-Condiciones | Contar con algún correo |
| Post-Condiciones | Recordar el usuario y contraseña |

* 1. **Requisitos No funcionales:**

**1.-Interfaz.**

La plataforma tendrá una interfaz sencilla para que cualquier usuario tenga la facilidad de entenderla.

|  |  |
| --- | --- |
| Identificación del requerimiento | RnF01 |
| Nombre del requerimiento | Interfaz del sistema |
| Características | La plataforma contara con una interfaz sencilla, la cual le facilitara el acceso a cualquier usuario. |
| Descripción | La plataforma deberá contar con una interfaz accesible. |
| Prioridad | Alta |
| Actor principal | Diseñador y programador |
| Co-Desarrolladores | //Tester, Analista y usuario final |
| Pre-Condiciones | Un diseño simple y accesible para el usuario. |
| Postcondiciones | Darle mantenimiento a la interfaz |

3.2 Resumen de los requerimientos

|  |  |  |
| --- | --- | --- |
| Referencia de Requerimiento | Nombre de Requerimiento. | Descripción |
| RF01 | Registro (Usuario). | Los usuarios se registrarán acorde a su función  (Alumno-Maestro). |
| RF02 | Ingreso a la plataforma (Alumno) | El alumno se registrará en la plataforma. |
| RFO4 | Ingreso a la plataforma (Maestro) | El profesor se registrará en la plataforma. |
| RFO4 | Entrega de tareas. (Alumno) | El alumno podrá entregar tares asignadas por el maestro. |
| RFO5 | Creación de cursos. (Maestro) | El maestro podrá crear cursos donde el inscribirá a los alumnos. |
| RFO6 | Crear equipos de trabajo. (Maestro) | El maestro podrá crear equipos dentro del curso. |
| RFO7 | Creación de tareas. (Maestro) | El maestro podrá crear tareas para los alumnos. |
| RFO8 | Creación de chats. (Maestro, Alumno) | La plataforma dará la opción a los usuarios de crear chats entre ellos |
| RF09 | Realización de llamadas. (Usuario) | Los usuarios podrán realizar llamadas entre ellos |
| RF010 | Creación de sesiones de trabajo. (Maestro) | El maestro creara sesiones de trabajo. |
| RF11 | Subir material relacionado al curso. (Maestro) | El maestro podrá subir elementos relacionados al curso |
| RnF01 | Interfaz. (Administrador) |  |
| RnF02 | Usabilidad. |  |
| RnFo3 | Elementos prediseñados |  |

# **9.-Síntesis del Proyecto**

**Imagen que contiene interior, cuarto, grande, tabla

Descripción generada automáticamente**

# **10.-Diseño del Proyecto**

# **11.-Conclusiones**

Con el paso de los años la demanda de nuevo software ha incrementado exponencialmente, lo hemos podido notar recorriendo su historia, además estos se han vuelto mucho más complejos por lo que la planeación y comunicación en el equipo que desarrolla software es vital así como tener requerimientos claros y concisos pues estos son las metas u objetivos a cumplir con el software por lo que su buena definición ayudara a una comunicación exitosa entre el cliente y los desarrolladores evitando retrasos o malos entendidos concluyendo en mejores sistemas que cumplan son los estándares actuales más exigentes.

Definiendo con exactitud los márgenes del proyecto hemos descrito la plataforma como un software de aplicación que será del tipo software educativo debido a la finalidad que tiene de generar foros para la comunicación entre alumnos y maestros, al generar los requisitos nos percatamos de que tenemos más aspectos que cuidar de los que pensamos en un inicio así que nos dimos a la tarea de detallar minuciosamente las funciones que permitirá nuestra plataforma.

Para desarrollar software en equipo, cumplir con las expectativas del cliente y entregar un producto del cual estar satisfechos es obligatoria una buena administración, la asignación de roles debe ser la correcta para que cada pieza del engranaje funcione óptimamente y facilite el trabajo de todos apoyándose entre sí yendo hacia un mismo objetivo; Los principios éticos complementan una buena organización favoreciendo un ambiente de confianza entre los integrantes del equipo que resulta en un trabajo más enriquecedor, también son reglas que dan un orden y dejan en claro la relación positiva que el software tendrá con la sociedad, siempre poniendo sobre todas las cosas el cuidado, bienestar y tranquilidad de las personas que hagan uso o se vean relacionados de cualquier manera con el software que producimos.

Conforme las semanas pasan hemos aprendido la importancia de una organización eficaz para realizar software a pequeña y gran escala pues no solo las megacorporaciones tienen necesidad de un orden también los proyectos personales o de equipos pequeños necesitan una gestión acertada de todo su proceso ya que resulta muy beneficioso y prácticamente es una necesidad seguir estas etapas para cumplir con los estándares de calidad que exige hoy en día el mercado del software.

La manera de organizar un proyecto depende de su contexto por lo que para decidir nuestro modelo ideal necesitamos entender nuestro software a crear, analizando el problema y planteamiento encontramos que al ser un entorno controlado no habrá realmente cambios de requisitos o de último minuto, identificamos que hemos estado llevando un proceso muy estructurado con un orden específico que ha sido planeado para seguir un proceso lineal, al reunir estas características e ha concluido que por el momento lo ideal será usar el modelo de cascada para producir nuestro software.

Los diagramas de flujos de datos son de gran relevancia e importancia para la programación de computadoras y diseño de software, ya que son un instrumento fundamental para el desarrollo de habilidades y destrezas lógicas. Igualmente, su utilización es tan importante, porque nos facilita la manera de representar visualmente el flujo de datos por medio de un sistema de tratamiento de información, en este realizamos un análisis de los procesos o procedimientos que requerimos para realizar un programa o un objetivo.

**Requerimientos funcionales:**

**1.-Registro de usuarios:**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF01 |
| Nombre del requerimiento | Registro de usuario. |
| Características | -Registro de Alumno.  -Registro como invitado.  -Registro de profesor. |
| Descripción | Los usuarios se registrarán según el correo que usen, es decir los alumnos deberán registrarse con el correo institucional para ser alumno, si esto no se hace se le registrara como invitado.  Los profesores deberán registrarse con su cedula. |
| Prioridad | Alta. |
| Actor principal | Usuario |
| Codesarrolladores | Plataforma y formulario de registro |
| Pre-Condiciones | -Correo electrónico.  -Matricula.  -Datos personales. |
| Postcondiciones | Inicio de sesión en plataforma. |

**2.-Ingreso a nuestra plataforma (Alumno)**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF02 |
| Nombre del requerimiento | Ingreso a la plataforma. |
| Características | Los alumnos deberán ingresar a nuestra plataforma para tener acceso a secciones específicas. |
| Descripción | El alumno gozara de privilegios según su nivel de importancia. |
| Prioridad | Alta. |
| Actor principal | Alumno. |
| Codesarrolladores | //Interfaz, Guía. |
| Pre-Condiciones | Contar con algún correo, estar conectado a una red de internet |
| Postcondiciones | Recordar el usuario y contraseña |

