INSTITUTO TECNOLÓGICO DE ZITÁCUARO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN SISTEMAS COMPUTACIONALES

“Diseño y desarrollo de un modelo de reingeniería de software para sistemas legados en una institución de educación superior: caso de estudio Instituto Tecnológico de Zitácuaro”

T E S I S

PARA OBTENER EL GRADO DE:

MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:

I.S.C. Cuautli David García López

DIRECTOR DE TESIS

M. en T.I. Darío Davalos Hernández

CO-DIRECTOR DE TESIS

M. en I.S.C Samuel Efrén Viñas Álvarez

LUGAR Y FECHA

H. Zitácuaro, Mich, Febrero de 2024

H. Zitácuaro, Mich., 21/Febrero/2024

CARTA DE AUTORIZACIÓN

ING. DANIEL HERNÁNEZ DURÁN
SUBDIRECTOR ACADÉMICO
PRESENTE

De acuerdo a los Lineamientos para la Operación de los Estudios de Posgrado en el Tecnológico Nacional de México, en donde se establecen los requisitos para la obtención del grado de Maestría; el H. Comité Tutorial del **C. CUAUTLI DAVID GARCÍA LÓPEZ,** estudiante del programa de **Maestría en Sistemas Computacionales**, con número de control: **M21650560**, después de haber realizado la revisión del contenido y formato de tesis "**DISEÑO Y DESARROLLO DE UN MODELO DE REINGENIERÍA DE SOFTWARE PARA SISTEMAS LEGADOS EN UNA INSTITUCIÓN DE EDUCACIÓN SUPERIOR: CASO DE ESTUDIO INSTITUTO TECNOLÓGICO DE ZITÁCUARO**" emite su consentimiento para continuar con el proceso de obtención de grado académico correspondiente.

Por ese motivo se solicita a usted Autorizar a la **C. CUAUTLI DAVID GARCÍA LÓPEZ** la impresión y reproducción de la tesis *in comento*.

ATENTAMENTE
Excelencia en Educación Tecnológica®
"Hagamos Tecnología Creativa, para ser útil a México" ®

H. Comité Tutorial

Director de tesis	Co-Director de tesis
Mtro. Darío Dávalos Hernández	Mtro. Samuel Efrén Viñas Álvarez
Dr. Noel Enrique Rodríguez Maya Revisor	Dra. Irna Zukeyt Garduño Jaimes Revisor

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Zitácuaro
División de Estudios de Posgrado e Investigación

H. Zitácuaro, Mich., 21/Febrero/2024

AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. CUAUTLI DAVID GARCÍA LÓPEZ
CANDIDATO AL GRADO DE MAESTRO EN SISTEMAS COMPUTACIONALES
NO. DE CONTROL: M21650560
PRESENTE

Conforme a los Lineamientos para la Operación de los Estudios de Posgrado en el Tecnológico Nacional de México y por recomendación del H. Comité Tutorial, esta División le autoriza imprimir y reproducir la tesis: **"DISEÑO Y DESARROLLO DE UN MODELO DE REINGENIERÍA DE SOFTWARE PARA SISTEMAS LEGADOS EN UNA INSTITUCIÓN DE EDUCACIÓN SUPERIOR: CASO DE ESTUDIO INSTITUTO TECNOLÓGICO DE ZITÁCUARO".**

Ruego a Usted dar puntual seguimiento al formato en vigor que, para tal caso, indica las características de diseño que deberá contener tan importante documento.

ATENTAMENTE
Excelencia en Educación Tecnológica®
"Hagamos Tecnología Creativa, para ser útil a México" ®

ING. DANIEL HERNÁNDEZ DURÁN
SUBDIRECTOR ACADÉMICO
INSTITUTO TECNOLÓGICO DE ZITÁCUARO

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE ZITÁCUARO
SUBDIRECCIÓN ACADÉMICA

CARTA DE CESIÓN DE DERECHOS

En H. Zitácuaro, Michoacán, a 28 de Febrero de 2024, el(la) que suscribe, **Cuautli David Garcia López** ,estudiante del programa de **Maestría en Sistemas Computacionales** del Instituto Tecnológico de Zitácuaro, con número de control **M21650560,** manifiesto que soy autor intelectual de la presente tesis, la cual fue dirigida por **M. en T.I Darío Dávalos Hernández** y cedo íntegramente los derechos de trabajo de tesis titulado: **"Diseño y desarrollo de un modelo de reingeniería de software para sistemas legados en una institución de educación superior: caso de estudio Instituto Tecnológico de Zitácuaro"** al Tecnológico Nacional de México / Instituto Tecnológico de Zitácuaro para su uso con fines académicos y de investigación.

Los usuarios pueden consultar y reproducir el contenido para todos los usos que tengan finalidad académica siempre y cuando sea citada la fuente información.

ATENTAMENTE

Cuautli David Garcia López
Nombre y firma del estudiante

DECLARACIÓN DE ORIGINALIDAD DE LA TESIS

En H. Zitácuaro, Michoacán, a 28 de Febrero de 2024, el(la) que suscribe, **Cuautli David Garcia López**, estudiante del programa de **Maestría en Sistemas Computacionales** del Instituto Tecnológico de Zitácuaro, con número de control **M21650560**, manifiesto que soy autor intelectual de la presente tesis, la cual fue dirigida por el **M. en T.I Darío Dávalos Hernández** con nombre: **"Diseño y desarrollo de un modelo de reingeniería de software para sistemas legados en una institución de educación superior: caso de estudio Instituto Tecnológico de Zitácuaro"**.

Declaro que la tesis es una obra original, que es de mi autoría y que toda la información y materiales extraídos de otras fuentes han sido debidamente referenciados. Que a la obra no ha sido previamente publicada y que, en caso de violación de derechos de autor, me hago responsable y exonero de toda responsabilidad al Instituto Tecnológico de Zitácuaro.

ATENTAMENTE

Cuautli David Garcia López

Nombre y firma del estudiante

Agradecimientos

“ Con un corazón lleno de gratitud, inicio dando gracias a Dios, cuya guía espiritual ha sido esencial en mi camino, y a mis padres, cuyo apoyo incondicional y confianza inquebrantable han sido la base de mi perseverancia y éxito. Extiendo un profundo agradecimiento a mi abuelito, cuyas historias y sabiduría han sido una fuente constante de inspiración para seguir mis sueños con determinación. A cada miembro de mi querida familia, primos, tíos y tías, les estoy inmensamente agradecido por sus valiosos consejos, por los momentos inolvidables compartidos y por un apoyo constante que ha sido fundamental en mi crecimiento personal.

A mi amada novia, eres el pilar de amor y serenidad en mi vida. Tu comprensión, paciencia y tu fe inquebrantable en mí han sido mi refugio y mi mayor motivación durante los momentos más desafiantes.

A los docentes que han dejado una huella imborrable en mi formación, por su pasión y dedicación al compartir su conocimiento. Un agradecimiento especial a los directivos de mi tesis, por su paciencia y rigurosidad, y a mis maestros de inglés, cuyo apoyo ha sido crucial en este camino.

Quiero reconocer a mis compañeros de trabajo en las diversas empresas que han sido parte de mi viaje profesional. Cada experiencia compartida y cada desafío superado juntos, han contribuido enormemente a mi crecimiento personal y profesional, de la misma manera, A mis amigos, gracias por su amistad sincera, sus risas y consejos. Ustedes han añadido alegría y significado a esta aventura, recordándome siempre la importancia de la camaradería.

Este logro académico es mucho más que un título; es el reflejo del amor, el apoyo y el esfuerzo de cada uno de ustedes. A todos y cada uno, les estoy profundamente agradecido. ¡Gracias por ser parte de mi historia! ”

ÍNDICE GENERAL

ÍNDICE DE FORMATOS

ÍNDICE DE FIGURAS

LISTA DE ABREVIATURAS Y/O SÍMBOLOS

1. Sistemas legados: Software heredado antiguo desactualizado.
2. DEP: División de Estudios Profesionales.
3. DGTV: Departamento de Gestión Tecnológica y Vinculación.
4. DA: Departamentos Académicos.
5. SE: Servicios Escolares.
6. ITZ: Instituto Tecnológico de Zitácuaro.
7. IES: Institución de Educación Superior.
8. INEGI: Instituto Nacional de Estadística y Geografía.
9. UNESCO: Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura.
10. IEEE: Instituto de Ingenieros Eléctricos y Electrónicos.
11. CMMI: Integración de modelos de madurez de capacidades (Capability Maturity Model Integration).
12. PO: Product Owner (dueño del producto).
13. PA: Project Administrator (Administrador de proyecto
14. Dev: Development Team (Equipo de desarrollo).
15. R: Reviewer.
16. Dir: director.
17. OAR: Options Analysis for Reengineering: Metodología diseñado para ayudar a las organizaciones a analizar y decidir entre diferentes opciones de reingeniería.
18. ORM: Object-Relational Mapping: Técnica de programación utilizada para convertir datos entre sistemas de bases de datos relacionales y objetos en programación orientada a objetos (POO)
19. Framework: Es un marco o estructura de software predefinida que sirve como base para desarrollar y organizar el código en aplicaciones o sistemas específicos.
20. Mindbox: Es un sistema en la nube que se encarga de automatizar los procesos de tu institución así como brindar las herramientas para que todos en tu institución puedan cumplir con sus actividades de la mejor manera.

RESUMEN

El presente trabajo se centra en la investigación desarrollada y aplicada en el ITZ, en la ciudad de H. Zitácuaro, Michoacán, México; esta Institución de Educación Superior (IES) se enfrenta ante el reto de producir software para sus principales procesos administrativos, así como actualizar y mejorar sus sistemas legados usando técnicas de reingeniería de software; para hacer frente a esta situación se analizaron estudios similares sobre este tema y a partir de ello se ha propuesto un modelo que se adapte a las necesidades del ITZ considerando su posible aplicación para otras IES con características similares. El modelo está compuesto por cuatro etapas: etapa de evaluación del sistema, etapa ágil, etapa de abstracción y etapa de aplicación. Al desarrollar y aplicar el modelo para un caso de estudio se obtuvieron resultados que pueden ser de interés para la comunidad científica académica, así como para los equipos de desarrollo de software que tengan la necesidad de aplicar un modelo de reingeniería de software que sea ágil y que les permita minimizar costos.

El modelo se aplicó a un requerimiento específico que implicaba un sistema legado sin documentación, solo se contaba con código. Al aplicar el modelo, se logró reducir el tiempo requerido para recopilar datos importantes sobre el funcionamiento del sistema, el proyecto fue documentado, beneficiando así al equipo de trabajo, fomentando las buenas prácticas. El equipo de desarrollo destacó que la redacción de la documentación fue breve y concisa gracias a las herramientas proporcionadas, siendo esto reflejado en los tiempos de levantamiento de requerimientos.

El modelo demostró ser viable para futuros mantenimientos y proyectos nuevos, se pretende seguir aplicando en otros sistemas con la finalidad de estandarizar el proceso de desarrollo del ITZ.

ABSTRACT

This paper focuses on research developed and applied at the ITZ, in the city of H. Zitácuaro, Michoacán, Mexico; This Higher Education Institution (HEI) faces the challenge of producing software for its main administrative processes, as well as updating and improving its legacy systems using software reengineering techniques; To deal with this situation, similar studies on this subject were analyzed and from this a model has been proposed that adapts to the needs of ITZ considering its possible application to other HEIs with similar characteristics. The model is composed of four stages: system evaluation stage, agile stage, abstraction stage, and application stage. By developing and applying the model for a case study, results were obtained that may be of interest to the academic scientific community, as well as to software development teams that need to apply a software reengineering model that is agile and that allows them to minimize costs.

The model was applied to a specific requirement that implied a legacy system without documentation, only code was available. By applying the model, it was possible to reduce the time required to collect important data on the operation of the system, the project was documented, thus benefiting the team of work promoting good practices. The development team highlighted that the writing of the documentation was brief and concise thanks to the tools provided, this being reflected in the requirements gathering times.

The model proved to be viable for future maintenance and new projects, it is intended to continue applying it in other systems in order to standardize the ITZ development process.

INTRODUCCIÓN

En diversa literatura se puede encontrar diferentes definiciones y conceptos de reingeniería de software, algunos conceptos la definen como una actividad que permite mejorar la comprensión del software, o bien, que mejora el trabajo de actualización, mantenimiento, reutilización o evolución (Arnold, 1993). Es decir, creando una serie de actividades en las que se incluye un análisis de inventario, restructuración de programas y datos, teniendo como objetivo crear versiones de programas existentes con mayor calidad y mantenibilidad (Pressman, 2022).

En general, la reingeniería de software está relacionada con el proceso de reutilizar lo ya existente para mejorar su calidad, es decir, sistemas que han sido heredados o legados que fueron desarrollados por personas utilizando técnicas y estilos de programación propia, las cuales ya no se encuentran en el equipo de desarrollo o en la empresa.

Los sistemas legados muy frecuentemente no cuentan con documentación o es muy escasa, además, de que el equipo que desarrollo el software ya no se encuentra en la empresa y es difícil realizar una actualización de acuerdo a las nuevas necesidades, por lo que iniciar de cero no es una opción y se requiere hacer uso de reingeniería de software para atender las demandas actuales.

El Instituto Tecnológico Nacional de México Campus Zitácuaro casa de estudios que ha mostrado forjado grandes exitosos profesionistas, ha tomado a bien dar un paso a la evolución de sus procesos administrativos, esto a causa de la pandemia vivida a partir del 2019 donde tuvo que migrar toda su forma educativa a las plataformas virtuales; sin embargo, su proceso administrativo no fue migrado de manera satisfactoria, ya que no se contaba con una plataforma la cual gestionará las reglas del proceso de cada área, provocando irregularidad en sus datos y en sus procesos.

El área donde se enfoca esta investigación tiende a ser de las más importantes para la institución, ya que es el proceso donde los alumnos acuden a las empresas y participan en proyectos, demostrando así el conocimiento adquirido en la institución.

Como plan de contingencia se empezó a desarrollar un sistema capaz de gestionar los documentos requeridos para llevar a cabo el proceso de residencias profesionales, lamentablemente no se pudo terminar el desarrollo por falta de tiempo.

Personal administrativo del área realizaron una investigación y encontraron que en diversas generaciones se había realizado sistemas que cubrían gran parte de las necesidades que se presentaban, lamentablemente los alumnos que desarrollaron los sistemas ya no se encuentran dentro de la institución y la documentación no se encuentra completa. Por dicha razón se optó por aplicar una reingeniería con el fin de actualizar los sistemas encontrados, sin embargo, se intentó aplicar las metodologías, pero no se obtuvieron buenos resultados, teniendo como surgimiento la idea de crear una metodología de reingeniería, la cual pueda no solo ayudar al proceso de residencias profesionales sino también a otros procesos, reutilizando proyectos generados por los mismos egresados del instituto.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA

En este capítulo se presenta a detalle el motivo de donde nace la investigación para el diseño de un modelo de reingeniería de software para sistemas legados en el ITZ, teniendo como componente clave el objetivo general de la investigación, justificación e hipótesis, además de datos estadísticos sobre el uso de sistemas en el sector educativo antes y después de la pandemia por COVID-19.

1.1 Descripción del caso de estudio

En la presente investigación se centra en el contexto del ITZ que como muchas instituciones educativas desde sus inicios de fundación tomó a bien migrar sus procesos administrativos a sistemas informáticos. El ITZ se encuentra organizado en varios departamentos que atienden diferentes procesos y ofrecen diversos servicios. Como parte del proceso de mejora continua que exige el Sistema de Gestión de Calidad (SGC), se optó por la adquisición de software comercial para el manejo de algunos procesos como el Sistema de Control Escolar a cargo de la empresa R2. Por otro lado, también se ha apoyado con personal adscrito y estudiantes de licenciatura para el desarrollo de sistemas a la medida.

Sin embargo, conforme transcurre el tiempo, estos sistemas han dejado de tener mantenimiento para seguir atendiéndolas con nuevas necesidades y cambios constantes que el modelo educativo demanda; en la actualidad esta institución ya cuenta con un equipo de desarrollo de software y uno de los retos a los que se enfrentan es la reingeniería de los sistemas actuales, ya que no se cuenta con el personal que desarrolló el sistema, tampoco se tiene documentación referente a diagramas de casos de uso, diseño de bases de datos o diccionarios de datos, tampoco se cuenta con el código fuente de los programas; aunado a lo anterior también existe la carga de trabajo para la creación de nuevos sistemas y la implementación de nuevos mantenimientos.

1.2 Antecedentes del caso de estudio

Los sistemas de información y la tecnología juegan un papel importante de las instituciones educativas, a medida que avanza el tiempo se han creado variedad de proyectos y programas los cuales han promovido el uso de tecnologías con el propósito de mejorar el proceso educativo por ejemplo el programa Escuelas 2.0 que fue adoptado a inicios del 2006 principalmente por escuelas de educación básica promoviendo así aulas con equipo de cómputo donde el alumno pudiera aprender de manera didáctica, posteriormente se continuo promoviendo en las escuelas de media superior y superior donde actualmente gran parte de las escuelas en México se encuentran con aulas multimedia beneficiando así el aprendizaje del alumno (UNESCO, 2022).

Durante el periodo 2017 al 2021 se registró una encuesta por parte del INEGI, la cual tenía como propósito obtener indicadores sobre el uso de tecnologías de información, donde se observó que ha habido más usuarios de telefonía celular que de internet y computadora, observándose un gran dominio en el uso de esta tecnología en la población mexicana, mientras que los usuarios de computadora han tenido un comportamiento totalmente contrario, disminuyendo año con año **véase Figura 1. Porcentajes de encuestas ENDUTIH. *Autor:*** (Instituto Nacional de Estadística y Geografía)**.**

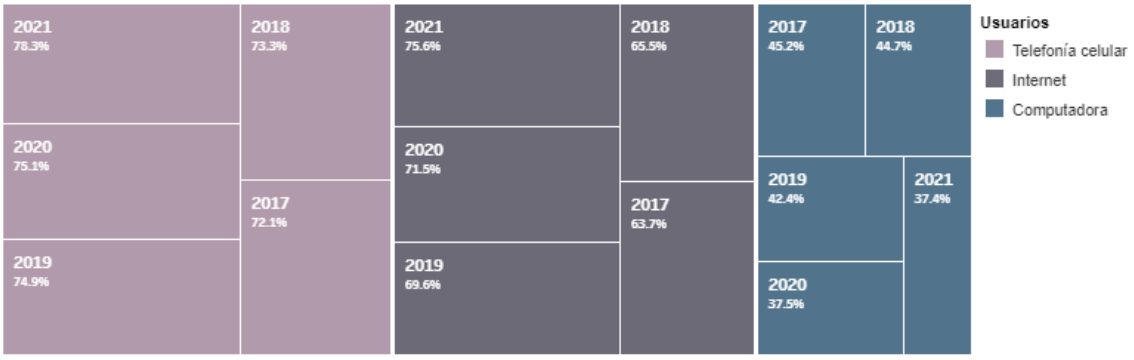


Figura 1. Porcentajes de encuestas ENDUTIH. Autor: (Instituto Nacional de Estadística y Geografía)

Con la llegada del COVID-19, el mundo entero se paralizó, forzando a modificar la forma de realizar diversas actividades, entre ellas las labores escolares, provocando el cese temporal de actividades presenciales, migrando todos sus servicios de atención y gestión, además de las clases a las plataformas digitales y a sistemas de información con el fin de sobrellevar algunas actividades.

Es cierto que la pandemia fue un periodo el cual afectó a la humanidad de una manera impresionante en diversos sectores, pues no estábamos preparados para el cese de actividades presenciales de forma indefinida, sin embargo, ayudó a revolucionar formas de trabajo.

La UNESCO en su publicación titulada "*COVID-19 y educación superior de los efectos inmediatos al día después"* (UNESCO, 2022), analiza los efectos causados por la pandemia haciendo mención sobre el esfuerzo que debe realizar el estudiante para adaptarse la nueva forma de enseñanza la cual es por medio de la conexión a internet, datos estadísticos proporcionados por la unión internacional de telecomunicaciones muestra que México es uno de los países que por cada 100 habitantes existen 60 constantemente conectados, sin embargo, se detectó que en la mayoría de los casos los habitantes tienen más de una línea móvil por dicha razón los resultados de la gráfica resultan ser una paradoja **véase Figura 2. Porcentaje de hogares con conexión a internet y de líneas móviles por cada 100 habitantes en una muestra de países de América Latina *Fuente:***

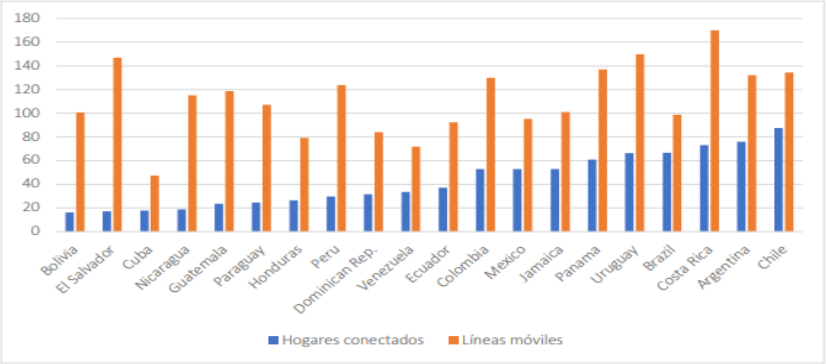


Figura 2. Porcentaje de hogares con conexión a internet y de líneas móviles por cada 100 habitantes en una muestra de países de América Latina Fuente: (IESALC, 2021)

Ahora bien, los datos analizados son a partir del 2020 donde la pandemia estaba en su mayor alcance, actualmente en 2022 se obtuvieron datos más exactos donde se demuestra que los jóvenes de 15 a 24 años de edad son el motor de la conectividad (IESALC, 2021) puesto que el 75% tiene acceso a internet rango de edades donde se pretende estudien su nivel medio superior y superior. Tomando en cuenta esto con los datos analizados por las diversas fuentes, podemos detectar que hay un área de oportunidad de mejora en soluciones tecnológicas, la cual promete revolucionar la forma de trabajo de instituciones educativas, facilitando el trabajo de personal administrativo y docentes, además de poner al alcance de alumnos herramientas y recursos educativos siendo que gran parte de los jóvenes tienen acceso a internet.

Desde la implementación del programa Escuelas 2.0 en 2006 hasta el aumento en el uso de dispositivos moviles y la migración a la enseñanza en línea debido a la pandemia de COVID-19, se demuestra que la tecnología ha sido un factor clave en la evolución de la educación.

Además, esta evolución tecnológica ha influido en la mejora de procesos administrativos en las instituciones educativas. La digitalización y automatización de tareas administrativas, como la gestión de registros, la comunicación con los estudiantes y la organización de recursos, han simplificado y agilizado la operación de las escuelas. Esto ha permitido al personal administrativo centrarse más en la toma de decisiones estratégicas y en mejorar la calidad de la educación.

Los desafíos planteados por la pandemia y la necesidad de adaptarse a la enseñanza en línea han resaltado aún más la importancia de las soluciones tecnológicas en las instituciones educativas, tanto para la enseñanza como para la administración. Además, el hecho de que los jóvenes de 15 a 24 años sean la población más conectada a Internet en México señala una oportunidad significativa para continuar mejorando y transformando la educación superior a través de la tecnología, beneficiando tanto a los estudiantes como a los procesos administrativos.

1.3 Formulación del problema

Tomando en cuenta que en la actualidad el equipo de desarrollo no tiene aplicada una metodología para realizar ingeniería y reingeniería de software, se estima generar un modelo que vaya acorde a las necesidades del ITZ, además que facilite tanto del desarrollo como la documentación de los proyectos estructurando de manera ordenada el proceso de ingeniería y reingeniería de software, es decir, llevar a cabo la formulación documentos correspondientes desde el levantamiento de requerimientos hasta la entrega de producto final con el fin de promover y crear una bitácora de históricos para futuros equipos de trabajo, ya que en la institución tiene rotación de personal muy constante, la finalidad es estandarizar la forma de trabajo en el área de desarrollo por medio del diseño de una metodología que tenga característica ágil de tal forma que no se dificulte para los equipos de trabajo.

Tomando en consideración la actual carencia de una metodología específica para llevar a cabo procesos de ingeniería y reingeniería de software dentro del equipo de desarrollo del ITZ, se visualiza una necesidad de implementar un modelo que se adapte con las exigencias y contexto particular del ITZ. Esta iniciativa busca ir más allá, aspirando no solo a facilitar el desarrollo y documentación de proyectos, sino también a estructurar de forma ordenada los procesos internos a la ingeniería de software y sus respectivas fases de reingeniería.

La propuesta se orienta en la concentración de un proceso sistemático, el cual abarcará desde la fase inicial de levantamiento de requerimientos hasta la culminación y entrega del producto final. Esta iniciativa no solo se concibe como una solución para proyectos actuales, sino que promueve la creación de una bitácora de históricos que servirá de guía para futuros equipos de trabajo. Dada la notoria rotación de personal que experimenta la institución, este registro histórico no sólo facilitará la inducción y adaptación de nuevos integrantes al equipo de desarrollo, sino que también servirá como un instrumento para evitar la pérdida de conocimientos y experiencias acumuladas.

La finalidad de este proyecto se enfoca en estandarizar la forma de trabajo dentro del área de desarrollo, no solo como una medida para asegurar la calidad y coherencia de los proyectos actuales, sino también como un mecanismo para asegurar la sostenibilidad y la continuidad de los esfuerzos de desarrollo a largo plazo. Se busca diseñar una metodología que, si bien esté fundamentada en principios sólidos y estructurados, también incorpore características ágiles, permitiendo de esta manera una adaptabilidad y flexibilidad que faciliten su implementación y seguimiento por parte de los diferentes equipos de trabajo.

