UNIVERSIDAD INTERSERRANA DEL ESTADO DE PUEBLA – CHILCHOTLA
ORGANISMO PÚBLICO DESCENTRALIZADO DEL GOBIERNO DEL ESTADO

PROGRAMA EDUCATIVO:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

"MANUAL PARA LA GESTIÓN DE PROYECTOS DE SOFTWARE DE LA UNIVERSIDAD INTERSERRANA DEL ESTADO DE PUEBLA - CHILCHOTLA"

QUE PRESENTA:

JOSÉ ROBERTO ROMERO RAMÍREZ

OPCIÓN DE TITULACIÓN:

TESIS

DIRECTOR:

MTRA. ROCÍO SOSA TORRES

GENERACIÓN:

2018 – 2023

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Rafael J. García Chilchotla, Puebla; septiembre de 2024

Chilchotla, Puebla a 24 de septiembre de 2024

Asunto: Se autoriza impresión y empastado

C. José Roberto Romero Ramírez

Pasante de la **INGENIERÍA EN SISTEMAS COMPUTACIONALES**

Matrícula: **201872020**

P r e s e n t e

Según el dictamen entregado a la Jefatura de Departamento Académico del programa educativo de INGENIERÍA EN SISTEMAS COMPUTACIONALES; por parte de la Comisión Revisora, de fecha **24 de septiembre** del año en curso, **No Existe Ningún Inconveniente** para que el trabajo **"Manual para la Gestión de Proyectos de Software de la Universidad Interserrana del Estado de Puebla - Chilchotla"**, pueda ser impreso y empastado en un ejemplar; debiendo adjuntar el trabajo recepcional en archivo electrónico debidamente rotulado a más tardar el **14 de octubre** del presente año, para proceder a tramitar su fecha de Examen profesional.

De antemano reciba una cordial felicitación por la conclusión de su trabajo recepcional.

A T E N T A M E N T E.

Titulo/Nombre	Cargo	Firma de conformidad
Mtra. Rocio Sosa Torres	Presidente	
M. C. A. Rafael Meneses Jimarez	Secretario	
Ing. Luis Alberto Solís Delgado	Vocal	

Manual para la Gestión de Proyectos de Software de la Universidad

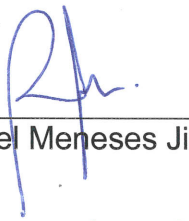
Interserrana del Estado de Puebla - Chilchotla

Tesis realizada por el C. José Roberto Romero Ramírez, bajo la dirección de la Mtra. Rocio Sosa Torres, ha sido revisada y aprobada por el H. jurado examinador, abajo indicado y aceptado como requisito parcial para obtener el título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

JURADO EXAMINADOR

PRESIDENTE Mtra. Rocio Sosa Torres



SECRETARIO M. C. A. Rafael Meneses Jimarez

VOCAL Ing. Luis Alberto Solís Delgado

Agradecimientos y dedicatorias

Dedico este trabajo, a mi madre, quien ha sido mi mayor fuente de inspiración y fortaleza. Gracias por tu apoyo incondicional y por enseñarme a enfrentar los desafíos con valentía. A pesar de los malos momentos, tu apoyo ha sido el pilar sobre el que he construido este camino.

A mi tío Ezequiel, quien me ha apoyado en incontables ocasiones, enseñándome el significado de trabajo duro, constancia, esfuerzo y apoyo incondicional. A sido mi ejemplo a seguir y una motivación para el desarrollo de esta tesis.

A mi hermana, que ha pesar de encontrarse lejos, me ha transmitido su cariño y confianza, dándome palabras de aliento cuando más las necesitaba.

A doña Irene por sus muchos consejos, regaños y sabiduría, pero también por haber sido una gran ayuda a lo largo de estos años cuando mi familia pasaba por momentos muy duros. A don Efraín por su sentido del humor y por su forma de darme consejos cuando sentía dudas o miedo. A Cyntia por ayudarme con mis tareas y regalarme buenas experiencias, como una gran amiga. A Efraín por sus consejos que me motivan a ser arriesgado y no tener miedo a nuevos desafíos y vivencias. En general, gracias familia Bretón por cada gesto de cariño y por hacerme sentir parte de su familia.

A don Enrique, un zapatero que considero como alguien de mucha experiencia y generoso, que sin pedir nada a cambio, me brindó su consejo y ayuda cuando más lo necesitaba. Su ejemplo de humildad y trabajo duro me ha inspirado más de lo que las palabras pueden expresar.

A Habib, mi amigo de la universidad que, aunque el destino nos haya llevado por caminos diferentes, aún seguimos creando momentos de risas, desafíos, logros e incluso filosóficos. Nuestra amistad ha sido un refugio además de una fuente de nuevas aventuras y

espero así sea por muchos años más.

A esa persona especial en mi vida, cuya presencia en mi vida ha sido una constante fuente de paz y motivación. Gracias por caminar a mí lado, por estar conmigo en mis momentos más difíciles, por ser tan comprensiva y amable conmigo, por hacerme sentir tan especial y por brindarme todo tu cariño. Aunque no la mencione de manera directa, sabe que cada paso que doy, lo hago con su recuerdo en mi corazón.

A la Doctora Brisol, mi maestra en la universidad, por sus sabias palabras, su apoyo incondicional, compartir conmigo sus logros transmitiéndome su felicidad y las largas charlas que hemos tenido que me han llevado a desarrollar una nueva forma de entender la vida. Su visión y orientación fueron fundamentales en mi formación académica, y su ejemplo me ha mostrado lo que significa ser un verdadero profesional.

A la maestra de bachillerato, la maestra Cleotilde, por compartir conmigo su amor por la poesía y por enseñarme que las palabras tienen el poder de transformar realidades. Sus consejos y su rectitud me ayudaron a cultivar un pensamiento crítico y creativo que ha sido esencial en mi formación.

A mi profesor de secundaria, el maestro Rodrigo, quien fuese mi guía en la búsqueda de conocimiento y la curiosidad por el aprendizaje. Sus enseñanzas trascendieron las aulas y se que al igual que a mí, muchos de sus alumnos quedaron profundamente marcados por el amor a su profesión, siempre guiándonos a buscar más allá de lo evidente.

A la maestra Rocio, mi asesora en esta tesis, por su paciencia, sus valiosos comentarios y su apoyo incondicional. Gracias por ayudarme a estructurar mis ideas, por impulsarme a dar siempre lo mejor de mí, y por confiar en mi capacidad para llevar a cabo este trabajo.

Al jefe de carrera, el M.C.A. Rafael, quien fuese clave en la definición y desarrollo del tema de esta tesis. Su experiencia y sus observaciones han sido cruciales para la calidad y

profundidad de este trabajo.

También agradezco a la Universidad Interserrana del Estado de Puebla - Chilchotla (UICh) por proporcionarme las herramientas y el espacio para desarrollarme tanto académica como personalmente. Cada paso que he dado dentro de sus aulas ha sido un cúmulo de experiencias, vivencias y nuevos desafíos que han sido fundamentales para la culminación de este sueño.

Finalmente, dedico este trabajo en memoria de la señora María Isabel de Picasso, una mujer extraordinaria que siempre tuvo una palabra amable y sobre todo un corazón enorme. Su presencia iluminó la vida de quienes la rodearon, y su partida dejó un vacío profundo. Este trabajo es también un tributo a su memoria a todo lo que representó para mí.

Índice general

Índice de tablas

Índice de ilustraciones

Glosario

Algoritmo: Se entiende por algoritmo a la secuencia o conjunto de instrucciones finitas las cuales tienen por objetivo dar solución a algún determinado problema. capítulo, se encarga de resolver cualquier problema empleando como medio una serie de instrucciones y reglas concisas para mostrar el resultado obtenido.

Código: Se define como código al conjunto de instrucciones los cuales hacen que un desarrollador de órdenes a una computadora en forma de datos organizados para que sea mostrado al usuario a través de una interfaz gráfica o una respuesta lógica.

Clases: Es donde se crean varios objetos individuales, es decir, es el conjunto de datos llamados atributos los cuales definen a los objetos. Además de que definen al conjunto de comportamientos, funciones o métodos del objeto.

Estándar: El término estándar aplicado a la informática hace referencia al conjunto de reglas, especificaciones y protocolos establecidos que garantizan la compatibilidad y su compatibilidad entre los sistemas y las tecnologías informáticas. Los estándares son creados por instituciones especializadas como el Instituto de Ingeniería Eléctrica y Electrónica (IEEE) o el Consorcio World Wide Web (W3C).

Funciones: También conocidos como métodos, son los que permiten dividir el trabajo en tareas más pequeñas, convirtiéndose así en un bloque de código diseñado para realizar tareas específicas. Esto se realiza mediante la toma de datos de entrada, su procesamiento y la devolución de un resultado.

Gestión: Se conoce como gestión al conjunto de acciones y procedimientos que buscan lograr un objetivo específico para optimizar al máximo los recursos disponibles, donde se incluyen la planificación, organización, dirección y control de estos con el fin de alcanzar metas de manera eficiente y efectiva.

Metodología: Dentro del desarrollo de software, se considera como metodología al marco o conjunto de procesos y prácticas los cuales guían la creación, diseño, despliegue y mantenimiento de un sistema informático. La metodología proporciona estructuras organizadas enfocadas en el desarrollo de software volviéndolo un proceso eficiente y eficaz ya que incluyen herramientas, modelos y métodos que facilitan la gestión durante el proceso de desarrollo.

Objetos: En el ámbito de la programación, los objetos son entidades que representan aspectos del mundo real o conceptos abstractos que almacenan características y comportamientos específicos. Los objetos se conforman de un estado y un comportamiento que a su vez se conforman de datos almacenados denominados como atributos o propiedades, además de tareas realizables conocidas como métodos que duran mientras el código se ejecute.

Programación: Consiste en escribir instrucciones empleando un lenguaje de programación que para que una computadora realice una tarea específica, las cuales se conoces como código y están organizada y se organiza de forma lógica para que la computadora pueda comprenderlo y ejecutarlo.

Protocolo: Dentro de la programación, se le denomina protocolo al conjunto de reglas que permiten la comunicación entre diversos elementos tales como computadoras, componentes de redes, entre otros.

Prototipo: Se entiende por prototipo a una versión preliminar o modelo de un proyecto o un producto generado antes de su desarrollo completo. Estos pueden ser representaciones visuales simples hasta modelos complejos en interactividad y funcionalidad, ya que buscan retroalimentar de forma temprana los conceptos antes del desarrollo final evitando comprometer recursos significativos.

Software: El software es aquel programa informático que contiene un conjunto de instrucciones, algoritmos y partes visuales que interactúan con el usuario por medio de un dispositivo electrónico de manera sencilla y gráfica. Este permite que equipos como computadoras o celulares funcionen, convirtiéndolos así en herramientas de uso común, ya que sin software serían obsoletos y su uso sería incomprensible para la mayoría de la población.

Resumen

La presente tesis se centra en el desarrollo del “Manual para la Gestión de Proyectos de Software para la UIEPCH”, con el objetivo de dar una guía que busca mejorar la eficiencia y estandarizar los procesos relacionados a proyectos de software universitario. A continuación, se presentará un breve resumen de los capítulos clave:

Durante el primer capítulo se establece la justificación de la investigación, a su vez se plantea el problema a abordar para definir los objetivos, tanto general como específicos que permitirán dar solución a la problemática. También se plantean hipótesis buscando guiar la investigación hacia la optimización de la gestión de proyectos de software dentro del ámbito universitario.

En el segundo capítulo se explora el marco teórico donde se sustenta la investigación, dando pie a abordar conceptos esenciales que permitirán familiarizarse con el tema de esta tesis. Dentro de estos conceptos se muestran elementos como la definición de un manual, los procesos de solicitud y aprobación, incluso las etapas de software, los paradigmas de programación, entre otros.

El tercer capítulo se enfoca en la información referente a la institución universitaria, incluyendo detalles como el nombre, razón social, ubicación, misión, visión y valores de la institución en donde se llevó a cabo la construcción de este documento.

En el capítulo 4 se exponen los objetivos y actividades que llevaron a la realización del manual, describiendo las metodologías empleadas, los procesos de investigación como

las entrevistas, el establecimiento del proceso de solicitud y creación de software. También se aborda la experiencia personal durante las prácticas profesionales.

Por último, en el capítulo 5 se presentan los resultados obtenidos, centrándose tanto en el proceso de solicitud como en el de aprobación, así como en la aplicación de las metodologías elegidas. A su vez, se tratan las etapas de desarrollo, proporcionando así una visión integral de la implementación buscada en el manual.

Palabras clave:

Manual, gestión, proyectos, software, metodología, desarrollo web.

Abstract

This thesis focuses on the development of the "Software Project Management Manualfor the UIEPCH", with the objective of providing a guide that seeks to improve efficiency and standardize processes related to university software projects. A brief summary of the keychapters will be presented below:

During the first chapter, the justification of the research is established, and the problema to be addressed is stated in order to define the objectives, both general and specific, that will allow the solution of the problem. Hypotheses are also proposed in order to guide the research towards the optimization of software Project management within the university environment.

The second chapter explores the theoretical framework where the research is based, giving rise to address essential concepts that will allow familiarization with the subject of this thesis. Within these concepts, elements such as the definition of a manual, the applicationand approval processes, including software stages, programming paradigms, among others, are shown.

The third chapter focuses on the information regarding the university institution, including details such as the name, company name, location, mission, vision and values of the institution where the construction of this document was carried out.

Chapter 4 presents the objectives and activities that led to the development of the manual, describing the methodologies used, the research processes such as interviews, the

establishment of the application process and the creation of software. Personal experience during the internship is also discussed.

Finally, Chapter 5 presents the results obtained, focusing on both the application andapproval processes, as well as on the application of the chosen methodologies. In turn, the development stages are discussed, thus providing a comprehensive view of the implementation sought in the manual.

Keywords:

Manual, management, projects, software, methodology, web development.

Capítulo 1

Introducción

La Universidad Interserrana del Estado de Puebla – Chilchotla está consciente de la realidad de la industria del software, por ende, se enfrenta al desafío de formar estudiantes con habilidades tanto prácticas como teóricas que sean relevantes para la industria del software. En este sentido, la creación de un manual que se enfoque en la gestión de software para convertirse en una herramienta clave que brinde a los estudiantes una comprensión más clara acerca de aspectos como la planificación, comunicación, coordinación, control de calidad, gestión de riesgos y monitoreo de proyectos.

El presente capítulo será el punto de partida en el desarrollo de dicho manual, que busca convertirse en una guía detallada que sea capaz de abarcar los aspectos relevantes en el ámbito de la gestión de proyectos de software universitario. Esto implica investigar de manera exhaustiva, basada en entrevistas realizadas a miembros de la universidad con la finalidad de evaluar el proceso de trabajo actual y el impacto que tendría la implementación del manual en la institución.

El planteamiento del problema ofrece una visión acerca de la problemática que da impulso a la creación de una guía: la falta de un protocolo claro para el proceso de solicitud de software que atrae consigo una seria de dificultades en la gestión de sistemas y que afecta a la eficiencia de los diversos procesos que se realizan. Esto lleva a definir objetivos que orienten a la elaboración del manual para que pueda adaptarse a las necesidades específicas de la UIEPCH.

Para ello, se investigarán la existencia de documentación similar para comprender como construir la composición del proceso de solicitud de software en la institución además de entender la construcción del software y como será desarrollado para finalmente, llegar a la redacción del manual de la forma más completa posible en base a las investigaciones previas.

Por último, la definición de hipótesis constituye a cuestionar si el desarrollo del manual contribuirá de forma significativa en la mejora del flujo de trabajo en la solicitud y construcción de software universitario. Con estas premisas, se procederá a abordar cada aspecto mencionado que ayudarán a sentar las bases de esta tesis y del Manual para la Gestión de Software de la Universidad Interserrana del Estado de Puebla – Chilchotla.

Justificación

La industria del software es considerada una de las más importantes y, por ende, tanto su crecimiento como su demanda por profesionales capacitados es exponencial generando así que la gestión de proyectos de software esté en aumento. Por ende, es clave que las universidades formen egresados con habilidades prácticas y teóricas relacionadas con esta disciplina.

La implementación de un manual de gestión de proyectos de software en la UIEPCH tiene como objetivo ayudar a los estudiantes a comprender el valor de aspectos como la planificación y sus técnicas, la comunicación, la coordinación, el control de calidad, la gestión de riesgos y el monitoreo de un proyecto de software desde el inicio hasta la entrega final. Además, brindará a las direcciones una comprensión completa del proceso y documentación necesaria para la solicitud de software en la institución.

La investigación realizada en esta tesis estará basada primeramente en la realización de entrevistas entre los miembros principales de cada dirección de la universidad para poder determinar el proceso actual de trabajo en el ámbito del desarrollo de actividades sin la presencia de software generado en la propia institución, además de conocer el impacto con respecto a la implementación de un manual de gestión de proyectos de software.

Los resultados de esta investigación permitirán justificar la necesidad de implementar un manual de gestión de proyectos de software, al igual que proporcionar recomendaciones prácticas acerca del contenido y la metodología elegida para el manual. La tesis también contribuirá a que los estudiantes de Servicio Social o Prácticas Profesionales para el

desarrollo de habilidades y la adaptación de técnicas adquiridas durante el desarrollo del proyecto.

Planteamiento del problema

La universidad en la actualidad carece de un protocolo claro para el proceso de solicitud de software, lo que ha generado problemas en la gestión de sistemas. La ausencia de una normativa en este aspecto genera problemas al momento de gestionar sistemas, porque no se conoce a quien dirigir las solicitudes además de no conocer a un comité encargado del análisis y aprobaciones de dichas solicitudes. Además, la falta de regulación para elaborar software ha dado a obtener diseños complejos, deficientes o "pobres" afectando negativamente la eficiencia de procesos productivos universitarios.

Por ende, es necesario contar con un conjunto de normas capaces de definir el proceso de forma clara, cómo funciona el proceso de solicitud de software, incluyendo, cómo se mencionó anteriormente, el señalamiento de quien será el responsable de recibir las solicitudes, además de esclarecer las reglas de diseño y la programación a considerar para la creación de los sistemas. Esto también implica conocer al comité encargado del análisis y las decisiones de las solicitudes, el cual garantice que los proyectos a realizar se ajusten a los estándares de calidad y seguridad establecidos por la universidad.

Cabe añadir la importancia de contar con un equipo de desarrollo que tenga las habilidades y conocimientos necesarios que sean aplicables en la creación y/o uso de herramientas para la solución de problemas permitiendo así no depender exclusivamente de software de terceros. Esto permitiría que la universidad cuente con soluciones más adecuadas y personalizadas, y al mismo tiempo, ayudar al fomento de habilidades técnicas y creativas dentro del equipo universitario.

Objetivos

Objetivo general

Desarrollar un Manual de Gestión de Proyectos de Software para la Universidad Interserrana del Estado de Puebla – Chilchotla.