**3.-Ingreso a nuestra plataforma (Maestro)**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF03 |
| Nombre del requerimiento | Ingreso a nuestra plataforma |
| Características | Los Maestros deberán ingresar a nuestra plataforma para tener acceso a secciones especificas |
| Descripción | El Maestro podrá crear equipos con el fin de añadir clases y programar las mismas |
| Prioridad | Alta |
| Actor principal | Maestro |
| Codesarrolladores | //Interfaz, Guía. |
| Pre-Condiciones | Contar con la clave de acceso al registro de profesores, estar conectado a una red de internet. |
| Postcondiciones | Recordar el usuario y contraseña. |

**4.-Entrega de tareas (Alumno)**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF04 |
| Nombre del requerimiento | Entregar Tareas |
| Características | -Subir archivos pdf, docx, png y links |
| Descripción | Los alumnos tendrán la capacidad de entregar archivos según la asignación del profesor. |
| Prioridad | Alta |
| Actor principal | Alumno |
| Codesarrolladores | Maestros |
| Pre-Condiciones | Tener una “Tarea” como asignación, compatibilidad de tipo de archivo. |
| Postcondiciones | Entrega de archivo |

**5.- CREACIÓN DE CURSOS**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF05 |
| Nombre del requerimiento | Creación de Cursos |
| Características | -Creación de cursos dependiendo la materia.  -Listado de Usuarios. |
| Descripción | El Maestro podrá crear un curso el cual generará un listado de alumnos para una organización mayor. |
| Prioridad | Alta |
| Actor principal | Administrativos |
| Codesarrolladores | Alumnos y Maestros |
| Pre-Condiciones | Alumnos registrados en la lista de inscripción dada por la escuela. |
| Postcondiciones | Registro de alumnos en su curso respectivo. |

**6.- Crear equipos de trabajo.**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF06 |
| Nombre del requerimiento | Crear equipos de trabajo. |
| Características | -Asignación de equipos.  -Crear espacios de trabajo grupal. |
| Descripción | El maestro tendrá la capacidad de crear grupos de trabajo según necesite siendo administrador de estos. |
| Prioridad | Alta |
| Actor principal | Maestro. |
| Codesarrolladores | Alumnos y administradores |
| Pre-Condiciones | -Alumnos inscritos en el curso  -Listado de alumnos |
| Postcondiciones | -Registro de alumnos en equipo de trabajo.  -Acceso al equipo de trabajo. |

**7.- Creación de tareas.**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF07 |
| Nombre del requerimiento | Creación de tareas |
| Características | -El maestro podrá crear asignaciones (Tareas) para los alumnos.  -Plazo de tiempo  -Tipo de tarea |
| Descripción | El maestro podrá crear tareas para los alumnos agregando un plazo de tiempo específico. |
| Prioridad | Alta |
| Actor principal | Maestro |
| Codesarrolladores | Alumnos |
| Pre-Condiciones | -Interfaz propia.  -Descripción de la tarea.  -Plan de tareas.  -Contenido de Curso.  -Adjunción de archivos. |
| Postcondiciones | -Interacción con la tarea.  -Adjunción de archivos.  -Entrega de archivos. |

**8.-Creacion de chats.**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF08 |
| Nombre del requerimiento | Creación de chats |
| Características | -opción de chats  -Enviar mensajes  -Recibir mensajes  -Adjuntar archivos. |
| Descripción | Todos los usuarios de la plataforma tendrán la opción de crear chats con cualquier usuario registrado en la plataforma. |
| Prioridad | Media |
| Actor principal | Plataforma |
| Codesarrolladores | Maestro y Alumnos. |
| Pre-Condiciones | -Estar registrado en la plataforma.  -Creación del chat. |
| Postcondiciones | -Usuario existente.  -Ultimo contacto con el usuario. |

**9.-Subir material relacionado al curso.**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF09 |
| Nombre del requerimiento | Subir material relacionado al curso |
| Características | -Subir contenido.  -Agregar secciones. |
| Descripción | El maestro podrá crear secciones en su curso principal donde podrá subir contenido relacionado al curso correspondiente. |
| Prioridad | Media |
| Actor principal | Maestro |
| Co-Desarrolladores | Plataforma- |
| Pre-Condiciones | -Ser administrador de un curso  -Adjunción de archivo.  -Otorgar permisos para su acceso  -Otorgar permisos para visualizar.  -Denegar permisos para editar. |
| Postcondiciones | -Acceso a contenido.  -Interacción con el contenido. |

**2.-Requerimientos no funcionales:**

**1.-Interfaz.**

|  |  |
| --- | --- |
| Identificación del requerimiento | RnF01 |
| Nombre del requerimiento | Interfaz del sistema |
| Características | La plataforma contara con una interfaz sencilla, la cual le facilitara el acceso a cualquier usuario. |
| Descripción | La plataforma deberá contar con una interfaz accesible. |
| Prioridad | Alta |
| Actor principal | Diseñador y programador |
| Codesarrolladores | //Tester, Analista y usuario final |
| Pre-Condiciones | Un diseño simple y accesible para cualquiera |
| Postcondiciones | -Uso de periférico “mouse” |