Este enfoque ágil no sólo se alinea con las tendencias actuales en desarrollo de software, sino que también se presenta como una estrategia viable para fomentar un ambiente de trabajo dinámico ante los retos y cambios que inevitablemente emergen en proyectos tecnológicos. A través de este modelo, se pretende que el equipo de desarrollo pueda navegar a través de las diversas etapas del ciclo de vida del software, asegurando no sólo la entrega oportuna de soluciones de alta calidad, sino también la documentación comprensiva de cada etapa del proceso, consolidando así una base de conocimientos que proporcione soporte y orientación para futuras iniciativas y equipos en la institución.

1.4. Justificación

De acuerdo con los fundamentos de Ingeniería de Software (AVELLA IBÁÑEZ, 2004), el hacer uso de un modelo o metodología para un proyecto de desarrollo de software permitirá obtener mejores resultados a los desarrolladores, a su vez se podrá entregar un producto de acuerdo a las necesidades y requerimientos del usuario final, sin embargo, al momento de iniciar un proyecto es necesario seleccionar la metodología que más se adapte al proyecto y al equipo de desarrollo, ya que de lo contrario lejos de ser una ventaja se puede convertir en un ciclo sin fin y puede elevar los costos del proyecto.

En el caso de los sistemas legados existen diversas técnicas para aplicar Reingeniería de Software y también es importante contar con los modelos que se puedan aplicar de acuerdo al entorno y necesidades.

La presente investigación propone el diseño, desarrollo y aplicación de un modelo de reingeniería de software que pueda ser utilizado por el equipo de programadores del ITZ para atender sistemas legados y de reciente creación.

Los beneficios al usar el modelo que se propone es que permitirá reconstrucción y evolución de los sistemas con los que cuenta la institución, asimismo permite a nuevo personal que se integre, el comprender como funcionan los sistemas existentes. Otro de los beneficios que aporta este trabajo es que permitirá la construcción de nuevos sistemas mediante la reutilización de código de los sistemas con los que ya se cuenta, permitiendo agilizar los tiempos de desarrollo.

La documentación del sistema es parte importante en el modelo de reingeniería, por lo que, a través del caso de estudio, se podrá reconstruir la arquitectura de un sistema legado, además de complementar la documentación de dicho sistema. Una aportación de esta investigación es presentar los resultados al aplicar el modelo propuesto en el instituto.

Esta propuesta de modelo, además de considerar a los sistemas que llevan varios años en la institución, puede ajustarse a nuevos sistemas, en los casos cuando la documentación es obsoleta o no existe, cuando se necesita hacer constantes cambios para solucionar fallas. Al aplicar reingeniería de software se busca reestructurar un sistema para que pueda evolucionar de acuerdo a nuevos objetivos, buscando la calidad del software cumpliendo con las funciones esperadas (funcionalidad), además mantener la facilidad para ser modificado y corregir defectos (mantenibilidad) (Sommerville, Galipienso, & Martinez, 2005).

Todo software tiene ciclo de vida en el cual si se desea extender la vida del producto será necesario aplicar reingeniería de software para reducir costos de inversión en cuanto a tiempo, dinero y esfuerzo, por esta y las otras razones mencionadas anteriormente es importante iniciar con este proceso ahora que se cuenta con el capital humano y los permisos necesarios por parte de las autoridades educativas de esta institución para llevar a cabo este estudio.

1.5. Objetivos

1.5.1. Objetivo general

Desarrollar y aplicar un modelo basado en reingeniería de software para mejorar la productividad y eficiencia de sistemas legados que operan en el ITZ.

1.5.2. Objetivos específicos

a) Investigar y analizar las diferentes metodologías y modelos de reingeniería de software para identificar principios a considerar en la elaboración de la propuesta del modelo.
b) Analizar y seleccionar un caso de estudio.
c) Analizar y diseñar mediante un enfoque de reingeniería de Software el proceso del caso de estudio seleccionado.
d) Estudiar las técnicas de Ingeniería de Dominio para describir características que puedan ser reutilizadas de los sistemas legados del ITZ.
e) Favorecer los tiempos de desarrollo y mantenimiento de sistemas legados que gestiona el equipo de desarrollo del ITZ.
f) Desarrollar y realizar pruebas del modelo de reingeniería de software.
g) Presentar los resultados obtenidos para identificar mejoras y áreas de oportunidad.

1.6. Hipótesis

a) A través del diseño y aplicación de un modelo de reingeniería de software se podrá mejorar la productividad y eficiencia en el mantenimiento de sistemas legados y nuevos desarrollos de software del ITZ.

CAPÍTULO II. ESTADO DEL ARTE

2.1 Antecedentes

Los sistemas legados representan una parte importante de la infraestructura informática en muchas organizaciones, ya que siguen siendo responsables de la ejecución de funciones empresariales críticas (Bisbal, Lawless, Wu, & Grimson, 1999), son el resultado de investigación y desarrollo, contienen gran parte del conocimiento empresarial codificado de una organización. A lo largo de este capítulo, se proporciona un análisis sobre el estado actual de los sistemas legados, abordando su definición, características, desafíos y enfoques de modernización.

En los últimos años varios investigadores han dedicado tiempo y esfuerzo a desarrollar sus propios métodos de reingeniería adecuándolos a sus necesidades, en algunos casos ha sido favorable el resultado, en otros se ha generado un conocimiento como parte del crecimiento en la ingeniería de software, a continuación, se muestran parte de los trabajos analizados para generar una propuesta del modelo de reingeniería de software. **Véase Tabla 1. Estudios realizados sobre reingeniería de software.**

Los sistemas legados, reconocidos por su durabilidad y robustez en múltiples organizaciones y campos académicos, enfrentan inevitablemente desafíos relacionados con su actualización y mantenimiento en el contexto de las rápidas evoluciones tecnológicas y de software. El trabajo (REINGENIERÍA DE SOFTWARE APLICADA A UN SISTEMA DE INFORMACIÓN ACADÉMICO DESARROLLADO SOBRE ORACLE-VISUAL BASIC, CON ANÁLISIS Y DISEÑO ESTRUCTURADO) de (AVELLA IBÁÑEZ, 2004) demuestra cómo la reingeniería de software y la ingeniería inversa se aplican a sistemas de información académica legados, logrando revitalizar su arquitectura y documentación sin invertir en un desarrollo completamente nuevo. En un enfoque complementario, (Franco Bianchiotti, 2014) en su trabajo (GUÍA PARA LA REINGENIERÍA DE SISTEMAS LEGADOS: UNA EXPERIENCIA PRÁCTICA Y REAL) aborda la necesidad de desarrollar marcos de trabajo para reingeniería, especificando criterios que

identifican cuándo los sistemas legados son aptos para tales procesos, proporcionando una guía práctica basada en características y factibilidad para la implementación de estrategias de reingeniería. Por su parte, el equipo formado (Álvarez García, Mateos Sánchez, & Moreno García, 2004), en su trabajo (METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACIÓN DE APLICACIONES CIENTÍFICAS HEREDADAS) proponen una metodología específica de reingeniería, especialmente enfocada en la remodelación de aplicaciones científicas heredadas, subrayando la importancia de la colaboración interdisciplinaria y la ejecución paralela de etapas durante el proceso de reingeniería, sin la completa dependencia de la automatización. Colectivamente, estos trabajos proveen un panorama rico y diverso sobre los esfuerzos y estrategias destinadas a preservar y modernizar sistemas legados, contribuyendo valiosos aportes y metodologías para enfrentar los retos inherentes en el manejo de tecnologías heredadas en entornos contemporáneos.

Título de trabajo	Objetivo	Resultados	Autor
REINGENIERÍA DE SOFTWARE APLICADA A UN SISTEMA DE INFORMACIÓN ACADÉMICO DESARROLLADO SOBRE ORACLE-VISUAL BASIC, CON ANÁLISIS Y DISEÑO ESTRUCTURADO	Analizar y rediseñar las técnicas apropiadas de reingeniería de software e ingeniería reversa a sistemas de información desarrollados a la medida con la metodología de análisis y diseño estructurado	Al implementar un proceso de reingeniería de software pueden obtenerse resultados como que el sistema legado se inicie desde cero, sin embargo, no deja de ser una muy buena opción cuando es todo lo contrario, ya que genera un ahorro de gasto en dinero y tiempo, la implementación de esa metodología propuesta permitió tener sistemas bien actualizados tanto en documentación como en arquitectura	Clara Patricia Avella Ibáñez
(Franco Bianchiotti, 2014)	Desarrollar un marco de trabajo de reingeniería que se adecuara a los sistemas legados de la organización con la finalidad de poder ser replicado a otros sistemas legados de la organización.	Detectaron que no es posible aplicar una reingeniería de software a cualquier sistema legado, en listaron características de un sistema legado que puede responder favorablemente a la aplicación de un modelo de reingeniería	Franco Bianchiotti
METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACIÓN DE APLICACIONES CIENTÍFICAS HEREDADAS	Reestructurar o transformar viejos sistemas en aplicaciones más fáciles de mantener, con entornos más agradables e integradas en nuevas plataformas de hardware/software	Se llegó a la conclusión que la mayoría de las metodologías para la reingeniería de software es buscar una completa automatización, es decir que no haya una intervención humana, como resultado la metodología propuesta tiene la ventaja de poder llevarse en paralelo con otras etapas de la misma, e inclusive puede formar parte del equipo personal de áreas diversas con el fin de reducir tiempos	Juan Carlos Álvarez García Montserrat Mateos Sánchez María N. Moreno García

Tabla 1. Estudios realizados sobre reingeniería de software.

2.2 Definición y Características de Sistemas Legados

En el ámbito de la tecnología de la información, los sistemas legados son una parte integral de muchas organizaciones, especialmente aquellas que han estado en funcionamiento durante un período prolongado. Estos sistemas, aunque pueden haber sido desarrollados hace décadas, siguen siendo utilizados para respaldar procesos críticos para el negocio. En general, estos sistemas se caracterizan por su arquitectura obsoleta, la falta de flexibilidad y la falta de documentación. (Aier, Riege, & Winter, 2009).

Existen diferentes razones por las cuales los sistemas legados se siguen utilizando y es que en la mayoría de los casos contienen reglas de negocio de gran importancia, las cuales generarían una reinversión de recursos, dinero y tiempo que las empresas no siempre quieren asumir y es que uno de los riesgos de reemplazar un sistema legado es que no cumpla con los requisitos de negocio esto puede provocar la perdida de datos o funcionalidades importantes. Es difícil migrar o dar mantenimiento a un sistema legado, pero es más complicado crearlo desde cero.

A pesar de los desafíos que presentan los sistemas legados, muchas organizaciones los siguen utilizando debido a su fiabilidad, estabilidad y capacidad para realizar tareas específicas de manera efectiva. En lugar de reemplazar por completo los sistemas legados, las organizaciones pueden optar por modernizarlos, lo que implica actualizarlos con tecnología más moderna y agregar nuevas características y funcionalidades. También se pueden integrar con otros sistemas y aplicaciones para mejorar su interoperabilidad y capacidad de integración.

La reingeniería de software es una de las disciplinas que estudia el mantenimiento y uso de sistemas ya realizados y que requieren ser utilizados son los sistemas legados o heredados, una definición para este tipo de sistema se menciona a continuación:

"*Sistema que ha resistido modificaciones significativas y ha evolucionado para satisfacer constantes cambios del ambiente de negocios, sin importar la tecnología con la que fue desarrollado*" (Brodie & Stonebraker, 1995).

El concepto va muy unido con la reingeniería de software, ya que se analiza un sistema el cual se pretende modificar para mejorarlo o reinventarlo para mejorar su eficiencia.

2.3 Importancia y Retos de los Sistemas Legados

Los sistemas legados tienen un nivel de importancia en las empresas o instituciones según satisfagan las necesidades utilizadas por la organización. En los años 60 y 70 marcaron el nacimiento de muchos sistemas legados. Construidos sobre arquitecturas mainframe y lenguajes de programación como COBOL y FORTRAN, estos sistemas fueron diseñados para satisfacer necesidades muy específicas (Bisbal, Lawless, Wu, & Grimson, 1999).A medida que la tecnología avanzaba, en lugar de ser reemplazados, muchos de estos sistemas se expandieron y adaptaron, consolidando aún más su posición en sus respectivas organizaciones. A pesar de su antigüedad, los sistemas legados desempeñan roles fundamentales en sectores como la banca, salud, transporte y gobierno. Estos sistemas, debido a su confiabilidad, resistencia y capacidad de adaptación, han demostrado ser invaluables, especialmente en tareas críticas donde la confiabilidad es esencial (Brown, McCormick, & Thomas, 2000).

Su importancia es identificada por tres características:

- **Funcionalidad comprobada:** Estos sistemas han sido probados y han demostrado su eficacia y confiabilidad en el curso de su operación durante muchos años. Muchas veces, están diseñados para resolver problemas muy específicos que podrían no ser fácilmente abordados por soluciones más modernas.

- **Datos históricos:** Los sistemas legados a menudo albergan una enorme cantidad de datos históricos. Los datos pueden ser vitales para la operación del negocio y pueden ser muy difíciles de transferir a un nuevo sistema.

- **Costo:** El reemplazo de un sistema legado puede ser costoso en términos de tiempo, dinero y recursos. Esto incluye el costo de adquisición de nuevos sistemas, la capacitación del personal, la migración de datos y la posible interrupción de las operaciones comerciales.

A pesar de su importancia, los sistemas legados plantean varios desafíos. Uno de los principales es la dificultad para mantener y actualizar estas aplicaciones a medida que las habilidades técnicas necesarias para su mantenimiento se vuelven menos comunes (Aversano, Grasso, & Tortorella, 2016). Otra dificultad es la integración de estos sistemas con nuevas aplicaciones y tecnologías, siendo uno de los principales desafíos (Bisbal, Lawless, Wu, & Grimson, 1999).

2.4 Soluciones y Estrategias para la Modernización de Sistemas Legados

Existen diversas estrategias para abordar el problema de los sistemas legados. Según Seacord (Seacord, Plakosh, & Lewis, 2003), estas pueden dividirse en tres categorías: reemplazar el sistema, convertirlo a una plataforma más moderna o encapsularlo para permitir su interacción con otras aplicaciones más nuevas.

Mientras que los sistemas legados continúan siendo un área activa de investigación, es importante notar que todavía existen muchos retos no resueltos. Algunos de los más prometedores incluyen la creación de técnicas automáticas para la conversión de sistemas legados a plataformas modernas (Comella-Dorda, Wallnau, Seacord, & Robert, 2000) y la mejora de las técnicas de encapsulación para facilitar la integración de sistemas legados con nuevas aplicaciones.

Aunque representan arquitectura obsoleta, continuarán siendo relevantes en el futuro previsible. Su modernización y adaptación se convierten en áreas de investigación activa, donde se busca equilibrar la preservación del valor empresarial y la adaptación a las demandas actuales.

CAPÍTULO III. FUNDAMENTO TEÓRICO

En el mundo de la ingeniería de software, es crucial entender los pilares que sustentan su práctica. Este capítulo se enfoca en aspectos esenciales del desarrollo de software, desde su arquitectura y fundamentos, hasta metodologías y modelos que han revolucionado la forma en que abordamos el diseño, mantenimiento y mejora de los sistemas. Exploraremos la importancia de la Arquitectura de Software, los principios detrás del Desarrollo de Software y la revitalización de sistemas mediante la Reingeniería de Software. Además, se abordarán modelos específicos de reingeniería, el concepto de Ingeniería de Dominio, y dos metodologías fundamentales en la industria: Scrum y CMMI. A través de este capítulo, se sentarán las bases teóricas para comprender la naturaleza del desarrollo de software moderno y sus buenas practicas.

3.1 Arquitectura de Software

Al explorar el concepto de arquitectura de software, un aspecto fundamental en el desarrollo de sistemas de software robustos y eficientes. La arquitectura de software se refiere al diseño y estructura del sistema, que proporciona una visión global de cómo los diferentes componentes interactúan entre sí y con el entorno en el que operan.
La arquitectura de software desempeña un papel crucial en el éxito de un proyecto, ya que sienta las bases para la construcción de un sistema de software sólido y flexible.

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos (Palomo, 2020), donde los elementos de software se refieren a las partes del sistema que se deben desarrollar.

La arquitectura de software es la base para construir un sistema que no carezca atributos de calidad como el desempeño y la fiabilidad. Un modelo reutilizable o genérico puede generar un beneficio al proyecto en mantenimientos futuros, siendo la esencia para que un proyecto de software sea exitoso.

Un buen diseño de arquitectura de software se ve reflejado cuando se tiene amplio conocimiento en los objetivos de negocio a cumplir, siendo esto la motivación para el desarrollo del sistema. Algunos autores describen la arquitectura de software como un puente entre objetivos de negocio y del sistema en sí mismo (Cervantes, castro, & Velasco-Elizondo, 2016).

La documentación de una arquitectura consiste en expresar de forma gráfica las vistas relevantes del sistema previamente definidas por el arquitecto, en esa actividad en su mayoría los arquitectos utilizan el lenguaje de modelado unificado (UML) siendo el estándar utilizado para documentar una arquitectura.

Continuando con (Cervantes, castro, & Velasco-Elizondo, 2016), algunas reglas para documentar una arquitectura son las siguientes:

1. Escribir la arquitectura desde el punto de vista de los lectores
2. Evitar ambigüedad
3. Mantener actualizada la documentación
4. Usar un estándar para la organización
5. Establecer un vocabulario corporativo común

Una vista representa una estructura y contiene por lo habitual un diagrama, además de información adicional que facilita la comprensión de este. Existen diversas vistas arquitecturales según (Bianchi, 2008):

- **Later kruchten:**
  - **Vista lógica:** Muestra los requerimientos funcionales del sistema. La abstracción del sistema en esta vista son los objetos o clases.

  - **Vista del proceso:** plasma, concurrencia y distribución, integridad del sistema y tolerancia a fallas.

- **Vista de desarrollo:** se enfoca en los módulos de software en el ambiente de desarrollo de este, es decir, provee la asignación de requerimientos y trabajo a los equipos e incluye la evaluación de costos, planificación, monitoreo del progreso del proyecto y razonamiento acerca de la reusabilidad, portabilidad y seguridad.

- **Vista física:** tiene en cuenta requerimientos no funcionales del sistema, tales como disponibilidad, fiabilidad, desempeño y escalabilidad. Esta vista hace mención en forma de mapa de los elementos de las vistas anteriores dentro del hardware asignado para el funcionamiento del sistema.

- **Vista de asignación:** describe la asignación o mapeo de unidades de software del sistema con respecto a los elementos del medio ambiente (hardware, sistemas de archivos, equipo de desarrollo).

- **Vista de módulos: plasma** la estructura modular de la arquitectura de software de un sistema mostrando sus capas y el uso según su relación con otros módulos.

Para que el proyecto obtenga un resultado satisfactorio debe ser evaluada la arquitectura, ya que juega un valor crucial en el desarrollo que puede evitar un costo elevado antes de ser implementada para realizar la codificación.

3.2 Fundamentos para el desarrollo del software

(Sommerville, Galipienso, & Martinez, 2005) mencionan que desarrollar software implica agrupar varias disciplinas relacionadas con la construcción y arquitectura de software, entre otros conceptos que están relacionados con la producción, mantenimiento y

reingeniería de software. El IEEE establece que la ingeniería de software es una disciplina formada por el conjunto de técnicas y prácticas que por medio de métodos y herramientas se utilizan para crear sistemas informáticos.

También indica que un compromiso organizacional con la calidad, refiriéndose a calidad, como la administración del proceso de construir un producto o sistema que cubra las necesidades de los usuarios, probarlo, desplegarlo en ambiente productivo y hacerlo evolucionar con futuros mantenimientos, por otra parte (Pressman, 2022) menciona que además de ser una diciplina la ingeniería de software es una tecnología con diversas capas como:

a) **Herramientas**: Proporcionan un apoyo automatizado para la generación de procesos, diagramas y métodos.

b) **Métodos**: Aportan la experiencia técnica en el proyecto, ya que incluyen un conjunto amplio de tareas, como lo son el análisis de requerimientos, modelación de diseño, construcción de programa y pruebas.

c) **Proceso**: La capa de proceso es el fundamento para la ingeniería de software, es la unión de las demás capas, es la base de la administración de proyectos de software que define los métodos técnicos a utilizar para generar productos de trabajo donde se busca siempre entregar el producto de software en forma oportuna y con calidad para satisfacer los requerimientos de los usuarios. Existen cinco actividades para una estructura de proceso general:

    1. **Comunicación**: siempre una buena comunicación entre el colaborador y el cliente ayuda a definir las características y funciones del software.

    2. **Planeación**: siendo la parte más complicada, puesto que se debe medir en tiempo y plasmar en un plan de proyecto de software todas aquellas a actividades técnicas a realizar por parte del equipo, considerando los posibles riesgos,

obteniendo así una programación de actividades para cada integrante del equipo de trabajo.

3. **Modelado**: Es la diagramación de los requerimientos presentados por el cliente con el fin de entenderlos de mejor manera.

4. **Construcción**: esta actividad es la generación del código y de las pruebas que se requieren para verificar su calidad.

5. **Despliegue**: se hace entrega del software al cliente para hacer una evaluación de este.

En el contexto de ingeniería de software, un proceso no es como tal una receta rígida sobre cómo se debe construir un producto de software. Más bien, es un enfoque adaptable que permite al equipo de trabajo elegir un conjunto apropiado de acciones con la intención de entregar en tiempo y forma el producto de software.

d) **Calidad del software**: Un aspecto que considerar de la calidad de software consiste en llevar adelante el proceso de la mejor forma que permita al proyecto asociado terminar en el tiempo planificado y dentro del presupuesto asignado. Un desafío para la calidad de software es la velocidad de desarrollo, calidad y velocidad no están reñidos, es decir para tener un producto de calidad se necesita tiempo al no tenerlo se hace presente una mala calidad de producto, esto se puede resolver con la aplicación de metodologías ágiles que se describen más adelante.

   Algunos factores para medir la calidad de software son los siguientes:

   1. Corrección: Es el grado en que un producto software cumple con sus especificaciones funcionales.

2. Fiabilidad**:** Es el grado de ausencia de fallos durante la operación del producto software.

3. Eficiencia**:** Es la relación entre la cantidad de resultados suministrados y los recursos requeridos durante la operación.

4. Seguridad**:** Es la dificultad para el acceso a los datos o a las operaciones por parte de personal no autorizado.

5. Facilidad De Uso: Es la inversa del esfuerzo requerido para aprender a usar un producto software y utilizarlo adecuadamente.

6. Mantenibilidad: Es la facilidad para corregir el producto en casos futuros.

El ciclo de vida del software es parte fundamental de la ingeniería de software, ya que es el proceso, tareas y actividades relacionas con el producto de software.

3.3 Reingeniería de Software

La reingeniería de software analiza los sistemas heredados o legados con la finalidad de reconstruirlos y publicarlos en ambientes productivos o bien realizar mantenimiento para evolucionar estos sistemas, reestructurando o migrando a nuevas tecnologías para obtener un alto grado de calidad en cuanto la ingeniería de software se refiere, la mayoría de las veces cuando se aplica cuando se detecta algún problema en producción o se quiere extender el software (AVELLA IBÁÑEZ, 2004).

Existen casos donde el software no ha sido documentado y solo se tiene el código donde en ocasiones no se encuentra documentado, en estos casos es donde se aplica la ingeniería inversa o reingeniería.

3.3.1 Fases de la reingeniería de software

Como tal, no existe un proceso a seguir en la reingeniería de software, pero hay algunos autores que definen buenas prácticas o series de pasos que pueden tomarse en cuenta tal como lo indica (Álvarez García, Mateos Sánchez, & Moreno García, 2004).

1. Primer paso: comprender el software existente, para eso se recomienda recuperar el diseño del sistema desde su código fuente, es muy importante que al estar en este paso se analice de forma detallada el código podríamos considerar también la base de datos como análisis para la obtención del diseño.

2. Segundo paso: entran todas las actividades que ayudarán a la transformación del software provocando que tenga una mejora, entre ellas está la descomposición, reestructuración, remodularización y redocumentación.

Los autores (Álvarez García, Mateos Sánchez, & Moreno García, 2004) en su informe detallan el proceso de reingeniería donde se podría distinguir las siguientes fases, de igual forma solo hacen una propuesta para el proceso:

a) Traducción del código fuente: migración de código fuente para mejor entendimiento del código y evitar la falta de soporte en algún futuro, basta con conocer la estructura, las equivalencias de las estructuras del código, es decir, entender bien al menos un lenguaje.

b) La ingeniería inversa: consiste en tomar el código fuente y con el tratar de construir la documentación de diseño o arquitectura para así poder entender que módulos se deben modificar en caso de un mantenimiento. Para llevar a cabo la ingeniería inversa se sigue el proceso mostrado en la **Véase Figura 3.Proceso de ingeniería inversa. Autor: .**

c) Mejora de la estructura del programa: hacer que sea más fácil de entender para su mantenimiento, estructurándolo de una mejor forma.

d) Modularización del programa: la creación de componentes como módulos para su reutilización generando componentes genéricos.

e) Reingeniería de datos: hacer más comprensible los valores de un sistema analizando y estructurando el mismo.