Objetivos específicos

- Investigar si existe un manual de gestión de proyectos aplicado con anterioridad.
- Preguntar cómo se realiza la petición de software dentro de la institución.
- Redactar el manual de gestión de proyectos en base a las investigaciones previamente realizadas.

Hipótesis

La creación de un manual de gestión de proyectos de software aplicable a la universidad permitirá establecer un flujo de trabajo eficiente tanto para la solicitud como para la construcción de software universitario.

H1: El manual detallará el proceso completo de solicitud de un sistema, incluyendo los formatos requeridos, el personal involucrado, la forma en que se decidirá sobre la solicitud y la comunicación del veredicto, mejorando así la transparencia y eficiencia del proceso.

H2: El manual seleccionará y describirá la metodología óptima para el desarrollo del sistema, incluyendo los formatos y productos necesarios en cada fase, garantizando una gestión de proyectos coherente y eficaz.

Capítulo 2. Marco teórico

Estado del arte

Investigaciones internacionales

Gestión de Proyectos de Software Basado en metodología PMP

Gómez Barbosa, J. S., Orozco Romero, C. J., Rico Marulanda, J. E., Zamora Valero, A. (2020). Gestión de proyectos de software basado en metodología PMP [Tesis de Especialización en Gestión de Redes de Datos, Universidad Santo Tomas] Repositorio Institucional - Santo Tomás de Aquino.

Objetivo: Abordar los temas más importantes que permiten realizar un paso a paso del planteamiento, ejecución, evaluación y conclusión de un proyecto.

Muestra: Son todos aquellos procesos requeridos para establecer el alcance del proyecto para refinar y ajustar los objetivos y definir las acciones para alcanzar la finalidad del proyecto.

Factores: El esfuerzo de cierre incluye actividades administrativas como la recolección y finalización de la documentación necesaria para completar el proyecto y requerirá de cierto trabajo técnico para confirmar que el producto final del proyecto es aceptable.

Resultados: En la construcción del proyecto, se abordaron múltiples temas de distintas áreas, estructurando una metodología para resolver el problema planteado, específicamente en la creación de una metodología aplicada a los proyectos de software. Al analizar las áreas, se identificaron debilidades, oportunidades, fortalezas y amenazas

(FODA), lo que permitió minimizar riesgos y así encontrar nuevas formas de desarrollo. Esto da como resultado una línea de conocimiento que facilitará la identificación de factores que afecten el proyecto, resaltando la importancia de implementar y asignar una metodología que reduzca los riesgos en la gestión del proyecto.

Conclusiones: La terminación de tareas en la elaboración y ejecución de un proyecto permite establecer tiempos, alcances y costos, lo que facilita la evolución bajo control y registro de actividades. La implementación de modelos de gestión mejora la calidad del producto además de que estandariza procesos, permitiendo así la asignación de recursos según la matriz de riesgos y estándares de calidad. Esta metodología brinda control enfocado en el alcance, costos y tiempo de duración del proyecto, permitiendo así adaptarse a su tamaño y duración. Las metodologías guían la gestión de proyectos, aplicándose según la necesidad de cada proyecto.

Gestión de proyectos de software

Rodríguez, Francisco & Hurtado, Julio & Arellano, Margarita & Muñoz, Jaime & Velázquez Amador, Cesar & Hernández Bieliukas, Yosly. (2014). Gestión de Proyectos de Software.

Objetivo: El objetivo del libro es brindar elementos conceptuales y técnicos para definir y modelar procesos de software de tal forma que sean utilizados por los distintos roles dentro de la ingeniería de software de la organización y que su representación, adaptación y análisis por parte del equipo de procesos sea apoyado a través de un procesamiento computacional.

Muestra: La principal aportación del libro será unificar contenidos para la Industria de Software Latinoamericana considerando las diferentes capacidades y experiencias de los académicos de las institucionales participantes.

Los niveles académicos que pretender abordar el libro son la diferentes Licenciaturas e Ingenierías de tengas tópicos en Ingeniería de Software, así como Posgrados relacionados con la temática. Aproximadamente se podría tener un impacto de al menos 200 alumnos de las instituciones participantes a nivel de licenciatura y de 40 alumnos a nivel de posgrado.

Factores: Durante el proceso de desarrollo de software se busca lograr un balance efectivo entre la productividad, la calidad, la armonía organización al y por tanto la competitividad. La industria de software requiere de la capacidad ingenieril para desarrollar sus proyectos, pero el éxito va más allá de la capacidad técnica, y una de esas relevantes capacidades es la de poder administrar adecuadamente sus proyectos.

Resultados: Este libro busca brindar los fundamentos y las herramientas conceptuales y técnicas para formar ingenieros de software orientados hacia la gestión práctica de procesos, con el ánimo de fortalecer la formación de los estudiantes de computación, de forma acorde a las necesidades de la creciente industria de software en Latinoamérica, quienes se desempeñarán como líderes de proyectos, de calidad, consultores, analistas de procesos, así como la investigación, innovación y desarrollo de nuevos modelos de gestión de proyectos de software.

Conclusiones: Derivado de la necesidad de definir un perfil específico para la Ingeniería de software acorde a los requerimientos de la Industria latinoamericana de Tecnologías e Información y Comunicación. La academia podrá utilizar el presente como

referente para la creación de programas y/o adaptación de su plan de estudios; a la Industria como referente para la selección y contratación de personal competente para cubrir sus necesidades de Ingeniería de Software; y para el aspirante como referente de los requisitos necesarios para obtener el conocimiento como "Ingeniero de Software" en el área de procesos y gestión de proyectos.

Implementación de un modelo de desarrollo de software orientado a la web para la universidad Simón Bolívar Sede Cúcuta

Angarita Sanguino, Carlos & Gallardo, O. (2015). Implementando un Modelo de Desarrollo de Software Orientado a la Web para la Universidad Simón Bolívar -Sede Cúcuta. investigación e Innovación en ingenierías. 3.10.17081/invinno.3.2.20 29.

Objetivo: Consolidar el aporte del Sistema de Información SIAA al desarrollo de los procesos académicos y administrativos del sistema de gestión institucional.

Muestra: El Sistema de Información Académico, creado en la sede principal de Barranquilla, consta de la aplicación cliente-servidor desarrollados en Visual Basic y una base de datos DB2, comunicados mediante la red institucional. Estos abarcan áreas administrativas y académicas empleándose en la sede Cúcuta como solución integral.

Además, se desarrolló el uso de una web desarrollada PHP para los estudiantes, desde la matrícula hasta la selección de horarios, comenzando en Barranquilla para llegar a Cúcuta con ajustes locales. Durante la implementación, se identificó la necesidad de gestionar ciertos datos con otra forma de aplicación aún no completamente implementado en la sede.

Factores: La nueva arquitectura plantea los siguientes componentes principales:

1) Base de Datos (Modelo): Permite realizar la persistencia de la información:

- Tablas: estructuras de datos que almacenan la información.
- Funciones: contiene lógica del negocio.
- Disparadores: funciones que se ejecutan ante cualquier cambio en los datos de las tablas.
- Clases de Entidad: objetos simples que representan las estructuras de los datos y son gestionados como objetos desde la aplicación.

2) Aplicaciones Académicas (Vista): Corresponden a todas las interfaces que permiten el ingreso y consulta de información.

3) Reportes: dentro del sistema se plantea un modelo de extracción de información a través de reportes PDF que serán diseñados y generados con la ayuda de Jasper Report.

Resultados: El proyecto consiste en una aplicación web que sirva para gestionar la información académica en la sede Cúcuta de la Universidad Simón Bolívar. Este incluye funcionalidades tales como administración de carga académica, infraestructura y actividades complementarias.

Desarrollado en PHP y HTML4, utiliza Bootstrap5 para la vista además de clases POO en PHP para el controlador. La metodología de desarrollo incluye la captura de requerimientos, pruebas de tecnología, desarrollo del módulo, prueba y documentación, y puesta en producción con capacitación a los usuarios. La implementación se realizó en dos etapas para los módulos de carga académica e información docente, con capacitaciones y ajustes basados en las observaciones de los usuarios.

Conclusiones: Este proyecto generó un modelo de desarrollo de aplicaciones, que se convierte en la base de todos los futuros sistemas que se realicen en la sede, y después de diversas reuniones, se evidencia que hay bastante trabajo aun por realizar y diversos sistemas que desarrollar, tanto para la gestión académica como parala gestión administrativa. Después de reuniones iníciales con los jefes de procesos, ya se estudia la posibilidad de desarrollar diversos módulos y sistemas:

- Sistema de Gestión de Congresos
- Sistema de Gestión y Seguimientos de Clientes (Estudiantes)
- Sistema de Gestión de Grupos de Investigación y Semilleros Al cierre de este artículo ya se encontraba en desarrollo un módulo para la gestión de proyectos de investigación formativa y un sistema de bitácora de docentes.

Igualmente se pudo evidenciar que el proceso de desarrollo de nuevos módulos es muy rápido, y los procesos y documentación realizados en el modelo inicial son muy útiles y permiten una apropiación de la tecnología de forma rápida.

Investigaciones nacionales

Guía Técnica de Gestión de Proyectos de Software

Chi, R. I. G. (2018). GUIA TECNICA DE GESTIÓN DE PROYECTOS DE SOFTWARE.89.

Objetivo: Elaborar un Manual de Prácticas para la asignatura de Gestión de Proyectos de Software de la carrera de Ingeniería en Sistemas computacional es, que proporcione al

alumno la habilidad y destreza para iniciar, planear, ejecutar y llevar a cabo el cierre de un proyecto, a través de la aplicación del PMBOK para la gestión de proyectos.

Muestra: La gestión de proyectos de software en el ámbito académico, específicamente en el Instituto Tecnológico de Ciudad Valles tocando temas como las metodologías, procedimientos, estándares, herramientas y técnicas utilizadas en la gestión de proyectos de software

Factores: Algunos de los factores a considerar en la gestión de proyectos de software son el identificar la importancia de la investigación para poder planear posibles formas de resolver sobre el problema investigado en la justificación indicando la factibilidad de las soluciones y la consideración de los tiempos, alcances y posibilidades al iniciar el proceso de investigación

Resultados: La gestión de proyectos de software busca lograr aspectos como establecer criterios de calidad y verificar su cumplimiento, identificar y gestionar los riesgos potenciales, resolver de forma inmediata los problemas surgidos, brindar una dirección clara al equipo, enfocar los resultados en los objetivos de los proyectos, ejecutar los planes de manera eficiente, priorizar y organizar el trabajo según las necesidades del proyecto, motivar al personal involucrado, comunicar claramente el propósito del proyecto y reconocer el éxito al concluir el proyecto.

Conclusiones: Se enfatiza la importancia de implementar un Plan de Aseguramiento de la Calidad en el desarrollo de software garantizando que se cumplan las características deseables del producto final, destacando la necesidad de ajustarse a modelos o estándares que sean capaces de medir estas características y establecer estrategias para alinearse a ellas

en caso de incumplimiento. También se resalta la relevancia de la gestión de calidad en el desarrollo de software que incluye la garantía y planificación de la calidad como actividades principales.

Tabla 1. Estado del arte

	Titulo	Objetivo	Muestra	Factores	Resultados	Conclusiones	Bibliografía
Investigaciones internacionales	Gestión de Proyectos de Software Basado en metodología PMP	Abordar los temas más importantes que permiten realizar un paso a paso del planteamiento, ejecución, evaluación y conclusión de un proyecto.	Son todos aquellos procesos requeridos para establecer el alcance del proyecto para refinar y ajustar los objetivos y definir las acciones para alcanzar la finalidad del proyecto.	El esfuerzo de cierre incluye actividades administrativas como la recolección y finalización de la documentación necesaria para completar el proyecto y requerirá de cierto trabajo técnico para confirmar que el producto final del proyecto es aceptable.	En la construcción del proyecto, se abordaron múltiples temas de distintas áreas, estructurando una metodología para resolver el problema planteado, específicamente en la creación de una metodología aplicada a los proyectos de software. Al analizar las áreas, se identificaron debilidades, oportunidades, fortalezas y amenazas (FODA), lo que permitió minimizar riesgos y así encontrar nuevas formas de desarrollo. Esto da como resultado una línea de conocimiento que facilitará la identificación de factores que afecten el proyecto, resaltando la importancia de implementar y asignar una metodología que reduzca los riesgos en la gestión del proyecto.	La terminación de tareas en la elaboración y ejecución de un proyecto permite establecer tiempos, alcances y costos, lo que facilita la evolución bajo control y registro de actividades. La implementación de modelos de gestión mejora la calidad del producto además de que estandariza procesos, permitiendo así la asignación de recursos según la matriz de riesgos y estándares de calidad. Esta metodología brinda control enfocado en el alcance, costos y tiempo de duración del proyecto, permitiendo así adaptarse a su tamaño y duración. Las metodologías guían la gestión de proyectos, aplicándose según la necesidad de cada proyecto.	Gómez Barbosa, J. S., Orozco Romero, C. J., Rico Marulanda, J. E., Zamora Valero, A. (2020). Gestión de proyectos de software basado en metodología PMP [Tesis de Especialización en Gestión de Redes de Datos, Universidad Santo Tomas] Repositorio Institucional - Santo Tomás de Aquino.
	Gestión de proyectos de software	El objetivo del libro es brindar elementos conceptuales y técnicos para definir y modelar procesos de software de tal forma que sean utilizados por los	La principal aportación del libro será unificar contenidos para la Industria de Software latinoamericana considerando las diferentes capacidades y experiencias de los	Durante el proceso de desarrollo de software se busca lograr un balance efectivo entre la productividad, la calidad, la armonía organización al y por tanto la	Este libro busca brindar los fundamentos y las herramientas conceptuales y técnicas para formar ingenieros de software orientados hacia la gestión práctica de procesos, con el	Derivado de la necesidad de definir un perfil específico para la Ingeniería de software acorde a los requerimientos de la Industria latinoamericana de	Rodríguez, Francisco & Hurtado, Julio & Arellano, Margarita & Muñoz, Jaime & Velázquez Amador, Cesar & Hernández Bieliukas, Yosly. (2014). Gestión de Proyectos de Software.

		distintos roles dentro de la ingeniería de software de la organización y que su representación, adaptación y análisis por parte del equipo de procesos sea apoyado a través de un procesamiento computacional.	académicos de las institucionales participantes. Los niveles académicos que pretender abordar el libro son la diferentes Licenciaturas e Ingenierías de tengas tópicos en Ingeniería de Software, así como Posgrados relacionados con la temática. Aproximadamente se podría tener un impacto de al menos 200 alumnos de las instituciones participantes a nivel de licenciatura y de 40 alumnos a nivel de posgrado.	competitividad. La industria de software requiere de la capacidad ingenieril para desarrollar sus proyectos, pero el éxito va más allá de la capacidad técnica, y una de esas relevantes capacidades es la de poder administrar adecuadamente sus proyectos.	ánimo de fortalecer la formación de los estudiantes de computación, de forma acorde a las necesidades de la creciente industria de software en Latinoamérica, quienes se desempeñarán como líderes de proyectos, de calidad, consultores, analistas de procesos, así como la investigación,innovación y desarrollo de nuevos modelos de gestión de proyectos de software.	Tecnologías e Información y Comunicación. La academia podrá utilizar el presente como referente para la creación de programas y/o adaptación de su plan de estudios; a la Industria como referente para la selección y contratación de personal competente para cubrir sus necesidades de Ingeniería de Software; y para el aspirante como referente de los requisitos necesarios para obtener el conocimiento como "Ingeniero de Software" en el área de procesos y gestión de proyectos.	
	Implementación de un modelo de desarrollo de software orientado a la web para la universidad Simón Bolívar sede Cúcuta	Consolidar el aporte del Sistema de Información SIAA al desarrollo de los procesos académicos y administrativos del sistema de gestión institucional.	El Sistema de Información Académico, creado en la sede principal de Barranquilla, consta de la aplicación cliente-servidor desarrollados en Visual Basic y una base de datos DB2, comunicados mediante la red institucional. Estos abarcan áreas administrativas y académicas empleándose en la sede Cúcuta como solución integral. Además, se desarrolló el uso de una web desarrollada PHP para los	La nueva arquitectura plantea los siguientes componentes principales: 1) Base de Datos (Modelo): permite realizar la persistencia de la información: □ Tablas: estructuras de datos que almacenan la información. □ Funciones: contiene lógica del negocio. □ Disparadores: funciones que se ejecutan ante	El proyecto consiste en una aplicación web que sirva para gestionar la información académica en la sede Cúcuta de la Universidad Simón Bolívar. Este incluye funcionalidades tales como administración de carga académica, infraestructura y actividades complementarias. Desarrollado en PHP y HTML4, utiliza Bootstrap5 para la vista además de clases POO en PHP para el controlador. La	Este proyecto generó un modelo de desarrollo de aplicaciones, que se convierte en la base de todos los futuros sistemas que se realicen en la sede, y después de diversas reuniones, se evidencia que hay bastante trabajo aun por realizar y diversos sistemas que desarrollar, tanto para la gestión académica como para la gestión administrativa. Después de reuniones	Angarita Sanguino, Carlos & Gallardo, O. (2015). Implementando un Modelo de Desarrollo de Software Orientado a la Web para la Universidad Simón Bolívar - Sede Cúcuta. investigación e Innovación en ingenierías. 3. 10.17081/invinno.3.2.20 29.

			estudiantes, desde la matrícula hasta la selección de horarios, comenzando en Barranquilla para llegar a Cúcuta con ajustes locales. Durante la implementación , se identificó la necesidad de gestionar ciertos datos con otra forma de aplicación aún no completamente implementado en la sede.	cualquier cambio en los datos de las tablas. Clases de Entidad: objetos simples que representan las estructuras de los datos y son gestionados como objetos desde la aplicación. 2) Aplicaciones Académicas (Vista): corresponden a todas las interfaces que permiten el ingreso y consulta de información. 3) Reportes: dentro del sistema se plantea un modelo de extracción de información a través de reportes PDF que serán diseñados y generados con la ayuda de Jasper Report.	metodología de desarrollo incluye la captura de requerimientos, pruebas de tecnología, desarrollo del módulo, prueba y documentación , y puesta en producción con capacitación a los usuarios. La implementación se realizó en dos etapas para los módulos de carga académica e información docente, con capacitaciones y ajustes basados en las observaciones de los usuarios.	iníciales con los jefes de procesos, ya se estudia la posibilidad de desarrollar diversos módulos y sistemas: Sistema de Gestión de Congresos Sistema de Gestión y Seguimientos de Clientes (Estudiantes) Sistema de Gestión de Grupos de Investigación y Semilleros Al cierre de este artículo ya se encontraba en desarrollo un módulo para la gestión de proyectos de investigación formativa y un sistema de bitácora de docentes. Igualmente se pudo evidenciar que el proceso de desarrollo de nuevos módulos es muy rápido, y los procesos y documentación realizados en el modelo inicial son muy útiles y permiten una apropiación de la tecnología de forma rápida.	