**4.-Usabilidad.**

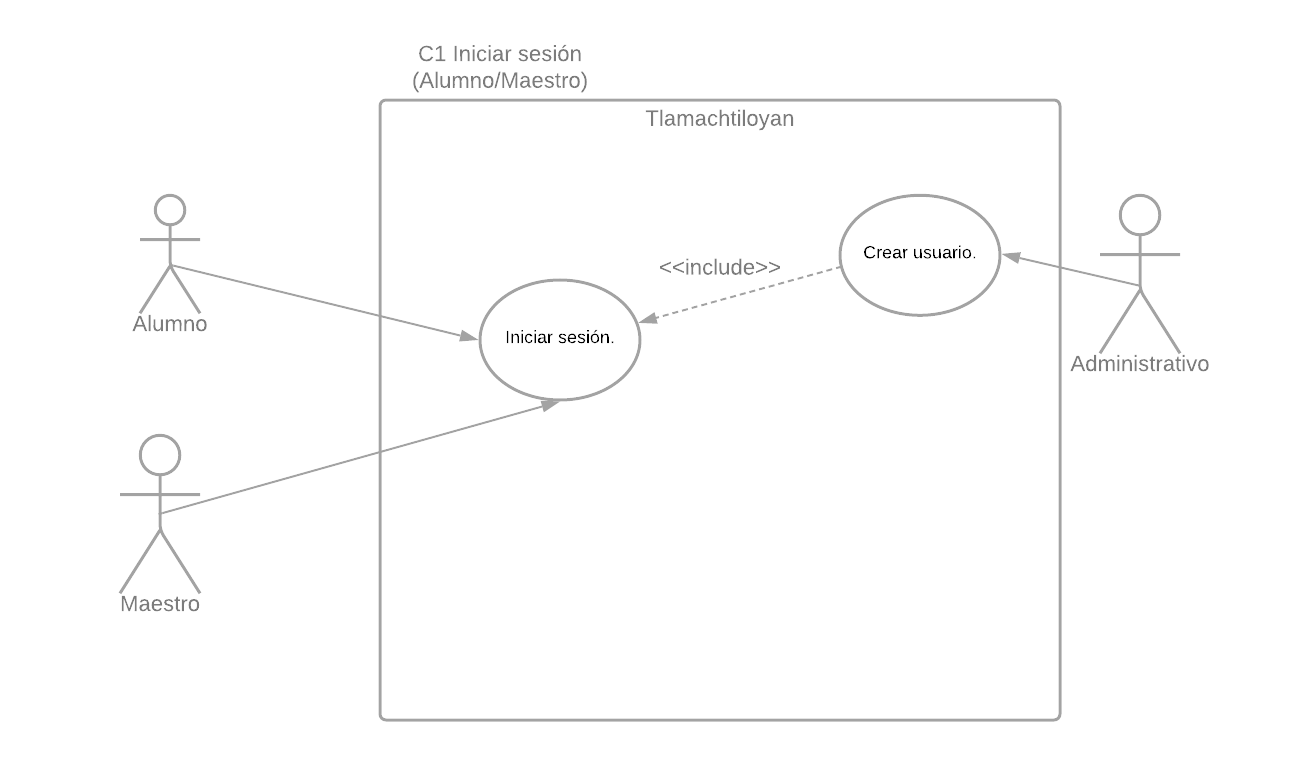
|  |  |
| --- | --- |
| Identificación del requerimiento | RnF02 |
| Nombre del requerimiento | Usabilidad |
| Características | Sencillez, Entendible, Intuitivo y Compresible. |
| Descripción | La plataforma debe ser fácil de usar con ayudas y interfaces intuitivas. |
| Prioridad | Alta. |
| Actor principal | Alumnos y Maestros. |
| Codesarrolladores | Usuarios externos. |
| Pre-Condiciones | Ejecución de programa, plataforma. |
| Postcondiciones | Darle mantenimiento al programa |

**3.-Elementos prediseñados**

|  |  |
| --- | --- |
| Identificación del requerimiento | RF03 |
| Nombre del requerimiento | Elementos Pre-Diseñados |
| Características | +Contar con elementos prediseñados para hacer uso de estos |
| Descripción | La plataforma será capaz de procesar distintas instrucciones de manera simultanea |
| Prioridad | Alta |
| Actor principal | Programador |
| Co-Desarrolladores | //Tester, Analista y usuario final |
| Pre-Condiciones | Conocimientos suficientes para el multi-Trehead de un Sw |
| Post-Condiciones | Darle mantenimiento al programa |

**Diagramas de caso de uso.**

*Primer caso de uso. (se resumieron dos casos de uso ya que tienen la misma función)*



*Segundo caso de uso.*

Diagrama

Descripción generada automáticamente

*Tercer caso de uso.*

**Diagrama

Descripción generada automáticamente**

*Cuarto caso de uso.*

Diagrama

Descripción generada automáticamente

*Quinto caso de uso.*

Diagrama

Descripción generada automáticamente

*Sexto caso de uso.*

Diagrama

Descripción generada automáticamente

*Séptimo caso de uso.*

Diagrama

Descripción generada automáticamente

**Diagrama de actividades**

*Iniciar sesión (Alumno/Maestro)*

Diagrama

Descripción generada automáticamente

*Entregar tarea (Alumno)*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Creación de cursos (Maestro)*