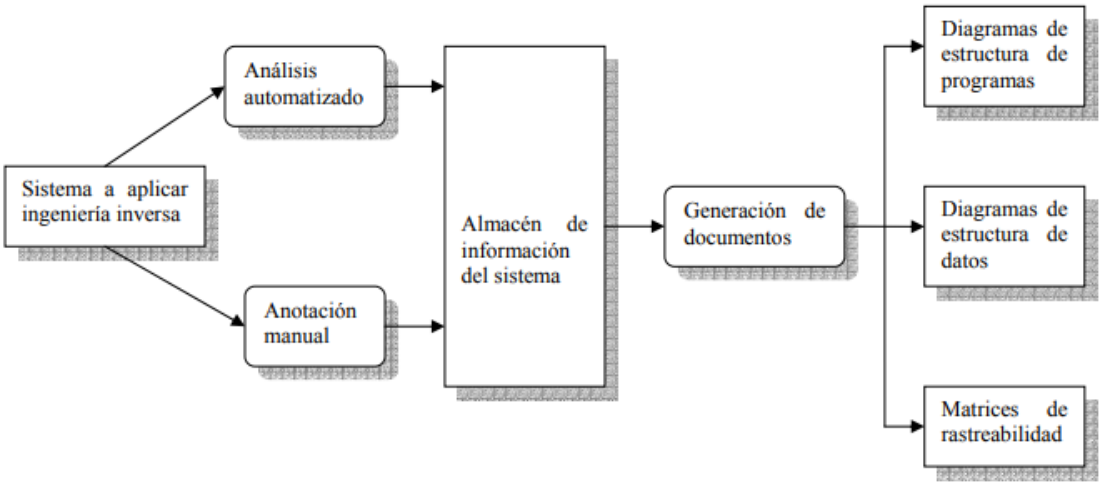


Figura 3.Proceso de ingeniería inversa. Autor: (Álvarez García, Mateos Sánchez, & Moreno García, 2004)

3.3.2 Métodos y Modelo de la Reingeniería de Software

El método y los modelos para llevar a cabo de manera satisfactoria la ingeniería inversa o la reingeniería son los siguientes (Atlantic Universal University, 2022):

- **Método de análisis de opciones para la reingeniería OAR (Options Analysis for Reengineering)**

  Es un método sistemático de toma de decisiones para la identificación y extracción de componentes dentro de grandes y complejos sistemas de software, los cuales son potencialmente relevantes y analiza los cambios necesarios para ser utilizados en nuevas arquitecturas.

Para poder entender el contexto de extracción se debe entrevistar a los creadores de código o en caso de no pertenecer al mismo equipo de trabajo se debe realizar un análisis del requerimiento para poder fijar metas definiendo así los componentes a extraer.

Una vez definido el contexto de la extracción se identifica los componentes heredados que se reutilizaran, generando así un inventario de componentes heredados. Analizar el componente de inventarios es un filtro necesario para determinar si se realizara alguna modificación a los componentes con el fin de cubrir las necesidades del requerimiento. En caso de encontrar modificaciones a realizar se debe medir el costo, riesgo y esfuerzo, determinando así si es viable reutilizar el componente o crearlo desde cero.

- **Modelo de Herradura**

  Se basa en tres procesos básicos:

  - **Análisis de un sistema existente:** En esta actividad se obtiene la arquitectura por medio de extracción de artefactos desde código fuente, donde se evalúa la seguridad, rendimiento y edificabilidad.

  - **Transformación lógica:** Una vez realizado el proceso anterior se aplica reingeniería a la arquitectura obtenida con el objetivo de evaluar si cumple con las metas y la calidad del sistema.

  - **Desarrollo de un nuevo sistema:** El alto proceso a seguir es presentar y aplicar los componentes seleccionados en los previos procesos para poder generar el nuevo sistema o su respectiva versión.

- **Modelo cíclico**

  El modelo cíclico tiene definidas seis actividades a realizar:

- **Análisis de inventario:** Las organizaciones de software deben inventariar todas sus aplicaciones activas donde especifiquen tamaño, edad e importancia del software en el negocio.

- **Reestructuración de documentos:** Documentar es una gran inversión hablando de costo y esfuerzo, sin embargo, si el sistema está relativamente estático, es decir, ya no ha sufrido cambios en los últimos años, puede ser índice de que su tiempo de vida útil está llegando a su fin, de lo contrario podemos realizar cambios en la documentación en caso de haber realizado algún mantenimiento.

- **Ingeniería inversa:** Siendo el proceso de análisis donde se extraerá la información existente del diseño arquitectónico y de proceso, e información de datos.

- **Reconstrucción de código:** Se analiza el código heredado para verificar si la arquitectura del software es sólida y fácil de entender, en caso contrario se modifica el código para estructurarlo correctamente y se actualiza la documentación.

- **Restructuración de datos:** Tiene que ver con la viabilidad a largo plazo del software, comenzando por una ingeniería inversa para comprobar los efectos de calidad de las estructuras de datos, siendo la parte más importante, ya que afecta a la arquitectura, o bien al código.

- **Ingeniería inversa**: Se le denomina renovación o reclamación donde su principal objetivo es recuperar la información de diseño del software ya existente para mejorar su calidad global.

3.4 Ingeniería de Dominio

Disciplina que se centra en el desarrollo de sistemas de software especializados para dominios específicos. Se basa en la idea de que muchos problemas y requisitos en un campo particular son comunes entre diferentes aplicaciones dentro de ese dominio.

La ingeniería de dominio tiene como objetivo aprovechar el conocimiento y la experiencia existente en un dominio particular para crear soluciones eficientes y reutilizables. En lugar de desarrollar sistemas desde cero para cada aplicación, se busca identificar los elementos comunes en el dominio y construir un conjunto de componentes y modelos reutilizables.

Siendo el acto de analizar y determinar qué es lo que se quiere hacer de una forma estructurada a partir de otros programas existentes, es decir, se analiza la forma de la forma de operación de software productivos para resolver requerimientos. (Red de Universidades con Carreras en Informática (RedUNCI) , 2022).

El objetivo de la ingeniería de dominio es identificar, construir y catalogar el conjunto de componentes que puedan estandarizarse, es decir, que puedan reutilizarse. (Castillo & Anaya, 2013).

La ingeniería de dominio tiene diversas fases:

- **Análisis:** Fase donde se realiza la recopilación de información, siendo posible el desarrollo de requisitos y arquitecturas configurables, es de gran importancia, ya que se considera la reutilización de código desde el inicio, dando oportunidad para aplicarse adecuadamente durante el desarrollo. Se tiene como objetivo que el desarrollador amplíe su conocimiento del dominio más allá de lo que conoce y categorizar similitudes y diferencias de la documentación, manuales y requisitos de otros proyectos (Reinhartz-Berger, Sturm, Clark, Cohen, & Bettin, 2013).

- **Diseño:** Tiene como objetivo producir una arquitectura genérica a la que todos los sistemas existentes se pueden acoplar. Resolviendo así problemas en común (Reinhartz-Berger, Sturm, Clark, Cohen, & Bettin, 2013) determina las reglas y configuración que deben llevar a cabo en el proyecto utilizando un patrón de diseño (Álvarez García, Mateos Sánchez, & Moreno García, 2004).

- **Implementación:** Es la creación de un proceso para generar de manera eficiente un proyecto (Reinhartz-Berger, Sturm, Clark, Cohen, & Bettin, 2013). Se concreta los lenguajes de programación para programar las clases definidas en el diseño de dominio.

- **Aplicación:** Una vez realizado las tres fases principales se realiza un ejemplo para presentar el diseño y codificación para ser aprobado (Álvarez García, Mateos Sánchez, & Moreno García, 2004).

Cuando el equipo determina las funcionalidades de otros programas pueden satisfacer el requerimiento del cliente, se plantea la creación y adaptación de los fragmentos de código, puede parecer que el desarrollador genera una plantilla de programación y lo reutilizara en todos los futuros proyectos, pero es completamente erróneo es un proceso que va más allá de copiar código o utilizar plantillas es enfocado a conocer y dominar las herramientas y algoritmos para resolver determinado problema. Una ventaja de esto es que incluyen un tiempo de producción más rápido y con menos errores, dando una mejor impresión a los clientes.

3.5 Metodología Scrum

Con el fin de cubrir las necesidades del departamento de desarrollo del ITZ según sus prioridades y poder realizar cambios en el momento que lo requiera en alguno de los proyectos que tienen a cargo, se propone la metodología SCRUM, proceso de dicha

metodología se realiza de forma iterativa e incremental, donde la iteración es denominada sprint la cual tiene una duración aproximada entre dos y cuatro semanas permitiendo entregar al cliente versiones de software funcional con nuevas mejoras según se avance en el desarrollo de cada sprint (Scrum Guia, 2022).

Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. El control de procesos empírico se rige por tres pilares:

- **Transparencia:** Los que aceptan el trabajo como los que lo desempeñan deben compartir una definición común de "terminado".

- **Inspección:** El equipo debe detectar variaciones en el proyecto que no lo desvíen del objetivo.

- **Adaptación:** Si al inspeccionar la metodología se determina que el producto final no será el aceptable, se deberá ajustar el proceso para evitar la desviación del objetivo.

Las principales razones para aplicar Scrum en el proyecto son:

- Las características del proyecto permiten desarrollar una base funcional mínima y sobre ella ir incrementando las funcionalidades o modificando el comportamiento o apariencia de las ya implementadas.

- Entregas frecuentes y continuas al cliente de los Sprints terminados, de forma que puede disponer de una funcionalidad básica en un tiempo mínimo y a partir de ahí un incremento y mejora continua del sistema.

- Previsible inestabilidad de requisitos.
    - Es posible que el sistema incorpore más funcionalidades de las inicialmente identificadas.

- Es posible que durante la ejecución del proyecto se altere el orden en el que se desean recibir los Sprints.

3.6.1 Características de Scrum

Scrum es identificado en el área de software por tener características peculiares para llevar a cabo la metodología (Scrum Guia, 2022)**.**

En Scrum, el trabajo se divide en ciclos llamados "Sprints", que suelen tener una duración de entre dos y cuatro semanas. Durante cada sprint, el equipo trabaja para completar un conjunto de tareas definidas al comienzo del sprint en una reunión llamada "planificación del sprint". Después de la planificación, el equipo se reúne diariamente en una reunión llamada "reunión diaria de Scrum" para discutir el progreso y cualquier problema o bloqueo que puedan estar experimentando.

Al final de cada sprint, el equipo realiza una revisión del sprint para discutir lo que se logró durante el sprint y cualquier problema que surgiera. También realizan una retrospectiva del sprint para evaluar cómo pueden mejorar su proceso de trabajo en el futuro (Scrum Guia, 2022).

- **Product Backlog**: Conjunto de requisitos denominados historias descritas en un lenguaje no técnico y priorizados por valor de negocio, o lo que es lo mismo, por retorno de inversión, considerando su beneficio y coste. Los requisitos y prioridades se revisan y ajustan durante el curso del proyecto a intervalos regulares.

- **Sprint Planning**: Reunión durante la cual se presenta las historias del backlog por orden de prioridad, junto con el equipo se determinan fechas de entrega de cada sprint.

- **Sprint:** Iteración de duración prefijada durante la cual el equipo trabaja para convertir las historias del Product Backlog a las que se ha comprometido, en una nueva versión del software totalmente operativo.

- **Sprint Backlog**: Lista de las actividades necesarias para llevar a cabo el sprint.

- **Daily sprint meeting:** Reunión diaria de como máximo 15 min. En la que el equipo se sincroniza para trabajar de forma coordinada. Cada miembro comenta que hizo el día anterior, que hará hoy y si hay impedimentos.

- **Demo y retrospectiva**: Reunión que se celebra al final del sprint y en la que el equipo presenta las historias conseguidas mediante una demostración del producto. Posteriormente, en la retrospectiva, el equipo analiza qué se hizo bien, qué procesos serían mejorables y discute acerca de cómo perfeccionarlos.

- **Roles de Usuario**: Scrum se basa en una serie de roles, incluyendo el "Scrum Máster" (responsable de garantizar que el equipo siga el proceso de Scrum), el "Product Owner" (responsable de definir los requisitos del producto y priorizar el trabajo) y el equipo de desarrollo o Development Team (responsable de crear el producto) pueden existir más actores involucrados en un proyecto de scrum, sin embargo, los tres anteriores son los más representativos (Frank Turley, 23 de octubre 2019).

En resumen, Scrum es un marco de trabajo ágil que se centra en la entrega de valor a través de iteraciones cortas y enfatiza la colaboración y la transparencia.

3.7 Capacidad de Maduración de Modelo de Integración (CMMI-DEV)

CMMI (Capability Maturity Model Integration) es un marco de referencia de mejora de procesos de software que ayuda a las organizaciones a mejorar continuamente sus procesos de desarrollo y gestión de software (Baldonedo, 2017). CMMI fue desarrollado por el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon, y proporciona un conjunto de mejores prácticas para la gestión de proyectos, la ingeniería de software y la gestión de procesos de negocio.

El modelo CMMI-DEV proporciona una serie de prácticas recomendadas que ayudan a las organizaciones a mejorar la calidad de sus productos de software y a gestionar mejor sus proyectos, van desde el nivel 1 (inicial) hasta el nivel 5 (optimizado). Cada nivel representa un grado de madurez en los procesos de la organización, y cada nivel se compone de un conjunto de áreas de proceso, que son conjuntos de prácticas que abordan un área específica de la gestión de procesos (Baldonedo, 2017):

- Nivel 1 - Inicial: Las organizaciones que se encuentran en este nivel tienen procesos ad-hoc, no tienen un enfoque sistemático y no pueden garantizar que los procesos se ejecuten de manera consistente. No hay capacidad para repetir éxitos pasados, y cualquier éxito se debe principalmente a la habilidad de las personas individuales que trabajan en la organización.

- Nivel 2 - Gestionado: En este nivel, las organizaciones han establecido procesos básicos y repetitivos que se utilizan para administrar proyectos. La gestión del proyecto se lleva a cabo de manera consistente, aunque aún puede haber variaciones en la forma en que se ejecutan los procesos. Las organizaciones a este nivel pueden repetir éxitos pasados.

- Nivel 3 - Definido: Las organizaciones que han alcanzado este nivel tienen procesos definidos para la gestión de proyectos y la ingeniería de software. Los procesos están documentados, estandarizados y se llevan a cabo de manera predecible. Las organizaciones a este nivel pueden adaptarse a las necesidades cambiantes del negocio mientras mantienen la calidad y la eficiencia de sus procesos.

- Nivel 4 - Cuantitativamente Gestionado: Las organizaciones que han alcanzado este nivel utilizan mediciones y análisis para controlar y mejorar continuamente sus procesos. Se establecen objetivos de calidad específicos y se miden regularmente para asegurarse de que se cumplan. Las organizaciones a este nivel tienen la capacidad de prever y prevenir problemas antes de que ocurran.

- Nivel 5 - Optimizado: En este nivel, las organizaciones han alcanzado la madurez máxima en cuanto a la gestión de procesos. Tienen una cultura de mejora continua y la innovación está integrada en sus procesos. Las organizaciones a este nivel se centran en la mejora constante de la calidad y la eficiencia de sus procesos y productos.

El objetivo del modelo CMMI es ayudar a las organizaciones a mejorar sus procesos y aumentar la calidad de su trabajo, así como a reducir los costos y el tiempo de desarrollo, la gestión de cambios y la gestión de riesgos, que son procesos críticos para el éxito de cualquier proyecto de desarrollo de software, apegado a la teoría que nos indica CMMI para generar un sistema maduro el cual dará resultado sistema de calidad (Baldonedo, 2017):

Gestión de requisitos:

- **Identificación de requisitos:** Este proceso consiste en identificar los requisitos del proyecto, estableciendo un proceso para la captura y documentación de los mismos.

- **Análisis de requisitos:** Este proceso implica el análisis y la verificación de los requisitos identificados para asegurar que sean claros, completos y coherentes.

- **Gestión de cambios en requisitos:** Este proceso implica establecer un proceso formal para la gestión de cambios en los requisitos identificados. Se deben definir las responsabilidades de las partes involucradas en la gestión de cambios, el proceso de aprobación de cambios y cómo se realizará la comunicación de los cambios a las partes interesadas.

Gestión de cambios:

- **Identificación de cambios:** Este proceso implica la identificación y documentación de los cambios solicitados por los interesados en el proyecto.

- **Análisis de cambios:** Este proceso implica el análisis y evaluación de los cambios solicitados para determinar su impacto en el proyecto.

- **Aprobación y control de cambios:** Este proceso implica establecer un proceso formal para la aprobación y control de los cambios solicitados. Se deben definir las responsabilidades de las partes involucradas en la gestión de cambios, el proceso de aprobación de cambios y cómo se realizará la comunicación de los cambios a las partes interesadas.

Gestión de riesgos:

- **Identificación de riesgos:** Este proceso implica la identificación de los riesgos potenciales que puedan afectar el proyecto.

- **Análisis de riesgos:** Este proceso implica el análisis y evaluación de los riesgos identificados para determinar su impacto en el proyecto.

- **Gestión de riesgos:** Este proceso implica establecer un proceso formal para la gestión de los riesgos identificados. Se deben definir las responsabilidades de las partes involucradas en la gestión de riesgos, el proceso de evaluación y priorización de los riesgos, el proceso de seguimiento y control de los riesgos y cómo se realizará la comunicación de los riesgos a las partes interesadas.

Al seguir las prácticas recomendadas por CMMI, las organizaciones pueden mejorar la eficiencia, la eficacia y la calidad de sus procesos, y obtener una mayor satisfacción del cliente. El modelo CMMI describe aquello que la organización debería hacer, pero no el cómo lo debería hacer, la organización en la que se implementa debe ser quien establezca como implementar las prácticas propuestas (Baldonedo, 2017).

CAPÍTULO IV. METODOLOGÍA

En este capítulo se detalla la metodología propuesta para el caso de estudio conforme a sus necesidades, nombrando roles, etapas de actividades y generando formatos de guía a manera de herramienta para dar seguimiento a las actividades nombradas en el modelo.

El capítulo también describe el proceso que se lleva a cabo en el caso de estudio, el cual se tomará para poder aplicar la metodología, especificando el proceso de trabajo que se lleva a cabo actualmente.

4.1 Referencias del modelo propuesto

El modelo propuesto se desarrolló con base en la documentación e investigación del fundamento teórico presentado en el capítulo anterior, donde se obtuvieron técnicas de trabajo y procedimientos utilizados en las diversas metodologías y modelos, de los cuales la mayoría se ha tenido o vivido una experiencia de trabajo determinando que gran parte de metodologías pueden resultar un poco tediosas en el momento de su aplicación ya que se genera documentación excesiva la cual afecta los tiempos de desarrollo o bien no se genera ningún antecedente, pero en ningún momento es equilibrada. Partiendo de eso se propone un modelo de mejora que cubra las necesidades observadas y analizadas en conjunto con el área de desarrollo, la cual tiene a cargo diversos proyectos que tienen relación con sistemas legados.

Para formular una propuesta se analizó las ventajas de diversos modelos o metodologías para extraer las mejores técnicas para el desarrollo por ejemplo de SCRUM siendo una metodología ágil la cual ha comprobado que al dividir el problema a resolver en fragmentos pequeños llamados Sprints se puede generar una mayor avance en el desarrollo, evitando la saturación de trabajo, además de adaptarse muy bien al iniciar un proyecto y en equipos de trabajo pequeños, sin embargo, como todas las metodologías

tiene desventajas las cuales sirvieron para fortalecer el modelo propuesta en el caso de SCRUM su principal desventaja es que no promete generar una documentación estructurada provocando un problema en la capacitación de usuarios o desarrolladores, puesto que solo fomenta la documentación en el código. Problema por el cual el ITZ se encuentra atravesando, ya que contiene varios sistemas heredados o legados, los cuales no contienen documentación necesaria para llevar a cabo un mantenimiento.

En la propuesta se realiza el llenado de formatos breves, los cuales obtienen datos principales para la documentación del sistema, facilitando la gestión del proyecto, sugiere la división del requerimiento en partes pequeñas para poder realizar un desarrollo ágil pero documentado.

Otro modelo analizado fue el modelo de Herradura, el cual es conocido como modelo en forma de U, se utiliza en diferentes contextos y disciplinas, como la gestión de proyectos, el liderazgo, la psicología y la toma de decisiones. En nuestro caso, promueve la identificación de las necesidades de un requerimiento de software de diversas perspectivas desde el usuario, el negocio y la tecnología, siendo su principal característica la evaluación de riesgos y obstáculos por medio de la exploración de los componentes de software, desarrollando una planificación bien estructurada. De acuerdo a lo investigado, documentar procesos o métodos ayuda a la reutilización de código, y a su vez, proporciona agilización en el desarrollo de código.

Ahora bien, documentar y planificar de manera exhaustiva puede generar retrasos, los cuales afectan al desarrollo. Por tal motivo, el modelo propuesto genera herramientas de trabajo para el equipo de desarrollo, facilitando la forma de documentar el proyecto sin demorar tiempo y esfuerzo, nivelando un proceso ágil con uno planificado y ordenado.

Los ciclos de desarrollo cortos permiten la entrega temprana y continua del producto, tomando como referencia el modelo cíclico el cual tiene el proceso de desarrollo, la

división en ciclos iterativos de análisis, diseño, implementación y prueba, además de promover la documentación detallada de cada ciclo y de los cambios realizados en el sistema. Al adaptar el enfoque del modelo cíclico podremos proporcionar una mayor flexibilidad y adaptabilidad al proyecto, permitiendo que el equipo de desarrollo se enfoque tanto en la entrega temprana del producto como en la mejora continua del mismo. Destacando que debe considerarse una buena planeación para que el proyecto no genere tiempos sobreestimados, gestión la cual podemos obtener con la técnica proporcionada por el modelo de herradura, siendo la evaluación de riesgos y la planificación bien estructurada.

Al unir las técnicas del fundamento teórico analizado, podemos observar que tenemos dos de las tres gestiones que propone el modelo de maduración CMMI, el cual promueve la mejora de procesos y aumenta la calidad de su trabajo por medio de la gestión del proyecto divida en tres etapas, gestión de requisitos que forma parte de la primera etapa del modelo puesto, la gestión de cambios que es más implícita en modelo propuesto, ya que es después de la entrega del proyecto, por último la gestión de riesgos la cual ya se obtuvo como buena técnica del modelo de herradura, sabemos que para poder llegar a un grado de maduración en un proyecto se deben llevar a cabo estas gestiones por un tiempo determinado, si desde un inicio del proyecto de reingeniería o ingeniería de software se aplican las técnicas propuestas por CMMI obtendremos resultados favorables a futuro y un mejor renombre o experiencia para el equipo de desarrollo, con esa visión se proponen diversos formatos para la documentación del proyecto cumpliendo con la gestión indicada en CMMI.

En la parte de desarrollo se propone aplicar las técnicas propuestas por la ingeniería de dominio y la metodología OAR las cuales son combinadas en ocasiones, ya que la ingeniería de dominio se enfoca en la identificación y análisis de conceptos clave de un dominio específico es decir reglas de negocio para poder generar modelos, patrones y componentes reutilizables para un dominio especifico. Ahora bien la metodología OAR proporciona un enfoque sistemático para identificar y modelar los objetos, acciones y

responsabilidades en un sistema, es decir, genera un listado de los principales módulos definiendo sus atributos y relaciones, identificando la responsabilidad de cada objeto en el sistema con el objetivo de reutilizarlo, por esa razón una etapa de abstracción es importante para que todo el equipo tenga conocimiento de como funcionara el sistema y poder así llevar a cabo esta técnica de forma correcta.

4.1.1 Roles

La identificación de roles es muy importante para desarrollar cada mantenimiento o requerimiento nuevo, ayuda a tener claras las actividades que realizara cada individuo, por esta razón se propone un formato de trabajo en el cual detalla quienes son los involucrados en el proyecto **Véase Tabla 2. Roles para el modelo propuesto** Automáticamente, pasan a ser miembros del equipo de trabajo, es decir, el usuario entrevistado y el desarrollador pertenecen al mismo equipo, siendo una técnica utilizada por la metodología SCRUM con la finalidad de crear una comunicación constante para aclaración de dudas que puedan surgir en el desarrollo.

Rol	Abreviatura	Descripción
Product Owner (dueño del producto)	PO	Responsable de definir los requisitos del producto y priorizar el trabajo
Project Administrator (Administrador de proyecto)	PA	Líder de equipo de desarrollo, encargado de administrar las actividades del equipo de desarrollo, adema de fungir como arquitecto de proyecto
Development Team (Equipo de desarrollo)	Dev	Desarrolladores encargados de generar el código para el proyecto
Reviewer (Revisor)	R	Responsable de aprobar el funcionamiento del requerimiento puede ser la misma persona Product Owner o encargado del área a la cual se desarrolló el requerimiento
Director	Dir.	Responsable Coordinación de Desarrollo de Sistemas y Servicios de Cómputo.