Investigaciones nacionales	Guía Técnica de Gestión de Proyectos de Software	Elaborar un Manual de Prácticas para la asignatura de Gestión de Proyectos de Software de la carrera de Ingeniería en Sistemas computacional es, que proporcione al alumno la habilidad y destreza para iniciar, planear, ejecutar y llevar a cabo el cierre de un proyecto, a través de la aplicación del PMBOK para la gestión de proyectos.	La gestión de proyectos de software en el ámbito académico, específicamente en el Instituto Tecnológico de Ciudad Valles tocando temas como las metodologías, procedimientos, estándares, herramientas y técnicas utilizadas en la gestión de proyectos de software.	Algunos de los factores a considerar en la gestión de proyectos de software son el identificar la importancia de la investigación para poder planear posibles formas de resolver sobre el problema investigado en la justificación indicando la factibilidad de las soluciones y la consideración de los tiempos, alcances y posibilidades al iniciar el proceso de investigación.	La gestión de proyectos de software busca lograr aspectos como establecer criterios de calidad y verificar su cumplimiento, identificar y gestionar los riesgos potenciales, resolver de forma inmediata los problemas surgidos, brindar una dirección clara al equipo, enfocar los resultados en los objetivos de los proyectos, ejecutar los planes de manera eficiente, priorizar y organizar el trabajo según las necesidades del proyecto, motivar al personal involucrado, comunicar claramente el propósito del proyecto y reconocer el éxito al concluir el proyecto.	Se enfatiza la importancia de implementar un Plan de Aseguramiento de la Calidad en el desarrollo de software garantizando que se cumplan las características deseables del producto final, destacando la necesidad de ajustarse a modelos o estándares que sean capaces de medir estas características y establecer estrategias para alinearse a ellas en caso de incumplimiento . También se resalta la relevancia de la gestión de calidad en el desarrollo de software que incluye la garantía y planificación de la calidad como actividades principales.	Chi, R. I. G. (2018). GUIA TECNICA DE GESTIÓN DE PROYECTOS DE SOFTWARE. 89.

Marco teórico

Manual

Dentro de un contexto general, un manual "es un libro o folleto en el cual se recopilan los aspectos básicos y esenciales de una actividad de la organización" (Olvera, 2021), por ende, nos permite acceder a conocimiento ordenado y conciso sobre algún tema o materia. Este tipo de documentos permite que las organizaciones naveguen entre procesos para así controlar cada uno de ellos permitiendo así el desarrollo de manera eficiente.

Enfocado al contexto de la gestión de software universitario, se le considera como un documento capaz de proporcionar una estructura y guía para la planificación, ejecución y control de proyectos de desarrollo de software. Para nuestra universidad, el manual se constituyó en dos grandes temas: la solicitud y construcción del software.

Norma ISO 9001:2015

La norma ISO 9001-2015 se considera como una norma internacional que establece los requisitos necesarios para un sistema de gestión de calidad de cualquier entidad. "Es la norma de SGC más utilizada en todo el mundo, con más de 1 millón de certificados emitidos en más de 178 países." (_Certificación ISO 9001 - Norma de gestión de la calidad | NQA_, 2019).

Esta norma no distingue entre organizaciones ni por su tamaño ni por el sector en donde se encuentre, ya que se establece como un marco de trabajo y un conjunto de principios que asegura un enfoque lógico capaz de satisfacer tanto a clientes como a cualquier parte interesada en la gestión de calidad. Esto lo hace sentando las bases para desarrollar procesos

y elementos efectivos dando como resultado un producto o servicios efectivos prolongados en el tiempo.

Proceso de solicitud y aprobación

El proceso de solicitud y aprobación presentado en este artículo se considera una serie de pasos que conducen a su definición e implementación, centrándose en las diferentes partes que componen el libro (solicitud, aprobación y desarrollo de software en la UIEPCH). Este proceso incluye una serie de pasos para garantizar una identificación de requisitos y una toma de decisiones efectivas.

El primer paso es identificar los requisitos que impulsan los requisitos de software de su empresa. Estos se revelan a través de entrevistas y análisis de datos para comprender cómo abordar el desarrollo de software y forman la base del libro.

Una vez aprobados los requisitos, comienza el proceso de solicitud. Siguiendo los documentos mencionados en el manual, que contienen la información más importante sobre el software necesario para una comprensión más profunda de los objetivos del proyecto y otra información relevante para la sección de evaluación y decisión.

Ya realizada la evaluación correspondiente, la solicitud será resuelta, aprobada o rechazada. De aprobarse, se continuará con la siguiente fase con el objetivo de aplicar la solución propuesta.

Por último, se hará el comunicado sobre la decisión al solicitante contemplando una explicación clara sobre los motivos que respaldan la decisión tomada. La transparencia

durante esta acción es crucial para mantener un mensaje claro y efectivo con los pasos involucrados durante el proceso.

Gestión de proyectos de software

Se considera a la gestión de proyectos como un conjunto de metodologías utilizadas para garantizar el acompañamiento y la conclusión exitosa de un proyecto. dicho de otra manera: "La gestión de proyectos es la práctica de coordinar procesos, herramientas, miembros del equipo y habilidades para entregar proyectos que cumplan con los objetivos y satisfagan los requisitos" (Atlassian, 2024).

Por otro lado, al enfocar este término a este documento, se consideró como un conjunto de técnicas y herramientas destinadas a la planificación, organización y supervisión del desarrollo de software apuntando a completar el desarrollo en tiempo y forma, además de cumplir con los requisitos especificados. Esto provocó que el manual se dividiera en dos grandes etapas: planificación, implementación y evaluación.

Metodología ágil Scrum – Lean

El siguiente punto trata de las metodologías ágiles, las cuales se definen como "aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno" (Sandra Garrido Sotomayor, 2023), sin embargo, también se pueden referir a estas metodologías como el conjunto de métodos capaces de evolucionar los cuales son basados en procesos de mejoras oportunistas e iterativas dentro del desarrollo.

Partiendo de esto, es preciso destacar las dos metodologías que son clave en la realización del manual. En primer lugar, la metodología Scrum ofrece un enfoque iterativo e incremental que tiene por base la transparencia, la inspección y la adaptación que busca predecir y controlar riesgos. Por otro lado, la metodología Lean se centra en la eliminación de "desperdicios" y la optimización de los procesos buscando la mejora continua de forma segura y sostenida.

Desarrollo de software

Para referirnos al desarrollo de software, debemos entenderlo cómo "un conjunto de actividades informáticas dedicadas al proceso de creación, diseño, despliegue y compatibilidad de software" (IBM, 2020). Es decir, un conjunto de actividades destinadas al diseño, creación y despliegue enfocado al software, el cual, contiene una serie de pasos relacionados. Este concepto está estrechamente relacionado con la selección de una metodología que permita dar una mejor definición de los pasos a realizar, las etapas que puede abarcar el proyecto e incluso en cómo se debe construir el software deseado.

Por ende, al combinar las metodologías Scrum y Lean con el desarrollo de software se ha dado como resultado el manual destinado a la UIEPCH, ya que, como se verá en temas posteriores, se buscó implementar un marco ágil que dé como resultado que el desarrollo de los softwares elaborados para la institución sean construidos de manera fluida y rápida, pero, sobre todo, que se conviertan en las mejores opciones para la universidad.

Etapas de desarrollo de software

Además del tema anterior, el desarrollo de software se divide en diferentes fases dependiendo del método elegido. En general, existen una serie de pasos que aseguran que

esta fase se lleve a cabo satisfactoriamente, incluyendo aspectos como la planificación, ejecución y gestión efectiva de los recursos disponibles para ejecutar el proyecto con el mayor mérito posible. Por lo tanto, estos aspectos hacen que el proceso sea rentable y eficaz para los equipos de desarrollo responsables de hacer realidad el software.

El desarrollo de software, como se mencionó anteriormente, tiene pasos definidos. relacionados con el método que se está implementando, aunque en general las fases pueden identificarse como planificación, análisis, diseño, programación, prueba, implementación, mantenimiento y documentación, incluidas actividades como definir requisitos y objetivos, comprender estos factores, elegir interfaces de usuario adicionales para herramientas y lenguajes necesarios para el proceso de construcción, codificación del software con las pruebas necesarias para poder brindar un producto satisfactorio, además del proceso de mantenimiento y la documentación sobre cómo se llevará a cabo el proyecto.

Maquetación de software

La maquetación de software se le considera como un proceso donde se trabaja el diseño y la presentación visual enfocada en la interfaz de usuario de un software. Este proceso crea lo que a menudo se llama un "prototipo" o "modelo" del software que sirve como retroalimentación tanto para los usuarios como para el equipo de desarrollo, permitiendo realizar los ajustes correspondientes antes de la fase de código.

Este proceso incluye el entendimiento de los requisitos del software, investigación y análisis de los requisitos dados en las primeras etapas, donde se trabajará con elementos como textos, imágenes, botones, íconos, formularios y muchos otros para que los usuarios tengan la mejor interacción con lo que será el software final. Como ejemplo de este proceso, está la

maquetación web que se entiende como "el procedimiento por el cual un profesional del diseño dispone de los elementos (como texto, títulos o imágenes) para hacer que un contenido pueda ser consumido de una manera óptima" (Tirosh, 2021).

Lenguaje de programación web

“Los lenguajes de programación no han dejado de ser un conjunto de símbolos con una estructura gramatical, reglas semánticas y de sintaxis" (Monterde, 2017).

Se le denomina lenguaje de programación a la herramienta empleada en la creación de software ya que permite diseñar, implementar y definir el comportamiento de dispositivos físicos y lógicos existentes en una computadora. Dicho de otra manera, es el conjunto de instrucciones necesarias en la interacción humano - máquina por medio de algoritmos e instrucciones escritas en una sintaxis que la computadora sea capaz de entender e interpretar en el famoso lenguaje máquina.

Esta manera de comunicación ayuda a las computadoras a procesar de manera efectiva y rápida información compleja y en grandes cantidades, por ejemplo, si un usuario ingresa una lista de nombres aleatorios de los alumnos de una institución, es muy probable que la cantidad de información sea muy grande e implique una considerable cantidad de tiempo si se desea buscar de manera manual un nombre en específico. Esto será más fácil empleando un lenguaje de programación, ya que proporcionará una respuesta más rápida y sin errores.

Existen múltiples lenguajes de programación utilizados de forma activa hoy en día, por ejemplo, lenguajes como C++, Python, Go, Ruby, JavaScript, Java, entre otros que

encapsulan en sí mismos métodos, funciones, objetos e inclusive fases de compilación dependiendo del tipo y el entorno donde se aplicará el lenguaje.

Paradigmas de programación

En cuanto a los paradigmas de la programación, se puede definir como "los principios fundamentales de la programación de software" (Ferdinandi, 2020), lo cual encapsula consigo los principios y directrices que definen como se debe diseñar, estructurar y escribir código. Aunque también pueden considerarse como marcos conceptuales que definen como se construye el software ya que como guías para determinar la estructura final del software.

Sin embargo, hay muchos paradigmas diferentes que se utilizan en el mundo de la programación. Estos se pueden definir en 8 clasificaciones, que van desde la programación iterativa (C y Pascal), la programación declarativa (Haskell y Prolog) que contiene a su vez la programación funcional (Lisp, Haskell y Earlang) y la lógica (Prolog), la programación orientada a objetos u OOP (Java, Python y C++), la programación reactiva (RxJava y ReactiveX) hasta la programación basada en eventos (JavaScript).

Documentación de código

El término documentación tiene distintos significados, sin embargo, aplicado al desarrollo de software, "se define como la información enfocada en la Descripción del sistema o producto para quienes se encargan de desarrollarlo, implementarlo y utilizarlo" (Team, 2022). Este tipo de documentación incorporan aspectos del tipo manual tales como las funciones de ayuda, versiones entre otros aspectos.

La documentación de código se emplea especialmente cuando existen errores que se deban reparar, se requiere actualizar el software con nuevas funcionalidades o que se requiera

escalarlo y/o adaptarlo para una nueva implementación. Este debe contener como primer nivel el uso adecuado del lenguaje de programación utilizado, específicamente los nombres, variables, métodos y clases demostrando la limpieza y orden del código.

Además, debe contener, de ser necesario, los comentarios realizados durante la codificación, es decir, pequeñas explicaciones de aspectos no evidentes que buscan no repetir lo que el código hace, sino más bien aclarar por qué el código hace lo que hace. Estos códigos pueden definir que se encarga una clase, para que sirve una función, cual es el uso de las variables, que tipo de algoritmos se emplearon al igual que sus límites, por mencionar algunos ejemplos.

Pruebas de software

Se considera como prueba de software al "proceso que se utiliza para garantizar la precisión, integridad y calidad del software" (Regulwar & Gulhane, 2010), con esto se busca que se cumplan con los requisitos especificados y se corrobore si funciona de manera correcta. Las pruebas de software consisten en ejecutar el software con la intención de detectar si existen errores y/o defectos, a su vez, se verifica que el comportamiento sea el que se esperaba.

Esta etapa "es una parte esencial del ciclo de vida del desarrollo de software, ya que ayuda a asegurar la calidad y confiabilidad del software" (Brown y otros, 2007), sin embargo, se tomaron en cuenta dos tipos de pruebas principales: las pruebas funcionales y las pruebas no funcionales. Las pruebas funcionales buscan verificar la funcionalidad del software bajo diversas entradas y condiciones que aseguren que se producen las salidas esperadas.

Generalmente están basadas en especificaciones funcionales o requerimientos del software anteriormente planteados.

Por otro lado, las pruebas no funcionales son aquellas que evalúan aspectos como la confiabilidad, usabilidad, compatibilidad y seguridad del software. También evalúan como se desempeña el software en términos de velocidad, capacidad de respuesta, estabilidad, facilidad de uso, compatibilidad con distintas plataformas y el nivel de protección que tienen contra posibles amenazas de seguridad.

Capitulo 3. Referencial

Nombre

Universidad Interserrana del Estado de Puebla – Chilchotla.

Razón social

Universidad Interserrana del Estado de Puebla - Chilchotla

Ubicación

Av. Miguel Hidalgo s/n, C.P. 75070 Chilchotla, Puebla.

Figura 1. Ubicación de la Universidad Interserrana del Estado de Puebla - Chilchotla. [Mapa]. Fuente: https://uich.edu.mx

Misión

“Formar profesionistas creativos, emprendedores y competitivos, a través de programas educativos pertinentes a la zona de influencia, con un diseño curricular tecnológico y

humanista, además de docentes comprometidos con la excelencia académica, para integrar egresados profesionales con vocación de servicio y valores; a los diferentes sectores de la sociedad."

Visión

"Ser una institución de educación superior que tenga reconocimiento regional, estatal y nacional, con programas educativos acreditados, aplicando tecnologías de la información y comunicación en el proceso enseñanza-aprendizaje, con enfoque sustentable y sostenible; impactando como agente promotor del desarrollo socioeconómico y cultural a través de la vinculación con los sectores de la sociedad."

Valores

- Solidaridad
- Responsabilidad
- Humildad
- Empatía
- Creatividad
- Respeto
- Lealtad
- Honestidad
- Ética
- Equidad

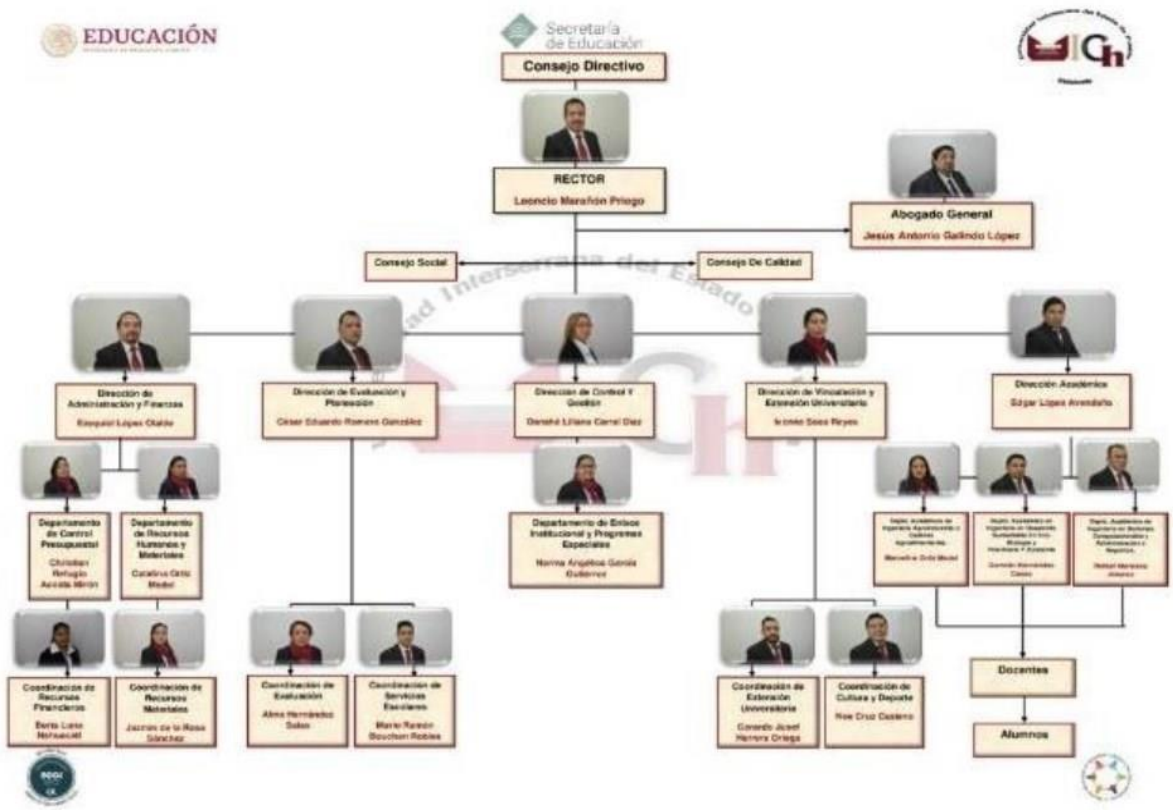


Figura 2. Organigrama de la Universidad Interserrana del Estado de Puebla - Chilchotla. [Imagen]. Fuente: https://uich.edu.mx/organigrama/

Capítulo 4. Metodología

En este capítulo se abordará el proceso de creación de la Guía de Gestión de Proyectos de Software UIEPCH, una herramienta diseñada para mejorar la consulta, planificación y creación de software en la Universidad Interserrana del Estado de Puebla - Chilchotla. La guía fue creada como respuesta a la necesidad de mejorar los procesos de gestión de proyectos de software específicos en una institución.