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

*Creación de equipos de trabajo (Maestro)*

Interfaz de usuario gráfica

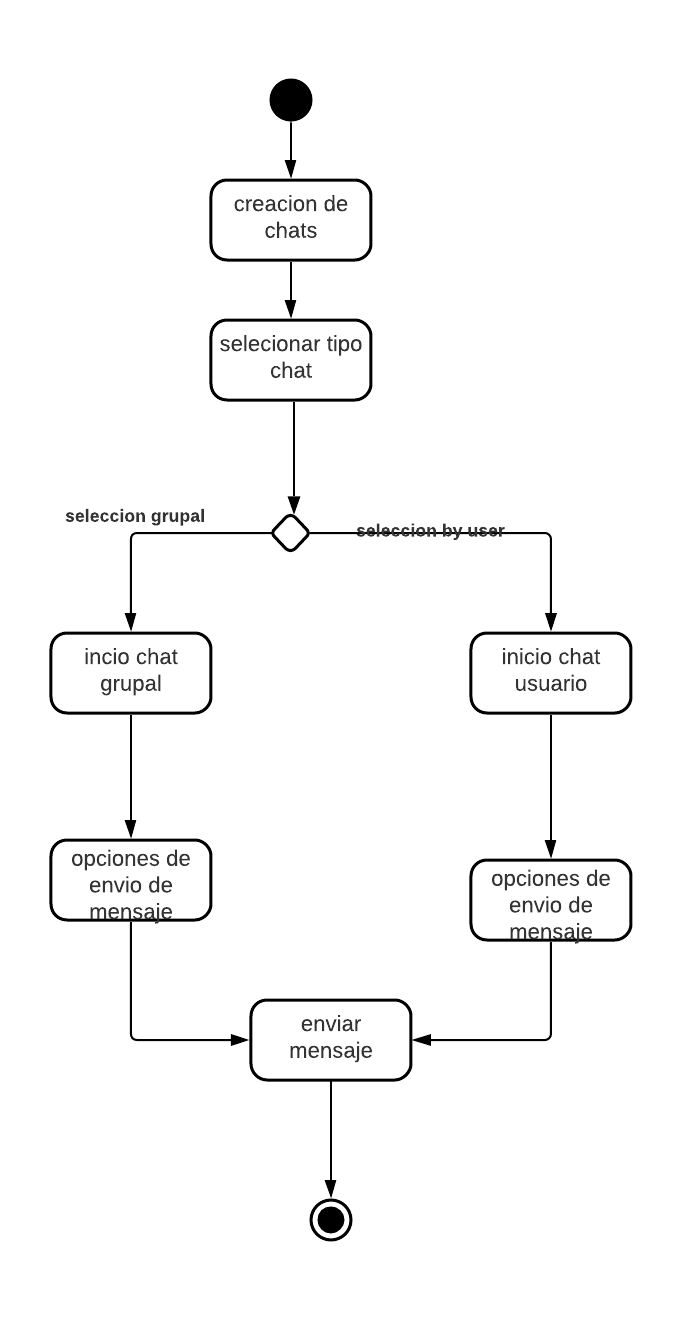
Descripción generada automáticamente

*Creación de tareas (Maestro)*

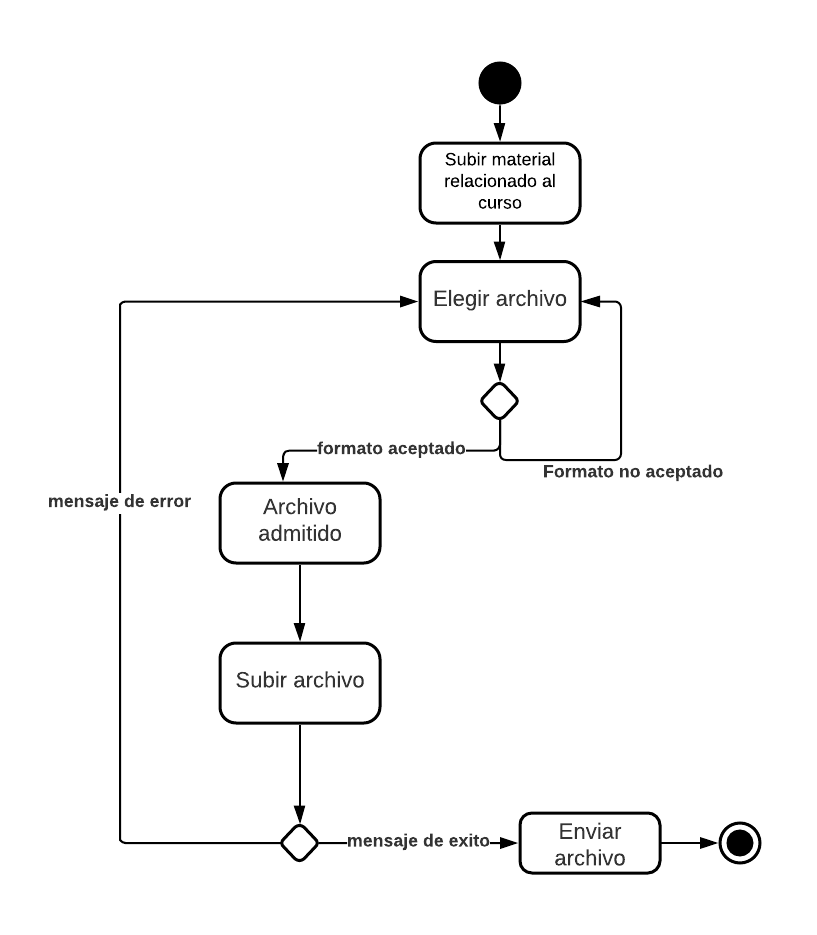
Interfaz de usuario gráfica, Aplicación

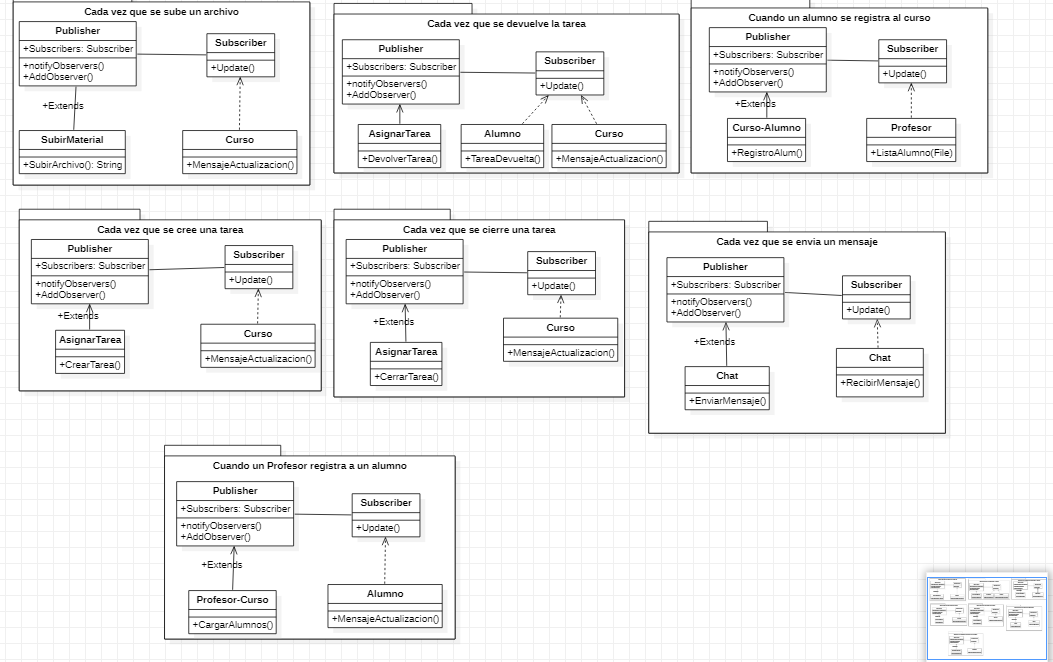
Descripción generada automáticamente

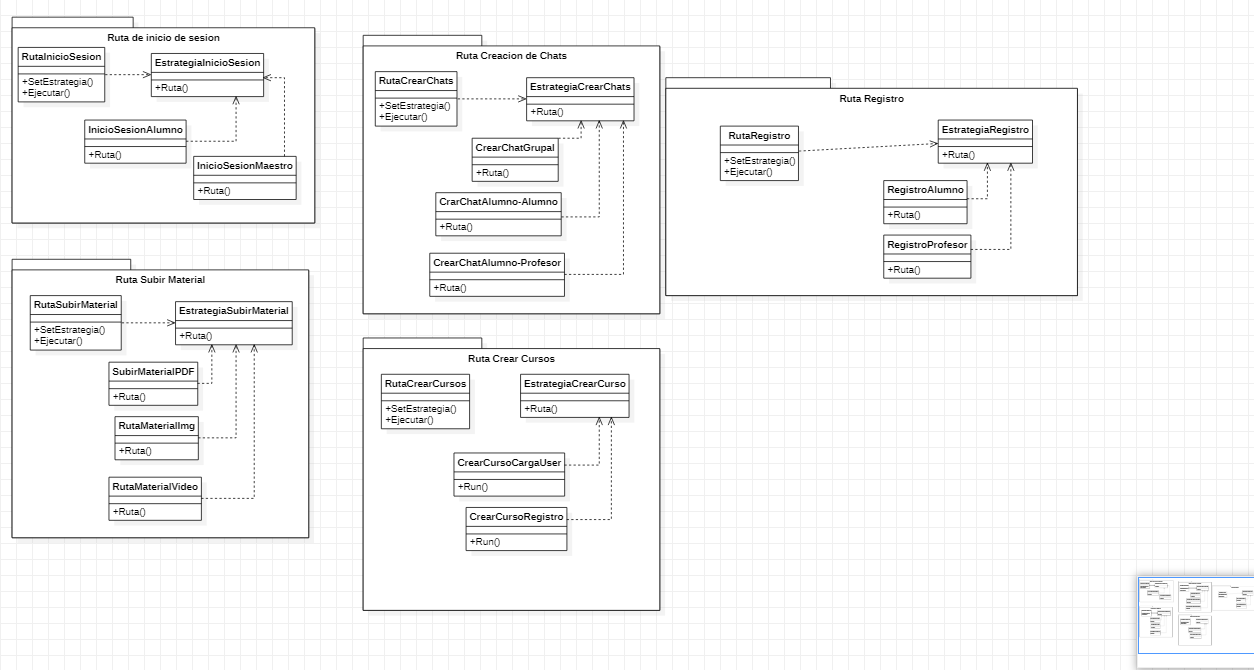
*Creación de chats*



*Subir material*



****

****

**Use Case Template**

Version 1.07

**Revision History**

|  |  |  |
| --- | --- | --- |
| Fecha | Autor | Descripción de cambio |
| 16/10/2020 | Manuel Alejandro Ramírez Salinas  Abraham Acosta Herrera  Abner Elías Cid Ramírez  Jaime Antonio Altamirano  Matías Hernández Hernández  Pablo García Bravo | Realización de casos de usos. |

----------------------------------------------------------------------------------------------

**Caso de uso:** <Creación de usuario>

**Id**: CU- <001>

**Descripción**:

<Los usuarios se registrarán según el correo que usen, es decir los alumnos deberán registrarse con el correo institucional para ser alumno, si esto no se hace se le registrara como invitado.

Los profesores deberán registrarse con su cedula.>

**Prioridad:** <Alta>

**Actores principales:**

Alumno.

Maestro.

**Actor de apoyo:**

Administrador.

**Interesados e intereses.**

Secretarias académicas. -Tiene el interés de aportar los datos de cada alumno para el registro de este

**Precondiciones.**

El usuario debe de aportar los datos correspondientes para su registro.

**Postcondiciones.**

Condición final de éxito.

El usuario entra al sistema correctamente. El usuario es registrado dentro del sistema.

Condición final de fracaso.

El usuario no logra entrar al sistema. El usuario no es registrado dentro del sistema.

Garantía mínima.

Para el registro del usuario en el sistema, la garantía mínima podría ser que los datos proporcionados no se almacenan dentro del sistema.

**Desencadenador**

Para el caso de uso de registro de uso **–** El usuario se registra dentro del sistema.

**Escenario principal de éxito.**

1. El usuario ingresa sus datos.

2. El usuario acepta registrarse al sistema.

3. El sistema registra al usuario con los datos del usuario.

**Extensiones.**

En el paso 1, si el usuario ingresa un dato no correspondiente al formato

1.El sistema desplegará un mensaje de error mencionando que el dato no corresponde al campo en donde se introduce.

En el paso 3, si el usuario ingresa datos no correspondientes a la institución académica en la que se encuentra registrado.

1.El sistema aceptará el registro del usuario, pero lo registrará como un invitado a la plataforma.

En el paso 2, si al momento de aceptar el registro al sistema se produce un error dentro del sistema.

1.El sistema desplegará mensaje de error mencionando que sucedió un error interno.