Tabla 2. Roles para el modelo propuesto

4.1.2 Metodología de Desarrollo Estructurado y Evaluación (MDEE)

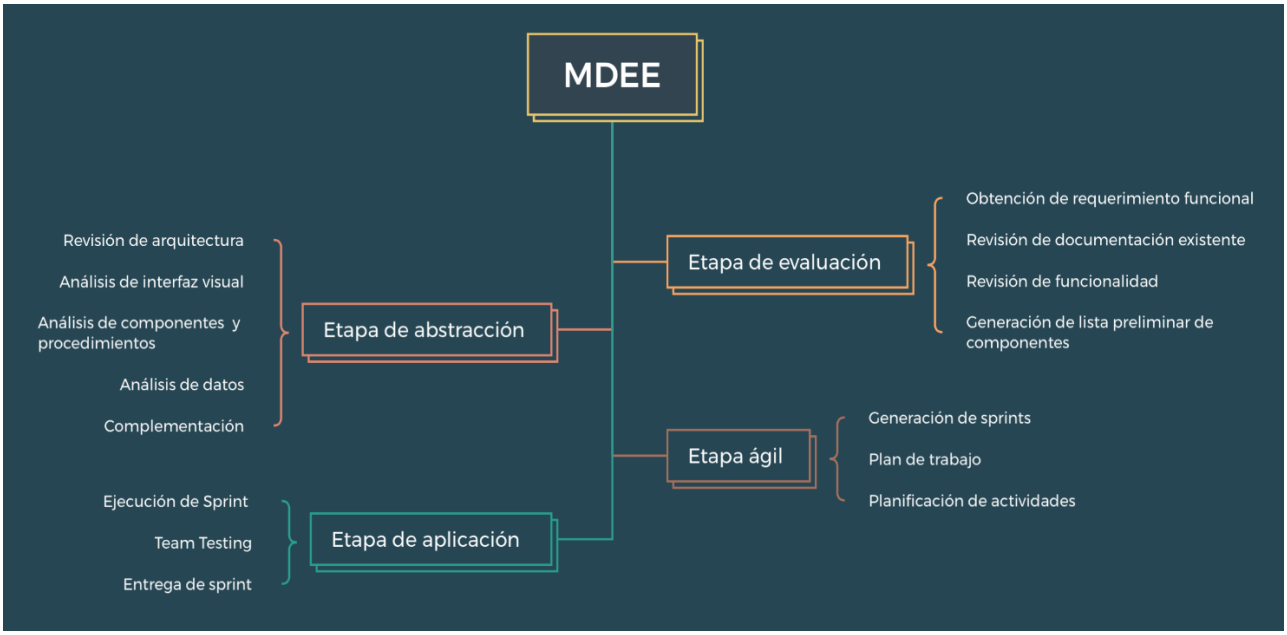


Figura 4. Metodología de Desarrollo Estructurado y Evaluación (MDEE) autor: contenido propio

La propuesta de desarrollo del modelo está comprendida en tres etapas y a su vez cada etapa contendrá una serie de actividades **Véase Figura 4. Metodología de Desarrollo Estructurado y Evaluación (MDEE**) autor: contenido propio:

4.1.2.1 Etapa de Evaluación

En la etapa de evaluación el sistema se somete a un análisis minucioso por parte del PO (Product Owner). Siendo uno de los objetivos principales de CMMI la gestión de requisitos, que a su vez nos ayuda a adquisición de componentes para posteriormente llevar a cabo la técnica de la metodología OAR y dar paso a la generación de la ingeniería de dominio donde obtendremos las reglas de negocio o bien del dominio específico. En esta etapa, se tienen las siguientes actividades:

- ***Obtención de requerimiento funcional***: Esta actividad se realiza por medio de entrevistas, reuniones o redacciones por parte del PO, especificando a detalle el

trabajo a realizar. Se propone dos herramientas **véase Tabla 3. Cuestionario para levantamiento de requerimiento del modelo propuesto y Tabla 4. Formato para la gestión del proyecto**, una manera de cuestionario como propuesta para realizar el levantamiento del requerimiento y la segunda a manera de bitácora con la finalidad de facilitar la documentación del requerimiento, dichas herramientas pueden ser modificadas según las necesidades.

Pregunta	Respuesta o comentario
¿Cuál es el objetivo principal del software?	
¿Qué problemas o necesidades específicas del usuario o el departamento espera que resuelva el software?	
¿Quiénes serán los usuarios finales?	
¿El sistema cuenta con un gestor de usuarios? Es decir, ¿maneja tipos de usuarios? ¿Es requerido?	
¿Cuál es el alcance del proyecto? Es decir, ¿Hay algún límite en cuanto a lo que el software debe hacer o no hacer?	
¿Qué plataformas se deben soportar? ¿Deben ser compatibles con varios sistemas operativos o dispositivos?	
¿En cuánto al requerimiento, que debe validar el software?	
¿Cómo se integrará el software con otros sistemas existentes?	
¿Qué nivel de mantenimiento y soporte se requiere después del lanzamiento del software? Es decir, ¿qué tan constante puede llegar a recibir cambios?	
¿Cuál es el plazo para la entrega del software?	
¿Tiene datos reales para realizar pruebas?	
¿Se requiere algún reporte?	
¿El sistema genera reportes o documentos?	

¿Cuáles son las principales debilidades y problemas del software actual?
¿Cuál es el alcance de la reingeniería del software?, es decir, el desarrollo tendrá nuevos módulos a los existentes
¿Cuáles son los principales requerimientos funcionales que se deben abordar en la reingeniería del software? Es decir, ¿Qué módulos son los principales a analizar?
¿Se requerirá la migración de datos del software actual a la nueva versión?
¿Qué tecnologías y herramientas se utilizan en el sistema (en caso de existir código)?
¿Tiene relación con otro sistema? Y si es sí, ¿cuál es ese sistema y qué funcionalidad tiene?

Tabla 3. Cuestionario para levantamiento de requerimiento del modelo propuesto

Datos del proyecto

Nombre del Proyecto:		a)	
Abreviatura		b)	
Encargado del proyecto:	c)		
Fase del Proyecto:	d)	**Fecha de Creación:**	g)
Documentado por:	e)		
Revisado por:	f)	**Fecha de Revisión:**	h)

Responsables

Nombre	Rol	Fecha	Correo Electrónico	Teléfono	Responsable de versión
i)	j)	k)	l)	m)	n)

Histórico

Nombre Requerimiento	o)

Objetivo	p)
Responsable	q)
Fecha de inicio	r)
Fecha fin	s)

Tabla 4. Formato para la gestión del proyecto

Indicador	Descripción
a	Nombre asignado para el proyecto
b	Abreviatura del proyecto, usar máximo 6 caracteres
c	Persona encargada de llevar a cabo la gestión del proyecto
e	Fase de proyecto: Nuevo proyecto, Nuevo requerimiento, Modificación de módulo, Proyecto detenido
f	Persona que autoriza cambios en documentos
g	Fecha de creación de documento y proyecto
h	Fecha de autorización del documento y proyecto
i	Persona perteneciente al equipo
j	Rol que efectuara la persona en el equipo, revisar tabla de roles adjunta (PO, PA, Dev, R, Dir)
k	Fecha de integración al equipo de la persona
l	Correo de contacto de la persona
m	Teléfono celular de la persona
n	Versión del proyecto en la que participo la persona
o	Nombre asignado para el requerimiento, se sugiere un nombre corto el cual contenga las inicialmente la abreviatura
p	Descripción breve del requerimiento
q	Responsable del desarrollo del requerimiento
r	Fecha de inicio del requerimiento
s	Fecha fin del requerimiento

- ***Revisión de documentación existente:*** Es importante revisar si existe documentación o no y evaluar en qué estado se encuentra, esto permite conocer más el sistema y generar una mejor estrategia para resolver el requerimiento funcional, para esto se propone una herramienta de evaluación **Véase Tabla 5. Formato para Evaluación de documentación,** la cual contiene un cuestionario

para determinar el estado en que se encuentra la documentación en caso de existir, generando así un panorama del estado del proyecto.

Núm.	Pregunta	X	Observación
1	¿Hay documentación disponible para el proyecto de software?	☐	
2	¿La documentación está organizada y estructurada de manera clara?, es decir, separa código de documentación a nivel carpetas	☐	
3	¿La documentación es fácil de leer y comprender?, maneja una estructura donde divide por secciones el tema a tratar	☐	
4	¿La documentación cubre todos los aspectos importantes del proyecto, incluyendo la arquitectura, el diseño y la implementación?	☐	
5	¿La documentación incluye ejemplos y casos de uso para ayudar a entender cómo funciona el proyecto?, estos se pueden encontrar en el manual de usuario	☐	
6	¿La documentación está actualizada con los últimos cambios en el proyecto?	☐	
7	¿La documentación incluye información sobre cómo instalar y configurar el software?, es decir, contiene algún manual de soporte	☐	
8	¿La documentación describe los requisitos del sistema necesarios para utilizar el software?	☐	
9	¿La documentación incluye información sobre cómo mantener y actualizar el software?	☐	
10	¿El software tiene comunicación con otro sistema? Escribir en el espacio de observaciones con qué sistemas tiene comunicación	☐	
11	¿El software maneja webservices o servicios rest?	☐	

12	¿Se tiene el código de webservices o servicios rest?	☐	
13	¿La documentación incluye información sobre cómo utilizar el código fuente del proyecto?, es decir, instalar ambientes de desarrollo para futuros mantenimientos	☐	
14	¿Utiliza algún repositorio de documentos? Indicar cuál es en el campo observaciones	☐	
15	¿Se tiene acceso al repositorio de archivos?	☐	
16	¿La documentación incluye información sobre las licencias y derechos de autor del software?	☐	
17	¿Se encuentra documentado el servicio rest o webservices que utiliza?	☐	
18	¿Existe información sobre la base de datos? Es decir, diagramas, entidad, relación o diccionario de datos	☐	
19	¿Se tiene el script o código SQL para generar la base de datos?	☐	
20	¿Existe en la institución un desarrollador con conocimientos del software? Es decir, miembro del equipo inicial de desarrolladores del software	☐	
Observaciones:			

Tabla 5. Formato para Evaluación de documentación

- ***Revisión de la funcionalidad***: al obtener el requerimiento funcional, analizar los módulos que son propensos a ser modificados con el fin de solo editar los que tengan relación con dicho requerimiento. Puesto a que, como se menciona en el fundamento teórico, un sistema legado se debe tratar con una buena estrategia más aún cuando este se encuentra en producción, se propone una herramienta a manera de cuestionario para evaluar la situación actual del sistema y poder

llevar a cabo el requerimiento **véase Tabla 6. Formato para evaluación del sistema.**

Preguntas	Respuesta	Observación
¿El software cumple con todas las funciones descritas en la documentación?	☐Si ☐No ☐En algunos casos	
¿El software se ejecuta sin errores?, es decir, muestra error en alguna función y qué tipo de error	☐Si ☐No ☐En algunos casos	
¿El software es fácil de instalar y configurar?, es decir, el proceso de instalación lo puede realizar quien sea	☐Si ☐No ☐En algunos casos	
¿El software es fácil de usar?, es decir, es intuitivo o fácilmente se encuentran las funciones prometidas	☐Si ☐No ☐En algunos casos	
¿El software es intuitivo y fácil de aprender? Es decir, se requiere alguna capacitación o con solo explicar su funcionamiento se puede usar	☐Si ☐No ☐En algunos casos	
¿El software es compatible con otros programas o sistemas operativos?	☐Si ☐No ☐En algunos casos	
¿El software cumple con los requisitos del sistema descritos en la documentación?	☐Si ☐No ☐En algunos casos	
¿El software es seguro y protege los datos del usuario? Es decir, maneja algún tipo de encriptación de datos	☐Si ☐No ☐En algunos casos	
¿El software funciona con rapidez y eficiencia?, funciona de manera ágil al consultar y realizar operaciones	☐Si ☐No ☐En algunos casos	
¿El software es escalable y puede manejar grandes cantidades de datos o usuarios?	☐Si ☐No ☐En algunos casos	
¿El software proporciona suficiente retroalimentación al usuario?, es decir, maneja información que ayude al usuario a saber qué hacer en la pantalla donde se encuentra	☐Si ☐No ☐En algunos casos	

¿El software incluye opciones de personalización para adaptarse a las necesidades del usuario?	☐Si ☐No ☐En algunos casos	
¿El software es confiable y estable?, tiene algún fallo anormal fuera de la funcionalidad que no permita realizar las actividades del usuario, por ejemplo, fallas de servidor o de red, fallas de conexión a base de datos	☐Si ☐No ☐En algunos casos	
¿El software incluye herramientas de generación de reportes para el usuario?, es decir, hace uso de APIs para la generación de reportes o documentos	☐Si ☐No ☐En algunos casos	
¿El software utiliza un login para acceder?	☐Si ☐No ☐En algunos casos	
¿El software incluye validaciones en campos? ¿Qué tipo de validaciones?	☐Si ☐No ☐En algunos casos	
¿Las validaciones que realiza se manifiestan en todos los campos donde se requieren? Por ejemplo, una validación de caracteres especiales en un campo de asignación de nombre de persona, en todos los campos donde se utiliza este campo se válida que no tenga caracteres especiales	☐Si ☐No ☐En algunos casos	
¿El software es compatible con dispositivos móviles y tabletas?	☐Si ☐No ☐En algunos casos	
¿El software incluye una ayuda contextual o una guía de usuario detallada?, es decir, maneja información de ayuda en los campos a llenar o hay algún manual de usuario	☐Si ☐No ☐En algunos casos	
¿El software puede integrarse con otras aplicaciones o servicios? Es decir, maneja servicios web o rest, y con qué sistemas tiene relación para llevar a cabo estos servicios	☐Si ☐No ☐En algunos casos	
¿El software es capaz de gestionar múltiples usuarios y roles de usuario?, es decir, tiene algún apartado donde gestione los usuarios	☐Si ☐No ☐En algunos casos	

¿El software incluye herramientas de respaldo y restauración de datos?	☐Si ☐No ☐En algunos casos	
¿El software incluye opciones de búsqueda y filtrado para facilitar la navegación del usuario?	☐Si ☐No ☐En algunos casos	
¿El software cumple con las expectativas del usuario y satisface sus necesidades?, existe alguna mejora que pueda facilitar las actividades que realiza el usuario que no esté contemplada	☐Si ☐No ☐En algunos casos	
¿El software ha sido probado rigurosamente para asegurar su calidad?, ha pasado por pruebas de testing en alguna ocasión, en caso de ser cierto, ¿se tiene los resultados?	☐Si ☐No ☐En algunos casos	

Tabla 6. Formato para evaluación del sistema

- ***Generación de una lista preliminar***: enlistar componentes y módulos que serán afectados. Esto puede ser con base a las entrevistas obtenidas o a la documentación existente e incluso puede ser generado en la obtención del requerimiento.

4.1.2.2 Etapa de abstracción

Aplicar ingeniería inversa, es decir, abstraer las reglas de negocio que maneja o intentaba manejar el sistema heredado y determinar sus propiedades o estructura a nivel código y arquitectura. La técnica de ingeniería de dominio también cobra fuerza, ya que podremos identificar las reglas de dominio específico. Los involucrados pueden revisar cada uno de los puntos para poder entender el funcionamiento del sistema o bien pueden seleccionar el punto en el que más tengan habilidad para desarrollar, siendo muy importante la unión del equipo de trabajo:

- ***Revisión de arquitectura:*** Revisar que si existe código y en caso de existir analizar su funcionalidad y compatibilidad con las tecnologías actuales como sistemas operativos, versiones de frameworks o librerías en caso de que las utilicen, en esta actividad es recomendable revisar el tiempo de soporte de cada tecnología

utilizada para en un futuro evitar un posible retrabajo o indicar el tiempo de vida que pude llegar a tener el sistema con la arquitectura actual y generar una arquitectura nueva como propuesta o bien reestructurar la actual de tal manera que pueda alargar el tiempo de vida del software. Es importante plasmar en diagramas UML donde se definen la interacción entre motores de base de datos, lenguajes de programación, frameworks, herramientas de mapeo objeto-relacional (ORM) y servidores que se pretendan utilizar o reutilizar según sea el caso siendo este el primer paso para la documentación del sistema legado. Se propone una herramienta para facilitar la documentación de la arquitectura **véase Tabla 7. Formato para a la generación de diagrama de arquitectura,** la cual es basada en modelo vista controlador, sin embargo, puede ser modificada según la arquitectura o el flujo de datos.

-

Diagrama de Arquitectura
Instrucciones: Diseñar a mano alzada la arquitectura del sistema , considerando: usuarios ,flujos y entidades: modelo vista controlador, según sea el caso

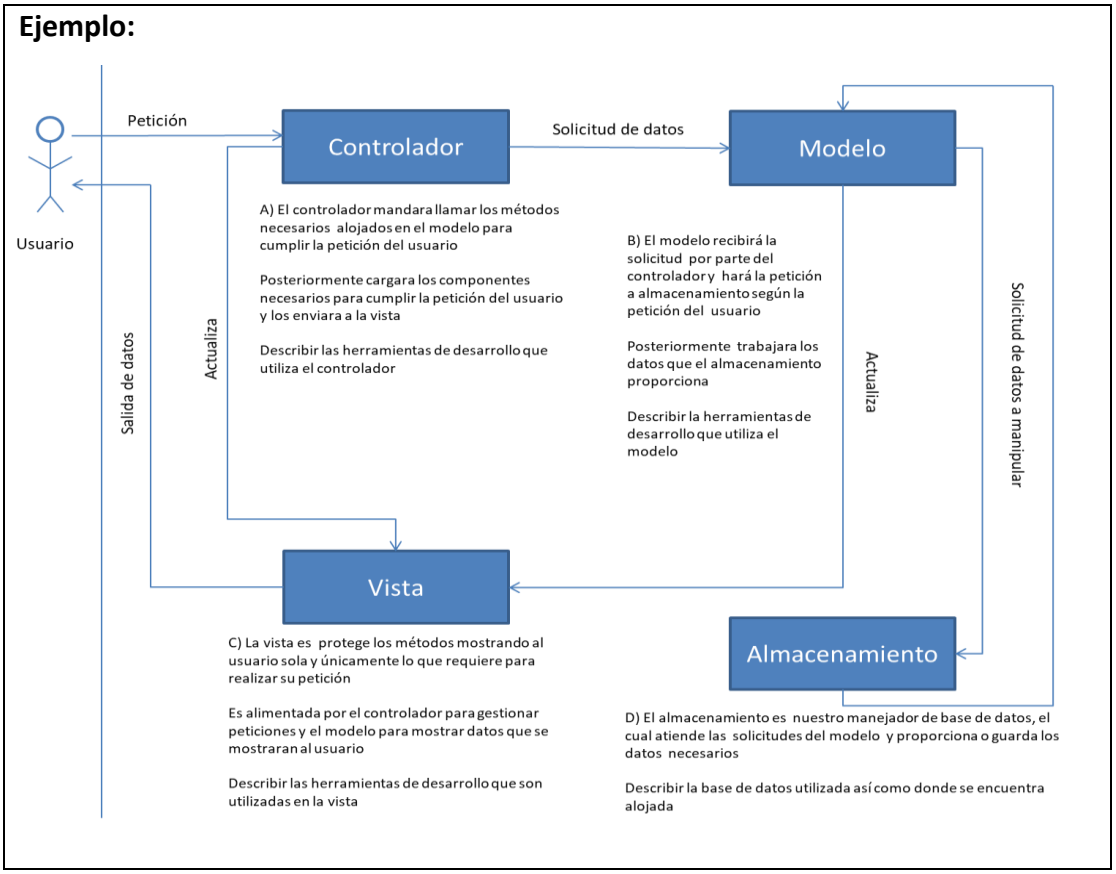


Tabla 7. Formato para a la generación de diagrama de arquitectura

- **Análisis de interfaz visual:** Se tendrán dos objetivos, siendo el primer una posible estrategia para hacer más amigable el entorno visual, además de facilitar el uso del módulo al usuario, como segundo objetivo identificar los componentes visuales que pueden llegar a ser genéricos como pueden ser botones, pantallas de alerta, grids, estilos, etc. Con el fin de documentarlos, esto podrá ser de gran ayuda en el momento de codificar, por ejemplo, tener una sola hoja de estilos donde facilitara el desarrollo, siendo una buena práctica la reutilización de código, ahora bien, si el sistema heredado no la contiene puede ser una buena propuesta de mejora.

- ***Analizar los componentes y procedimientos*** haciendo uso del código tener como objetivo entender la estructura del código y además detectar componentes genéricos o reutilizables, de esta forma se complementa la documentación,

además de facilitar el desarrollo de otros sistemas convirtiendo código en herramientas de trabajo. Agregar notas en el código en caso de no existir documentación es parte de documentar, siendo recomendación de scrum de esta forma facilitamos futuros mantenimientos y desarrollos.

- ***Análisis de datos:*** Entender el flujo de datos que existe en el almacenador de datos es parte importante del proceso para poder cumplir con el requerimiento solicitado, si en la documentación no existe un diagrama es recomendable generarlo para poder entender el flujo por el cual se envían los datos.

- ***Complementación:*** al término de esta etapa es importante el reforzamiento de conocimiento, esta actividad se recomienda unir al equipo para poder compartir los hallazgos encontrados en cada actividad, puede ser a manera de presentación o en forma de plática, muy impórtate que todos los involucrados participen con el fin de obtener la información posible sobre el funcionamiento legado.

4.1.2.3 Etapa Ágil

El PA generará un listado con los módulos detectados conforme lista de módulos generados de la etapa anterior para poder crear una planeación, percibiendo los posibles riesgos:

- ***Generación de Sprint:*** se dividirá las actividades a realizar según el tamaño del requerimiento, es decir, puede existir requerimientos muy simples que solo requieran un sprint, pero al contrario pueden existir requerimientos que son muy extensos como hacer funcionar un módulo completo ahí podrán salir uno o más Sprint, este deberá contener un nombre para su fácil identificación, un objetivo, una breve descripción de lo que se realizará y una fecha de entrega **Véase Tabla 8. Formato para generación de bitácora de cada Sprint.** En la generación de Sprint es importante tener en cuenta el listado de módulos y componentes

generado en las anteriores etapas, porque puede definir la cantidad de Sprint a realizar.

Nombre:	a)
Descripción	
b)	
Comprobable	
c)	
Entregable	
d)	
Actividades	
e)	

Tabla 8. Formato para generación de bitácora de cada Sprint

Instrucciones

a	Nombre del módulo principal, por ejemplo: Administrador de usuarios
b	Descripción la funcionalidad que debe entregarse en el sprint
c	Descripción de cómo se comprobará que la funcionalidad está terminable
d	Descripción de lo que se espera entregar en el sprint
e	Descripción de las actividades a realizar durante el sprint para poder alcanzar el entregable

Nombre del Módulo	Funcionalidad	Prioridad	Descripción	Última Modificación	Estado	Riesgo o contratiempo	Plan de acción	Documentación URL	Sprint	Duración Estimada	Duración Real	Repositorio	versión
a	b	c	e	f	g	h	i	j	k	l	m	n	o

Tabla 9. Formato para la generación de un plan de trabajo

INSTRUCCIONES	
A	Nombre del módulo principal, por ejemplo: Administrador de usuarios
B	Nombre de la funcionalidad perteneciente al módulo principal, por ejemplo, Alta, baja, consulta
C	Categoría de prioridad (Alta, Media, Baja)
E	Descripción breve de la funcionalidad
F	Fecha de la última modificación de la funcionalidad
G	Estado actual de la funcionalidad (Producción, Desarrollo, Pruebas, Cancelado)
H	Indicar si existe un riesgo o contra tiempo que pueda afectar la funcionalidad, por ejemplo, falta de información por parte del usuario, investigación sobre cómo desarrollar la funcionalidad
I	Plan de acción para resolver el riesgo o contra tiempo
J	Escribir la URL de donde se encuentra la documentación referente a la funcionalidad ejemplo (C:\Trabajo\Sistema\Doc\ Manual de soporte y administración)
K	Nombre del Sprint, es recomendable usar la abreviatura del proyecto y una numeración o letra del alfabeto, por ejemplo: numeración
L	Duración estimada de desarrollo del sprint en horas
M	Duración real de desarrollo del sprint
N	Escribir la URL de donde se encuentra la documentación referente al código, por ejemplo (C:\Trabajo\Sistema\ Modelo de construcción\sistema\src\com\itz\sistema\usuarios)
O	Escribir la versión del proyecto donde se encuentra esa funcionalidad

ACTIVIDAD	Mes	b)					b)				b)					b)			
	Semana	1	2	3	4	5	1	2	3	4	1	2	3	4	5	1	2	3	4
a)	Estimado	c)																	
	Real	d]																	
	Estimado																		
	Real																		
	Estimado																		
	Real																		
	Estimado																		
	Real																		
	Estimado																		
	Real																		
	Estimado																		
	Real																		
	Estimado																		
	Real																		
OBSERVACIONES	e)																		

Tabla 10. Formato para la generación de cronograma de actividades

Instrucciones	
a)	Nombre del Sprint , es recomendable usar la abreviatura del proyecto y una numeracion o letra del alfabeto por ejemplo : Sprint_SISE_A
b)	Nombre del mes que implicara el sprint
c)	Colorear celda para especificar la semana estimada , se propone un color para el estimado
d)	Colorear celda para especificar la cantiad de semanas reales que se tomo para realizar el sprint , se propone un color para el tiempo real
e)	Campo para describir alguna observacion a considerar o reflejada en el cronograma

- ***Plan de trabajo***: conforme a los módulos detectados categorizando por prioridades a elección del usuario se irán resolviendo los Sprints cada sprint debe tener menos de una semana de duración. **Véase Tabla 9. Formato para la generación de un plan de trabajo.**

- ***Calendario de actividades*** Generar un calendario el cual debe incluir las fechas de entrega de cada sprint y las reuniones para dar avances del proyecto, estas pueden ser a consideración del equipo, lo recomendable es generar al menos una cada 15 días con una duración de 30 min, reuniones extraoficiales pueden llevarse a cabo para resolver dudas durante el desarrollo, sin embargo, estas no deben durar más de una hora para no interrumpir en las actividades del equipo de esta forma solo se tratara el tema a consultar. **Véase Tabla 10. Formato para la generación de cronograma de actividades.**

4.1.2.4 Etapa de aplicación

Para cada sprint generado en la planeación aplicaremos la siguiente etapa en forma cíclica con el fin de avanzar a la entrega de los Sprints.

- ***Ejecución de Sprint***: en esta etapa se aplica la ingeniería directa en la cual se realiza la codificación del sprint en curso, por lo cual es importante que el desarrollador agregue notas al código que elabora, al menos poner una breve descripción antes de cada método. La codificación se debe estandarizar tanto en base de datos como en el código, es decir, determinar por medio de notación camello el nombrado de variables y métodos o procedimientos sin mencionar que deberán ser brevemente comentados o descritos antes de su declaración. Además de generar o reutilizar los procesos genéricos previamente analizados, teniendo como objetivo facilitar el mantenimiento y futuros desarrollos. Como propuesta de buenas prácticas, realizar el nombrado uniforme de carpetas que contengan código de la misma tipología, es decir:

NombreSistema/JS/AngularJS/AdministradorUsuarios/

NombreSistema/JS/AngularJS/AltaDeUsuarios

NombreSistema/HTML/AdministradorDeUsuarios.html

NombreSistema/HTML/AltaDeUsuarios.html

- ***Team Testing***: Al término del sprint antes de ser entregado al PO realizar una breve reunión al menos un día antes con el equipo de desarrollo con el fin de aprobar internamente en sprint y este no lleve errores, es mejor hablar con la verdad con el POy notificarle si hay un retraso en la entrega, así no hará quedar mal a equipo.