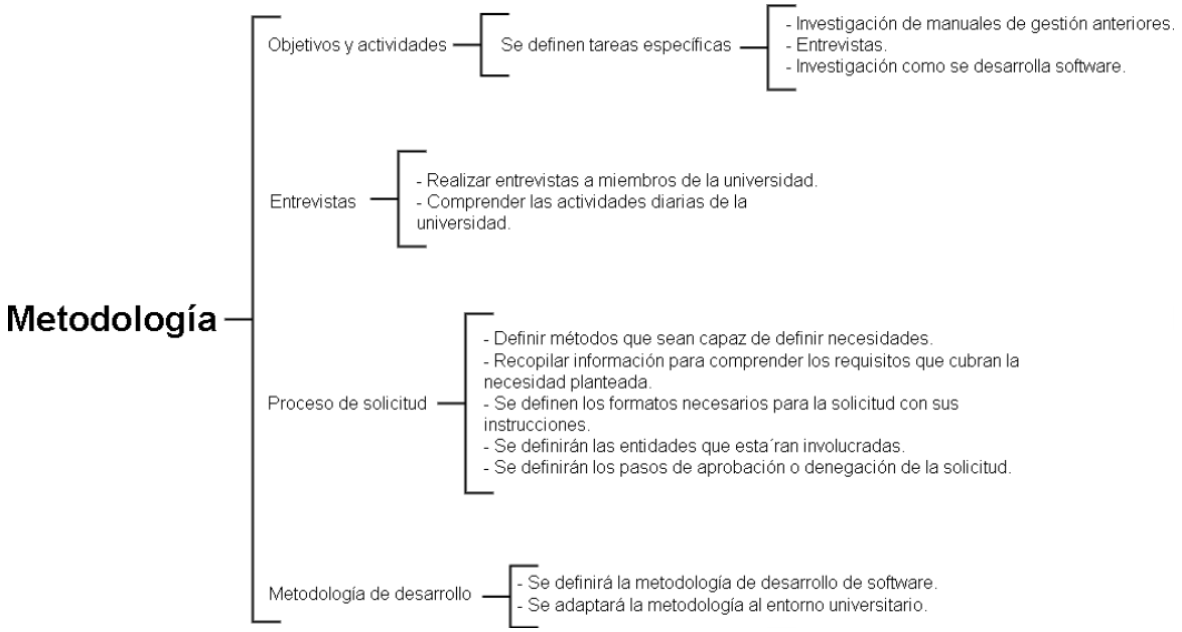


Figura 3. Diagrama de los pasos que conforman la metodología elegida. [Imagen]. Fuente: Elaboración propia.

Objetivos y actividades

El objetivo principal fue el desarrollo del "Manual para la Gestión de Proyectos de Software de la UIEPCH". Para lograr este objetivo, se propusieron tareas específicas, como investigar la disponibilidad de manuales de gestión utilizados anteriormente para comprender cómo se realizan las solicitudes de software en la organización y escribir un manual de gestión de proyectos basado en investigaciones previas.

Las actividades incluyeron realizar entrevistas, investigar la necesidad de software en la universidad e investigar los elementos necesarios para el desarrollo de software y su adaptabilidad en el entorno universitario.

Entrevistas

Para evaluar la acogida y necesidad de implementación del software en la UIEPCH, se realizaron entrevistas a personal clave de los diversos departamentos de la institución. La entrevista se centró en comprender el nivel de impacto de los softwares en las actividades diarias que enfrentan cada día la universidad, a través de las siguientes preguntas:

- ¿Área o departamento?
- ¿Titular o responsable del área?
- ¿Ha experimentado en algún momento la necesidad de utilizar algún tipo de software para realizar sus actividades?
- En caso de responder sí, ¿Qué actividades ha querido realizar con software?
- De esas necesidades, ¿Cuáles se han realizado con software?
- ¿Dicho software ha sido diseñado para ese uso particular o es un software de terceros?
- En caso de ser software de terceros, ¿Cuál es?
- ¿Qué ha pasado con las actividades que no se pudieron realizar con software?
- ¿A qué cree que se deba que no se haya podido realizar con software dichas actividades?
- ¿Los proyectos de software mejorarían la productividad de sus actividades?
- ¿Cree que es fundamental que las organizaciones adopten una metodología de gestión de proyectos de software estándar?

Proceso de solicitud

Este proceso comenzará con una exploración del proceso que dará origen al manual. Aquí se sientan las bases para iniciar una gestión efectiva de proyectos de software, asegurando que cada iniciativa esté alineada con las necesidades y objetivos de la institución.

En primer lugar, para iniciar con el proceso de solicitud, se utilizarán métodos como encuestas, entrevistas y análisis de datos para poder identificar las necesidades que se resuelvan con la ayuda de un software. La recopilación y documentación de requisitos serán el paso crucial que permita comprender de forma clara como identificar necesidades.

También se exponen los campos y secciones de los formatos necesarios para la solicitud, con instrucciones claras para su llenado que incluyen ejemplos para información concreta que sea requerida, brindando una guía completa que va desde la conceptualización de la idea hasta su presentación formal.

El papel del Comité de Tecnologías de la UIEPCH tendrá relevancia gracias a que serán los encargados de realizar la revisión y evaluación de las solicitudes de software. Este comité establecerá los criterios clave las cuales influirán en la decisión que enfatizará la importancia de alinearse con los objetivos de la institución.

A su vez se describirán los pasos que seguirá el Comité de Tecnologías en torno a la decisión de aprobar o rechazar una solicitud con la ayuda de ejemplos de criterios transparentes para evaluar la viabilidad del proyecto, los recursos necesarios y el impacto dentro de la universidad.

Por último, se presentará el proceso de comunicación de la decisión al solicitante, incluyendo los pasos a seguir en caso de aprobación o rechazo, ejemplificándolo por medio de un diagrama que dará un contexto más fácil de entender del proceso completo.

Metodología de desarrollo

Para elegir una metodología adecuada a los proyectos de software de la UIEPCH, se evaluaron diversos métodos y se eligió la agilidad como método principal, prestando especial atención a las metodologías Scrum y Lean adaptadas a la dinámicas y necesidades de la universidad.

La elección de los métodos ágiles se basó en la necesidad de flexibilidad y adaptación al entorno académico cambiante. Las metodologías Scrum y Lean, son reconocidas por la capacidad que tienen de gestionar proyectos de forma ágil y eficiente, gracias a ello es que fueron elegidas.

Estas metodologías se pueden adaptar dentro de la universidad, alineándolas con las estructuras de desarrollo de software universitario. Esto implica realizar ajustes considerando las necesidades de las direcciones, así asegurando que todas las fases, roles y recursos del desarrollo se integren de manera armoniosa en el contexto institucional.

Esta combinación permite beneficiarse de lo mejor de ambas metodologías. Por un lado, Scrum proporciona una estructura clara de roles y responsabilidades que permiten una mejor planificación, por otro lado, Lean se enfoca en eliminar desperdicios o elementos innecesarios en el proceso de desarrollo, garantizando así una eficiencia continua y una reducción significativa del tiempo.

Por ende, esta metodología compuesta buscará ser implementada en los nuevos proyectos de software permitiendo así una mejora y evaluación continua para ajustarse a las nuevas necesidades específicas que surjan en el futuro. La retroalimentación constante de las direcciones escolares además de los equipos de desarrollo será crucial para asegurar una integración exitosa y la mejora continua.

Capítulo 5. Resultados

Resultados

Proceso de solicitud y aprobación

En esta sección, se presentarán los resultados obtenidos a partir del análisis de datos recopilados durante la investigación sobre el impacto de un manual para la gestión de proyectos de software dentro de la UIEPCH.

Como principio, por medio de entrevistas a diversos miembros clave de diferentes direcciones de la universidad, se evaluó como se pide, crea y en que usos se aplican el software. Esto evidenció la necesidad de implementar un enfoque de gestión de proyectos de software que permita desarrollar software dentro de la institución. Los resultados de la entrevista son los siguientes:

La siguiente tabla presenta un cuadro categórico de las entrevistas realizadas a las distintas direcciones de la universidad. Esta abarca aspectos como el área o departamento, el titular o responsable del área, la necesidad de software, algunas de las actividades que realizan de las cuales se definen si son realizados con software, su origen, si son propios o de terceros, que razones existen para que no se realicen las actividades con la ayuda de software, sí existe la oportunidad de mejorar la productividad con el apoyo de software y la aceptación de adoptar una metodología estándar.

Este cuadro proporciona una visión general de las necesidades y algunas prácticas relacionadas con el uso de software dentro de las direcciones de la institución, destacando los casos en donde se utiliza o no software como apoyo y las razones por las cuales no se llega a utilizar software. Además, se resalta la percepción en la mejora de productividad

además de la importancia de adoptar una metodología estándar aplicada a la gestión de proyectos de software, lo que sugiere la necesidad de creación de software institucional quesea capaz de satisfacer las necesidades y desafíos de la universidad.

Tabla 2. Cuadro categórico de las entrevistas realizadas

Área o Departamento	Titular o responsable del área	Necesidad de software	Actividades requeridas	Actividades realizadas con software	Origen del software	Software de terceros	Razones para no realizar actividades con software	Mejora de productividad con proyectos de software	Importancia de adoptar metodología estándar
Dirección de Vinculación y Extensión Universitaria	Lic. Ivonne Sosa Reyes	Si	Control de Seguimiento de Egresados	Si	Externo	Office	N/A	Si	Si
Coordinación de Extensión Universitaria	C. Gerardo Jusef Herrera Ortega	Si	Gráficos automatizados	Si	Externo	Programas de diseño	Resultados no esperados	Si	Si
Coordinación de Cultura y Deporte	Lic. Noé Cruz Casiano	Si	Control de Inventario de Material, Control de Roles de Juego	No	N/A	N/A	Se realiza de forma manual y falta de conocimiento	Si	Si
Dirección de Planeación y Evaluación	Lic. César Eduardo Romero González	Si	Biblioteca Virtual, Programa Presupuestario, Sistema de Control Escolar	Si	Externo	SUNI	N/A	Si	Si
Coordinación de Evaluación	Lic. Alma Hernández Salas	Si	Seguimiento del Programa Presupuestario	Si	Externo	Sistema Integral de Administración Financiera (SIAF), Excel	N/A	Si	Si
Coordinación de Servicios Escolares	Lic. Mario Ramón Bouchan	Si	Control de Asistencia de Alumnos	Si	Externo	SUNI	No hay un buen control	Si	Si
Dirección de Control y Gestión	Lic. Dahané Liliana Carral Díaz	Si	Reinscripciones, Solicitud de Almacén, Comprobación de Gastos, Informes de Planeación	Si	Externo	Excel	Se realiza de forma lenta	Si	Si

Área o Departamento	Titular o responsable del área	Necesidad de software	Actividades requeridas	Actividades realizadas con software	Origen del software	Software de terceros	Razones para no realizar actividades con software	Mejora de productividad con proyectos de software	Importancia de adoptar metodología estándar
Departamento de Enlace Institucional y Programas Especiales	Lic. Norma Angélica García Gutiérrez	Si	Creación de Bases de Datos, Diseño de Reconocimiento, Constancias	Si	Externo	Corel y Office (Access)	Falta de un programa específico y falta de licencias	Si	Si
Dirección de Administración y Finanzas	Lic. Ezequiel López Olalde	Si	Inventario, Presupuesto, Contabilidad, Nómina	Si	Externo	Excel y privados	No hay softwares diseñados para esta área	Si	Si
Departamento de Control Presupuestal	Lic. Christian Refugio Acosta Mirón	Si	Sistema Contable, Administración de Flujo de Efectivo	Si	Externo	SIAP, SUNI	N/A	Si	Si
Departamento de Recursos Humanos y Materiales	Mtra. Catalina Ortiz Medel	Si	Inventario, Manejo de Datos, Informes, Expedientes	Si	Externo	Office, Sistema Contable	N/A	Si	Si
Coordinación de Recursos Financieros	Lic. Berta Luna Nahuacatl	No	Contabilidad	Si	Externo	Excel	N/A	Si	Si
Coordinación de Recursos Materiales	Lic. Jazmín de la Rosa Sánchez	Si	Timbrado de Nóminas	Si	Externo	Folios Digitales	N/A	Si	No
Dirección Académica	Lic. Edgar López Avendaño	Si	Estadísticas y Sistematización de Información	Si	Externo	SIAP y Office	Excedente en tiempo	Si	Si
Departamento Académico de Ingeniería Agroindustrial y Cadenas Agroalimentarias	Ing. Marcelina Ortiz Medel	Si	Diagramas, Mapeo, Horarios	Si	Externo	Draw.io y Office	Falta de capacitación para usarlos	Si	Si

Área o Departamento	Titular o responsable del área	Necesidad de software	Actividades requeridas	Actividades realizadas con software	Origen del software	Software de terceros	Razones para no realizar actividades con software	Mejora de productividad con proyectos de software	Importancia de adoptar metodología estándar
Departamento Académico en Ingeniería en Desarrollo Sustentable en Eco Biología y Veterinaria y Zootecnia	Lic. Germán Hernández Casas	Si	Planeación Académica, Impartir Clases	Si	Externo	Office	Se realizan de forma manual	Si	Si

Como complemento de la tabla anterior, se muestra de manera gráfica los resultados obtenidos en las entrevistas realizadas a cada dirección de la universidad, mostrando que la implementación de una metodología de gestión de software es aceptada en un 99%, dando como resultado la creación del Manual para la Gestión de Proyectos de Software de la UIEPCH.

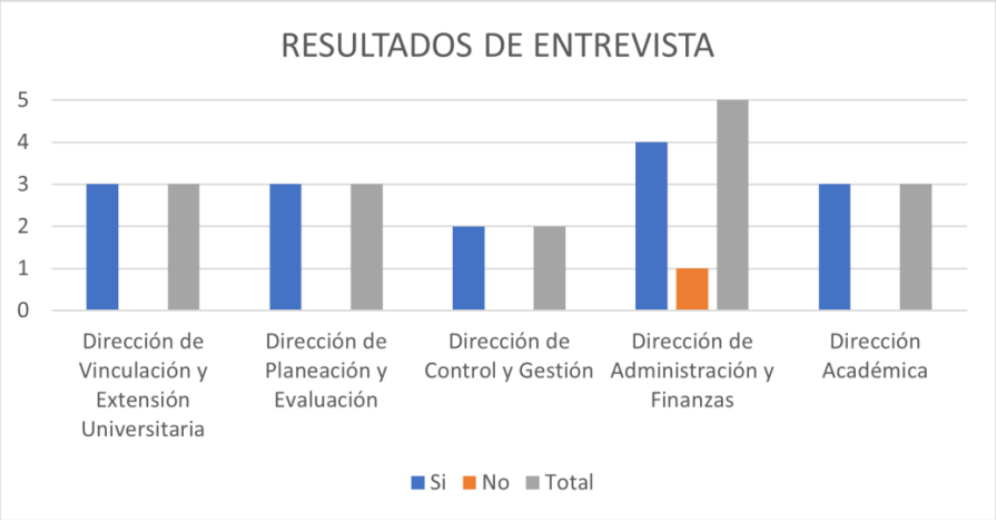


Figura 4. Gráfico de los resultados de las entrevistas realizadas. [Gráfico]. Fuente: Elaboración propia.

La tabla 3 representa las respuestas más destacadas que se obtuvieron de la tabla anterior, en relación con el uso de software en la universidad. Cada fila representa el área donde se realizó la entrevista, además de proporcionar información sobre si han experimentado la necesidad de software, que actividades realizan y cuáles de ellas realizaron con el uso de software, el origen y nombre del software ya sea desarrollado en la universidad o si es de terceros, que sucedió con las actividades que no se pudieron realizar con la ayuda de software, si los proyectos de software mejorarían la productividad y si creen fundamental adoptar como organización una metodología de gestión de proyectos de software.

Estas respuestas proporcionan una visión más detallada del como las diferentes direcciones emplean software en sus actividades diarias, así como los desafíos y necesidades que han surgido, así como las percepciones y opiniones sobre la necesidad de software universitario

Tabla 3. Respuestas más sobresalientes.

¿Área o departamento?	¿Titular o responsable del área?	¿Ha experimentado en algún momento la necesidad de utilizar algún tipo de software?	En caso de responder sí, ¿Qué actividades ha querido realizar con software?	De esas necesidades, ¿Cuáles se han realizado con software?	¿Dicho software ha sido diseñado para ese uso particular o es un software de terceros?	En caso de ser software de terceros, ¿Cuál es?	¿Qué ha pasado con las actividades que no se pudieron realizar con software?	¿A qué cree que se deba que no se haya podido realizar con software dichas actividades?	¿Los proyectos de software mejorarían la productividad de sus actividades?	¿Cree que es fundamental que las organizaciones adopten una metodología de gestión de proyectos de software estándar?
Coordinación de Extensión Universitaria	C. Gerardo Jusef Herrera Ortega	Si	Gráficos automatizados	La anterior	Terceros	N/A	N/A	Resultados no esperados	Si	Si, porque funcionarían como apertura, además de que sería algo estandarizado
Coordinación de Evaluación	Lic. Alma Hernández Salas	Si	Seguimiento del Programa Presupuestario	Programación del Programa Presupuestario	Terceros	Sistema Integral de Administración Financiera (SIAF) y Excel	Se realiza de forma manual	No se ha completado el desarrollo de software necesario	Por supuesto	Por supuesto, es indispensable
Dirección de Control y Gestión	Lic. Dahané Liliana Carral Díaz	Si	Reinscripción, Solicitud de Almacén, Control de Gastos, Informes de Planeación	Ninguna	N/A	N/A	Se realiza de forma manual	Falta de mantenimiento y/o presupuesto	Si	Si, facilitaría el trabajo y generaría una comunicación rápida
Dirección de Administración y Finanzas	Lic. Ezequiel López Olalde	Si	Inventarios, Presupuestos, Contabilidad	Las anteriores	Terceros	Excel y privados	N/A	No hay software diseñado para el área en la universidad	Si	Pudiera ser
Coordinación de Recursos Materiales	Lic. Jazmín de la Rosa Sánchez	Si	Timbrada de Nóminas	La anterior	Terceros	Folios Digitales	N/A	N/A	Si	No
Departamento Académico en Ingeniería en Desarrollo Sustentable en Eco Biología y Veterinaria y Zootecnia	Lic. Germán Hernández Casas	Si	Establecer mejores formatos para migrar información	Planeación Académica, Impartir Clases	Terceros	Office	Se realiza de forma manual	Falta de presupuesto	Si	Si

Como se puede observar, los resultados de las entrevistas afirman que el 99% de los entrevistados estuvieron de acuerdo en el nuevo enfoque, lo cual dio luz verde a la creación del "Manual para la Gestión de Proyectos de Software de la Universidad Interserrana del Estado de Puebla - Chilchotla" el cual busca convertirse en la guía definitiva que abra la puerta para el desarrollo de software universitario buscando la máxima calidad de desarrollo.