2.El sistema limpiará los campos de datos dejándolos en blanco.

**Variaciones.**

**Frecuencia:**

Al inicio de ciclo escolar. (6 meses).

**Supuestos:**

¿Qué pasaría si es un usuario invitado?

----------------------------------------------------------------------------------------------

**Caso de uso:** <Ingreso a la plataforma (Alumno)>

**ID**: CU- <002>

**Descripción:**

<Una vez el alumno está en la plataforma debe ingresar su nombre de usuario y contraseña.>

**Prioridad**: <Alta>

**Actor principal:**

Alumno.

**Actor de apoyo:**

Administrador.

**Interesados e intereses.**

Plataforma- Interesada en dar acceso.

Alumno- Interesado en ingresa a la plataforma.

**Precondiciones.**

El alumno debe estar registrado en la plataforma.

**Postcondiciones.**

Condición final de éxito.

El alumno entra al sistema correctamente.

Condición final de fracaso.

El alumno no logra entrar al sistema.

Garantía mínima.

Puede intentarlo de nuevo o si no recuerda su contraseña se le envía un correo con un código único para cambiarla.

**Desencadenador**

Para el caso de uso de Ingreso a la plataforma (Alumno) **–** El alumno ingresa al sistema.

**Escenario principal de éxito.**

1. El alumno ingresa sus datos para ingresar.

2. El alumno acepta ingresar al sistema.

3. El sistema admite al alumno en la plataforma.

**Extensiones.**

En el paso 1, si el alumno ingresa un dato no correspondiente al formato

1.El sistema desplegará un mensaje de error mencionando que el dato no corresponde al campo en donde se introduce.

En el paso 2, si el alumno ingresa datos no correspondientes a la institución académica en la que se encuentra registrado.

1.El sistema no admitirá el ingreso del alumno, pero le permitirá intentar nuevamente.

En el paso 3, si al momento de ingresar al sistema se produce un error dentro del sistema.

1.El sistema desplegará mensaje de error mencionando que sucedió un error interno.

2.El sistema limpiará los campos de datos dejándolos en blanco.

**Variaciones.**

No existen variaciones en como ingresar datos para ingresar al sistema.

**Frecuencia:**

Ingresara según sus horarios y días de clases, como lo indique su profesor o como le sea necesario (24hrs los 7 días de la semana).

**Supuestos:**

El alumno habla español, sabe usar el teclado y mouse.

----------------------------------------------------------------------------------------------

**Caso de uso**: <Iniciar sesión(maestro)>

**Id:** CU- <003>

**Descripción:**

<El usuario Maestro deberá iniciar sesión para así poder ingresar a la plataforma

Los profesores deberán registrarse con su cedula.>

**Prioridad:** <Alta>

**Actor principal:**

Maestro.

**Actor de apoyo:**

Ninguno

**Interesados e intereses.**

Plataforma interesada en dar acceso

Alumno es principal interesado para poder obtener clases

**Precondiciones.**

El Maestro deberá ingresar su cuenta con los datos correctos

**Postcondiciones.**

Condición final de éxito.