- ***Entrega de Sprint***: El testing se realizan pruebas con el PO basándose en el requerimiento o especificación funcional que se elaboró, dando como entrega el formato creado en la etapa ágil en la generación del sprint, validando que se cumpla lo establecido.

Es importante no liberar un producto el cual no cumple con lo requerido, ya que puede causar complicaciones al sistema o bien alguna molestia a los usuarios.
Al término se obtendrá un sistema documentado y con sus módulos desarrollados, lo que, facilitada futuros mantenimientos, además de que estará planificado de forma ágil.

4.2 Descripción de las Residencias Profesionales

La residencia profesional es parte del programa educativo de nivel licenciatura con valor curricular, siendo una estrategia que permite al alumno vivir la experiencia en el sector productivo donde pone en práctica los conocimientos adquiridos durante su estancia en la institución, impulsando así al alumno para participar activamente en el sector productivo como futuro profesionista.

El programa se cursa en un periodo de 4 a 6 meses, acumulando un aproximado de 500 horas laborales, en los horarios que el sector productivo le indique, un dato a considerar

es que el alumno solo puede acceder al programa una sola vez, generando así un sentido de responsabilidad en el alumno para con la empresa dando en renombre al Instituto Tecnológico y al mismo alumno al término del programa.

El alumno tiene como obligación sujetarse a las disposiciones especificadas en el acuerdo entre el Instituto Tecnológico y el organismo donde se desarrollará la residencia, al igual deberá mantener la confidencialidad de la información generada durante la realización del proyecto, además debe concluir satisfactoriamente todas actividades planeadas en su residencia. (Instituto Tecnologico de Zitacuaro, 2023)

Para poder acceder a este programa el alumno debe cumplir los siguientes requisitos:

- Acreditación del Servicio Social.
- Acreditación de todas las actividades complementarias.
- Tener aprobado al menos el 80% de créditos del plan de estudios.
- No contar con alguna materia en este momento en condición de “curso Especial”.
- Asistencia al curso de inducción de residencias profesionales.

El proceso de residencia profesional se lleva a cabo en varias etapas. **Véase:**

- **Figura 5. Diagrama de proceso de residencias parte 1**
- **Figura 6. Diagrama de proceso de residencias parte 2**
- **Figura 7. Diagrama de proceso de residencias parte 3**

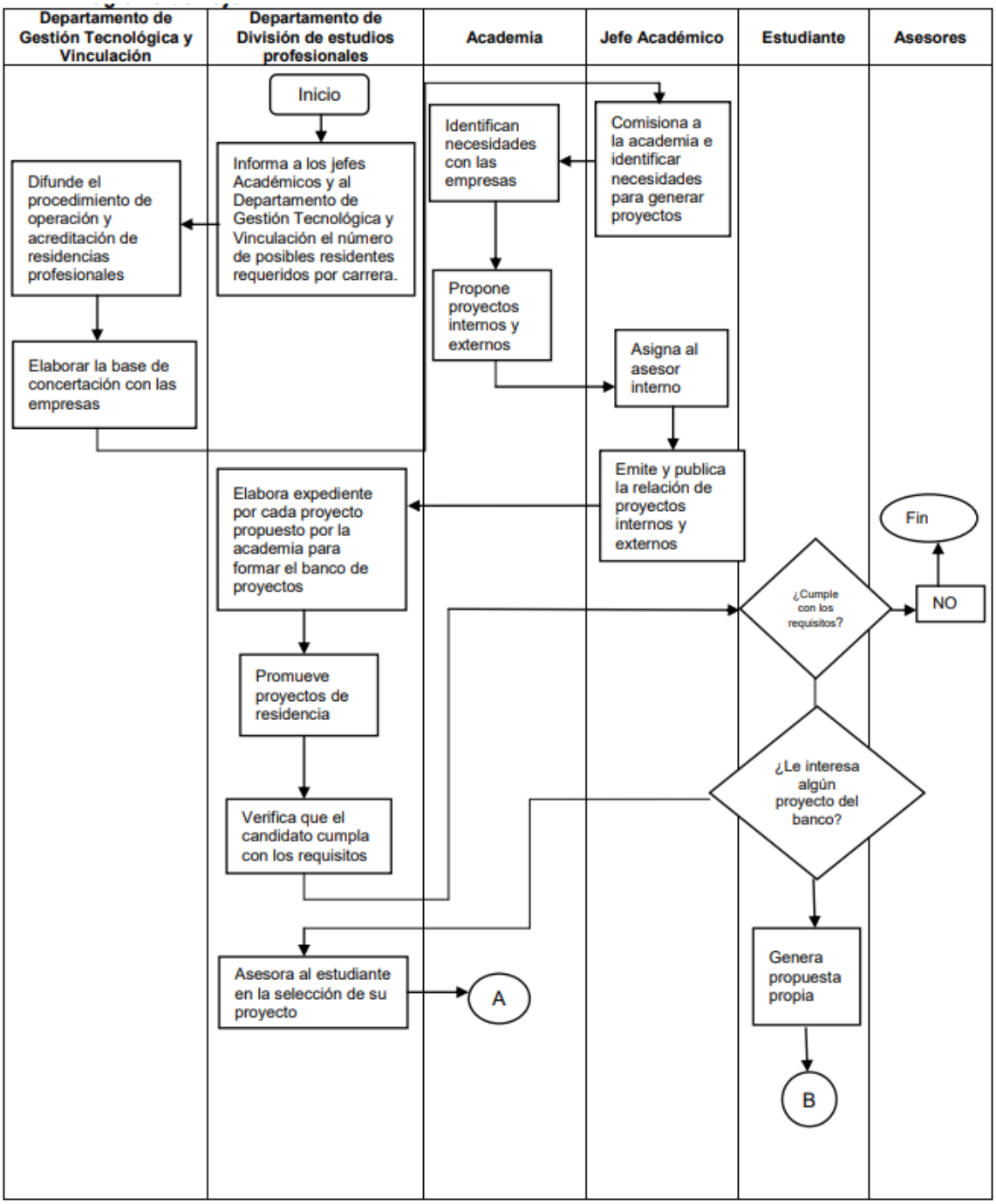


Figura 5. Diagrama de proceso de residencias parte 1 (Instituto Tecnologico de Zitacuaro, 2023)

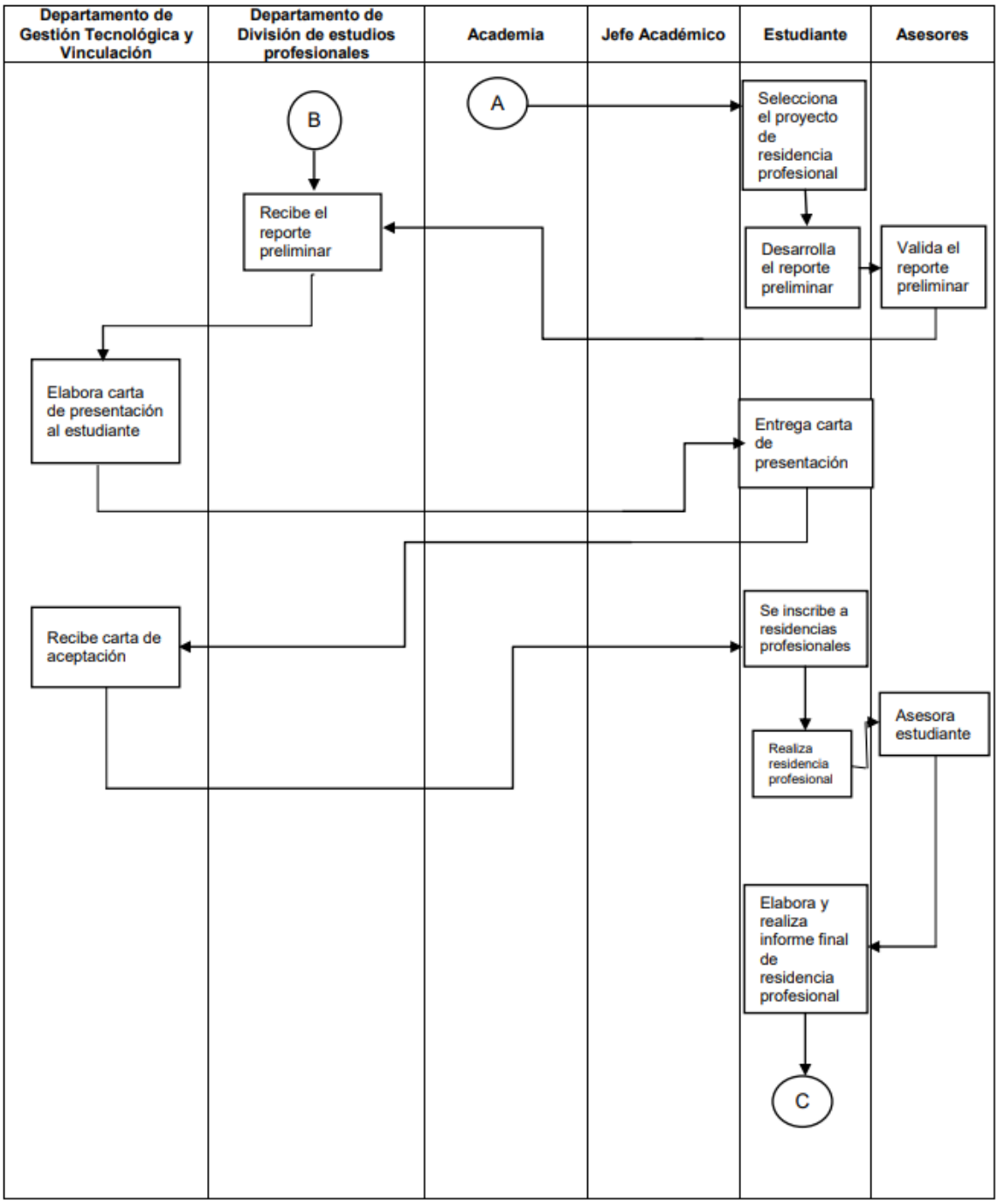


Figura 6. Diagrama de proceso de residencias parte 2 (Instituto Tecnologico de Zitacuaro, 2023)

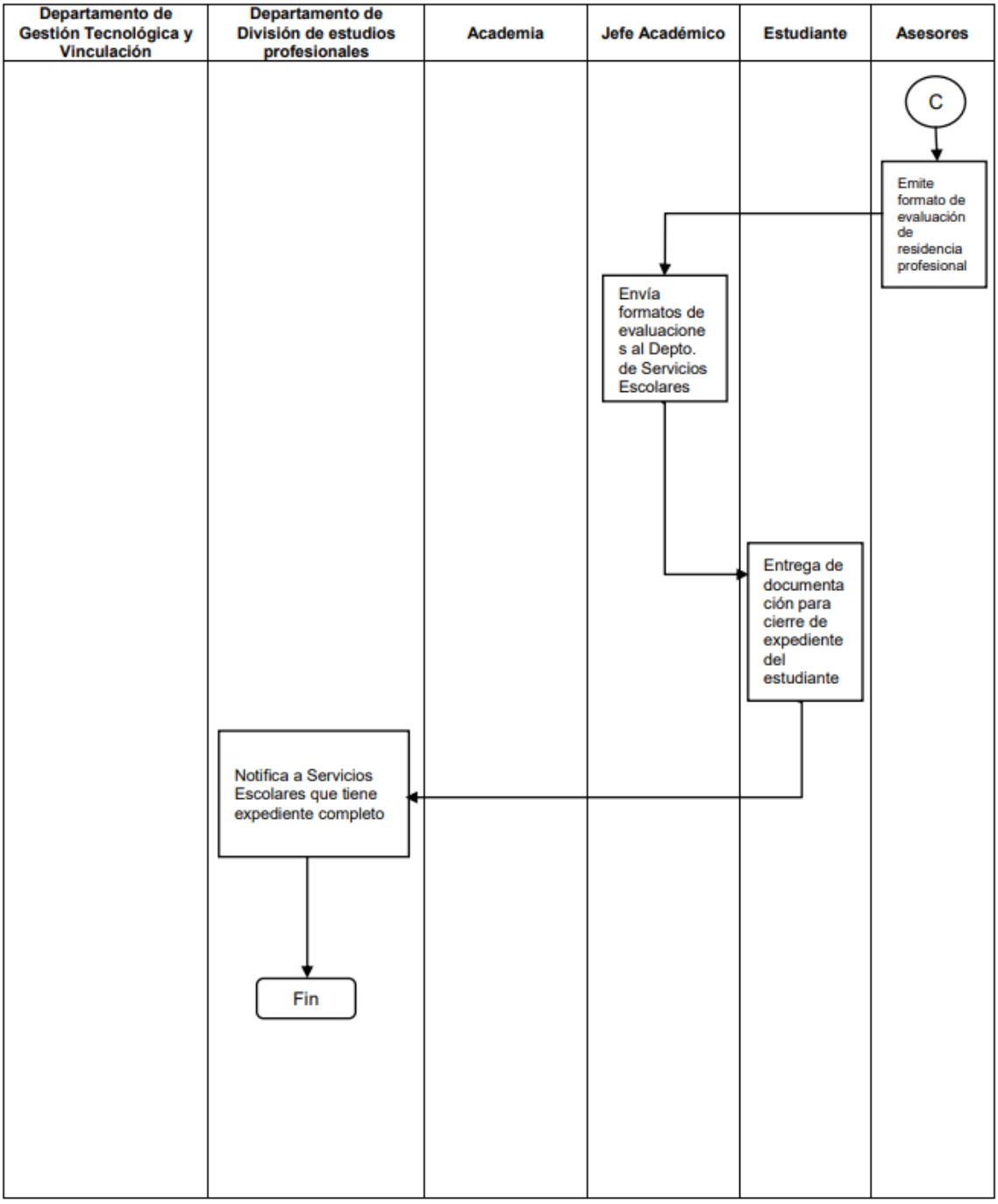


Figura 7. Diagrama de proceso de residencias parte 3 (Instituto Tecnologico de Zitacuaro, 2023)

- **Curso de inducción:** Se convoca a los estudiantes que son aspirantes a realizar RP (Residencias Profesionales) a una reunión o Curso de Inducción donde se explica el proceso administrativo y se proporciona un momento para despejar dudas de los candidatos.

- **Documentos para inicio de trámite:** El alumno debe entregar al coordinador de carrera por medio de correo electrónico los siguientes cuatro formatos debidamente requisitados:

    1. **Formato 1**. Avance de residencia profesional. Este formato es una lista de cotejo que le permite al aspirante conocer los pasos a realizar durante el proceso de residencia profesional.

    2. **Formato 2**. Cumplimiento de requisitos previos. En este formato el estudiante declara cumplir con todos los requisitos necesarios para iniciar la residencia.

    3. **Formato 3.** Solicitud de residencia profesional. Este documento contiene información general del estudiante, así como datos del proyecto y empresa o lugar donde realizará la residencia profesional.

    4. **Formato 4**. Reporte preliminar. Contiene la información detallada del proyecto o trabajo a realizar.

    5. **Formato 5.** Oficio COVID (Solo en caso de emergencia sanitaria). Este documento se utilizó para casos especiales como la pandemia a causa del COVID-19.

- **Asignación de asesores internos**: Los coordinadores de carrera envían el reporte preliminar al Departamento Académico (DA) en donde en de forma colegiada se asignan a los asesores internos y revisores para cada proyecto.

  Una vez que se tienen los asesores internos, el DA publica la lista de asesores internos, los estudiantes deben contactar con sus asesores para que en conjunto actualicen, y mejoren la redacción del reporte preliminar, finalmente el asesor y autoriza el reporte preliminar mediante su firma autógrafa en la portada de dicho

documento. Posteriormente, los asesores entregan al DA los reportes preliminares autorizados.

El DA realiza dictamen con los datos de los estudiantes, asesores y nombres de proyectos aceptados y los envía a los coordinadores de carrera en DEP.

- **Recepción de Carta de Asignación:** los coordinadores de carrera elaboran la carta de asignación, la cual indica de manera oficial la fecha de inicio y término de la residencia profesional y entregan tres ejemplares: para el residente, para GTV y otra para el DA.

- **GTV envía carta de presentación y Acuerdo Tripartita:** El Departamento de Gestión Tecnológica y Vinculación (GTV) envía al residente la carta de presentación oficial y acuerdo tripartita para que los entregue a la empresa o entidad donde realizará la residencia profesional.

- **Entregar en GTV carta de aceptación:** el residente deberá entregar la carta de aceptación y acuerdo tripartita previamente firmados y sellados por la empresa.

- **Entregar evaluaciones a DEP:** El alumno es evaluado en la empresa por un asesor externo y por el asesor interno con base al cumplimiento de las actividades asignadas y realizadas; estas evaluaciones se realizan en dos ocasiones durante el tiempo que dure la residencia, las evaluaciones se realizan en un formato y tienen las firmas del asesor interno y externo.

- **Entrega de Evaluación Final en DEP:** El residente deberá entregar la evaluación final de su reporte, así como el Reporte Final.

- **Entrega en GTV Carta de Terminación:** El residente deberá entregar en GTV la carta de terminación de residencia profesional.

- **Servicios Escolares entrega Boleta de Calificación a DEP:** en este paso Servicios Escolares entrega boletas de calificaciones a los coordinadores de carrera con la finalidad de que ellos entreguen a los asesores internos.
- **Asesores Internos entregan boletas a DEP:** los asesores requisita la boleta de calificación final y entregan a coordinadores de carrera.

- **Entrega de Calificación Final a SE:** El coordinador de carrera entrega a SE las boletas de calificación de los residentes profesionales.

Una vez concretado todo el proceso en tiempo y forma, el estudiante podrá participar en otros procesos como el de titulación. Las residencias profesionales son las estrategias educativas con un proceso largo, pero de alta importancia, con el fin de beneficiar a todas las partes involucradas (empresa o entidad, alumno, instituto tecnológico).

4.3 Análisis del sistema de residencias profesionales

De acuerdo al proceso de residencias profesionales analizado se determinó que el sistema tiene una importante participación con otras áreas del ITZ donde algunas de ellas no cuentan con un sistema de gestión, el cual pueda proporcionar la información necesaria para poder llevar a cabo el desarrollo completo del proyecto, es decir, el proceso de residencias profesionales gestiona alumnos autorizados para la solitud del mismo proceso los cuales deben cubrir requisitos mínimos para poder realizar dicho trámite, a su vez requiere los docentes y personal administrativo para poder llevar a cabo la asignación de solicitudes al personal administrativo que asignara a los docentes con los conocimientos y aptitudes necesarias que puedan apoyar al alumno por medio de ideas basadas en su experiencia profesional, con el fin de entregar un proyecto viable a la empresa con la que se estableció contacto. Por tanto, para poder realizar un sistema completo sobre residencias profesionales previamente se debe realizar desarrollos de las áreas involucradas. **VéaseFigura 7. Diagrama de proceso de residencias parte 3**

Figura 8. Flujo de departamentos involucrados en el sistema de gestión de residencias profesionales.

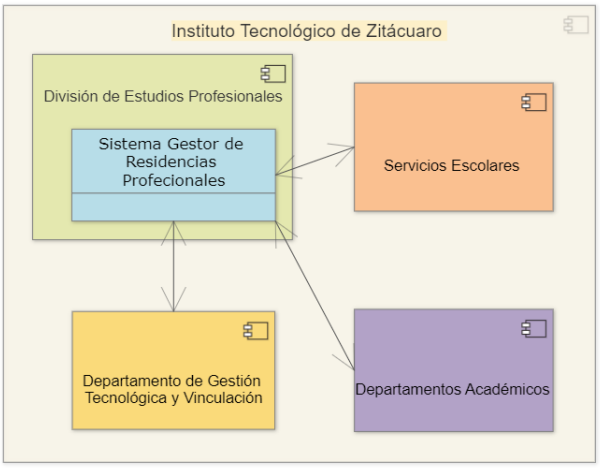


Figura 8. Flujo de departamentos involucrados en el sistema de gestión de residencias profesionales

Al realizar el análisis se propone la realización de tres módulos para el Sistema gestor de residencias profesionales, de los cuales cada uno contiene reglas de negocio o procesos independientes con el fin de llevar a cabo el proceso. **Véase Figura 9. Módulos de Sistema Gestor de Residencias Profesionales (SGRP)**

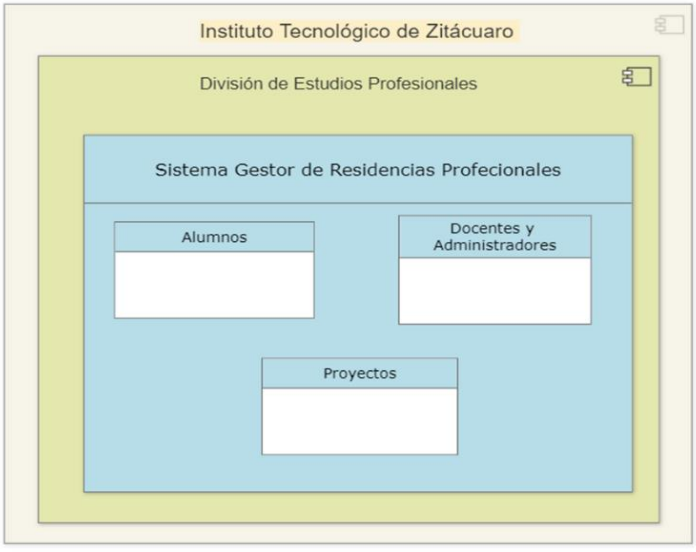


Figura 9. Módulos de Sistema Gestor de Residencias Profesionales (SGRP)

Se propone por parte del equipo de desarrollo generar los siguientes módulos:

- **Alumnos**: Gestión de los alumnos autorizados para aplicar al proceso de residencias profesionales, el cual estará dividido en submódulos para su funcionamiento:
    - **Alta**: Dar de alta a los alumnos autorizados para el proceso de residencias, importando la información desde archivo Excel generado por Mindbox siendo la fuente de información que tiene el registro de todos los alumnos. Una vez realizada la importación, se deberá guardar en la base de datos del sistema teniendo como número de identificación único para el alumno su número de control.

    - **Baja**: Eliminar los usuarios dados de alta en el sistema, que no puedan realizar el proceso de residencias profesionales, esto por medio de su número de control. Cuando un estudiante deja de formar parte de la institución educativa o solicita la baja del proceso de residencias profesionales, se debe realizar dicho proceso. El usuario con los privilegios necesarios podrá acceder al sistema y buscar al alumno que se dará de baja.

    - **Consulta**: Consultar los datos del alumno accediendo a la información almacenada en el sistema, buscando por medio de filtros, pero principalmente por su número de control.

    - **Modificación**: Consultar los datos del alumno mostrándolos en pantalla para poder editar la información almacenada, al editarla deberá actualizarse en la base de datos.

- **Docentes y administradores**: Gestión de personal docente y administrativo autorizados para llevar a cabo el proceso de residencias profesionales, donde estará compuesto por los siguientes submódulos:

- **Alta**: Dar de alta a los docentes autorizados para el proceso de residencias, importando la información desde archivo Excel generado por el sistema de Recursos Humanos, siendo la fuente de información que tiene el registro de todos los docentes y administradores que pertenecen al ITZ. Una vez realizada la importación, se deberá guardar en la base de datos del sistema, teniendo como número de identificación único para el docente su número de empleado.

- **Baja**: Eliminar los usuarios dados de alta en el sistema, que no puedan realizar el proceso de residencias profesionales o por motivo de baja del plantel, esto por medio de su número de empleado. Cuando un docente deja de formar parte de la institución educativa o solicita la baja del proceso de residencias profesionales reasignando a otro docente en caso de ser requerido. El usuario con los privilegios necesarios podrá acceder al sistema y buscar al docente que se dará de baja del sistema.

- **Consulta**: Consultar los datos del docente accediendo a la información almacenada en el sistema, buscando por medio de filtros, pero principalmente por su número de empleado.

- **Modificación**: Consultar los datos del docente mostrándolos en pantalla para poder editar la información almacenada, al editarla deberá actualizarse en la base de datos.

- **Proyectos**: Gestión de solicitudes y proyectos activos de residencias profesionales, siendo este el módulo más complejo del sistema, ya que involucra varias áreas del ITZ, generando una propuesta del funcionamiento, esto sin ser las reglas de negocio final:
  - **Creación de la propuesta de proyecto**: Un alumno podrá crear una propuesta detallada que describe los objetivos, alcance del proyecto a

realizar en una empresa de la región o fuera de ella. Esta propuesta se ingresa en el sistema gestor de residencias profesionales a través de un formulario o interfaz de usuario específico. El sistema puede requerir que se adjunten documentos adicionales, como planes de trabajo o cronogramas de actividades, entre otros.

- **Enrutamiento de la propuesta:** Una vez que la propuesta se ha creado y registrado en el sistema, el sistema la enruta automáticamente al flujo de trabajo correspondiente. El flujo de trabajo puede estar basado en reglas predefinidas que determinan los docentes que son los responsables de revisar y aprobar la propuesta. El sistema puede enviar notificaciones por dentro del sistema y por medio del correo a los revisores designados informándoles sobre la nueva propuesta que deben evaluar.

- **Evaluación de la propuesta:** Los revisores designados acceden al sistema autorizador de proyectos y revisan la propuesta de manera individual.
  Pueden realizar análisis detallados y evaluar la viabilidad técnica, aportando ideas de mejora o sugiriendo alternativas de proyecto.
  Los revisores pueden dejar comentarios, hacer preguntas o solicitar modificaciones al solicitante a través del sistema.

- **Aprobación o rechazo de la propuesta**: Una vez que los revisores han realizado su evaluación, el sistema recopila los resultados y determina el estado de la propuesta. Si todos los revisores aprueban la propuesta, el sistema la marca como aprobada y pasa al siguiente paso. En caso de que alguno o varios revisores rechacen la propuesta, el sistema la marca como rechazada y se notifica al alumno, quien puede hacer ajustes y volver a enviarla para su revisión.