Tras la evaluación, se comenzó el desarrollo del manual contemplando los siguientes apartados:

- Proceso de solicitud
- Proceso de aprobación
- Metodología aplicable para la universidad
- Etapas de desarrollo de software

Para el proceso de solicitud, se establecieron varias etapas clave que al estar anidadas permiten crear un método más eficiente para poder solicitar y evaluar la viabilidad de un proyecto de software. Estas etapas son las siguientes:

- **Identificación de necesidades:** Es parte del proceso es donde se acentúan los diversos desafíos y dificultades que existen en la institución por las cuales se necesita de la implementación de un software capaz de resolver problemas específicos y mejorar la eficiencia de las operaciones.
- **Formato de Solicitud de Software:** Al haber realizado el paso anterior, el solicitante debe completar la solicitud de software, proporcionando información detalla acerca del software requerido, tal como el objetivo y funcionalidades que se necesiten.

- **Proceso de revisión y evaluación:** En esta fase el Comité de Tecnologías será el encargado de la revisión y evaluación de la propuesta realizada en el paso anterior. Esto implica realizar una entrevista con el solicitante además de un cuestionario interno entre los miembros del comité para determinar la viabilidad del proyecto.

Tras ser evaluada la solicitud del solicitante a través de la etapa anterior, el Comité de Tecnologías tomará la decisión de aprobar o denegar el proyecto de software. En caso de aprobación, se podrá continuar con el resto del proceso, en caso contrario se brindará la oportunidad de volver a presentar la solicitud con las respectivas correcciones señaladas. Al finalizar la evaluación, el Comité de Tecnologías será el encargado de comunicar la decisión tomada por medio del Formato de Aprobación de Desarrollo de Software.

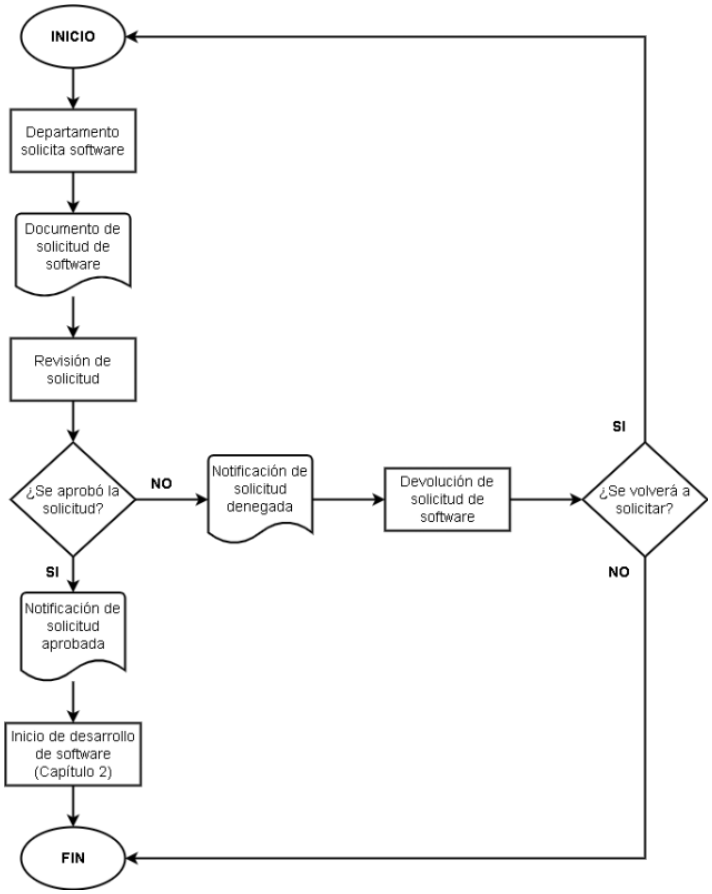


Figura 5. Diagrama del proceso de solicitud y evaluación de un proyecto de software en la universidad. [Imagen].

Fuente: Elaboración propia.

Metodología Scrum – Lean

Tras haber investigado entre diversas metodologías, tanto tradicionales y ágiles, se optaron por estas últimas, destacando las metodologías Scrum y Lean, las cuales veremos a continuación:

- "Scrum es un marco ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos" (Schwaber & Sutherland, 2020). Esto permite fomentar un entorno donde haya un trabajo ordenado, se desarrollen Sprints de valor incremental además de inspeccionar resultados para realizar los ajustes necesarios. De esta metodología se pueden rescatar

elementos como los roles Scrum, el Backlog de productos, los Sprints junto con su planificación, revisión y retrospectiva.

- ” La metodología Lean es esencialmente un proceso de transformación, metódico y ordenado, encaminado a la creación de valor añadido a través de la eliminación de desperdicio o despilfarros" (García Ortega, s/f). De esta metodología es rescatable la eliminación de desperdicios, el pull (tirar) y la perfección.
- Combinando ambas metodologías, el manual establece los siguientes apartados:
- Roles involucrados: El Comité de Tecnologías será quien tome las decisiones en el desarrollo del proyecto, el profesor designado será el Scrum Máster, los alumnos encargados de construir el software serán el equipo de desarrollo y el Product Owner o cliente será el solicitante del software.
- Fases de desarrollo: Las fases extraídas de esta combinación de metodologías son la planificación, la implementación y evaluación, y por último la retrospectiva.
- Eficiencia y adaptabilidad: Esta combinación busca maximizar la eficiencia, adaptabilidad y valor entregado en cada etapa del proyecto convirtiéndose en una poderosa herramienta de gestión de proyectos.
- Enfoque híbrido: Al combinar Scrum y Lean, se destacan los roles clave de Scrum, las fases de desarrollo de ambas y los formatos o productos aplicados a cada fase. Este enfoque busca el mejor aprovechamiento de ambas metodologías para mejorar la gestión de proyectos de software en la universidad.

A continuación, se describirán las etapas de desarrollos seleccionadas para el manual.

Etapa 1: Planificación

Esta es la etapa inicial en donde el equipo de desarrollo realizará el análisis del proyecto tomando como acción inicial una entrevista no estructurada con el solicitante que estará basada en las siguientes preguntas:

- ¿Cómo se llama el proceso donde se desea implementar un software?
- ¿El proceso se encuentra en el Sistema de Gestión de la Calidad?
- ¿Existe un Formato de Reporte obtenido de dicho proceso?
- ¿Cómo funciona el proceso específicamente?

Además de que se definirán los siguientes aspectos:

- Objetivos del Sprint: Se definirán los objetivos a cumplir en cada inicio de Sprint.
- Requerimientos: Se clasificarán los requerimientos del solicitante para darles su orden de importancia.
- Definición de tareas y actividades: Se plasmarán las actividades a realizar en cada Sprint.
- Registro de riesgos: Se llevará un control de riesgos que vayan surgiendo durante el desarrollo del proyecto para poder darles solución.
- Resultados del Sprint: Se identificará lo aprendido de cada Sprint para poder mejorar el flujo de trabajo en el futuro.

Es preciso resaltar la maquetación de software, el cual es una parte fundamental del desarrollo de software ya que es la plantilla inicial que determinará tanto el aspecto como las funcionalidades esperadas del software a través de un diseño construido por los lenguajes

HTML como estructura base y CSS que le dará el aspecto buscado. Para ello, se realizará una reunión en donde se determinen aspectos como el comportamiento del diseño (diseño responsivo), la selección de colores enfocados a la universidad, el tipo de letra a utilizar en sus diversos elementos, el diseño de las cajas contenedoras de la información requerida (cabecera, cuerpo y pie de página), las imágenes y multimedia que sea necesaria y que sea accesible para personas con alguna discapacidad (por ejemplo, visual).

Etapa 2: Implementación y evaluación

Esta etapa es donde se creará, por medio de codificación, la planificación de la etapa anterior. Para ello, se implementarán los lenguajes de programación web, en conjunto con sus librerías de apoyo, además de sus respectivos paradigmas de programación, la documentación de código y las pruebas de software.

Dentro de los lenguajes de programación, se contemplaron lenguajes que sean capaces de darle una mayor funcionalidad a los proyectos además de poder anidarlos con bases de datos:

- Javascript: Este lenguaje será empleado para webs dinámicas e interactivas por medio de eventos.
- PHP: Este lenguaje será empleado para poder comunicar elementos de las webs, como son los formularios, con las bases de datos que se necesiten.

Estos lenguajes serán apoyados por librerías externas que funcionarán como herramientas conocidas como complementos que otorgarán funcionalidades extras:

- Bootstrap: Empleado para el font-end gracias a sus herramientas como plantillas, componentes y utilidades prediseñadas que aportan funcionalidad visual.
- jQuery: Al ser escrito en JavaScript, se emplea para cosas como el recorrido y manipulación de documentos HTML, el manejo de eventos y animaciones.
- Laravel: En conjunto con el lenguaje PHP, apoyará en dar una mejor comunicación con las bases de datos y apoyará en las autenticaciones seguras usadas en los inicios de sesión.

Estos lenguajes contienen en sí mismos paradigmas de programación que son reglas que les permiten identificar para que usos pueden ser utilizados. Tanto para JavaScript como para PHP, sus paradigmas óptimos para la universidad son la Programación Orientada a Objetos (OOP).

Por otra parte, es preciso señalar que toda la codificación realizada en esta etapa debe ser documentada de manera clara ya que servirá de referencia cuando sea preciso darle continuidad o al momento de realizar algún cambio importante por parte de algún miembro del equipo de desarrollo. Para ello se requiere aclarar cómo se construyó cada elemento del código utilizado, su funcionalidad y el uso que este tiene ayudando así a que los desarrolladores y partes interesadas puedan comprender la funcionalidad interna del software. Esto se realizará en tres pasos:

- Se analizará el código fuente del software.
- Se realizarán los resúmenes que describan la funcionalidad de cada clase y método.
- Se tomarán en cuenta las dependencias del código para resumir las clases y métodos.

Por último, en esta etapa, están las pruebas de software, las cuales se realizarán al final de cada Sprint de manera obligatoria para determinar si las funciones implementadas si tiene una ejecución esperada o si existe algún error inesperado. Tras analizar distintos métodos de pruebas de software, se optó que el método de la "Caja Negra" sea utilizado por la universidad, ya que este permite comprobar la funcionalidad de los productos generados en cada Sprint desconociendo la estructura y los detalles de implementación, dicho de otro modo, se realizarán pruebas de los elementos del software de manera externa, desconociendo su código fuente y su funcionamiento interno, ya que se buscará que funcionen de la manera definida en los objetivos de cada Sprint.

Etapa 3: Retrospectiva

Esta última fase será ejecutada al momento de haber finalizado la construcción de cualquier proyecto de software. En esta fase todos los involucrados en el desarrollo analizarán que cosas funcionaron y cuales no, con la finalidad de detectar oportunidades de mejora para los desarrollos futuros, ya que se invita a la reflexión grupal donde se expresarán opiniones libres para discutir aspectos positivos y negativos del desarrollo.

Para ello, se realizará un documento que encapsule estas opiniones en conjunto a sus soluciones propuestas tomando en cuenta aspectos como dar a conocer la descripción del software realizado al igual que sus alcances, objetivos, resultados y logros alcanzados.

También se enlistarán los entregables cumplidos y los resultados de sus evaluaciones dando como resultado el análisis de la eficacia y eficiencia obtenidos en conjunto con las lecciones aprendidas y los desafíos del desarrollo del proyecto.

Conclusiones

Como se vio en apartados anteriores, se abordó la importancia de contar con un protocolo que sea claro para el proceso de solicitud y creación de software dentro de la UIEPCH, destacando la necesidad de establecer una guía que conduzca al desarrollo de sistemas eficientes y de calidad. La investigación realizada se basa en entrevistas con personal clave de las direcciones de la universidad para conocer y comprender los procesos actuales y las necesidades que dan pie a la implementación de un manual de gestión de proyectos de software.

Se identificaron problemas en la gestión de sistemas provocado por una falta de regulación en el proceso de solicitud de software, lo que ha llevado a la creación de diseños deficientes, complejos, el no usar software o incluso preferir el uso de software generado por algún tercero, lo cual afecta de alguna manera la eficiencia de las operaciones universitarias.

A su vez, se destacó la importancia de la adopción de una mezcla de metodologías ágiles con la finalidad de adaptarlas a la universidad, creando así, la metodología ágil Scrum – Lean para que contribuya a un mejor desarrollo de software que sea capaz de mejorar la eficiencia, colaboración y adaptabilidad a los cambios. Con ello, se introducen las etapas que conforman esta metodología: planificación, implementación y evaluación acompañada de retrospectiva convirtiendo estas etapas en la parte fundamental del proceso de mejora continua en el desarrollo de cualquier software universitario.

Recomendaciones

Durante el desarrollo de este documento, se identificaron varias recomendaciones a considerar, a continuación, se presentan las sugerencias más destacables:

- Proporcionar capacitación en metodologías ágiles: Se sugiere la capacitación en la metodología a implementar Scrum – Lean a los equipos de desarrollo de software. Esto con la finalidad de ayudar a la mejora de la eficiencia, colaboración y adaptabilidad a los cambios que vayan dándose durante la creación de software universitarios.
- Familiarizarse con el sistema de retroalimentación: Es importante la familiarización con un sistema de retroalimentación durante la fase de retrospectiva para que la mejora continua ayude al desarrollo de mejores proyectos de software en la institución. Esto permitirá detectar áreas de mejora, correcciones, desviaciones, entre otros elementos que aseguren la calidad de los productos entregados.
- Revisar y actualizar periódicamente el manual: Es prescindible la revisión y actualización regular del Manual de Gestión de Proyectos de Software. Esto permitirá la incorporación de nuevas prácticas, herramientas o la adopción de cambios en la metodología que mejoren la eficiencia y la calidad en el desarrollo de sistemas universitarios.

Discusión

En esta etapa se presentará una interpretación exhaustiva de los hallazgos mencionados en el estado del arte y las hipótesis planteadas sobre la creación del manual gestión de proyectos de software, además se explorarán las implicaciones y contribuciones de esta investigación en el campo de la gestión de software. A través de una discusión detallada, se examinará la coherencia de los resultados con la literatura existente, así como las posibles implicaciones para cualquier discrepancia detectada.

Al examinar las investigaciones internacionales, se encontró que existe un interés significativo en comprobar la eficiencia de la metodología PMP en la gestión de proyectos de software. Particularmente, la tesis de especialización de Gómez Barboza destaca la importancia de establecer alcances, definir objetivos y concluir satisfactoriamente los proyectos de software. Sus hallazgos indican que se pueden establecer tiempos, alcances y costos para los proyectos de software, permitiendo así un mejor control y un registro de actividades que facilitan la adaptación a las necesidades de cada proyecto. Estos aspectos son fundamentales y están alineados con la hipótesis general sobre la creación de un manual que permitirá establecer un flujo de trabajo eficiente tanto para la solicitud como para la construcción de un sistema.

Por otro lado, la documentación proporcionada por Rodríguez complementa los hallazgos internacionales al enfatizar la gestión de proyectos de software desde el punto de vista latinoamericano. Este estudio ha permitido explorar diferentes metodologías, tanto clásicas como ágiles, resultando en la adopción de metodologías híbridas. Este enfoque es relevante para la segunda hipótesis (H1), ya que detalla el proceso completo de solicitud de un sistema, incluyendo los formatos requeridos, el personal involucrado y la forma en que se

decidirá sobre la solicitud y su respectiva comunicación del veredicto, mejorando así la transparencia y eficiencia del proceso.

La implementación del modelo de desarrollo de software aplicado a la Universidad Simón Bolívar es un ejemplo práctico que demuestra cómo una gestión eficiente puede contribuir al éxito de los proyectos de software. Los hallazgos coinciden con la tercera hipótesis (H2) sobre la selección y descripción de la metodología óptima para el desarrollo del sistema, incluyendo los formatos y productos necesarios en cada fase, garantizando una gestión de proyectos coherente y eficaz.

En el contexto de las investigaciones nacionales, se encontró en el trabajo de Rosa Imelda Chi la aportación de un manual de prácticas enfocado en la gestión de proyectos de software, lo cual coincide con la creación de un plan que asegure la calidad del desarrollo de software implementando las metodologías seleccionadas para una mejor gestión de calidad. Estos hallazgos refuerzan la relevancia y necesidad de desarrollar un manual específico que abarque todas las fases y procesos críticos en la gestión de proyectos de software en el entorno universitario.

En resumen, aunque el manual aun no se ha implementado, se espera que su desarrollo y futura aplicación mejoren significativamente el flujo de trabajo en la solicitud y construcción de sistemas en la universidad. Esto permitirá una gestión más eficiente, una mejor comunicación y un incremento en la garantía de todo el proceso de desarrollo, alineándose con las hipótesis planteadas y ofreciendo una contribución valiosa al campo de la gestión de proyectos de software.

Referencias

Alegsa, L. (31 de 07 de 2023). Definición de Protocolo (en redes). Retrieved 11 de 03 de 2024, from Alegsa.com.ar: https://www.alegsa.com.ar/Dic/protocolo.php

Amazon Web Services, Inc. (01 de 02 de 2021). ¿Qué es el SDLC? - Explicación del ciclo de vida del desarrollo de software - AWS. Retrieved 05 de 02 de 2024, from Amazon Web Services, Inc.: https://aws.amazon.com/es/what-is/sdlc/

Angarita Sanguino, C., & Gallardo, O. (Julio - Diciembre de 2015). IMPLEMENTATION OF MODEL DEVELOPMENT OF WEB ORIENTED SOFTWARE TO UNIVERSIDAD SIMÓN BOLÍVAR SEDE CÚCUTA. Investigación e Innovación en Ingeniería, 2(3), 38 - 45. Retrieved 2023 de 11 de 10, from https://www.researchgate.net/publication/318875037_Implementando_un_Modelo_de_Desarrollo_de_Software_Orientado_a_la_Web_para_la_Universidad_Simon_Bolivar_-_Sede_Cucuta

Asana. (13 de 11 de 2022). Guía de 6 pasos para la recopilación de requisitos para asegurar el éxito de tu proyecto [2022] • Asana. Retrieved 05 de 02 de 2024, from Asana: https://asana.com/es/resources/requirements-gathering

Atlassian. (s.f.). La guía definitiva de gestión de proyectos | The Workstream. Retrieved 05 de 02 de 2024, from Atlassian: https://www.atlassian.com/es/work-management/project-management

Chi, R. I. (2018). GUIA TECNICA DE GESTION DE PROYECTOS DE SOFTWARE. www.academia.edu, 89. Retrieved 05 de 02 de 2024, from

https://www.academia.edu/33443141/GUIA_TECNICA_DE_GESTI%C3%93N_DE_PROYECTOS_DE_SOFTWARE

Desarrollo de Software. (18 de 03 de 2021). ¿Qué es y para qué sirve una metodología de desarrollo de software? Retrieved 11 de 03 de 2024, from Desarrollo de software: https://desarrollodesoftware.dev/metodologia/

Equipo de Enciclopedia Significados. (22 de 03 de 2016). Significado de Manual. Retrieved 05 de 02 de 2024, from Significados: https://www.significados.com/manual/

Fernandini, C. (20 de 04 de 2020). Paradigmas de programación: principios básicos de programación. Retrieved 05 de 02 de 2024, from IONOS Digital Guide: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/paradigmas-de-programacion/

Hernández, J. (25 de 02 de 2024). ¿Qué es una función? Características, Tipos y Funciones de llamada. Retrieved 11 de 03 de 2024, from https://saberpunto.com/programacion/que-es-una-funcion-caracteristicas-tipos-y-funciones-de-llamada/

IBM. (2020). ¿Qué es el desarrollo de software? | IBM. Retrieved 05 de 02 de 2024, from https://www.ibm.com/mx-es/topics/software-development

Javier Álvarez, F., Hurtado Alegría, J. A., Mondragón Arellano, M., Muñoz Arteaga, J., Velázquez Amador, C. E., & Hernández Bieliukas, Y. C. (2014). Gestión de Proyectos de Software - Proyecto LATIn (1a. ed. ed.). LAtinoamerica: Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn). Retrieved 2024 de 03 de 24, from

https://studylib.es/doc/9361905/gesti%C3%B3n-de-proyectos-de-software-autor-francisco-javier-%C3%A1...