El maestro entra al sistema correctamente.

Condición final de fracaso.

El maestro no logra entrar al sistema debido a que no se acuerda de su cuenta o contraseña

Garantía mínima.

La garantía es que puede volver ingresar los datos de nuevo o si se olvidó de su contraseña se le puede enviar un código para recuperarla

**Desencadenador**

Para el caso de uso de inicio de sesión **–** El maestro incia sesion dentro del sistema

**Escenario principal de éxito.**

1. El maestro ingresa sus datos.

2. El maestro inicia sesión correctamente.

3. El sistema permite acceder a el maestro.

**Extensiones.**

En el paso 1, si el maestro ingresa un dato no correspondiente a su usuario

1.El sistema desplegará un mensaje de error mencionando que el dato no corresponde al campo en donde se introduce.

En el paso 2, si persiste con el error de inicio sesión

1.El sistema desplegará mensajee pidiendo si quiere que este se le envié un código a su email para recuperarlo

**Variaciones.**

**Frecuencia:**

Las 24 horas, los siete días de la semana.

**Supuestos:**

ninguna

----------------------------------------------------------------------------------------------

**Caso de uso:** <Entregar Tareas>

I**d:** CU- <004>

**Descripción:**

<Los estudiantes tendrán la opción de enviar sus trabajos, tareas o avances del curso que estén tomando>

**Prioridad**: <Alta>

**Actor principal:**

Alumno.

**Actor de apoyo:**

Maestro

**Interesados e intereses.**

El estudiante es el más interesado en entregar su avance para que el profesor lleve el control de s calificación

**Precondiciones.**

1. El profesor debe activar la opción de entrega de actividades
2. La entrega de actividades debe activarse con el tiempo necesario para enviar la tarea.
3. El usuario debe haber iniciado sesión.
4. El usuario debe enviar formatos permitidos como pdf,dox,rar,pnj,jpg.
5. El usuario debe estar conectado a una red de internet.

**Postcondiciones**

Condición final de éxito.

El estudiante debe presionar la opción de enviar.

El sistema debe confirmar la entrega mediante un mensaje por correo.

La entrega debe quedar registrado en la cuenta del estudiante.

Condición final de fracaso.

Si el estudiante no está conectado una red de internet no podrá enviar su trabajo.

Si el estudiante manda su trabajo después del tiempo límite no será recivido.

Garantía mínima.

**El estudiante debe recibir un correo de confirmación.**

El trabajo enviado debe registrarse en la sección de trabajos enviados.

**Desencadenador**

Para este caso de uso, debe ejecutare el caso de uso Ingreso a la plataforma (Alumno).

**Escenario principal de éxito.**

1. El estudiante debe dar click en la función de entregar tareas.
2. El sistema debe mostrar la opción de cargar archivo.
3. El estudiante debe seleccionar sus archivo.
4. El sistema debe cargar el archivo.
5. El estudiante debe dar click a la opción de de enviar.
6. El sistema muestra un mensaje de éxito.

**Extensiones:**

1. En caso de que el tiempo límite haya terminado no se podrá enviar nada.

4. En caso de que el estudiante no cargue el formato soportado por el sistema no se podrá enviar el trabajo.

5. Sí el sistema no pude enviar el archivo por mala conexión de red entonces mostrara un mensaje de error.

**Variaciones**

No existen variaciones

**Frecuencia**

Cada que el profesor deje una actividad para enviar.

**Supuestos:**

1.- ¿Qué pasaría sí en el momento que se esté enviando el archivo se va el internet, o se apaga la computadora?

2.- ¿Qué pasaría si todos los alumnos envían su trabajo al mismo tiempo?

**--------------------------------------------------------------------------------------------------------------**

**Caso de uso:** Creación de cursos(maestro)>

**Id:** CU-<005>

**Descripción:**

<El maestro tendrá la capacidad de crear cursos a los cuales los alumnos podrán ingresar y participar en las sesiones de clase.>

**Prioridad:** <Alta>

**Actor principal:** Maestro.

**Actor de apoyo:** Plataforma

**Interesados e intereses.**

Plataforma interesada en el uso de sus capacidades

Alumno es principal interesado para poder obtener clases

**Precondiciones:**

El Maestro deberá ingresar su cuenta con los datos correctos

**Postcondiciones:**

Condición final de éxito.

El maestro entra al sistema correctamente.

El maestro logra crear sus cursos.

Condición final de fracaso.

El maestro no logra entrar al sistema debido a que no se acuerda de su cuenta o contraseña

El maestro no Logra crear su curso

Garantía mínima.

La garantía es que puede volver ingresar los datos de nuevo o si se olvidó de su contraseña se le puede enviar un código para recuperarla

La garantía es que puede volver a intentar un crear un curso.

**Desencadenador**

Para el caso de uso de crear cursos—El maestro crea sus cursos.

**Escenario principal de éxito.**

1.-El maestro decide crear un curso.

2.-Selecciona la opción de “Crear curso”.

3.-El sistema permite crear el curso.

4.El maestro crea el curso

**Extensiones.**

En el 3.- si el maestro introduce un valor incorrecto al llenar los datos el sistema enviara un mensaje con la leyenda “No se creara el curso hasta corregir el campo” “ERRONEO”.

1.El sistema desplegará mensajee pidiendo si quiere que este se le envié un código a su email para recuperarlo

En el paso 3.- Si el maestro no rellena los datos correctos el curso no se creara por lo cual al maestro le aparecerá un mensaje emergente avisando el error.

**Variaciones.**

**Frecuencia:**

Al inicio de ciclo escolar hasta su culminación de semestre (6meses).

**--------------------------------------------------------------------------------------------------------------**

**Caso de uso:** Crear equipos de trabajo. (Maestro)

**Id**: CU- <006>

**Descripción:**

El usuario (Maestro) tendrá la capacidad de crear grupos de trabajo según lo necesite.

**Prioridad**: <Alta>

**Actor principal:**

Alumno.

**Actor de apoyo:**

Administrador.

**Interesados e intereses.**

Plataforma- Interesada en dar acceso.

Maestro- Interesado en la creación de equipos de trabajo.

**Precondiciones.**

El maestro debe de iniciar sesión en la plataforma.

**Postcondiciones.**

Condición final de éxito.

Registro de alumnos en equipos de trabajo.

Acceso por parte de los alumnos al equipo.

Condición final de fracaso.

No es posible el acceso del alumno al equipo de trabajo aun cuando fue un registro exitoso.

No existe tal usuario (Alumno) al momento de registrar.

Garantía mínima.

Para el caso de uso *Crear equipos de trabajo*, una garantía mínima podría ser que el alumno puede abandonar el equipo por decisión propia.

**Desencadenador**

Para el caso de uso *Crear equipos de trabajo* **–** La creación del equipo de trabajo por parte del maestro.