- **Seguimiento y gestión del proyecto aprobado:** Una vez que la propuesta es aprobada, se considera como un proyecto oficial y se inicia su seguimiento y gestión con los demás departamentos. El sistema debe permitir al alumno subir el reporte de avance, para generar evidencia del trabajo realizado.

Basado en el análisis realizado del caso de uso por parte del equipo de desarrollo del ITZ, se determinó que el proyecto es extenso e involucra diferentes áreas del ITZ, sin embargo, para poder aplicar el modelo propuesto de reingeniería de software se seleccionó el módulo inicial para poder visualizar resultados, siendo este el módulo de alumnos, el cual carece de una carga masiva de usuarios tomando esto como un mantenimiento al sistema actual favoreciendo al proyecto en general, documentado el aplicativo actual para que el equipo de desarrollo pueda aprender el funcionamiento del mismo y así poder integrar el avance actual con el sistema general el cual involucra las demás áreas. Además de poder visualizar si beneficia al equipo en la gestión de nuevos sistemas o mantenimientos.

La limitación del tiempo es un factor el cual no permite ver resultados más allá del módulo de alumnos, sin embargo, es el inicio de aplicar un modelo que permita estandarizar el proceso que se lleva a cabo en el ITZ.

4.5 Aplicación de MDEE

Actualmente, el equipo de desarrollo del ITZ no lleva a cabo una metodología o modelo para la gestión de proyectos, debido a que es reciente la creación del área y cuenta con un equipo pequeño de alumnos egresados del ITZ y alumnos que se encuentran en el proceso de residencias profesionales, siendo este un escenario ideal para llevar a cabo la aplicación de un modelo de reingeniería de software que a su vez puede convertirse en el modelo de ingeniería para adquirir nuevos desarrollos, dando experiencia en buenas prácticas al equipo, ayudándoles en su futuro profesional.

4.5.1 Etapa de Evaluación

Basado en la propuesta del modelo de reingeniería, se llevó a cabo una entrevista con el Product Owner (PO) y el Revisor (R) de la entrega final, donde el Project Administrator (PA) y desarrollador en conjunto generaron la arquitectura y el flujo de datos a manera de propuesta para generar un plan de trabajo, documentando esta actividad con el siguiente cuestionario. **Véase Tabla 11. Resultado de Cuestionario para levantamiento de requerimiento del modelo propuesto.**

4.5.1.1 Obtención de requerimiento funcional

Pregunta	Respuesta o comentario
¿Cuál es el objetivo principal del software?	La gestión del proceso de residencias profesionales de manera virtual, para facilitar a los alumnos y a los docentes el trámite, además de mejorar los tiempos del proceso (el usuario proporciono un diagrama donde explica todo el proceso de residencias profesionales)
¿Qué problemas o necesidades específicas del usuario o el departamento espera que resuelva el software?	Que pueda gestionar todo el trámite de residencias profesionales en las diferentes áreas del ITZ
¿Quiénes serán los usuarios finales?	Alumnos residentes y docentes que tomen el papel de revisores y tutores, así como personal administrativo para la gestión de los proyectos
¿El sistema cuenta con un gestor de usuarios? Es decir, ¿maneja tipos de usuarios? ¿Es requerido?	El actual sistema, sí, contiene un apartado para generar usuarios de tipo maestro o alumno, sin embargo, se agrega de manera manual
¿Cuál es el alcance del proyecto? Es decir, ¿Hay algún límite en cuanto a lo que el software debe hacer o no hacer?	No hay limitante, ya que el sistema debe cubrir la gestión completa de residencias profesionales
¿Qué plataformas se deben soportar? ¿Deben ser compatibles con varios sistemas operativos o dispositivos?	Dispositivos que tengan acceso a un navegador

¿En cuánto al requerimiento, que debe validar el software?	De inicio debe validar los siguientes puntos sobre el alumno • Acreditación del Servicio Social. • Acreditación de todas las actividades complementarias. • Tener aprobado al menos el 80% de créditos del plan de estudios. • No contar con alguna materia en este momento en condición de “curso Especial”. • Asistencia al curso de inducción de residencias profesionales (Instituto Tecnologico de Zitacuaro, 2023)
¿Cómo se integrará el software con otros sistemas existentes?	Al informar a otros sistemas que alumnos son los candidatos al proceso de residencias profesionales
¿Qué nivel de mantenimiento y soporte se requiere después del lanzamiento del software? Es decir, ¿qué tan constante puede llegar a recibir cambios?	Al menos cada cambio de directivos puede presentarse una mejora, ya que las reglas de negocio persisten, pero pueden cambiar en otros departamentos es decir la forma de llevar a cabo sus procesos
¿Cuál es el plazo para la entrega del software?	Inicios de 2024
¿Tiene datos reales para realizar pruebas?	Sí, la generación del listado de alumnos candidatos a residencias profesionales
¿Se requiere algún reporte?	Sí, reportes para notificar los índices de residentes y futuros egresados, además de la generación de documentación oficial para los alumnos
¿El sistema genera reportes o documentos?	Si el actual sistema maneja la generación de la solitud del alumno y el proceso que debe llevar a cabo

¿Cuáles son las principales debilidades y problemas del software actual?	No genera, lleva a cabo la gestión del proceso como tal, debemos realizar el registro del alumno, a veces son más de 100 alumnos, no tenemos los docentes registrados ni tampoco la generación de todos los documentos, además no hay comunicación con otros sistemas de las áreas, es decir en las otras áreas debe realizarse la documentación en físico
¿Cuál es el alcance de la reingeniería del software?, es decir, el desarrollo tendrá nuevos módulos a los existentes	Integrar el sistema actual con todas las áreas para poder tener información en tiempo real de los alumnos y del proceso de residencias
¿Cuáles son los principales requerimientos funcionales que se deben abordar en la reingeniería del software? Es decir, ¿Qué módulos son los principales a analizar?	El módulo de alumnos, el módulo de docentes y el módulo de proyectos
¿Se requerirá la migración de datos del software actual a la nueva versión?	Tal vez de datos sí, pero de estructura de base de datos no
¿Qué tecnologías y herramientas se utilizan en el sistema (en caso de existir código)?	PHP 7, Mariadb, HTML, CSS
¿Tiene relación con otro sistema? Y si es si, ¿cuál es ese sistema y qué funcionalidad tiene?	No tiene relación con otro sistema, pero sí debe tener con los siguientes departamentos DGTV: Departamento de Gestión Tecnológica y Vinculación DA: Departamentos Académicos SE: Servicios Escolares

Tabla 11. Resultado de Cuestionario para levantamiento de requerimiento del modelo propuesto.

Después de realizar la entrevista se determinó lo ya mencionado en el capítulo anterior, véase capítulo III en el apartado cuarto, siendo el sistema extenso y por limitantes de tiempo solo se generó la documentación para el módulo de alumno donde sé estableciendo los datos de los involucrados y el histórico de los futuros desarrollos, siendo este el punto de partida para el proyecto en general. **Véase** Tabla Resultado de Formato de gestión de proyecto

(*Por confidencialidad de los involucrados se omitió el número de teléfono para el presente trabajo de tesis*)

Datos del proyecto

Nombre del Proyecto:	Sistema Gestor de Residencias Profesionales		
Abreviatura	SGRP		
Encargado del proyecto:	Darío Davalos Hernández		
Fase del Proyecto:	Modificación de Módulo	**Fecha de Creación:**	3/Febrero/2023
Documentado por:	Cuautli David García López		
Revisado por:	Darío Davalos Hernández	**Fecha de Revisión:**	26/Mayo/2023

Responsables

Nombre	Rol	Fecha	Correo Electrónico	Teléfono	Responsable de versión
Julio Joel Darío	Dev	16/Enero/2023	Julio.joel.dm@zitacuaro.tecnm.mx	-	V1.0
Samuel Efrén Viñas Álvarez	PO	16/Enero/2023	samuel.va@zitacuaro.tecnm.mx	-	V1.0
Darío Davalos Hernández	PA	16/Enero/2023	dario.dh@zitacuaro.tecnm.mx	-	V1.0
Cuauhtémoc Jiménez	R	16/Enero/2023	cuauhtemoc.jo@zitacuaro.tecnm.mx	-	V1.0
Darío Davalos Hernández	Dir	16/Enero/2023	dario.dh@zitacuaro.tecnm.m	-	V1.0

Tabla 12. Resultado de Formato de gestión de proyecto

Histórico

Nombre Requerimiento	Alta de Usuarios para sistema gestor de residencias profesionales
Objetivo	Dar de alta los alumnos candidatos al proceso de residencias profesionales, importar los alumnos desde un archivo Excel proporcionado por el sistema mindbox
Responsable	Julio Joel Darío
Fecha de inicio	06/Marzo /2023
Fecha fin	31/Marzo/2023
Autorizado por	Darío Davalos Hernández

Nombre Requerimiento	Consulta de Usuarios para sistema gestor de residencias profesionales
Objetivo	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y mostrarlos en la interfaz
Responsable	Julio Joel Darío
Fecha de inicio	29/Marzo/2023
Fecha fin	14/Abril/2023
Autorizado por	Darío Davalos Hernández

Nombre Requerimiento	Baja de Usuarios para sistema gestor de residencias profesionales
Objetivo	Dar de baja a los alumnos que: solicitado detener el proceso de residencias profesionales Han sido dados de baja del ITZ Que no pertenecen o no cumplen los requisitos para el proceso de residencias profesionales
Responsable	Julio Joel Darío
Fecha de inicio	15/Abril/2023
Fecha fin	1/Mayo/2023
Autorizado por	Darío Davalos Hernández

Nombre Requerimiento	Modificación de Usuarios para sistema gestor de residencias profesionales
Objetivo	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y tener opción de modificar sus datos en caso de ser requerido, almacenando la modificación en el sistema
Responsable	Julio Joel Darío

Fecha de inicio	2/Mayo/2023
Fecha fin	14/Mayo/2023
Autorizado por	Darío Davalos Hernández

4.5.1.2 Evaluación de documentación

Posterior a la generación del documento con los datos principales del proyecto, se solicitó la documentación existente, la cual no fue proporcionada, por motivo se extravió, sin embargo, se realizó parte del formato de evaluación como parte del modelo.

Véase Tabla 12.Resultado de evaluación de proyecto.

Núm.	**Pregunta**	**X**	Observación
1	¿Hay documentación disponible para el proyecto de software?	☐	No existe documentación y la poca existente no se proporcionó para revisión, se encuentra extraviada
2	¿La documentación está organizada y estructurada de manera clara?, es decir, separa código de documentación a nivel carpetas	☐	Según la entrevista, la documentación no fue terminada
3	¿La documentación es fácil de leer y comprender?, maneja una estructura donde divide por secciones el tema a tratar	☐	No existe
4	¿La documentación cubre todos los aspectos importantes del proyecto, incluyendo la arquitectura, el diseño y la implementación?	☐	No, solo incluía un manual de usuario
5	¿La documentación incluye ejemplos y casos de uso para ayudar a entender cómo funciona el proyecto?, estos se pueden encontrar en el manual de usuario	☐	No, en el manual de usuario solo se encontraba una descripción breve de cómo utilizar el sistema sin casos de uso solo capturas de pantalla
6	¿La documentación está actualizada con los últimos cambios en el proyecto?	☐	no
7	¿La documentación incluye información sobre cómo instalar y configurar el software?, es decir, contiene algún manual de soporte	☐	no

8	¿La documentación describe los requisitos del sistema necesarios para utilizar el software?	☐	no
9	¿La documentación incluye información sobre cómo mantener y actualizar el software?	☐	no
10	¿El software tiene comunicación con otro sistema? Escribir en el espacio de observaciones con qué sistemas tiene comunicación	☐	No contiene comunicación con ningún sistema
11	¿El software maneja webservices o servicios rest?	☐	no
12	¿Se tiene el código de webservices o servicios rest?	☐	no
13	¿La documentación incluye información sobre cómo utilizar el código fuente del proyecto?, es decir, instalar ambientes de desarrollo para futuros mantenimientos	☐	No, la instalación es realizada como proyecto PHP
14	¿Utiliza algún repositorio de documentos? Indicar cuál es en el campo observaciones	☐	No
15	¿Se tiene acceso al repositorio de archivos?	☐	no
16	¿La documentación incluye información sobre las licencias y derechos de autor del software?	☐	no
17	¿Se encuentra documentado el servicio rest o webservices que utiliza?	☐	no
18	¿Existe información sobre la base de datos? Es decir, diagramas, entidad, relación o diccionario de datos	☐	No solo se tiene acceso a manera de consulta a la base de datos
19	¿Se tiene el script o código SQL para generar la base de datos?	☒	si
20	¿Existe en la institución un desarrollador con conocimientos del software? Es decir, miembro del equipo inicial de desarrolladores del software	☒	Si Samuel Efrén Viñas Álvarez Fue el encargado de realizar parte de ese sistema

Observaciones:	La documentación fue extraviada, el código sí fue proporcionado al igual que el código en SQL para la base de datos

Tabla 12.Resultado de evaluación de proyecto

4.5.1.3 Evaluación de Funcionalidad

De igual forma se solicitó el código de desarrollo del sistema y su base de datos para poder analizar la funcionalidad, determinando el estado del proyecto, obteniendo la siguiente evaluación. **Véase Tabla 13.Resultado de evaluación de sistema.**

Preguntas	Respuesta	Observación
¿El software cumple con todas las funciones descritas en la documentación?	☐Si ☒No ☐En algunos casos	Le hace falta funcionalidad, solo genera la solicitud para generar las residencias profesionales
¿El software se ejecuta sin errores?, es decir, muestra error en alguna función y qué tipo de error	☒Si ☐No ☐En algunos casos	
¿El software es fácil de instalar y configurar?, es decir, el proceso de instalación lo puede realizar quien sea	☒Si ☐No ☐En algunos casos	Es un sitio Web
¿El software es fácil de usar?, es decir, es intuitivo o fácilmente se encuentran las funciones prometidas	☐Si ☐No ☒En algunos casos	Algunas pantallas no son tan intuitivas,
¿El software es intuitivo y fácil de aprender? Es decir, se requiere alguna capacitación o con solo explicar su funcionamiento se puede usar	☐Si ☐No ☒En algunos casos	Puede ser fácil de entender, pero esto alca bode horas
¿El software es compatible con otros programas o sistemas operativos?	☒Si ☐No ☐En algunos casos	
¿El software cumple con los requisitos del sistema descritos en la documentación?	☐Si ☒No ☐En algunos casos	
¿El software es seguro y protege los datos del usuario? Es decir, maneja algún tipo de encriptación de datos	☐Si ☐No ☒En algunos casos	
¿El software funciona con rapidez y eficiencia?, funciona de manera ágil al consultar y realizar operaciones	☒Si ☐No ☐En algunos casos	Genera consultas rápidas, además de realizar la exportación de todos de manera rápida

¿El software es escalable y puede manejar grandes cantidades de datos o usuarios?	☐Si ☐No ☒En algunos casos	Depende también de los servidores donde ese encuentre la aplicación
¿El software proporciona suficiente retroalimentación al usuario?, es decir, maneja información que ayude al usuario a saber qué hacer en la pantalla donde se encuentra	☐Si ☒No ☐En algunos casos	
¿El software incluye opciones de personalización para adaptarse a las necesidades del usuario?	☐Si ☐No ☒En algunos casos	
¿El software es confiable y estable?, tiene algún fallo anormal fuera de la funcionalidad que no permita realizar las actividades del usuario, por ejemplo, fallas de servidor o de red, fallas de conexión a base de datos	☐Si ☒No ☒En algunos casos	
¿El software incluye herramientas de generación de reportes para el usuario?, es decir, hace uso de apis para la generación de reportes o documentos	☒Si ☐No ☐En algunos casos	
¿El software utiliza un login para acceder?	☒Si ☐No ☐En algunos casos	Se accede a donde login uno para docentes y otro para alumnos
¿El software incluye validaciones en campos? ¿Qué tipo de validaciones?	☐Si ☒No ☐En algunos casos	
¿las validaciones que realiza se manifiestan en todos los campos donde se requieren? **Por ejemplo, una validación de caracteres especiales en un campo de asignación de nombre de persona, en todos los campos donde se utiliza este campo se válida que no tenga caracteres especiales**	☐Si ☒No ☐En algunos casos	
¿El software es compatible con dispositivos móviles y tabletas?	☒Si ☐No ☐En algunos casos	
¿El software incluye una ayuda contextual o una guía de usuario detallada?, es decir, maneja información de ayuda en los campos a llenar o hay algún manual de usuario	☐Si ☒No ☐En algunos casos	
¿El software puede integrarse con otras aplicaciones o servicios? Es decir, maneja	☒Si ☐No	

servicios web o rest, y con qué sistemas tiene relación para llevar a cabo estos servicios	☐En algunos casos	
¿El software es capaz de gestionar múltiples usuarios y roles de usuario?, es decir, tiene algún apartado donde gestione los usuarios	☒Si ☐No ☐En algunos casos	Tiene la opción de dar alta y baja usuarios con el administrador de usuarios
¿El software incluye herramientas de respaldo y restauración de datos?	☐Si ☒No ☐En algunos casos	
¿El software incluye opciones de búsqueda y filtrado para facilitar la navegación del usuario?	☒Si ☐No ☐En algunos casos	
¿El software cumple con las expectativas del usuario y satisface sus necesidades?, existe alguna mejora que pueda facilitar las actividades que realiza el usuario que no esté contemplada	☐Si ☐No ☒En algunos casos	Hace falta funcionalidad para llevar a cabo
¿El software ha sido probado rigurosamente para asegurar su calidad?, ha pasado por pruebas de testing en alguna ocasión, en caso de ser cierto, ¿se tiene los resultados?	☐Si ☒No ☐En algunos casos	

Tabla 13.Resultado de evaluación de sistema

4.5.1.4 Generación de una lista preliminar de componentes

Se generó el siguiente listado de funcionalidad que deberá tener el módulo de alumnos, este servirá para poder generar la planeación en las siguientes etapas, Además de ser funcionalidad que presenta fallas en el sistema actual**. Véase Tabla 15. Listado preliminar**

Nombre del Módulo	Funcionalidad	Prioridad	Descripción
Alumnos	Alta	alta	Dar de alta los alumnos candidatos al proceso de residencias profesionales, importar los alumnos desde un archivo Excel proporcionado por el sistema mindbox
Alumnos	Consulta	alta	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y mostrarlos en la interfaz

Alumnos	Baja	media	Dar de baja a los alumnos que: solicitado detener el proceso de residencias profesionales Han sido dados de baja del ITZ Que no pertenecen o no cumplen los requisitos para el proceso de residencias profesionales
Alumnos	Modificación	media	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y tener opción de modificar sus datos en caso de ser requerido, almacenando la modificación en el sistema

Tabla 15. Listado preliminar

4.5.2 Etapa de abstracción

Debido a la falta de documentación, no fue posible realizar una revisión completa del sistema, por lo que el área de desarrollo ha tenido que recurrir a la exploración detallada del código fuente. Mediante la ingeniería inversa, este enfoque implica examinar las interacciones entre los componentes del proyecto, el flujo de datos y la lógica de programación. Al realizar esta revisión detallada, el desarrollador puede obtener una comprensión del funcionamiento del proyecto, aunque no disponga de la documentación original.

4.5.2.1 Revisión de arquitectura

El equipo de desarrollo revisó el código para determinar la arquitectura utilizada en el proyecto, generando un Diagrama de UML donde se explica brevemente los lenguajes de programación utilizados en el aplicativo.

En el Diagrama se puede visualizar en el modelo Laravel 7.0, siendo esta la propuesta para organizar el código, ya que el sistema actual solo fue realizado en PHP, **Véase Tabla 14. Arquitectura generada a partir del sistema actual.**

Diagrama de Arquitectura

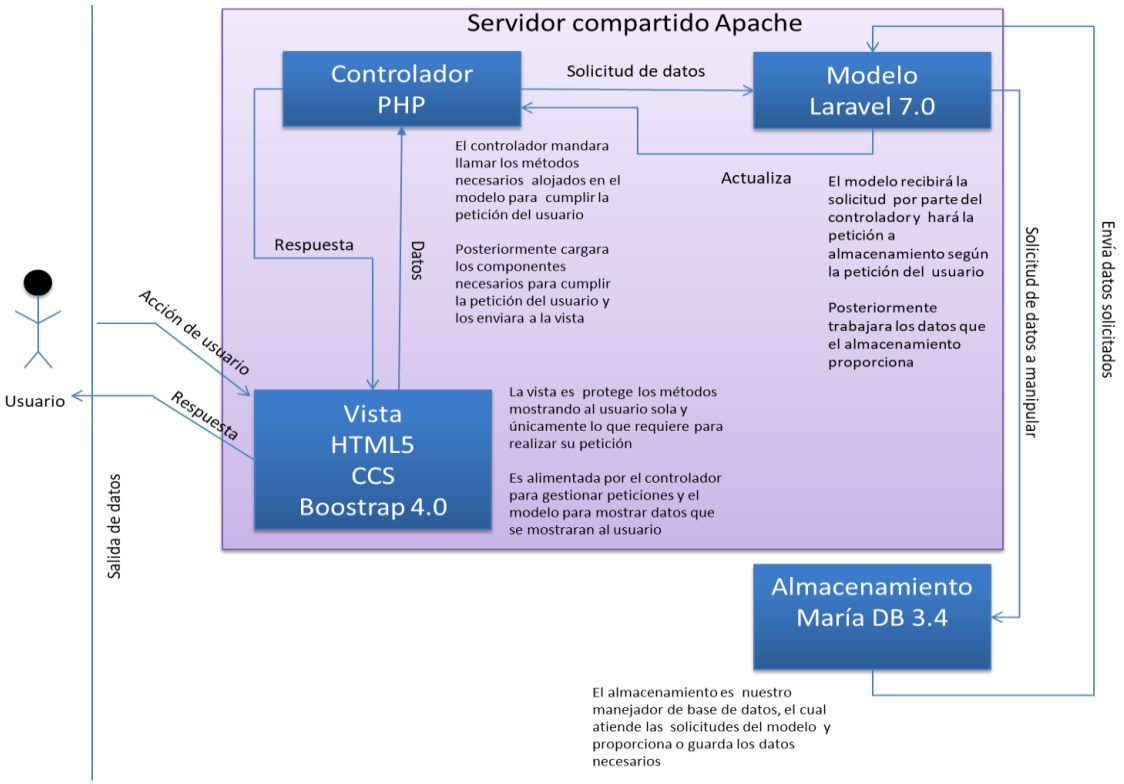
Instrucciones: Diseñar a mano alzada la arquitectura del sistema , considerando: usuarios ,flujos y entidades: modelo vista controlador, según sea el caso


Tabla 14. Arquitectura generada a partir del sistema actual

4.5.2.2 Análisis de interfaz visual

Al analizar la interfaz visual se determinó, es muy intuitiva y fácil de usar para el usuario, es probable que solo requiera reajustar algunos botones en futuros desarrollos, por lo que la generación de registro de usuarios es sencilla. Se muestra captura de cómo se agregan los usuarios. **Véase Figura 10. Alta de alumnos en sistema actual.**

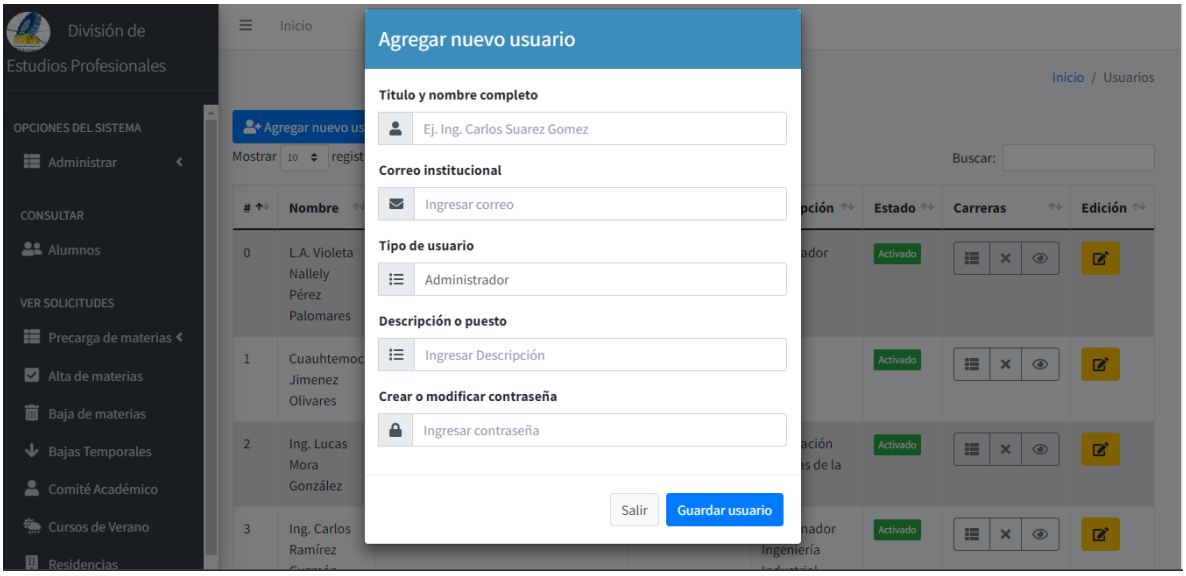


Figura 10. Alta de alumnos en sistema actual

4.5.2.3 Analizar los componentes y procedimientos

Después de escudriñar el código, se pudo determinar que este estaba acompañado de breves comentarios que contribuían a facilitar su comprensión. Además, se encontró que el código estaba bien estructurado en términos de la organización de las capas.