KeepCoding Team. (08 de 02 de 2022). ¿Qué es la documentación de software? Retrieved 05 de 02 de 2024, from https://keepcoding.io/blog/que-es-la-documentacion-de-software/

manuel.cillero.es. (27 de 04 de 2020). Documentación del código. Retrieved 05 de 02 de 2024, from manuel.cillero.es: https://manuel.cillero.es/doc/apuntes-tic/documentacion-proyectos/documentacion-codigo/

Mendoza, M. L. (16 de 07 de 2020). Qué es un lenguaje de programación. Retrieved 05 de 02 de 2024, from OpenWebinars.net: https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/

Monterde, U. M. (2017). Lenguajes de Programación. Retrieved 05 de 02 de 2024, from https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

nqa. (07 de 30 de 2019). Certificación ISO 9001 - Norma de gestión de la calidad | NQA. Retrieved 05 de 02 de 2024, from nqa. ORGANISMO DE CERTIFICACION GLOBAL: https://www.nqa.com/es-mx/certification/standards/iso-9001

Olvera, D. (15 de 03 de 2021). ¿Qué es y cómo hacer un manual de procedimientos? Retrieved 05 de 02 de 2024, from Entorno Empresarial: https://coparmexjal.org.mx/entorno-empresarial/que-es-y-como-hacer-un-manual-de-procedimientos/

Orozco Romero, C., Enrique Rico, J., Zamora Valero, A., & Gómez Barbosa, J. S. (s.f.). Gestión de Proyectos de Software Basado en metodología PMP. Retrieved 2023 de 11 de 09,

from Universidad Santo Tomas - Primer Claustro Universitario de Colombia: https://repository.usta.edu.co/handle/11634/30429

Ortega, B. G. (s.f.). Introducción a la metodología Lean. Retrieved 05 de 02 de 2024, from https://riunet.upv.es/bitstream/handle/10251/165994/Garc%C3%ADa%20-%20Introducc%C3%B3n%20a%20la%20metodolog%C3%ADa%20Lean.pdf?sequence=1

Palomares, K. (s.f.). ▶ ¿Qué es un OBJETO en Programación? | Kiko Palomares. Retrieved 11 de 03 de 2024, from https://kikopalomares.com/clases/que-es-un-objeto-en-programacion

Pes, C. (s.f.). Definición de Protocolo (en informática) - Concepto y Significado. Retrieved 11 de 03 de 2024, from https://www.carlospes.com/minidiccionario/protocolo.php

Prieto, E. (16 de 11 de 2023). ¿Cuáles son las etapas del Desarrollo de Software? Retrieved 05 de 02 de 2024, from Tiffin University: https://global.tiffin.edu/noticias/cuales-son-las-etapas-del-desarrollo-de-software

Raeburn, A. (08 de 11 de 2023). Introducción al mundo de las soluciones de gestión de proyectos [2023]. Retrieved 05 de 02 de 2024, from Asana: https://asana.com/es/resources/best-project-management-software

Redacción KeepCoding. (18 de 08 de 2022). ¿Qué son los paradigmas de programación y qué tipos hay? Retrieved 05 de 02 de 2024, from https://keepcoding.io/blog/paradigmas-de-programacion/

Redacción KeepCoding. (10 de 01 de 2024). Prototipado en gestión de proyectos: La clave del éxito. Retrieved 13 de 03 de 2024, from https://keepcoding.io/blog/prototipado-en-gestion-de-proyectos/

Regulwar, G., & Gulhane, V. (25 de 02 de 2010). Software Testing Practices. International Journal of Computer Applications, 01(02), 06. https://doi.org/10.5120/68-165

Robledano, A. (18 de 06 de 2019). Qué es un algoritmo informático. Retrieved 11 de 03 de 2024, from OpenWebinars.net: https://openwebinars.net/blog/que-es-un-algoritmo-informatico/

Rootstack. (08 de 11 de 2022). Etapas en el proceso de desarrollo de software | Rootstack. Retrieved 05 de 02 de 2024, from Rootstack: https://rootstack.com/es/blog/etapas-en-el-proceso-de-desarrollo-de-software

Rubrika. (26 de 05 de 2022). Qué es el Código en programación - PoperLab, Big Data & Data Science. Retrieved 11 de 03 de 2024, from PiperLab: https://piperlab.es/glosario-de-big-data/codigo/

Santander Universidades. (30 de 09 de 2022). ¿Qué es el software? Ejemplos, definición y tipos. Retrieved 13 de 03 de 2024, from https://www.santanderopenacademy.com/es/blog/que-es-software-y-ejemplos.html

Schwaber, K., & Sutherland, J. (2020). La Guía Scrum - La Guía Definitiva de Scrum: Las Reglas del Juego. Retrieved 05 de 02 de 2024, from https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

sistemas.com. (s.f.). Código. Retrieved 11 de 03 de 2024, from https://sistemas.com/codigo.php

Sotomayor, S. G. (14 de 11 de 2023). Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa. Thinking for Innovation. Retrieved 05 de 02 de 2024, from https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

Sulbarán, I. (28 de 08 de 2023). ¿Qué es la programación, sus lenguajes y tipos de enfoque? Retrieved 13 de 03 de 2024, from Tiffin University: https://global.tiffin.edu/noticias/que-es-programacion

Tirosh, O. (14 de 07 de 2021). Maquetación: Todo lo que debes saber sobre esta disciplina milenaria. Retrieved 05 de 02 de 2024, from Tomedes: https://es.tomedes.com/blog-de-traduccion/maquetacion

Upadrista, V. (2015). Agile Methodology. (págs. 99 - 106). Productivity Press. https://doi.org/10.1201/b18065-15

Verou, L. (10 de 07 de 2018). ▷ ¿Qué es una FUNCIÓN en programación? 【definición y ejemplos】. Retrieved 11 de 03 de 2024, from Lenguajes de programación: https://lenguajesdeprogramacion.net/diccionario/que-es-una-funcion-o-metodo-en-programacion/

Verou, L. (31 de 01 de 2023). ¿Qué es una clase en programación 【orientada a objetos】? Retrieved 11 de 03 de 2024, from Lenguajes de programación: https://lenguajesdeprogramacion.net/diccionario/que-es-una-clase-en-programacion-orientada-a-objetos/

Westreicher, G. (07 de 08 de 2020). ¿Qué es la gestión? Para qué sirve, pasos a seguir y tipos. Retrieved 11 de 03 de 2024, from Economipedia: https://economipedia.com/definiciones/gestion.html

Zona Green. (31 de 10 de 2023). ¿Qué es un estándar en informática? Aprende sobre su importancia - Zona Green. Retrieved 11 de 03 de 2024, from https://zonagreen.com.mx/que-es-un-estandar-en-informatica/

Anexos

Manual para la Gestión de **Proyectos de Software de la UIEPCH**

De la carrera de Sistemas Computacionales
José Roberto Romero Ramírez

26 de septiembre de 2023

Índice

Tabla de ilustraciones

Anexos

Introducción

Bienvenido(a) al Manual para la Gestión de Proyectos de Software de la UIEPCH, que tiene como propósito ilustrar sobre los diversos procesos que conllevan a la obtención de software destinado para cualquier miembro o dirección perteneciente a nuestra institución.

Como institución, las direcciones que constituyen a la universidad desempeñan un papel fundamental al brindar apoyo tanto administrativo como los recursos necesarios para así garantizar el funcionamiento eficiente de la misma. Sin embargo, conforme evolucionan las necesidades académicas, se presentan nuevos desafíos abriendo así, la importancia de identificar y abordar de manera proactiva dichas necesidades.

Este manual tiene por objetivo presentar la solución para identificar, analizar y satisfacer las necesidades de nuestras direcciones escolares. Se centrarán en los procesos de solicitud y elaboración de software informático.

A través del contenido de este manual, se busca mejorar la calidad y eficacia de las diversas operaciones de las direcciones, además de impulsar la construcción de sistemas dentro de nuestra universidad.

En el primer capítulo, se abordarán aspectos necesarios para solicitar software, incluyendo los formatos necesarios para realizar esta tarea, el proceso sobre la decisión de aprobar o rechazar la solicitud y, por último, como se comunicará al solicitante sobre la decisión tomada.

En el segundo capítulo, se explicará la metodología elegida, además de sus diferentes fases, formatos obtenidos y aspectos a considerar para el desarrollo de software.

Capítulo 1. Primeros pasos

Durante el desarrollo de este primer capítulo, se expondrán los pasos necesarios para realizar la solicitud de un software empleando las siguientes preguntas como base:

- ¿Existe una necesidad identificable?
- ¿Qué documentación será necesaria durante este proceso?
- ¿A quién será dirigida la solicitud?
- ¿Cómo será el proceso de decisión de la solicitud?
- ¿Cómo se notificará si fue aprobada o denegada la solicitud?
- En caso de ser aprobada, ¿cuáles serán los siguientes procesos que se habrán de realizar a después?
- En caso contrario, ¿qué es lo que se podría hacer?

Para responder lo anterior planteado, es preciso conocer los dos procesos importantes que darán comienzo a este manual: el proceso de solicitud y el proceso de decisión, los cuales serán abordados a continuación.

Proceso de solicitud

Para dar comienzo a este proceso, es preciso aprender a identificar si existe una necesidad que deba ser resuelta con el apoyo de un software, por lo tanto, es necesario conocer la definición de "necesidad". Según la Kemerman English Multilingual Dictionary, se define como necesidad a un "hecho o circunstancia en que alguien o algo es necesario". Partiendo de esto, podemos afirmar que las direcciones que conforman nuestra institución se enfrentan a diversos retos día a día, los cuales inevitablemente resaltan las diversas necesidades que exigen de apoyo el cual sea capaz de dar soluciones efectivas.

Para poder identificar una necesidad, será preciso como solicitante realizar una serie de preguntas para poder entender si es precisa la implementación de un software como apoyo:

- ¿Cuáles son los desafíos o dificultades que existen en la dirección o área?
- ¿Cuáles son los procesos que realiza la dirección o área?
- ¿Qué actividades se consideran podrían ser más eficientes, automatizadas o simplificadas con el apoyo de un software?
- ¿Cuáles son las características y funcionalidades clave que debería tener un software para resolver los problemas específicos identificados?

- ¿La dirección o área ha consultado con otras áreas para evaluar la viabilidad técnica y obtener un asesoramiento sobre el desarrollo o implementación de un software personalizado?

Una vez aplicado este cuestionario y haber identificado la existencia de una o varias necesidades las cuales sean elegidas para implementar un software, se puede pasar a la siguiente parte del procedimiento: la solicitud del sistema.

Para este proceso, se precisará del llenado de la documentación necesaria, comenzando con el Formato de Solicitud de Software (Anexo A). Este formato tiene como función que el solicitante, a través de sus respectivos apartados, pueda solicitar un software tomando en cuenta lo siguiente:

- Se deberán rellenar los campos del formato de manera digital conforme a la guía que lo acompaña.
- Al acabar el proceso anterior, el formato se entregará al jefe(a) del área/dirección al que pertenezca quien realice la solicitud.
- El jefe(a) será el encargado de entregar el formato al Comité de Tecnologías, específicamente al secretario técnico, quien será único responsable de recibir el formato.
- El formato será revisado por el Comité de Tecnologías, el cual a través de un proceso de decisión (que se explicará más adelante) determinará si la solicitud será o no aceptada.
- Una vez concluido el punto anterior, el Comité de Tecnologías dará a conocer al solicitante el resultado de la decisión (este punto también será explicado más adelante).

Es importante aclarar que todos los formatos mencionados en este manual deberán ser rellenados de manera digital.

Proceso de aprobación

Al haber entregado el Formato de Solicitud de Software, se dará inicio a este nuevo proceso que consiste en la aprobación de la solicitud. Para ello, el Comité de Tecnologías se encargará de la evaluación de la solicitud a través de dos filtros importantes: la realización de una entrevista al solicitante y un cuestionario interno entre los miembros del comité, con el fin de llegar a un veredicto.

Primer filtro – La entrevista

Este filtro se realizará tras la entrega del Formato de Solicitud de Software. Esto será realizado por un miembro elegido por el Comité de Tecnologías, el cual se reunirá con el solicitante con la finalidad de obtener más información sobre el software requerido, permitiendo dar un mejor entendimiento de los requisitos del solicitante y ofrecer así un mejor panorama al comité.

En la entrevista se contemplarán los siguientes puntos:

- Funciones necesarias en el software solicitado: Se identificarán las funciones consideradas como necesarias para que el software cumpla su función, por ejemplo, "necesito un software que me ayude a generar, editar y ordenas muchas notas".
- Alternativas aplicadas (en caso de existir): Se explicarán si se han implementado software de terceros o si se ha recurrido a solventar la necesidad por medio de procesos manuales, es decir, por medio de medios no electrónicos como, por ejemplo, el uso de libretas u hojas donde se lleven anotaciones.
- Funciones del área en donde se encuentra el solicitante. Se buscará clasificar la función o funciones que realiza el solicitante en donde sea necesario el uso de un software.
- Requerimientos específicos (funcionales y no funcionales): Se clasificarán los requerimientos que el solicitante necesita en el software. Esto con la finalidad de determinar cuales los requerimientos funcionales o necesarios y cuáles serán los requerimientos no funcionales que no afecten al funcionamiento del software en caso de no ser implementados.
- Usos y límites del sistema: Se definirán aspectos como cuantas personas usarán el software, que medidas de seguridad se busca tenga el sistema, si deberá tener integración con otros dispositivos, entre otros aspectos.
- Visualización del uso futuro del sistema: Se explorarán aspectos como la vida futura del software, si será necesario la implementación de nuevos requerimientos para adaptarse a nuevas necesidades, entre otros.
- Comentarios complementarios del solicitante: Serán tomados en cuenta los diversos comentarios que puedan proporcionar información extra que no haya sido contemplada en los puntos anteriores.

Es preciso aclarar que esta entrevista estará incluida en el Formato de Entrevista de Conocimiento del Sistema (Anexo B).

Segundo filtro: Cuestionario interno

Si la solicitud ha superado el primer filtro exitosamente, se someterá a una nueva evaluación, que consta de un cuestionario realizado en el Comité de Tecnologías. El objetivo es verificar y comprobar la viabilidad de la solicitud.

Este nuevo cuestionario estará basado en los siguientes aspectos:

- Necesidad identificable: Si se ha detectado una necesidad clara.
- Cumplimiento de requisitos: Si es posible cumplir los requisitos, tanto los funcionales como los no funcionales.
- Disponibilidad de recursos: Si se cuenta con el equipo y la experiencia para su desarrollo.
- Implementación: Cuestionar si funcionará el sistema una vez implementado.
- Continuidad: Verificar si es necesario darle continuidad al sistema.

El cuestionario formará parte del Formato de Evaluación de Viabilidad de Desarrollo de Software (Anexo C), el cual será determinante para el avance a la última fase del proceso.

Respuesta a la solicitud

Tras concluir los filtros mencionados anteriormente, además de haberse tomado una decisión por parte del Comité de Tecnologías, se dará paso al lapso final de este primer proceso. Esta parte consiste en la notificación al solicitante del veredicto tomado por el comité, para ello, se hará uso del Formato de Aprobación de Desarrollo de Software (Anexo D).

Para poder realizar la notificación mencionada, será utilizada la información de contacto proporcionada en el Formato de Solicitud de Software. Esta tarea será realizada por un responsable elegido por el Comité de Tecnologías, el cual mandará un correo electrónico al solicitante dando aviso sobre el veredicto.

En caso de haber sido rechazada la solicitud, se le dará oportunidad al solicitante de realizar nuevamente su solicitud, esto con la finalidad de corregir aspectos que hayan sido señalados por el Comité de Tecnologías y reconstruir la solicitud conforme a lo planteado en temas anteriores.

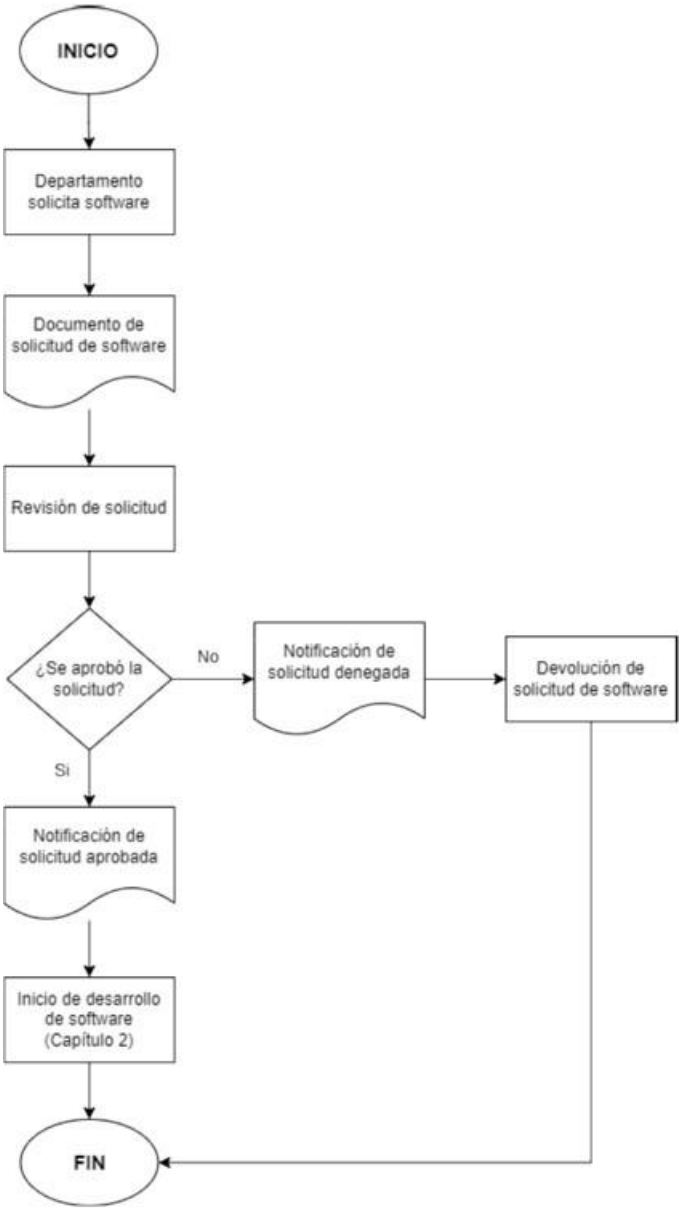


Ilustración 1. Proceso de solicitud de software dentro de la UIEPCH.