**Escenario principal de éxito.**

1. El maestro crea el equipo de trabajo.

2. El maestro registra a los alumnos en el equipo de trabajo.

3. El alumno es registrado en el equipo y accede correctamente a este.

**Extensiones.**

En el paso 2, si el maestro quiere registrar un alumno que no existe

1.El sistema desplegara una leyenda que describe que no se encontraron coincidencias con su búsqueda.

En el paso 3, si el alumno es registrado, pero no puede acceder

1. El sistema desplegará un mensaje mencionando que su acceso no es permitido.

**Variaciones.**

No existen variaciones en como acceder al equipo de trabajo.

**Frecuencia:**

Ingresara según sus horarios y días de clases, como lo indique su profesor o como le sea necesario (24hrs los 7 días de la semana).

**Supuestos:**

El alumno habla español, sabe usar el teclado y ratón.

--------------------------------------------------------------------------------------------------------------

**Caso de uso:** <Creación de tareas (Maestro)>

**ID**: CU- <007>

**Descripción:**

<El maestro puede crear tareas para los alumnos quienes tendrán un plazo de tiempo específico definido por el maestro.>

**Prioridad**: <Alta>

**Actor principal:**

Maestros.

**Actor de apoyo:**

Alumno.

**Interesados e intereses.**

Maestro- Interesado en crear y asignar tareas.

Alumno- Interesado en recibir la tarea.

**Precondiciones.**

El maestro debe haber ingresado en la plataforma.

**Postcondiciones.**

Condición final de éxito.

El maestro crea y asigna correctamente la tarea.

Condición final de fracaso.

El maestro no logra crear la tarea.

Garantía mínima.

Puede intentarlo de nuevo.

**Desencadenador**

Para el caso de uso de Creación de tareas (Maestro) **–** El maestro elige la opción de crear una tarea.

**Escenario principal de éxito.**

1. El maestro ingresa los datos para crear la tarea.

2. El maestro acepta asignar la tarea.

3. El sistema admite la tarea del maestro y las asigna.

**Extensiones.**

En el paso 1, si el maestro ingresa un dato no correspondiente al formato.

1.El sistema desplegará un mensaje de error mencionando que el dato no corresponde al campo en donde se introduce.

En el paso 2, si el maestro ingresa datos no correspondientes a la asignación de tarea.

1.El sistema no admitirá la creación de la tarea, pero le permitirá intentar nuevamente.

En el paso 3, si al momento de ingresar al sistema se produce un error dentro del sistema.

1.El sistema desplegará mensaje de error mencionando que sucedió un error interno.

2.El sistema limpiará los campos de datos dejándolos en blanco.

**Variaciones.**

No existen variaciones en como ingresar datos para crear/asignar tareas.

**Frecuencia:**

Creará tareas según sus horarios y días de clases (24hrs los 7 días de la semana).

**Supuestos:**

El maestro ya sabe que tarea asignará, que archivos incluirá y los plazos de tiempo que dará.

--------------------------------------------------------------------------------------------------------------

**Caso de uso:** <Creación de chats (Maestro/Alumnos)>

**ID**: CU- <008>

**Descripción:**

<Cualquier usuario podrá crear un chat para hablar con alguien.>

**Prioridad**: <Alta>

**Actor principal:**

Maestros.

Alumno

**Actor de apoyo:**

Plataforma.

**Interesados e intereses.**

Maestro- Interesado en estar en contacto con sus alumnos.

Alumno- Interada en estar contacto con maestro y otros alumnos.

**Precondiciones.**

El maestro debe haber ingresado en la plataforma.

Los alumnos deben estar ingresados en la plataforma.

**Postcondiciones.**

Condición final de éxito.

Los usuarios están en contacto entre si

Condición final de fracaso.

Nadie crea chats entre si.

Garantía mínima.

Puede interntarlo de nuevo.

**Desencadenador**

Para el caso de uso de Creación de chats (Maestro/Alumno) **–** Los usuarios crean chats entre si.

**Escenario principal de éxito.**

1. El maestro/Alumno crea el chats.

2. El Maestro/Alumno envia mensaje.

3. El Maestro/Alumno recibe el mensaje y lo devuelve.

**Extensiones.**

1.-EL usuario crea un chat

2.-EL usuario envia un chat.

3.-El usuario recibe un mensaje

4.-El usuario reenvía el mensaje.

5.-Se crea una comunicación.

**Variaciones.**

No existen variaciones en como ingresar datos para enviar mensajes.

**Frecuencia:**

Cuando necesite comunicación.

**Supuestos:**

El usuario tiene la necesidad de enviar mensajes y estar en comunicación.

--------------------------------------------------------------------------------------------------------------

**Caso de uso:** <Subir material relacionado al curso (Maestro)>

**ID**: CU- <009>

**Descripción:**

El profesor tendrá la opción de cargar material de la clase para sus alumnos mediante una opción que le permita cargar archivos de texto.

**Prioridad**: <Alta>

**Actor principal:**

Maestros.

**Actor de apoyo:**

Alumno.

**Interesados e intereses.**

Maestro- Interesado en crear y asignar tareas.

Alumno- Interesado en recibir la tarea.

**Precondiciones.**

El maestro debe haber ingresado en la plataforma.

**Postcondiciones.**

Condición final de éxito.

El maestro asigna correctamente el material de clase.

Condición final de fracaso.

El maestro no logra asignar el material de clase.

Garantía mínima.

El profesor puede intentarlo de nuevo

**Desencadenador**

Para el caso de uso subir material relacionado al curso(Maestro) **–** El maestro elige la opción de subir material.

**Escenario principal de éxito.**

1. El maestro inicia sesión.

2. El maestro elije el curso.

3. El maestro selecciona la opción de subir material.

4. El sistema permite al maestro cargar su archivo.

5. El maestro elije su archivo.

6. El maestro sube el archivo.

7. El maestro envía el archivo.

8. El sistema envía un mensaje de éxito.

**Extensiones.**

En la parte 5, el profesor debe elegir el formato adecuado que sea soportado por el sistema.

En la parte 8, el sistema no puede cargar el material por problemas de red y muestra un mensaje de error.

**Variaciones.**

No existen variaciones.

**Frecuencia:**

El profesor subirá material a su clase de acuerdo a su plan de estudios y su manera de trabajar.

**Supuestos:**

El profesor cuenta con una computadora, ya tiene el material a cargar en el sistema y además cuenta con una red de internet.

**Requerimientos especiales.**

**Seguridad:**

1.El sistema encriptará los datos personales del usuario.