La presencia de comentarios en el código resultó ser de gran utilidad, ya que brindaron información adicional y explicaciones sobre el propósito y funcionamiento de diferentes secciones del código. Esto permitió a los desarrolladores comprender más fácilmente el flujo y la lógica de la aplicación. **Véase Figura 10. Estructura de proyecto actual y Figura 12. Clase Actual para dar de alta al alumno.**

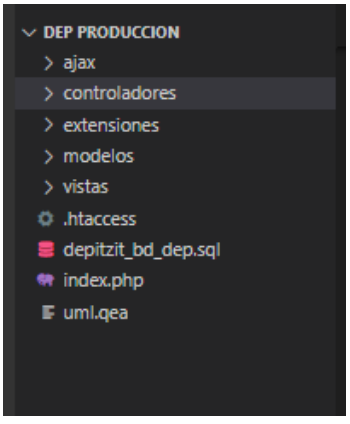


Figura 10. Estructura de proyecto actual

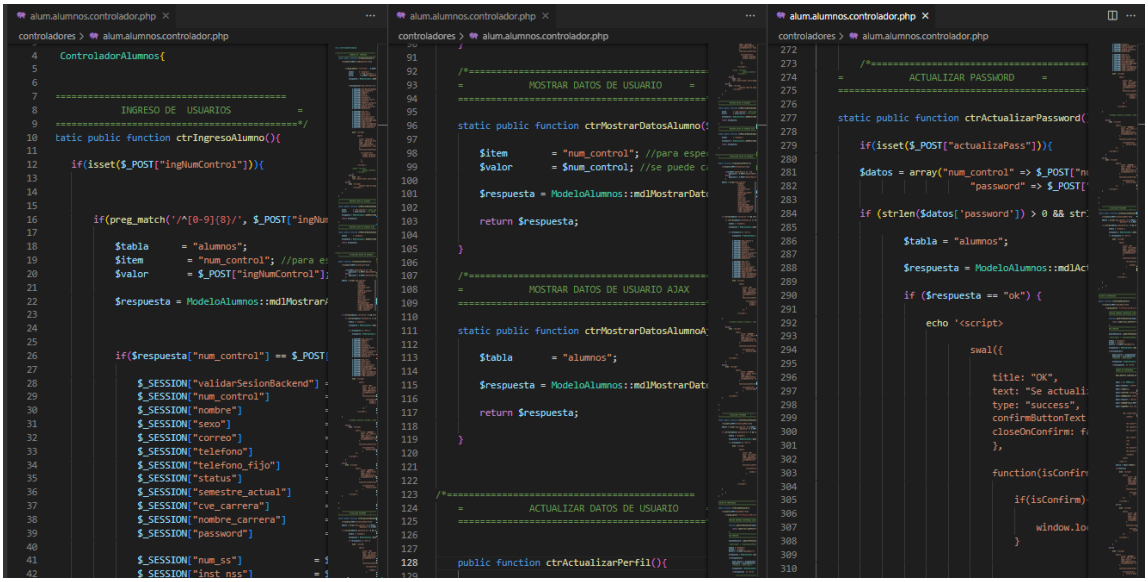


Figura 12. Clase Actual para dar de alta al alumno

4.5.2.4 Análisis de datos

En la etapa de análisis de datos se detectó que no existe ninguna relación entre las tablas del sistema, esto causando redundancia de datos o datos repetidos, además de estar en una versión más antigua de María dB siendo MySQL su predecesor **véase Figura 11. Diagrama de base de datos de sistema actual.**

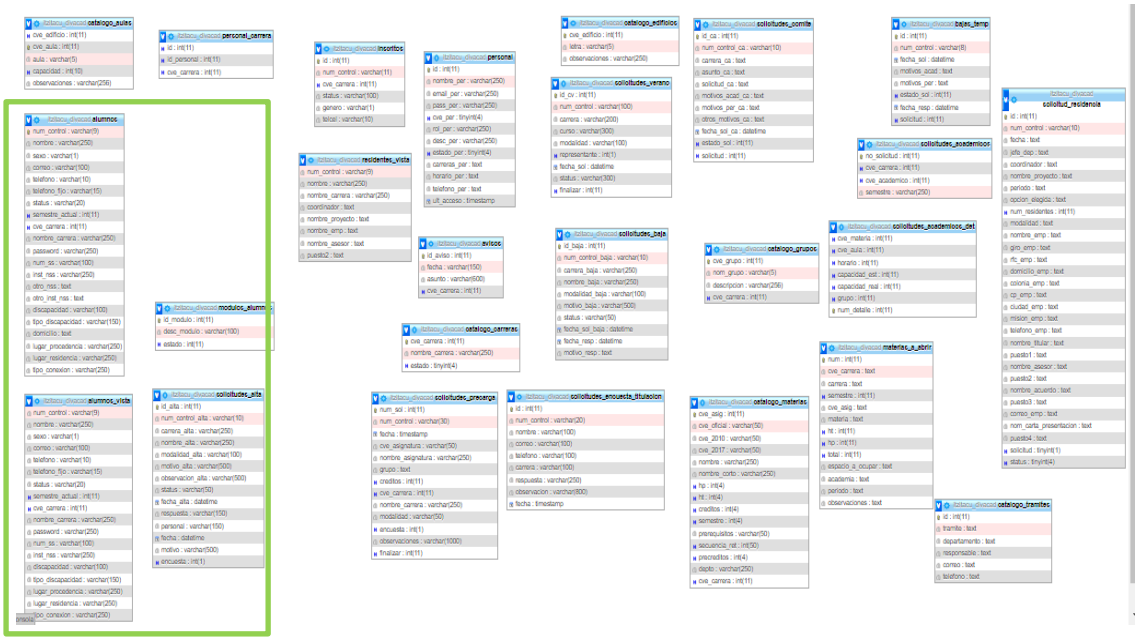


Figura 11. Diagrama de base de datos de sistema actual

4.5.2.5 Complementación

Se determinó que el sistema se encuentra brevemente documentado en el código, no existe documentación y la base de datos no se encuentra normalizada para el módulo de alumnos. El proceso actual de registro funciona de manera correcta, además de ser útil para generar alumnos de manera unitaria, sin embargo, para reducir el trabajo del usuario se desarrollará una carga masiva de usuarios por medio de Excel que agiliza el proceso en lo que se desarrollan los módulos restantes que solventan la carga de alumnado por este proceso y departamento. Además de reutilizar el código existente para generar el alta del alumno, es necesario normalizar las tablas en base de datos relacionadas con el módulo para evitar redundancia de datos y esta pueda incrementar su volumen, evitando así un futuro problema para el sistema.

4.5.3 Etapa Ágil

Para el mantenimiento solicitado se generaron cuatro Sprint, los cuales describen las actividades que realizaran mientras se lleva a cabo el desarrollo del Sprint, siendo una bitácora para el desarrollador.

4.5.3.1 Generación de Sprints

Los sprints son períodos de tiempo cortos y enfocados en los cuales se ejecutan tareas específicas del proyecto. Se explican las fases de preparación, selección de tareas, asignación de recursos y seguimiento de cada sprint. A continuacion se destaca cómo la metodología ágil y la gestión por sprints contribuyen a la eficiencia y flexibilidad en la ejecución del proyecto, permitiendo una adaptación ágil a los cambios y una entrega incremental de resultados.

Nombre:	Alta de alumnos de manera masiva
Descripción	
Dar de alta los alumnos candidatos al proceso de residencias profesionales, importar los alumnos desde un archivo Excel proporcionado por el sistema mindbox	
Comprobable	
El usuario al entrar al apartado para cargar un archivo de tipo Excel, el aplicativo le permitirá realizar la acción y mostrará los alumnos cargados según el archivo	
Entregable	
Apartado en la interfaz para poder cargar archivos de tipo Excel	
Actividades	
• Analizar el modulo de alta de alumnos • Investigar el api para poder cargar archivos excel • Definir el formato que llevara el excel, es decir las columnas deben coincidir con los datos requeridos en la base de datos • Obtener el Excel del sistema mindbox • Cargar el archivo Excel al aplicativo • Mostrar los alumnos cargados	

Tabla 17. Sprint_SGRP_01

Nombre:	Consulta de alumnos
Descripción	
Dar de baja a los alumnos que: solicitado detener el proceso de residencias profesionales Han sido dados de baja del ITZ Que no pertenecen o no cumplen los requisitos para el proceso de residencias profesionales	
Comprobable	
El usuario al entrar al apartado de consulta, deberá poder realizar la consulta del usuario por medio del número de control	
Entregable	
El usuario podrá visualizar la información del alumno que ingreso en el apartado de numero de control	
Actividades	
• Obtener los alumnos registrados en base de datos • Poder filtrar los alumnos por número de control	

Tabla 15. Bitácora de Sprint_SGRP_02

Nombre:	Baja de alumnos de Alumnos
Descripción	
Dar de baja a los alumnos que: • Solicitado detener el proceso de residencias profesionales • Han sido dados de baja del ITZ • Que no pertenecen o no cumplen los requisitos para el proceso de residencias profesionales	
Comprobable	
El usuario al entrar a la aparta de consulta, deberá tener una opción para deshabilitar de alumno del proceso de residencias, si ser mostrado ya en el aplicativo, al dar de alta de nuevo solo deberá habilitar al alumno	
Entregable	
El usuario al eliminar del listado al alumno, no deberá visualizarlo	
Actividades	
• Validar el estado del alumno • Deshabilitar al alumno seleccionado • Volver a cargar al alumno y validar que no se registre el alta nuevamente, solo deberá habilitarlo para se muestre en el listado	

Tabla 19. Bitácora de Sprint_SGRP_03

Nombre:	Modificación del Alumno
Descripción	
Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y tener opción de modificar sus datos en caso de ser requerido, almacenando la modificación en el sistema	
Comprobable	
El usuario al entrar al apartado de consulta, deberá poder tener la opción para modificar los datos del alumno y poder almacenar en base de datos	
Entregable	
Al momento de realizar la modificación se deberá visualizar la modificación al volver a consultar al usuario	
Actividades	
• Analizar el módulo de alumnos • Generar método para modificar al alumno • No deberá poder modificar el numero de control	

Tabla 20. Bitácora de Sprint_SGRP_04

4.5.3.2 Plan de trabajo

Reutilizando la lista preliminar definida en la etapa de evaluación se generó el siguiente plan de trabajo por motivo de visibilidad, se agrega en la siguiente Hoja la tabla del plan de trabajo **Véase Tabla 16.**

Funcionalidad	Prioridad	Descripción	Última Modificación	Estado	Riesgo o contratiempo	Plan de acción	Documentación URL	Sprint	Duración Estimada	Duración Real
Alta	alta	Dar de alta los alumnos candidatos al proceso de residencias profesionales, importar los alumnos desde un archivo Excel proporcionado por el sistema mindbox	06/03/2023	Desarrollo	El Excel con los datos de los alumnos no sea generado por mindbox Curva de aprendizaje del framework Laravel	Generar un Excel prototipo para poder realizar la importación en caso de que el sistema mindobox no genere el Excel a cargar, una vez que se tenga se podrá modificar el formato de entrada al requerido Actualizar al desarrollador para la curva de aprendizaje	C:\Centro_de_Informacion\SGRP\Documentacion	Sprint_SGRP_01	100hrs	160hrs
Consulta	alta	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y mostrarlos en la interfaz	06/03/2023	Producción	ninguno	no es requerido	C:\Centro_de_Informacion\SGRP\Documentacion	Sprint_SGRP_03	80hrs	120hrs

Baja	media	Dar de baja a los alumnos que: solicitado detener el proceso de residencias profesionales Han sido dados de baja del ITZ Que no pertenecen o no cumplen los requisitos para el proceso de residencias profesionales	06/03/2023	Producción	ninguno	no es requerido	C:\Centro_de_Informacion\SGRP\Documentacion	Sprint_SGRP_02	80hrs	120hrs
Modificación	media	Consultar por medio de filtros, pero en especial por número de control los alumnos dados de alta en el sistema y tener opción de modificar sus datos en caso de ser requerido, almacenando la modificación en el sistema	06/03/2023	Producción	ninguno	no es requerido	C:\Centro_de_Informacion\SGRP\Documentacion	Sprint_SGRP_04	80hrs	120hrs

Tabla 16.Resultado de plan de trabajo

4.5.3.3 Calendario de actividades

A través del calendario de actividades se analiza y compara cómo los datos reales difieren de las estimaciones iniciales. Esta comparación proporciona una visión crítica de la gestión del tiempo en el proyecto y permite identificar desviaciones, retrasos o adelantos en relación con lo planificado. Mostrando la importancia de esta evaluación para mejorar la precisión en la planificación de proyectos futuros.

ACTIVIDAD	Mes	Marzo					Abril				Mayo				
	Semana	1	2	3	4	5	1	2	3	4	1	2	3	4	5
Sprint_SGRP_01	Estimado														
	Real														
Sprint_SGRP_02	Estimado														
	Real														
Sprint_SGRP_03	Estimado														
	Real														
Sprint_SGRP_04	Estimado														
	Real														
OBSERVACIONES	Se tuvieron contratiempos en la realización de modulo por la curva de aprendizaje en laravel , esto afectando el primer sprint														

Tabla 22. Resultado de cronograma de actividades con tiempos planeados

4.5.4 Etapa de aplicación

4.5.4.1 Ejecución de Sprint

Antes de Ejecutar el primer Sprint, se reajustó el código para poder desarrollar de manera estructurada, integrando el framework laravel en su versión 7.0, Siendo una ventaja por sus múltiples bondades, sin embargo, se tuvo un atraso por la curva de aprendizaje el desarrollador para poder realizar esta actividad. **Véase Tabla .**

ACTIVIDAD	Mes	Marzo					Abril				Mayo				
	Semana	1	2	3	4	5	1	2	3	4	1	2	3	4	5
Sprint_SGRP_01	Estimado														
	Real														
Sprint_SGRP_02	Estimado														
	Real														
Sprint_SGRP_03	Estimado														
	Real														
Sprint_SGRP_04	Estimado														
	Real														
OBSERVACIONES	Se tuvieron contratiempos en la realización de módulo por la curva de aprendizaje en laravel, esto afectando el primer sprint														

Tabla 23. Resultado de Cronograma de actividades con tiempos reales

4.5.4.2 Team Testing

El equipo de desarrollo, en conjunto con el PA, realizaron pruebas para revisar la funcionalidad que se espera según lo comentado en el levantamiento de requerimiento, completando satisfactoriamente las pruebas aplicadas, cumpliendo con el requerimiento. **Véase, Figura 14. Inicio del portal general del ITZ , Figura 15. Resultado final de los Sprint programados, Figura 16. Opción para importar Excel, Figura 17. Interfaz después de dar de alta a los alumnos.**

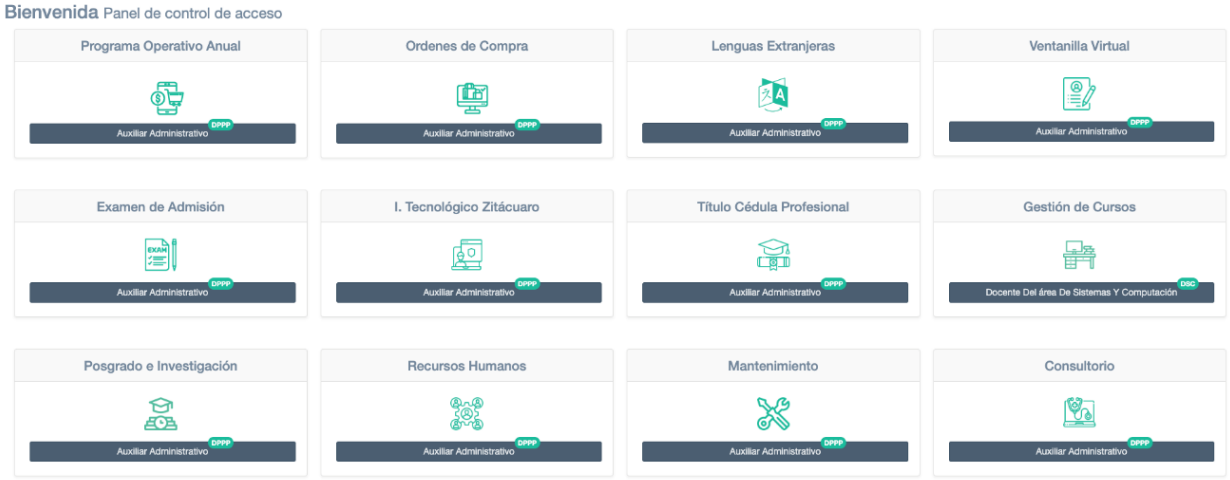


Figura 14. Inicio del portal general del ITZ

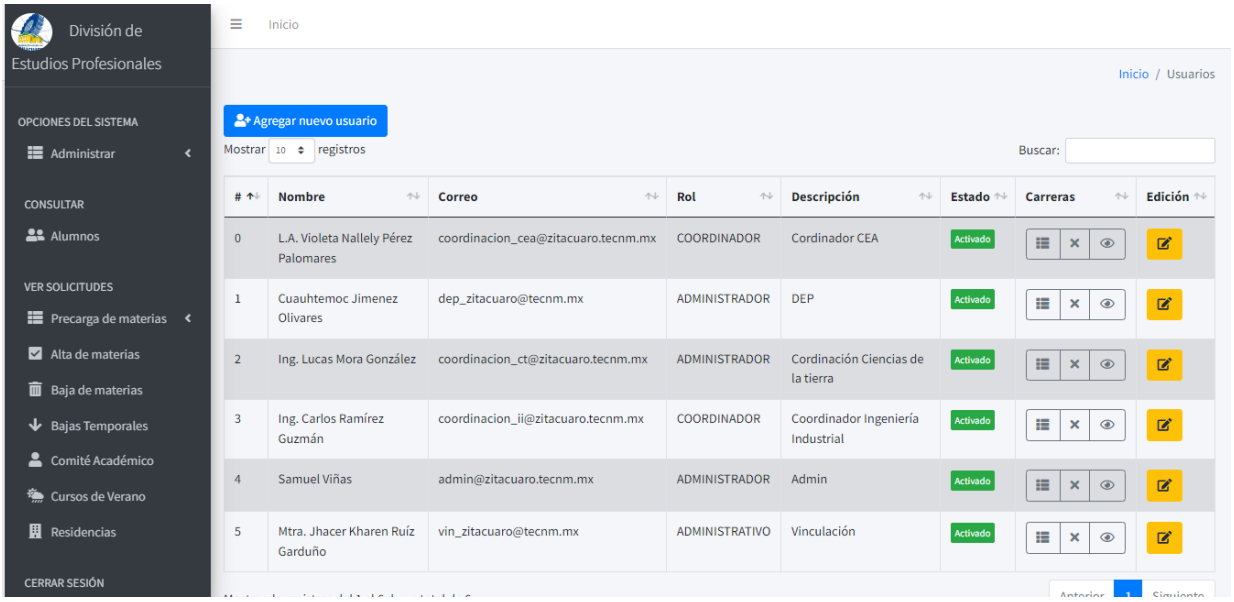


Figura 15. Resultado final de los Sprint programados

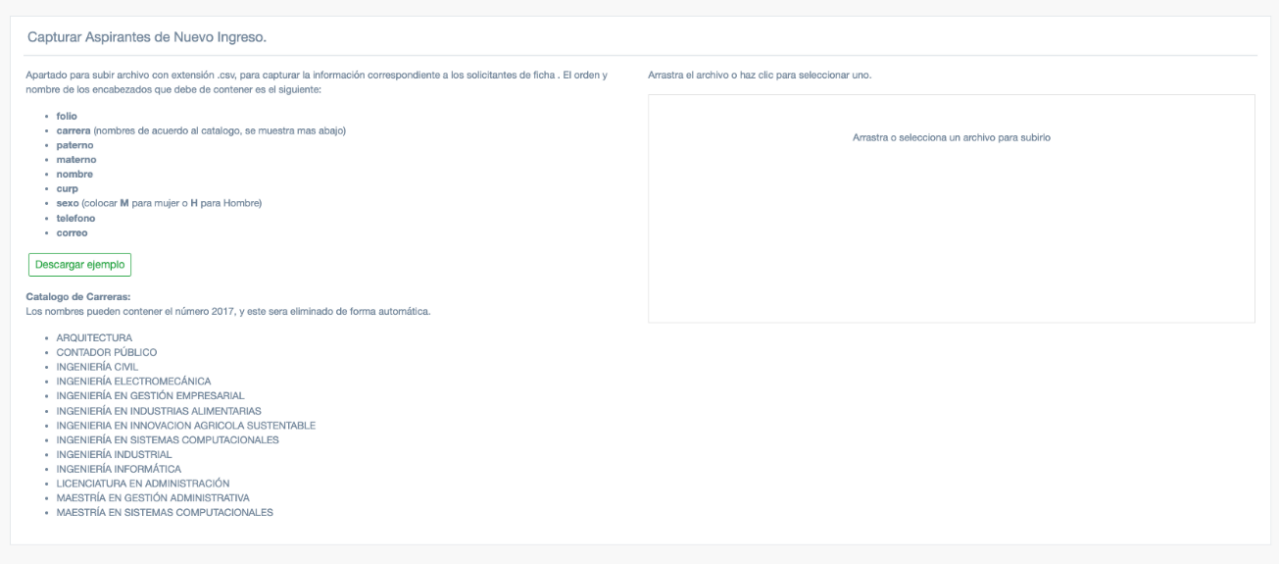


Figura 16. Opción para importar Excel

No	No.Solicitud	Nombre	Carrera	Fecha	Periodo	Contraseña
1	2816	Gabriel Hernández Mendoza	I.G.E.	02-02-2021	2021-1	
2	2817	Erick Dominguez Barrera	I.G.E.	02-02-2021	2021-1	
3	2818	Kimberly Avendaño Garcia	I.I.A.S.	02-02-2021	2021-1	
4	2819	Juan Daniel Suarez Iturbe	I.G.E.	02-02-2021	2021-1	
5	2820	David Pérez Hernández	I.I.	02-02-2021	2021-1	
6	2748	Jose Manuel Rosales Cruz	I.I.A.	02-02-2021	2021-1	
7	2749	Braulio Jesús Aguilar Morales	I.I.	02-02-2021	2021-1	
8	2750	Angel Armando Urbano Sánchez	I.S.C.	02-02-2021	2021-1	
9	2751	Lizeth Tapia Figueroa	I.G.E.	02-02-2021	2021-1	
10	2752	Jesús Asaf Viñas Alvarado	I.S.C.	02-02-2021	2021-1	
11	2753	Paola Samantha Sánchez Salazar	I.C.	02-02-2021	2021-1	
12	2754	Luis Fernando Hernández Martínez	ARQ.	02-02-2021	2021-1	
13	2756	Gustavo Alberto Dominguez Perez	I.E.	02-02-2021	2021-1	
14	2757	David Josué González Esquivel	L.A.	02-02-2021	2021-1	
15	2768	Diego Sebastian Maya Navarro	I.E.	02-02-2021	2021-1	
16	2769	Yahir Gómez Sánchez	I.E.	02-02-2021	2021-1	
17	2771	Omar Ramírez García	L.A.	02-02-2021	2021-1	
18	2772	Maria Guadalupe Cruz Soto	C.P.	02-02-2021	2021-1	
19	2773	Alvaro Manuel Zacarias Sanchez	I.I.	02-02-2021	2021-1	
20	2774	Andrea Miranda Pérez	I.I.	02-02-2021	2021-1	
21	2775	Juan Carlos Gomez Hernandez	I.S.C.	02-02-2021	2021-1	
22	2776	Alvaro Martinez Méndez	I.I.	02-02-2021	2021-1	
23	2777	Juan Manuel Gómez Hernández	I.G.E.	02-02-2021	2021-1	
24	2778	Jair Rivera Garcia	I.C.	02-02-2021	2021-1	
25	2779	David Frasco Leon Frasco Materno	I.S.C.	02-02-2021	2021-1	
26	2780	Rodolfo Colin Medina	I.G.E.	02-02-2021	2021-1	
27	2781	Yahir Bustos Garcia	I.I.A.S.	02-02-2021	2021-1	
28	2782	Luna Estefania Medina Zacarias	I.I.A.	02-02-2021	2021-1	
29	2784	Cesar Jair Contreras Lopez	I.C.	02-02-2021	2021-1	
30	2786	Jeremi Joel Ramirez Jiemenez	I.E.	02-02-2021	2021-1	
31	2788	Dilan Rivas Bautista	I.S.C.	02-02-2021	2021-1	
32	2789	Luis Manuel Hernández Salazar	I.S.C.	02-02-2021	2021-1	
33	2790	Estefany Jordan Mendoza	I.S.C.	02-02-2021	2021-1	
34	2791	Axel Bernal Sarmiento	I.G.E.	02-02-2021	2021-1	
35	2792	Marcos Uriel Irineo Mendoza	I.C.	02-02-2021	2021-1	

Figura 17. Interfaz después de dar de alta a los alumnos

4.5.4.3 Entrega de Sprint

La entrega de los Sprints ha sido pospuesta hasta la entrega del módulo de proyectos, por cuestiones de agenda de los involucrados, sin embargo, el usuario está en el entendido de las funciones generadas, ya que se mantiene en pláticas la funcionalidad del módulo de proyectos.

CAPÍTULO V RESULTADOS Y ANÁLISIS

En este capítulo se muestra los resultados obtenidos al implementar el modelo propuesto en un caso de estudio real con el equipo de desarrollo del ITZ.

El equipo de desarrollo del ITZ realizó la aplicación de la metodología desarrollada con el fin de comprobar la viabilidad de la mejora de tiempos de desarrollo y fomentar la documentación de proyectos en el ITZ.

Acorde al análisis de requerimientos y necesidades, se determinó que el caso de estudios es extenso, siendo esto una limitante para obtener un muestreo de resultados para el presente trabajo; por lo que se decidió solo la implementación al módulo de alumnos que es parte del Sistema de Gestión de Residencias Profesionales. Teniendo como propósito empezar a documentar el desarrollo del proyecto y como visión aplicarlo en todos sus módulos que puedan generarse a futuro una vez que se determinen todas las reglas de negocio que abarcara con las diferentes áreas del ITZ.