Scrum-Lean, la metodología institucional

Es necesario explicar esta parte, ya que será la base de la construcción del cualquier software dentro de la institución, convirtiéndose así, en la puerta de entrada al segundo capítulo. A partir de aquí, se explorará esta metodología que será implementada en nuestra institución y que será la combinación dos metodologías ágiles: Scrum y Lean.

Metodología Scrum

Scrum es una metodología ágil que sirve para gestionar y completar proyectos complejos. Es un marco capaz de ayudar a los equipos a trabajar de manera conjunta para lograr sus objetivos. Esta metodología se basa en los principios de la transparencia, la inspección, la adaptación y la entrega iterativa de productos, ya que su diseño es flexible y puede adaptarse a diversos tipos de proyectos y equipos.

Este proceso funciona dividiendo el proyecto en ciclos cortos y manejables denominados como "Sprints", los cuales tienen en promedio una duración de 1 a 4 semanas. En cada sprint, el equipo trabaja en una serie de objetivos específicos donde se crea un incremento de producto potencialmente lanzable, es decir, se espera entregar una versión funcional del producto. Este proceso a su vez está dividido en roles, artefactos y eventos:

Roles de Scrum:

- El propietario del producto quien es el responsable de la cartera de productos, que es una lista priorizada de funciones y requisitos del producto.
- El maestro de Scrum o Scrum Máster quien es responsable de asegurar que se siga la metodología Scrum y de ayudar al equipo de desarrollo para que se haga un trabajo eficaz.
- El equipo de desarrollo quien se responsabiliza de crear el producto.

Artefactos de Scrum:

- Cartera de productos: Lista donde se priorizan las características y requisitos del producto.
- Cartera de pedidos del Sprint: Lista con los elementos de la cartera de productos que el equipo de desarrollo completará en cada Sprint.
- Incremento: Suma de los elementos de la cartera de productos ya completados durante el sprint y el valor de cada incremento de los Sprints anteriores.

Eventos de Scrum:

- Planificación del sprint: Reunión de horario fijo que se realiza al comienzo de cada sprint para planificar el trabajo a realizar.
- Scrum diario: Reunión de 15 minutos o menos para planificar el trabajo de las próximas 24 horas.
- Revisión de sprint: Reunión calendarizada celebrada al final de cada sprint para inspeccionar el incremento además de adaptar la cartera de productos en caso de ser necesario.

- Retrospectiva del sprint: Reunión celebrada al final de cada sprint para inspeccionar los procesos del equipo e identificar oportunidades de mejora.

Metodología Lean

Es una metodología ágil denominada como "producción justo a tiempo", el cual se enfoca en la disminución de "desperdicios", en la mejora continua de los procesos y en la maximización del valor, además de la importancia de la mejora continua, el respeto por las personas y la eliminación de actividades que no aporten ningún valor. Esta fue definida por Bob Charette's a raíz de su experiencia en proyectos con la industria japonesa en los años 80.

Esta metodología tiene los siguientes principios:

- Eliminación o reducción de todos los residuos.
- Entender los residuos como cualquier actividad que no llega a aportar al producto o servicio final.
- Uso del ciclo DMAIC (definir, medir, analizar, mejorar y controlar) para identificar residuos y realizar mejoras.
- Énfasis en el trabajo en equipo además de la definición clara de las funciones y las responsabilidades de los miembros.

Algunas herramientas de la metodología Lean:

- Mapeo de flujo de valor: Es una herramienta visual empleada para analizar y mejorar el flujo de materiales e información necesarios para llevar un producto o servicio a un cliente.
- Kanban: Es un sistema utilizado para la gestión y control del flujo de materiales e información en un proceso de producción.
- 5S: Es un método de organización del lugar de trabajo usado en la mejora de eficacia, seguridad y calidad.
- Poka-yoke: Es una técnica de detección de errores que se emplea para evitar riesgos y defectos.
- Kaizen: Es un enfoque de mejora continua el cual implica la realización de cambios pequeños a incrementables en un proceso de mejora y eficacia a lo lardo del tiempo.

Combinando Scrum + Lean – Metodología Scrum-Lean

Una vez abordados las metodologías de manera individual, es momento de adentrarse en un enfoque que combina lo mejor de ambas. En este apartado se explorará en la metodología híbrida Scrum-Lean, donde se consideran los roles clave de Scrum, las fases de desarrollo extraídas de ambas metodologías y los formatos o productos que se obtendrán en cada fase del proceso. Esta combinación busca maximizar la eficiencia, la adaptabilidad y el valor entregado en cada etapa del proyecto, brindando así una poderosa herramienta para la gestión ágil de proyectos para nuestra universidad.

Para comenzar, es necesario definir cuáles serán los roles involucrados en el desarrollo de proyectos realizados en la universidad:

- Comité de Tecnologías: Será el encargado de tomar la decisión de autorizar o denegar el desarrollo del proyecto, esto con base a lo visto en el capítulo 1.
- Scrum Máster: Será quien tome la responsabilidad de dirigir y garantizar que el equipo de desarrollo concluya el desarrollo del software. Este rol será desempeñado por un(a) profesor(a) el cual tendrá a su cargo uno o varios alumnos de Servicio Social o Prácticas Profesionales de la carrera de Ingeniería en Sistemas Computacionales.
- Equipo de desarrollo: Serán los alumnos que estén en situación de Servicio Social o Prácticas Profesionales pertenecientes a la carrera de Ingeniería en Sistemas Computacionales.
- Product Owner o Cliente: Serán cualquiera de las direcciones integrantes de la Universidad Interserrana del Estado de Puebla – Chilchotla que necesiten de un software y lo hayan solicitado.

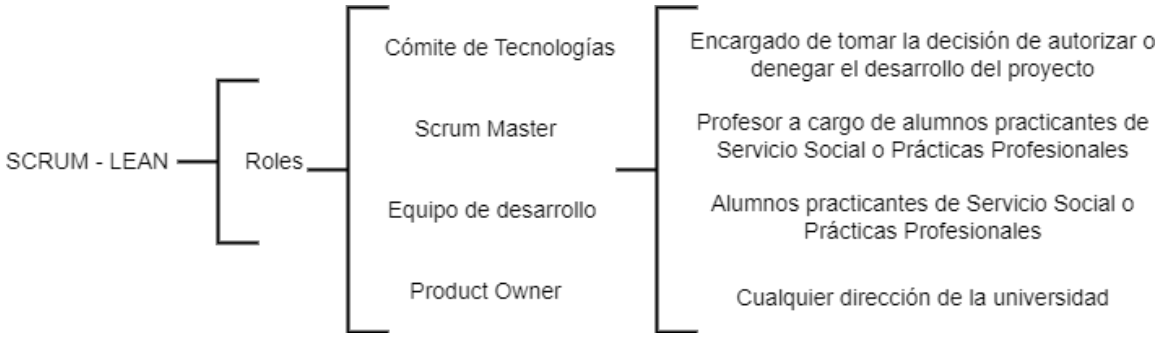


Ilustración 2. Roles Scrum-Lean dentro de la UIEPCH.

Ya identificados los roles, se mostrarán las fases del desarrollo de los sistemas creados dentro de la universidad, las cuales se profundizarán más adelante:

- Fase de Planificación
- Fase de Implementación y Evaluación
- Fase de Retrospectiva

Capítulo 2. Etapas de desarrollo de software

En apartados anteriores se ha explorado como realizar la solicitud de software y el proceso de aprobación de este. A partir de este punto, se abordará el proceso para el desarrollo de software elegido para nuestra universidad, explorando las etapas de desarrollo, la documentación generada, los productos obtenidos y las pruebas de software.

Fase 1: Planificación

Esta fase es fundamental gracias a que da paso a la generación del software. Aquí es donde se establecen las bases y la dirección del proyecto. Durante la realización de esta fase, el equipo designado para el desarrollo en conjunto con el Scrum Máster serán los encargados de la definición de los objetivos, la identificación de las necesidades del solicitante además de establecer la cantidad de sprints necesarios y un plan de trabajo.

El primer paso será tener una entrevista con el solicitante, esto será ejecutado por un integrante del equipo y buscará entender mejor las características que deban ser implementadas en el software. Esta entrevista será no estructurada, es decir, las preguntas que sean mencionadas durante este proceso serán elegidas por el entrevistador mientras vaya conociendo las necesidades del solicitante, sin embargo, se plantearán a continuación algunas preguntas bases que el entrevistador considerará:

- ¿Cómo se llama el proceso donde se desea implementar un software?
- ¿El proceso se encuentra en el Sistema de Gestión de la Calidad?
- ¿Existe un Formato de Reporte obtenido de dicho proceso?
- ¿Cómo funciona el proceso específicamente?

Con la información recabada en la entrevista, se dará inicio al proceso de llenado del Formato de Product Backlog (Anexo E), el cual estará conformado por los siguientes apartados importantes:

- Objetivos del sprint: Se expresarán los objetivos a cumplir en cada inicio de los Sprints, esto con el objetivo de tener una mejor claridad en las tareas y actividades que se deberán cumplir en cada fase.
- Requerimientos: Se clasificarán los requerimientos del solicitante en dos grupos: requisitos funcionales y no funcionales. Los requisitos funcionales son aspectos básicos que el software debe ofrecer, mientras que los requisitos no funcionales son aquellos que no son requeridos por el usuario o no son importantes para el desarrollo, por ejemplo, elementos extras en la estética del software.
- Definición de tareas y actividades: Se plasmarán las diversas actividades que pertenezcan a su respectivo sprint donde deben llevarse a cabo. Estas actividades deben ser acompañadas por la descripción de la tarea a realizar, el nombre del o los responsables de la ejecución de la tarea, el estado de desarrollo, la estimación de tiempo de realización y su nivel de importancia.
- Registro de riesgos: Se llevará a cabo un control de los riesgos que hayan surgido durante el proceso de desarrollo de cada sprint, esto con la finalidad de identificar los fallos y darles solución en Sprints posteriores.
- Resultados del sprint: Se identificará lo aprendido en cada sprint para poder entender de mejor manera si el flujo de trabajo elegido es el correcto además de resaltar los resultados positivos y el aprendizaje que estos conllevan.

Maquetación de software

Esta actividad se refiere a la generación de un diseño gráfico que, en términos de programación, sea capaz de ser entendido por el navegador o algún dispositivo específico. Durante este proceso se consideran aspectos como la planificación, conceptualización y organización del contenido del software considerando a su vez como se establecen los textos, imágenes, vídeos y todo lo que sea requerido para ofrecer una buena experiencia al solicitante.

Es necesario aclarar que la maquetación de software deberá ser la actividad posterior al proceso de entrevista con el cliente.

Para realizar la maquetación del sistema en los proyectos desarrollados en la universidad, se realizarán los siguientes pasos:

- Planeación e investigación: Se realizará utilizando la entrevista con el solicitante, explicada en apartados anteriores.
- Wireframing: Se construirá un boceto básico del sistema utilizando formas y líneas simples que representarán los diversos elementos necesarios. Para ello se pueden utilizar programas como Draw.io.
- Diseño: Se creará el diseño visual del software, incluyendo el diseño que será

otorgado por el Comité de Tecnologías del cual se hablará más adelante.

Al haber ejecutado los pasos anteriores y obtenido la maqueta del software, se realizará la especificación de la interfaz de usuario, donde se detallarán los comportamientos de cada uno de los elementos que conforman la interfaz del software que se desarrollará. Estos elementos serán identificables en la maqueta de software.

Para ello, se enumerarán los elementos funcionales, es decir, los elementos que interactuarán con el usuario para poder ser identificados de manera fácil en la presentación de la interfaz permitiendo así especificar sus características y funcionalidades.

Para realizar este proceso, los elementos funcionales serán relacionados a un número identificador por cada interfaz del software con el fin de evitar la confusión entre funcionalidades de los elementos, esto permitirá establecer una referencia única para cada elemento en su propia interfaz.

Ya enumerados los elementos, se creará una lista o tabla que contendrá lo siguiente:

- Nombre del elemento
- Interfaz donde se encuentra
- Propósito
- Acciones que se espera realice
- Descripción del elemento
- Descripción de su funcionalidad

Por ejemplo, si en la interfaz el primer elemento recibirá como dato números, se le debe asignar el número identificador "1" y en la lista o tabla de especificaciones se indicará su función, la cual podría ser la recopilación del número telefónico de un usuario y que, al hacer clic en el botón de registro, se realizará el guardado de dicha información.

Este método de identificación permitirá tener una visión más clara y ordenada de cada uno de los elementos que contendrá cada interfaz del software, previniendo confusiones y ayudando a la comprensión sencilla tanto para el solicitante como para el equipo de desarrollo.

Retomando un poco el tema del diseño, es preciso conocer que lenguajes serán utilizados durante su construcción:

- HTML: Considerado como un lenguaje de marcado estándar, será utilizado para crear la estructura base del software por medio de etiquetas y atributos necesarios.
- CSS: Este lenguaje será utilizado para dar presentación y formato a los elementos

HTML por medio de una descripción de semántica.

Como ya se mencionó anteriormente, el Comité de Tecnologías será el responsable de la creación y aprobación de la estructura general del diseño, específicamente donde se utilizará el lenguaje CSS. Esto quiere decir que el diseño generado con código CSS aprobado por el comité, será utilizado en todos los proyectos de desarrollo de software dentro de la universidad, para así generar un sello distintivo capaz de identificar los proyectos desarrollados por la UIEPCH.

Para ello, el Comité de Tecnologías llevará a cabo una reunión donde se determinarán los siguientes aspectos de diseño:

- Diseño responsivo: Se comprobará si el software se ve bien y es funcional en cualquier dispositivo donde será ejecutado.
- Selección de colores: Se elegirá una paleta de colores específica que tenga relación con los colores institucionales.
- Tipografía: Se seleccionarán fuentes que sean legibles y adecuadas, considerando aspectos como la jerarquía de las fuentes (títulos, subtítulos y texto principal) y factores que pudieran frustrar el entendimiento claro (problemas de visión, tamaño de letra, entre otros).
- Diseño de cajas: Se tomarán en cuenta aspectos como el tamaño, margen y bordes de los elementos que tendrá cada software.
- Imágenes y multimedia: Se determinará la localización, tamaño, peso y cantidad de imágenes que sean necesarias para no saturar la interfaz del software.
- Accesibilidad: Se asegurará que el diseño sea lo más accesible posible para personas que sufran de alguna discapacidad visual o de otra índole.

Para enfatizar más en estos puntos, se dará un mejor contexto para así enfatizar que elementos se tomarán en cuenta para el diseño de los proyectos de la institución:

- Diseño responsivo: Dentro del diseño, se considerarán aspectos como
    - Tamaño de visualización: El espacio que será visualizado contenido en el dispositivo.
    - Tamaño de navegación: El contenido navegable para el usuario dependiendo del dispositivo.
    - Contenidos textuales: Las letras, números, títulos y párrafos tanto en tipografía como en tamaño.
    - Contenidos visuales: Las distintas imágenes, infografías, vídeos, gráficos o cualquier.
    - Identificadores de marca: El logo y nombre de nuestra institución
    - Botones de acción: Elementos como menús, botones, chats visibles o cualquiera que sea necesario y fácil de encontrar en la interfaz.

- Selección de colores: Se tomarán en cuenta los colores representativos de la universidad distribuidos en los diversos elementos que conformen el sistema.
    - Tipografía: Se contemplarán aspectos como tamaño de letra: Esto será en base a una jerarquía de información que permita diferenciar entre títulos, subtítulos, texto en general y textocomplementario o de menús.
    - Ancho de línea: Considerando alrededor de 70 caracteres de anchura de un texto que es lo más popular.
    - Contraste entre el color de letra y fondo: Fijar una selección de colores que permita al usuario visualizar mejor el contenido, por ejemplo, letras negras sobre un fondo blanco.
    - Número de fuentes de letra: Se optará por no más de una o dos tipos de letra.
    - Capitales: Se corroborará el no usar mayúsculas en los títulos o en otros apartados donde no sean necesarias.
    - Fuentes estándar de letra: Se elegirán tipos de letra como Arial, Verdana, Georgia, Helvética y Sans-Serif.
- Diseño de cajas: Se elegirán que "modelos de caja" se utilizarán en los softwares que se desarrollen, esto teniendo en cuenta el contenido que contengan y para que sean construidos.
- Imágenes y multimedia: Se tomarán en cuenta aspectos como
    - Tonos de color: Se determinarán si es válido el uso de tonos cálidos o fríos.
    - Uso de imágenes nítidas: Se hará énfasis en evitar las imágenes borrosas o poco nítidas.
    - Resolución: Se elegirán resoluciones multimedia adaptativa o con tamaños definidos.
    - Formato: Se tomarán en cuenta formatos de imagen del tipo JPG, PNG y GIF.
    - Peso: Se hará énfasis en el peso de las imágenes, optando por un menor peso para así ofrecer una velocidad de navegación óptima.
- Accesibilidad: Se contemplarán elementos como
    - Uso de colores adecuados: Esto se cumplirá al proponer el contraste adecuado para los proyectos.
    - Texto sea legible: Se analizarán las diversas fuentes y tipografía para ofrecer un contenido fácil de leer.
    - Ofrecimiento a alternativas de medios visuales: Se contemplarán a personas con discapacidades auditivas o visuales, además de la descripción de imágenes y subtítulos en vídeos.
    - Navegación fácil: Se determinará en el diseño responsivo.
    - Velocidad de carga: Se determinará en las imágenes y multimedia.

Fase 2: Implementación y evaluación

Esta fase se considera crucial dentro del proceso de desarrollo de software ya que es donde se llevará a cabo la implementación práctica de los requisitos y funcionalidades definidas en la primera fase. Además, se realizará una evaluación continua del progreso del proyecto que permitirá tomar medidas correctivas de ser necesario.