**Interfaz del usuario:**

1. Barra scroll lateral para su navegación.

2. Acceso visible a cursos y contenidos a la izquierda.

3. Acceso visible a chats personales.

4. Acceso visible a los equipos de trabajo.

5. En la creación de chats se desplegará una ventana donde se pueda escribir y adjuntar archivos.

**Problemáticas**

1. ¿Como expresar la frecuencia de registro de un profesor en el sistema?

2. ¿Cómo pasara un usuario invitado a un usuario normal?

3. ¿Cómo se sabrá si un alumno no puede acceder a un equipo de trabajo?

**A realizar**

-Modificación de la interpretación de casos de uso después de su revisión

-Modificación de requerimientos especiales después de su revisión.

## ***Bibliografía***

Content, R. R. (30 de abril de 2019). *rockcontent.* Obtenido de https://rockcontent.com/es/blog/tipos-de-software/

*MARCO TEÓRICO*. (s.f.). Obtenido de https://www.marcoteorico.com/curso/91/ingenieria-de-software/848/la-importancia-de-la-ingenieria-del-software

Msg, W. F. (s.f.). *Historia y Evolución de la Ingeniería de Software.* Obtenido de https://ingswusa.files.wordpress.com/2014/08/historia-de-ing-software.pdf

s.n, (30 de JUNIO de 2020). *WIKIPEDIA*. Obtenido de https://es.wikipedia.org/wiki/Problema\_del\_a%C3%B1o\_2000

Palponto. (28 de diciembre de 2010). *Historia de la informática*. Obtenido de https://histinf.blogs.upv.es/2010/12/28/ingenieria-del-software/#:~:text=OR%C3%8DGEN,de%20la%20crisis%20del%20software.&text=El%20software%20no%20se%20produc%C3%ADa,definido%20y%20dividido%20en%20fases.

Pérez, G. A. (19 de 02 de 2019). *YouTube*. Obtenido de https://www.youtube.com/watch?v=9etxPJHX260&feature=youtu.be

programacion4toaitjo. (11 de 10 de 2011). *Programación comentarios.* Obtenido de https://comentariosprogramacion.wordpress.com/2011/10/15/mitos-y-leyendas-de-software-libre/

Quintilla, D. (08 de 30 de 2019). *OurAcademy*. Obtenido de https://our-academy.org/posts/software-engineering-history

Tello, D. E. (12 de septiembre de 2012). Obtenido de https://www.tamps.cinvestav.mx/~ertello/swe/sesion01.pdf

web, l. (2017). *#2 Mitos del software | Curso de ingeniería de software.*

*Roger S. PressMan. (2010). Ingeniería del Software Un enfoque Práctico. México, D.F: Mc Graw Hill.*

Ch1 Introduction Software Engineering book, 10th edition. (2014, 12 diciembre). [Diapositivas]. Slideshare. https://www.slideshare.net/software-engineering-book/ch1-introduction-42645973

Jaramillo, O. (2017, 25 octubre). El código de ética y práctica profesional de ingeniería del software [Diapositivas]. Slideshare. https://www.slideshare.net/OmarJaramillo13/el-cdigo-de-tica-y-prctica-profesional-de-ingeniera-del-software

Icesi.edu.co. n.d. *Administración De La Configuración Del Software*. [online] Disponible en: <https://www.icesi.edu.co/departamentos/tecnologias\_informacion\_comunicaciones/cursos/09561/102/home/\_media/unidad1/cap\_10.pdf> [Accedido el 26 de agosto de 2020].

Pressman, R., 1996. *Ingeniería Del Software*. 7th ed. Madrid: McGraw-Hill.

Ramírez, I., 2020. *Ciclo De Vida De Desarrollo De Software*. [online] Efectodigital.online. Disponible en: <https://www.efectodigital.online/post/2018/04/23/ciclo-de-vida-de-desarrollo-de-software> [Accedido el 26 de agosto 2020].

Terrera, G., 2012. *Testingbaires» Pruebas De Caja Negra Y Un Enfoque Práctico*. [online] Testingbaires.com. Disponible en: <https://testingbaires.com/2017/02/26/pruebas-caja-negra-enfoque-practico/> [Accedido el 26 de agosto de 2020].

Terrera, G., 2012. *Testingbaires» Testing De Caja Blanca – Parte I*. [online] Testingbaires.com. Disponible en: <https://testingbaires.com/2014/01/03/testing-de-caja-blanca-parte/> [Accedido 26 de agosto de 2020].

Ramón Rodríguez, J., García Mínguez, J. and Lamarca Orozco, I., 2007. *Gestión De Proyectos Informáticos: Métodos, Herramientas Y Casos - Capítulo I. La Gestión De Proyectos. Conceptos Básicos*. [En línea] Editorialuoc. Disponible en: <https://www.academia.edu/38677442/Gestion\_de\_Proyectos\_Inform%C3%A1ticos\_M%C3%A9todos\_Herramientas\_y\_Casos\_1a\_ed\_2007\_Rodriguez\_Jose\_R > [Accedido el 4 septiembre 2020].

Salazar, G., 2009. *Estimación De Proyectos De Software: Un Caso Práctico*. [En línea] Dialnet. Disponible en: <https://dialnet.unirioja.es/servlet/articulo?codigo=3101335> [Accedido el 4 Septiembre 2020].

*Modelo V*. (s. f.). Ciclo de vida del software. Recuperado 11 de septiembre de 2020, de <https://cdvmv.blogspot.com/2016/02/modelo-v.html>

Campderrich Falgueras, Benet (2002): Ingeniería de software. Barcelona: Editorial UOC, 2002. 320 páginas.

Ch2 Software processes. Software Engineering Book. 10th Edition. (2016, 2 junio). [Diapositivas]. Slideshare. <https://www.slideshare.net/software-engineering-book/ch2-sw-processes>

Es.wikiversity.org. 2020. *Metodologías Ágiles De Desarrollo Software - Wikiversidad*. [online] Disponible en: <https://es.wikiversity.org/wiki/Metodolog%C3%ADas\_%C3%A1giles\_de\_desarrollo\_software> [Accedido 18 Septiembre 2020].

Fariño R, G., 2020. Modelo Espiral De Un Proyecto De Desarrollo De Software. [online] Ojovisual.net. Disponible en: <http://www.ojovisual.net/galofarino/modeloespiral.pdf> [Accedido 18 Septiembre 2020].

Gonçalves, L., 2020. *Qué Es La Metodología Ágil, Todo Lo Que Necesitas Saber*. [online] ADAPT METHODOLOGY®. Disponible en: <https://adaptmethodology.com/es/que-es-la-metodologia-agil/> [Accedido 18 Septiembre 2020].

Padilla Monge, E., n.d. *Modelo RUP*. [online] Ing. en Software. Disponible en: <https://softwarerecopilation.wordpress.com/modelo-rup/> [Accedido 18 Septiembre 2020].