Se espera comparar el proceso que lleva a cabo el área de desarrollo actualmente con el modelo que se propone, para así concluir con un dictamen, el cual de la implementación del modelo para gestionar los proyectos de desarrollo del ITZ. En caso de no ser viable, se analizarán los resultados y se actuará con una propuesta de mejora con el fin poderlo adaptar en su totalidad y pueda ser aplicado en los diferentes sistemas.

5.1 Resultados

Después de aplicar el modelo de reingeniería de software propuesto, se determinó que es viable aplicarlo por los siguientes puntos:

- Al momento de realizar la entrevista se detectó que el PA y el Dev se vieron beneficiados, ya que las preguntas propuestas fueron de guía para poder extraer información al usuario. Anteriormente, estas entrevistas duraban de una a dos semanas, ya que no se tenía un objetivo sobre que preguntar. Al momento de aplicar la entrevista se redujo a solo dos sesiones, donde la primera se enfocó a la entrevista y la segunda para resolver dudas. **Véase Tabla 17. Gráfica comparativa de tiempos de entrevista.** Podemos observar que solo se realizaron 2 sesiones donde gracias a tener un objetivo claro de entrevista, los tiempos pueden variar por el tiempo de respuesta del entrevistado, sin embargo, las reglas de negocio se entienden mejor cuando se evita la divagación

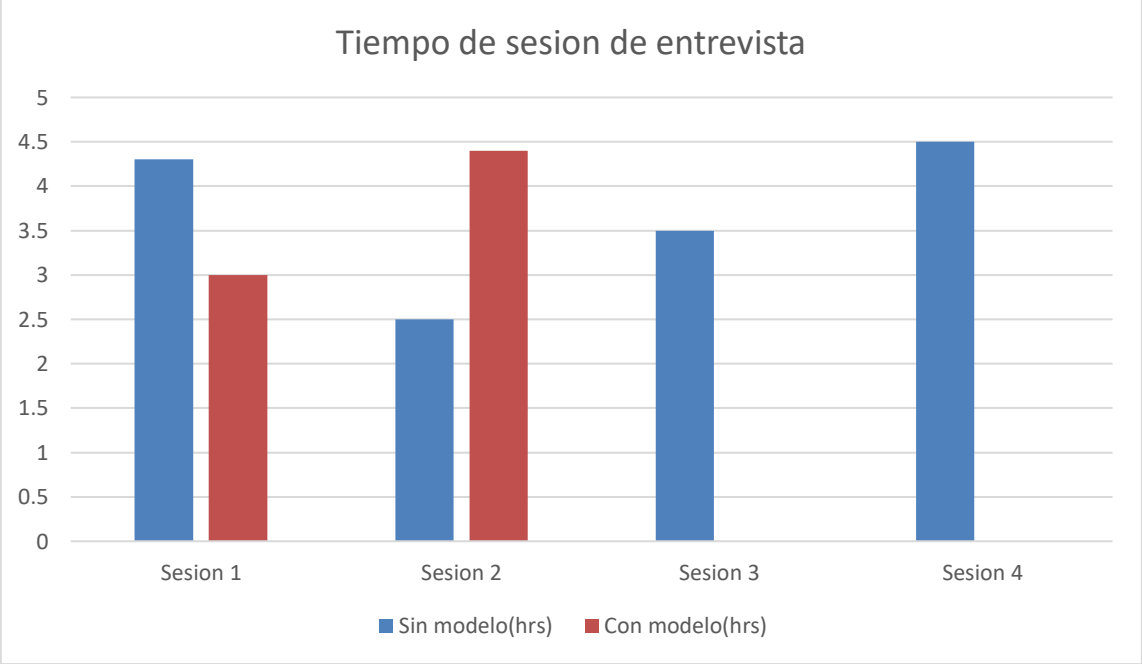


Tabla 17. Gráfica comparativa de tiempos de entrevista

- Los formatos de evaluación ayudaron a entender las funciones básicas del sistema a pesar de no tener la documentación, así mismo también ayudaron a cuestionar el funcionamiento del proyecto, siendo esta la razón por la cual se extendió el desarrollo para tomar en cuenta las reglas de negocio de las demás áreas involucradas.

- Se generó y fomento la documentación para el proyecto, siendo un punto favorable para el desarrollo de futuros mantenimientos. En palabras del equipo de desarrollo, mencionan que el llenado de documentos fue breve, limitándose a lo esencial sin olvidar el objetivo general.

- La importación masiva de alumnos fue una propuesta derivada del análisis correcto de requerimientos propuesto por el modelo en cuestión, el cual fue generado de manera que pueda ser reutilizado para el módulo de docentes.

- Se obtuvo conocimiento de diferentes técnicas de desarrollo, gracias a la investigación realizada y expuesta al equipo.

5.2 Comparativa entre el Modelo Propuesto y Modelos Tradicionales de Reingeniería de Software: Enfoques y Diferencias Centrales

El modelo propuesto enfatiza una interacción profunda con los usuarios mediante entrevistas estructuradas. Estas entrevistas, reducen drásticamente el tiempo que anteriormente podría haberse gastado en múltiples sesiones sin dirección clara. Esta técnica contrasta especialmente con el Modelo de Herradura, donde el inicio se basa en un estudio y documentación detallada del sistema existente, antes de embarcarse en una transformación y reconstrucción.

A diferencia del modelo propuesto, que pone un fuerte énfasis en la documentación esencial, Scrum, que es más un marco ágil que un modelo de reingeniería puro, se centra en un proceso iterativo e incremental. A través de sprints, el equipo de desarrollo, guiado por roles específicos como el Scrum Master y el Product Owner, trabaja en la realización de tareas priorizadas. Si bien hay un elemento de revisión y adaptación en Scrum, la estructuración dirigida de entrevistas y la creación de documentación no son centrales como en el modelo que se propone.

El método OAR es un viaje desde la observación hasta el rediseño, pasando por el análisis, el cual se lleva a cabo después de la observación directa del sistema actual, llevando eventualmente a un rediseño basado en las fortalezas y debilidades identificadas. El modelo propuesto, mientras tanto, integra una interacción directa con los usuarios desde el principio, garantizando que las necesidades se comprendan y se aborden desde el inicio.

Por otro lado, el Modelo Cíclico lleva a los equipos a través de ciclos de identificación de problemas, planificación, implementación y evaluación. Es un enfoque repetitivo que se asemeja al aspecto iterativo de Scrum, pero con un enfoque más marcado en la reingeniería. Aunque el modelo propuesto puede tener iteraciones implícitas, su énfasis no está en un ciclo repetido, sino en la claridad desde el comienzo mediante entrevistas y documentación.

Finalmente, una característica destacada del modelo es el enriquecimiento del equipo. A través de la investigación y el aprendizaje de los proyectos, el equipo no solo trabaja en el proyecto, sino que también mejora sus propias habilidades y conocimientos, algo que no se destaca tanto en los otros modelos mencionados.

Basado en los resultados y contribuciones mencionados, es evidente que el modelo de reingeniería de software propuesto ha tenido un impacto positivo en el proceso de desarrollo y gestión de proyectos del ITZ. Analicemos de forma cualitativa los beneficios y contribuciones del modelo:

Eficiencia en las Entrevistas

- Antes: Las entrevistas se alargaban por varios días debido a la falta de estructura y claridad en las preguntas.
- Después: La estructuración de preguntas resultó en entrevistas reducidas a solo dos sesiones, lo que indica una comunicación más efectiva y eficiente.

Documentación enfocada

- Antes: La falta de documentación era un problema para el desarrollo y mantenimiento del software.
- Después: El modelo propuesto enfatizó la documentación, y se logró crear una que es concisa, enfocada y esencial para el mantenimiento futuro.

Comprensión mejorada del Proyecto

- Antes: Había confusión o falta de claridad sobre ciertas funciones básicas del sistema.
- Después: Los formatos de evaluación proporcionaron una herramienta esencial para entender y cuestionar el proyecto, incluso en ausencia de documentación.

Reutilización y Extensibilidad: El modelo propuesto no solo abordó las necesidades actuales, sino que también pensó en el futuro. La importación masiva de alumnos es un ejemplo de una característica que fue diseñada con la reutilización en mente para futuros módulos.

Formación Continua del Equipo: El modelo no solo sirvió como una herramienta para mejorar el software existente, sino también como una oportunidad de aprendizaje para el equipo, familiarizándose con diferentes técnicas de desarrollo.

La implementación del modelo de reingeniería de software en el ITZ ha demostrado ser no solo viable, sino altamente beneficioso. Los puntos destacados anteriormente subrayan las áreas de mejora y optimización que el modelo introdujo en el proceso de desarrollo. Es evidente que el modelo puede actuar como una valiosa herramienta en proyectos similares, permitiendo una recopilación de requerimientos más estructurada, una documentación más robusta y una mejora general en la eficiencia del desarrollo de software.

5.2 Contribuciones a la reingeniería de software

Se identificó una mejora significativa en el proceso de entrevistas para la recopilación de requerimientos gracias a la estructuración de preguntas, se redujo el tiempo de entrevista, facilitando una comunicación más directa y eficaz con los usuarios. Las entrevistas pasaron de durar días a solamente dos sesiones demostrando que es una metodología eficaz para extracción de información.

Los formatos de evaluación se destacaron como herramientas cruciales para entender las funciones básicas del sistema, incluso en ausencia de documentación previa. Estas herramientas también ayudaron a cuestionar y comprender a fondo el funcionamiento del proyecto.

Una de las principales contribuciones fue el énfasis y el impulso dado a la documentación de proyectos. No solo se generó documentación, sino que se hizo de manera concisa y relevante, atendiendo lo esencial sin perder de vista el objetivo general. Esta documentación es crucial para el mantenimiento y la escalabilidad del software en el futuro. Ademas con la metodologia se fomenta la investigación y el aprendizaje de nuevas técnicas de desarrollo es esencial para mantener la relevancia de los sistemas legados en un mundo tecnológico en constante evolución. Mantenerse al tanto de las mejores prácticas actuales y explorar cómo estas pueden aplicarse para mejorar la calidad y la eficiencia del sistema es esencial para el éxito a largo plazo de sistemas heredados.

La implementación del modelo no solo permitió la reingeniería del software en sí, sino que también facilitó el aprendizaje y la adquisición de diferentes técnicas de desarrollo por parte del equipo. Este conocimiento fue adquirido a través de la investigación y compartido con todo el equipo, potenciando sus habilidades y capacidades. La implementación exitosa del modelo propuesto en el ITZ es un testimonio de su eficacia

y de las contribuciones significativas que puede aportar al campo de la reingeniería de software.

La reutilización de componentes se demostró como una práctica beneficiosa. El caso de la importación masiva de alumnos ejemplifica cómo identificar componentes reutilizables puede acelerar el desarrollo y mejorar la eficiencia del sistema. Este enfoque puede aplicarse a sistemas legados para identificar partes del sistema que pueden ser utilizadas en diferentes áreas, acelerando así el proceso de mejora y actualización.

En resumen, las lecciones extraídas de la reingeniería de software ofrecen valiosas perspectivas para sistemas legados. Al optimizar procesos, enfocarse en documentación esencial, reutilizar componentes y mantener un enfoque en el aprendizaje continuo, es posible mejorar significativamente la gestión de sistemas heredados y mantenerlos relevantes en un entorno tecnológico en constante evolución.

Conclusiones

La aplicación del modelo de reingeniería de software propuesto en el Instituto Tecnológico de Zitácuaro ha marcado un hito significativo en la modernización de sistemas legados, demostrando no solo la viabilidad de tal enfoque sino también sus beneficios tangibles en términos de productividad y eficiencia operacional. La metodología adoptada, basada en una exhaustiva revisión de las prácticas actuales y emergentes en el campo de la reingeniería, ha permitido no solo alcanzar los objetivos específicos planteados sino también superar las expectativas iniciales en algunos aspectos.

El proceso comenzó con una meticulosa investigación y análisis de metodologías de reingeniería existentes, lo que permitió construir una sólida base teórica y adaptar las estrategias más efectivas a las necesidades y desafíos específicos de los sistemas legados que existen en el ITZ. Esta fase inicial fue crucial para establecer un marco de trabajo que guiaría todo el proyecto, asegurando que cada paso estuviera alineado con nuestro objetivo general de mejorar la productividad y eficiencia.

La selección de un caso de estudio relevante y representativo proporcionó un campo de prueba ideal para aplicar y evaluar el modelo de reingeniería. Al sumergirnos en el análisis y diseño del sistema de residencias profecionales, no solo pudimos demostrar la aplicabilidad del enfoque sino también identificar oportunidades para la optimización y reutilización de componentes, un aspecto subrayado por nuestro estudio de técnicas de Ingeniería de Dominio.

La implementación de prácticas orientadas a mejorar los tiempos de desarrollo y mantenimiento ha sido particularmente de gran beneficio para los futuros desarrollos del ITZ. A través de este enfoque, hemos podido experimentar de primera mano cómo la reingeniería puede facilitar la adaptación de sistemas legados a las demandas modernas, manteniendo al mismo tiempo la integridad y funcionalidad esenciales del

sistema original. Esto no solo ha beneficiado al equipo de desarrollo del ITZ en términos de eficiencia operativa sino que también ha establecido un precedente para futuras iniciativas de modernización.

El desarrollo y las pruebas del modelo han sido etapas fundamentales, ofreciendo conocimientos valiosos sobre su robustez y flexibilidad. A pesar de enfrentar contratiempos temporales, los aprendizajes obtenidos de estas experiencias han enriquecido enormemente nuestro entendimiento de cómo abordar y mitigar desafíos similares en el futuro.

La documentación generada y los conocimientos compartidos con el equipo de desarrollo no solo han mejorado la gestión actual de sistemas legados, sino que también han proporcionado una base sólida para la evolución continua de estas plataformas. Este aspecto es crucial para asegurar la sostenibilidad a largo plazo de los sistemas de software, en un entorno tecnológico que está en constante evolución.

En última instancia, la aplicación del modelo de reingeniería ha demostrado que es posible trascender las limitaciones tradicionalmente asociadas con los sistemas legados, abriendo nuevas vías para su evolución y adaptación en la era digital. Este proyecto ha subrayado la importancia de la reingeniería no solo como una herramienta para la modernización tecnológica sino también como un enfoque estratégico para la gestión de sistemas de software, enfatizando la necesidad de adaptabilidad, eficiencia y sostenibilidad.

Mirando hacia el futuro, este proyecto sienta las bases para una exploración más profunda de cómo las tecnologías emergentes y las metodologías de desarrollo ágil pueden integrarse aún más en la reingeniería de sistemas legados, potenciando la innovación y asegurando que estas plataformas críticas puedan continuar apoyando las operaciones del ITZ de manera efectiva y eficiente. La travesía de modernización emprendida con este proyecto no solo valida nuestra hipótesis inicial, sino que también

abre el camino hacia futuras transformaciones, marcando un punto de inflexión en cómo abordamos la evolución de los sistemas de software legados, donde marcamos con el éxito del modelo propuesto, representando una reevaluación fundamental de cómo abordamos la modernización de lo que antes considerábamos reliquias digitales o de software. Lejos de ser los vestigios obsoletos de eras pasadas, hemos descubierto en los sistemas legados un potencial latente, esperando ser liberado mediante enfoques innovadores y colaborativos.

El viaje hacia este descubrimiento no ha sido simplemente técnico; ha sido una travesía de transformación cultural dentro de nuestra organización. A través de este proyecto, hemos aprendido que los sistemas legados, con todas sus peculiaridades y desafíos, no solo pueden integrarse en la presente digital, sino que pueden servir como cimientos sobre los cuales construir un futuro aún más innovador. Las metodologías de modernización que hemos adoptado y adaptado, desde la reutilización de componentes hasta la integración de tecnologías emergentes, han redefinido lo que significa revitalizar la tecnología antigua.

Más allá de la técnica, hemos fomentado un habito de colaboración y conocimiento compartido, creando un ambiente donde el intercambio de ideas no es solo bienvenido sino esencial. Esta cultura de innovación abierta ha sido clave para superar los desafíos inherentes a los sistemas legados, permitiéndonos no solo ver más allá de sus limitaciones sino también encontrar maneras de convertir esos obstáculos en oportunidades.

El enfoque adaptable y sostenible que se ha desarrollado prepara al desarrollador no solo para el presente sino también para el futuro, asegurando que los sistemas que hoy se modernizan puedan seguir evolucionando. Este enfoque garantiza que la tecnología no sólo sobreviva, sino que prospere, adaptándose a las necesidades cambiantes y a las nuevas posibilidades que el futuro depara al ITZ.

En definitiva, este proyecto ha sido mucho más que la reingeniería de software; ha sido una reinvención de cómo se visualiza el legado tecnológico. Al marcar este punto de inflexión, no solo ha cambiado la trayectoria del sistema de residencias profesionales, sino también la forma en que nuestra organización aborda la innovación y el crecimiento. Hemos aprendido que, con la perspectiva y las herramientas adecuadas, incluso los sistemas más antiguos pueden transformarse en activos valiosos para el futuro, demostrando que la evolución tecnológica es tanto una cuestión de perspectiva como de práctica.

Recomendaciones y Trabajos Futuros

Basado en los resultados y conclusiones obtenidas, determinamos que la implementación y la mejora continua del modelo propuesto de reingeniería de software puede tener un gran futuro en la Institución, tomando en cuenta los siguientes puntos:

- **Ampliación de la aplicación del modelo:** A partir de los resultados exitosos obtenidos en la aplicación del modelo de reingeniería de software en el módulo de alumnos, se recomienda extender su implementación a otros aplicativos. Esto permitirá estandarizar y mejorar los procesos de desarrollo en todo el ITZ, maximizando así los beneficios del modelo.

- **Análisis detallado de las limitaciones identificadas:** Para superar las limitaciones y contratiempos en el desarrollo identificados en el texto, se sugiere realizar un análisis detallado de cada uno de ellos. Identificar las causas subyacentes y explorar posibles soluciones permitirá implementar estrategias más efectivas en futuras aplicaciones del modelo.
- **Evaluación de los beneficios a largo plazo:** Es recomendable realizar un seguimiento de los beneficios obtenidos a largo plazo a través de la implementación del modelo de reingeniería de software. Esto implicaría realizar mediciones y recopilar datos relevantes para evaluar cómo se han traducido los beneficios en mejoras tangibles, como la eficiencia del desarrollo, la calidad del software y la satisfacción de los usuarios.

- **Retroalimentación y mejora continua:** Fomentar un ciclo de retroalimentación y mejora continua será fundamental para perfeccionar el modelo de reingeniería de software. Involucrar a todo el equipo de desarrollo en sesiones de retroalimentación periódicas, donde se compartan lecciones aprendidas, se

identifiquen áreas de mejora y se propongan acciones correctivas, permitirá optimizar el proceso y asegurar un crecimiento continuo.

- **Investigación y adopción de nuevas técnicas de desarrollo:** Dado que el equipo de desarrollo ha obtenido conocimiento de diferentes técnicas de desarrollo a través de la investigación, se recomienda fomentar una cultura de aprendizaje constante y la exploración de nuevas metodologías y enfoques. Esto incluiría la capacitación continua del equipo en áreas relevantes y la evaluación de prácticas ágiles o nuevas tecnologías que puedan mejorar aún más el proceso de desarrollo.

- **Comunicación y colaboración interdisciplinaria:** Dado que el proyecto general involucra diferentes áreas, se sugiere promover una comunicación fluida y una colaboración efectiva entre los equipos de desarrollo y los usuarios finales. Esto facilitará la comprensión de las necesidades y requisitos de todas las partes interesadas, mejorará la coordinación y asegurará una implementación exitosa del modelo en todo el instituto.

Al enfocarse en estos aspectos, el ITZ podrá fortalecer la aplicación del modelo de reingeniería de software, mejorar sus prácticas de desarrollo y lograr un impacto positivo y sostenible en la gestión de proyectos y mantenimientos de software.

Referencias

Castillo, A., & Anaya, M. (June de 2013). Desarrollo de Aplicaciones WEB por Componentes – Código Libre. *Publicaciones e Investigación, 7*, 33. doi:10.22490/25394088.1092

Cervantes, H., castro, l., & Velasco-Elizondo, P. (April de 2016). 2015 ArquitecturaDeSoftware-C1-3. *2015 ArquitecturaDeSoftware-C1-3*.

Comella-Dorda, S., Wallnau, K., Seacord, R., & Robert, J. (2000). *A Survey of Legacy System Modernization Approaches.* Tech. rep., Software Engineering Institute, Carnegie Mellon University, Pittsburgh. Obtenido de http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5093

IESALC. (September de 2021). Informe del IESALC analiza los impactos del #COVID19 y ofrece recomendaciones a gobiernos e instituciones de educación superior. *Informe del IESALC analiza los impactos del #COVID19 y ofrece recomendaciones a gobiernos e instituciones de educación superior*. Obtenido de https://www.iesalc.unesco.org/2020/04/14/iesalc-insta-a-los-estados-a-asegurar-el-derecho-a-la-educacion-superior-en-igualdad-de-oportunidades-ante-el-covid-19/

Álvarez García, J. C., Mateos Sánchez, M., & Moreno García, M. N. (January de 2004). METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACIÓN DE APLICACIONES CIENTÍFICAS HEREDADAS. *METODOLOGÍA DE REINGENIERÍA DEL SOFTWARE PARA LA REMODELACIÓN DE APLICACIONES CIENTÍFICAS HEREDADAS*. Obtenido de https://gredos.usal.es/bitstream/handle/10366/21762/DPTOIA-IT-2004-003.pdf;jsessionid=046B234106D51BA5A92312C05467E203?sequence=1

Aier, S., Riege, C., & Winter, R. (2009). Classification of Enterprise Architecture Scenarios with regard to Knowledge Integration. *Journal of Enterprise Architecture, 5*, 11–20.

Arnold, R. S. (1993). *Software reengineering.* Los Alamitos, California: IEEE Computer Society Press.

Atlantic Universal University. (11 de July de 2022). *Metodos y modelos de la reingeniería de software.* Obtenido de Atlantic Universal University: https://cursos.aiu.edu/REINGENIER%C3%8DA%20EN%20SISTEMAS/9/PDF/Reingenieria%20de%20sistemas%20sesion%209.pdf

AVELLA IBÁÑEZ, C. (2004). REINGENIERÍA DE SOFTWARE E INGENIERÍA REVERSA SOFTWARE REENGINEERING AND REVERSE ENGINEERING.

Aversano, L., Grasso, C., & Tortorella, M. (2016). Managing the alignment between business processes and software systems. *Inf. Softw. Technol., 72*, 171–188.

Baldonedo, J. A. (2017). *Modelo CMMI y metodos Agiles en la gestion de proyectos de software.* Obtenido de Repositorio Institucional de la Universidad de Oviedo: http://hdl.handle.net/10651/43638

Bianchi, A. (April de 2008). Framework de mejora de procesos de desarrollo de software. *Framework de mejora de procesos de desarrollo de software*. Obtenido de http://sedici.unlp.edu.ar/handle/10915/4075

Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy information systems: issues and directions. *IEEE Software, 16*, 103-111. doi:10.1109/52.795108

Brodie, M., & Stonebraker, M. (1995). *Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach* (1 ed.). Morgan Kaufmann Pub.

Brown, W., McCormick, H., & Thomas, S. (2000). *Anti-patterns in Project Management.* Nashville, TN, Estados Unidos de América: John Wiley & Sons.

Franco Bianchiotti, S. C. (2014). Guía para la reingeniería de sistemas legados: una experiencia práctica y real. *Revista latinoamericana de ingeniería de software*, 99. Obtenido de https://doi.org/10.18294/relais.2014.99-106

Frank Turley, N. K. (23 de octubre 2019). *Los Fundamentos de Agile Scrum.* Van Haren Publishing; Edición 1st.

Instituto Nacional de Estadística y Geografía. (s.f.). Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2021. *Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares (ENDUTIH) 2021*.

Instituto Tecnologico de Zitacuaro. (14 de 01 de 2023). *TecNM | Instituto Tecnologico de Zitacuaro.* Obtenido de zitacuaro.tecnm.mx: https://zitacuaro.tecnm.mx/descargarAcademico?url=/pdf/sistema_gestion/academico/Procedimientos_1570750782.%20RES.%20RES.%20RES.pdf

Palomo, G. S. (2020). *Aproximación a la ingeniería del software* (2 ed.). Madrid : Centro de Estudios Ramón Areces, 2020.

Pressman. (2022). *Ingenieria De Software* (7 ed.). MCGRAW HILL EDDUCATION.

Red de Universidades con Carreras en Informática (RedUNCI) . (April de 2022). *Técnicas de reuso dentro de la Ingeniería del Dominio.* Tech. rep. Obtenido de http://sedici.unlp.edu.ar/handle/10915/21531

Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., & Bettin, J. (2013). Ingeniería de dominio: líneas de producto, lenguajes y modelos conceptuales. *Ingeniería de dominio: líneas de producto, lenguajes y modelos conceptuales*. Springer Science + Business Media . ISBN 978-3-642-36654-3. Obtenido de https://hmong.es/wiki/Domain_engineering

Scrum Guia. (22 de 12 de 2022). *Scrum Guia*. Obtenido de Scrum Guia: https://scrumguides.org/scrum-guide.html

Seacord, R. C., Plakosh, D., & Lewis, G. A. (2003). *Modernizing legacy systems: Software technologies, engineering processes, and business practices.* Boston, MA, Estados Unidos de América: Addison-Wesley Educational.

Sommerville, I., Galipienso, M. I., & Martinez, A. B. (2005). *Ingenieria del Software.* Pearson Educacion.

UNESCO. (November de 2022). Comunicados de prensa. *Comunicados de prensa*. Obtenido de https://www.itu.int/es/mediacentre/Pages/PR-2022-11-30-Facts-Figures-2022.aspx