En el desarrollo de esta fase, se pondrá especial atención a la construcción funcional del software, esto con ayuda de los resultados obtenidos en la fase anterior. Durante los sprint, los encargados del desarrollo se dedicarán a implementar las funcionalidades planificadas.

A su vez, se deben emplear los lenguajes necesarios, así como sus respectivos paradigmas de programación que se abordará en los siguientes temas.

Lenguajes de programación web

Los lenguajes de programación web se consideran como un conjunto de elementos, códigos y reglas capaces de crear instrucciones que permiten comunicar a los sistemas informáticos las acciones que realizará un sitio o aplicación web. Esto los hace fundamentales para el desarrollo web dando pie a que hayan proliferado en los últimos años, lo que ha permitido fomentar su accesibilidad, potenciar su uso y optimizar los procesos.

A continuación, se mencionarán los lenguajes a utilizar para el desarrollo de proyectos dentro de la institución que tendrán como función darle funcionalidad al software:

- JavaScript: Será utilizado en la creación de páginas web dinámicas e interactivas, es decir, será empleado para la creación de eventos, por ejemplo, el comportamiento de algunos elementos de la interfaz.
- PHP: Lenguaje de scripting que será usado en la comunicación entre el software y la base de datos que sea necesaria implementar.

Estos lenguajes serán apoyados por herramientas, mejor conocidas como complementos, los cuales otorgarán funcionalidades extras que no puedan ser conseguidas con los lenguajes anteriormente mencionados:

- Bootstrap: Comúnmente empleado en el front-end, para los proyectos de software de la universidad será utilizado cuando sean requeridas plantillas, componentes y utilidades extras ya prediseñadas para el funcionamiento visual.

- jQuery: Siendo una biblioteca de JavaScript, será empleada para aspectos como el recorrido y manipulación de documentos HTML, manejo de eventos, animaciones y el uso de Ajax como apoyo a los complementos del software.
- Laravel: Este framework será utilizado en conjunto con PHP para una mejor comunicación con las bases de datos y el apoyo de autenticaciones seguras cuando se requieran inicio de sesiones.

Para trabajar con estos lenguajes, es preciso conocer los paradigmas de programación que serán base en cada lenguaje. Estos serán considerados como un conjunto de reglas y pautas en la construcción de código.

Paradigmas de la programación

Al hablar de un paradigma de programación se habla de un conjunto de reglas que ayudan a determinar cómo debe usarse un lenguaje de programación para la resolución de problemas. Sin embargo, estos lenguajes tienen su propia sintaxis y semántica que apoyan a definir un paradigma óptimo.

Considerando que existen diversos paradigmas de programación, es importante señalar que paradigmas serán los elegidos para cada uno de los lenguajes utilizados en el desarrollo de software de la universidad:

- JavaScript: Aunque es considerado como un lenguaje multiparadigma, para la institución, se utilizará la Programación Orientada a Objetos (OOP).
- PHP: Aunque es considerado como un generador de marcas HTML, es también un lenguaje multiparadigma, por lo tanto, para el desarrollo de software de nuestra institución, se utilizará la Programación Orientada a Objetos (OOP).

Documentación de código

Una vez realizadas las tareas y actividades correspondientes a cada sprint, es necesaria la documentación del código generado, esto para tener una clara referencia al momento de darle continuidad o al producirse un cambio dentro del equipo desarrollo.

Se entiende como documentación de código o documentación de software a la descripción escrita de un software, esto incluye el diseño, funcionalidad y uso de este. Esto convierte a este proceso en parte esencial del desarrollo de los proyectos en nuestra institución, ya que ayudará a los desarrolladores, usuarios y otras partes interesadas en la comprensión del sistema del software.

Aplicado para el desarrollo de los proyectos de software dentro de la universidad, se empleará el enfoque de resumen de código fuente conocido como "Suncode". Este método producirá breves descripciones para cada clase y método implementado en el software. Con esto se buscará una mejor documentación del software al proporcionar mejor información acerca de las dependencias del código.

Este enfoque funcionará de la siguiente manera:

- Se analizará el código fuente del software.
- Se realizarán los resúmenes que describan la funcionalidad de cada clase y método.
- Se tomarán en cuenta las dependencias del código para resumir las clases y métodos.

Pruebas de software

Al final de cada sprint, es estrictamente necesario realizar pruebas de software para poder determinar si las funciones implementadas realizan su respectiva ejecución esperada o si existe algún error inesperado.

Las pruebas de software son procesos que garantizan la precisión, integridad y calidad de cualquier software que sea desarrollado. Esto implica la prueba de los componentes y sus respectivos conjuntos a los que pertenecen para determinar la integridad, capacidad, confiabilidad, eficiencia, portabilidad, mantenibilidad, compatibilidad y usabilidad.

De entre las distintas pruebas existentes, se implementará el método de la "Caja Negra" para la realización de proyectos de software dentro de la UIEPCH. Este método permitirá comprobar la funcionalidad de los productos de cada sprint al igual que del software construido en su totalidad sin la necesidad de conocer su estructura interna ni los detalles de implementación, es decir, se realizarán las comprobaciones necesarias desconociendo tanto el código fuente como el funcionamiento interno del software.

Para ejecutar esta prueba, previamente el Comité de Tecnologías elegirá a uno o más elementos que conformarán al equipo de pruebas, el cual será encargará de verificar las entradas y salidas a fin de identificar defectos en el comportamiento general o especifico de los elementos que conforman el software, asegurando así, que se cumplan los requisitos especificados en la etapa de planeación.

Para ello, serán tomadas en cuenta los siguientes puntos clave:

- Esta prueba se enfocará en el comportamiento externo de los elementos del software.
- El equipo de pruebas no tendrá acceso al código fuente ni al funcionamiento interno del software.
- El equipo de pruebas tendrá de antemano información sobre los resultados que se esperan de cada uno de los elementos a evaluar.
- Está prueba tiene por objetivo identificar defectos de comportamiento del software que no concuerden con la información obtenida en el punto anterior, por ejemplo, entradas o salidas erróneas o comportamientos inesperados.
- Se realizará esta prueba en cada final de sprint y antes de la entrega del software al solicitante.

A continuación, se describirán los pasos a realizar para la ejecución de esta prueba:

- Planificación de pruebas: El equipo de pruebas, con el apoyo de la información de los requisitos funcionales, identificarán los comportamientos esperados de cada elemento y/o conjunto de elementos.
- Desarrollo de casos de pruebas: El equipo de pruebas contemplará todos los escenarios posibles de interacción con los elementos del software con la ayuda del punto anterior.
- Seguimiento de defectos: El equipo de pruebas rastreará los defectos localizados, es decir, que dará evidencia del (posible) origen que ocasionen los defectos encontrados previamente.
- Cierre de prueba: En esta última etapa, el equipo de pruebas reportará al Scrum Máster acerca de las actividades realizadas y los resultados obtenidos durante las pruebas.

Capítulo 3. Retrospectiva

La fase de retrospectiva es el último paso llevado a cabo al finalizar la construcción de cualquier proyecto de software. Durante esta fase, los miembros involucrados en el proyecto reflexionarán sobre su trabajo reciente, analizando que cosas funcionaron bien y cuáles no. Esto será con la finalidad de buscar oportunidades de mejora para un siguiente desarrollo.

Esta fase invita a que todos los miembros participen expresando sus opiniones de forma libre, discutiendo aspectos positivos y negativos, generando acciones concretas de mejora promoviendo la transparencia, colaboración y aprendizaje continuo, todo dentro de un ambiente seguro.

Estos aspectos serán encapsulados dentro del Formato de Informe de Cierre del Proyecto (Anexo F) tomando en cuenta los siguientes puntos clave:

- Se explicará el objetivo del informe y la descripción del software.
- Se resumirán los alcances y objetivos del software.
- Se describirán los logros y resultados obtenidos realizando una comparativa con los objetivos y alcance planificados.
- Se enlistarán los entregables cumplidos de cada sprint además de su respectivo análisis de calidad y relevancia.
- Se analizará la eficiencia y eficacia de los recursos utilizados en la realización del software.
- Se describirán las lecciones aprendidas durante el desarrollo además de recomendaciones para futuros proyectos.
- Se generará un resumen de los puntos anteriores destacando los principales logros, desafíos y lecciones aprendidas añadiendo reflexiones y recomendaciones de ser necesario.

Anexos

Anexo A

FORMATO DE SOLICITUD DE SOFTWARE	Fecha:	(1)
NOMBRE DEL PROYECTO		
(2)		
DATOS DEL SOLICITANTE		
Nombre completo:	(3)	
Dirección:	(4)	
Cargo:	(5)	
Correo electrónico:	(6)	

DESCRIPCIÓN DEL SOFTWARE
Descripción del problema o necesidad
(7)
Requisitos del sistema
(8)
Plazo
(9)

INSTRUCTIVO DE LLENADO

1.-	FECHA	Anotar la fecha de elaboración del documento.
2.-	NOMBRE DEL PROYECTO	Anotar el nombre del sistema que se desea solicitar.
3.-	NOMBRE COMPLETO	Anotar el nombre completo del solicitante.
4.-	DIRECCIÓN	Anotar el nombre de la dirección al que pertenece.
5.-	CARGO	Anotar el cargo dentro de la dirección.
6.-	CORREO ELECTRÓNICO	Anotar el correo electrónico del solicitante o de la dirección.
7.-	DESCRIPCIÓN DEL PROBLEMA O NECESIDAD	Describir el problema que haya provocado que se requiera de un sistema.
8.-	REQUISITOS DEL SISTEMA	Anotar los requisitos que se necesitan para el sistema. Comúnmente se describe como se requiere que funcione el sistema además de que elementos se necesita que contenga.
9.-	PLAZO	Anotar el plazo de tiempo en que se necesita se entregue el sistema.

Anexo B

FORMATO DE ENTREVISTA DE CONOCIMIENTO DEL SISTEMA	Fecha:	(1)
NOMBRE DEL PROYECTO		
(2)		
ENTREVISTA		
¿Cuál es el objetivo principal del departamento para el cual se requiere el software?	(3)	
¿Cómo se maneja el proceso actualmente?		
¿Cuáles son las principales funciones que necesita el software?		
¿Actualmente se maneja un software de terceros? (Office, Suni u otros)		
¿Quiénes serán los usuarios principales?		
¿Qué tipo de datos o información necesita ser manejada y almacenada por el software?		
¿Se requerirán integraciones con otros sistemas o aplicaciones escolares? (Plataforma móvil, trabajar en paralelo con otros sistemas)		
¿Se necesitará la integración de una base de datos?		
¿Qué nivel de seguridad y privacidad de la información será necesaria?		
¿Cuál es el plazo de tiempo que se espera se entregue el proyecto?		

INSTRUCTIVO DE LLENADO

1.-	FECHA	Anotar la fecha de elaboración del documento.
2.-	NOMBRE DEL PROYECTO	Anotar el nombre del sistema que se desea solicitar.
3.-	ENTREVISTA	Anotar las respuestas que del solicitante a cada pregunta.

Anexo C

FORMATO DE EVALUACIÓN DE VIABILIDAD DE DESARROLLO DE SOFTWARE

Aspectos para evaluar	Decisión		Comentarios (opcional)
	Si	No	
¿Existe una necesidad clara?			
¿Es comprobable la necesidad?			
¿Existe algún software de terceros que pueda solucionar la necesidad?			
¿Se pueden realizar los requerimientos funcionales?			
¿Se pueden realizar los requerimientos no funcionales			
¿Son suficientes?			
¿Se cuenta con equipo como computadoras?			
¿Están en buenas condiciones?			
¿Son suficientes?			
¿Se cuenta con recursos humanos (programadores)?			
¿Se cuentan con las herramientas necesarias para su desarrollo?			
¿Son suficientes?			
¿Se tiene conocimiento del uso de esas herramientas?			
¿Se necesita enseñar a los programadores sobre las herramientas?			
¿Se conoce la plataforma de desarrollo?			
¿Se tiene conocimiento de la plataforma de desarrollo?			
¿Se necesita enseñar a los programadores sobre la plataforma?			
¿Se puede aplicar mantenimiento al software?			
¿Se puede actualizar el software en un futuro?			
¿Se necesitará un plan de seguridad para protección de datos?			
¿Se necesitará retroalimentación de los usuarios para mejorar el software?			
¿Se necesitará un plan de seguimiento del software?			
¿Se necesitará un plan de medición de rendimiento del software?			

Anexo D

FORMATO DE APROBACIÓN DE DESARROLLO DE SOFTWARE	**Nombre de proyecto:**	(1)
	Fecha:	(2)

Descripción del proyecto
(3)
Requisitos del proyecto
(4)
Evaluación y aprobación
(5)
Comentarios adicionales
(6)

Presidente del Comité de Tecnologías

Secretario Técnico

INSTRUCTIVO DE LLENADO

1.-	NOMBRE DE PROYECTO	Anotar el nombre del sistema solicitado.
2.-	FECHA	Anotar la fecha cuando se realiza el proceso.
3.-	DESCRIPCIÓN DEL PROYECTO	Anotar el objetivo del sistema y los beneficios esperados.
4.-	REQUISITOS DEL PROYECTO	Anotar los requisitos, incluyendo las características y cualquier otro aspecto relevante.
5.-	EVALUACIÓN Y APROBACIÓN	Anotar los nombres, roles, firmas y fechas de aprobación de los representantes del Comité de Tecnologías.
6.-	COMENTARIOS ADICIONALES	Anotar los comentarios, aclaraciones o cualquier información relevante relacionada con la aprobación (o denegación según sea el caso) del desarrollo de software.

Anexo E

FORMATO DE PRODUCT BACKLOG	Nombre del Proyecto:	(1)
	Fecha de Comienzo:	(2)

OBJETIVOS DEL SPRINT
(3)
REQUERIMIENTOS
Requerimientos funcionales (4) **Requerimientos no funcionales** (5)
DEFINICIÓN DE TAREAS Y ACTIVIDADES

ID	No. Sprint	Descripción	Responsable	Estado	Estimación	Nivel de importancia
(6)	(7)	(8)	(9)	(10)	(11)	(12)

REUNIONES Y EVENTOS
(13)

REGISTRO DE RIESGOS DEL SPRINT ANTERIOR							
ID	**No. Sprint**	**Autor del riesgo**	**Fecha de registro**	**Categoría del riesgo**	**Causa principal**	**Impacto**	**Estado**
(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)

REGISTRO DE RIESGOS PARA EL NUEVO SPRINT					
ID	**No. Sprint**	**Encargado del riesgo**	**Fecha de término**	**Respuesta al riesgo**	**Observaciones**
(22)	(23)	(24)	(25)	(26)	(27)

RESULTADOS DEL SPRINT ANTERIOR
(28)

INSTRUCTIVO DE LLENADO

1.-	NOMBRE DEL PROYECTO	Anotar el nombre del sistema a elaborar.
2.-	FECHA DE COMIENZO	Anotar la fecha de cuando se realiza el proceso.
3.-	OBJETIVOS DEL SPRINT	Anotar los objetivos que se esperan alcanzar durante el sprint.
REQUERIMIENTOS		
4.-	REQUERIMIENTOS FUNCIONALES	Anotar los requerimientos funcionales detectados.
5.-	REQUERIMIENTOS NO FUNCIONALES	Anotar los requerimientos no funcionales detectados.
DEFINICIÓN DE TAREAS Y ACTIVIDADES		
6.-	ID	Anotar el número representativo (1, 2, 3, etcétera).
7.-	NO. SPRINT	Anotar el número del sprint actual.
8.-	DESCRIPCIÓN	Anotar una breve descripción sobre la tarea o actividad a realizar.
9.-	RESPONSABLE	Anotar el nombre del o los representantes encargados de cumplir el punto anterior
10.-	ESTADO	Anotar el estado en que se encuentra la actividad, ya sea COMPLETADO, NO COMPLETADO o EN PROCESO.
11.-	ESTIMACIÓN	Anotar la estimación de tiempo que se necesitará para completar la tarea/actividad a realizar.

12.-	NIVEL DE IMPORTANCIA	Anotar el grado de importancia de la actividad en una escala del 1 al10, tomando en cuenta que 1 es poco grado de importancia y 10 es máximo grado de importancia.
13.-	REUNIONES Y EVENTOS	Anotar las reuniones acordadas con fecha y duración.
REGISTRO DE RIESGOS DEL SPRINT ANTERIOR		
14.-	ID	Anotar el número representativo (1, 2, 3, etcétera).
15.-	NO. SPRINT	Anotar el número del sprint anterior.
16.-	AUTOR DEL RIESGO	Anotar el o los nombres de las personas que hayan identificado el riesgo.
17.-	FECHA DE REGISTRO	Anotar la fecha de cuando se detectó el riesgo.
18.-	CATEGORIA DEL RIESGO	Anotar donde influye el riesgo, por ejemplo, en la seguridad, organización u otra área.
19.-	CAUSA PRINCIPAL	Describir el riesgo anotando la causa principal.
20.-	IMPACTO	Anotar la calificación del riesgo del 1 al 5, considerando que 1 es bajo y 5 es crítico/alto.
21.-	ESTADO	Anotar si el riesgo está ACTVIO o RESUELTO.
REGISTRO DE RIESGOS PARA EL NUEVO SPRINT		
22.-	ID	Anotar el número representativo (1, 2, 3, etcétera).
23.-	NO. SPRINT	Anotar el número del sprint que dará comienzo.
24.-	ENCARGADO DEL RIESGO	Anotar el o los nombres de las personas que se encargarán del riesgo.
25.-	FECHA DE TÉRMINO	Anotar en qué fecha se estima se resolverá el riesgo.

26.-	RESPUESTA AL RIESGO	Anotar que alternativa se realizará para contrarrestar el riesgo.
27.-	OBSERVACIONES	Anotar los motivos que no permiten resolver el riesgo.
28.-	RESULTADOS DEL SPRINT ANTERIOR	Anotar un resumen de los resultados positivos obtenidos durante el sprint.

Anexo F

FORMATO DE INFORME DE CIERRE DE PROYECTO	**Nombre de proyecto:**	
	Fecha:	

Instrucciones:

1. Introducción: Objetivo del informe + Descripción del proyecto
2. Alcance y objetivos: Resumen de los objetivos del proyecto + Descripción del alcance del proyecto
3. Registro: Descripción detallada de los logros y resultados obtenidos en el proyecto + Comparación con los objetivos planteados y el alcance establecido
4. Hitos y entregables: Lista de los hitos y entregables cumplidos + Análisis de su calidad y relevancia para el proyecto
5. Recursos: Análisis de la eficiencia y eficacia en el uso de recursos
6. Lecciones aprendidas: Identificación y descripción de las lecciones aprendidas en el proyecto + Recomendaciones para futuros proyectos similares
7. Conclusiones: Resumen de los principales logros, desafíos y lecciones aprendidas en el proyecto + Reflexiones finales y recomendaciones