

Doctorado en Informática

Gestión de deuda técnica en modelos ágiles de desarrollo de software bajo enfoque basado en valor

Edwin Fabricio Lozada Torres

Director: Dr. Rodrigo Cadena Martínez

Ecuador, Ambato, mayo 2026

Universidad Americana de Europa (ROR: https://ror.org/03698ym41)

Resumen

El principal desafío para los desarrolladores de software es garantizar un producto de calidad, no obstante, la presión por el cumplimiento de cronogramas y los principios ágiles conlleva a una serie de problemas que aumentan el riesgo de un software ineficiente. Al producir código y documentación con errores se adquiere una deuda de trabajo que, tiene como consecuencia demoras en la entrega y un producto de baja calidad, esto genera trabajo extra al corregirlo que es conocido como Deuda Técnica, que se incrementa con el tiempo generando intereses en algunos casos impagables. El estudio desarrolla un marco para la gestión de la deuda técnica en modelos de desarrollo ágiles, que proporciona un marco conceptual para controlarla y reducirla, lo que permitirá obtener un enfoque para comprenderla y gestionarla. Se realiza un caso de estudio con el equipo de desarrollo de una institución de educación superior, con datos recopilados de una entrevista semiestructurada en la cual se aplicó el método cualitativo para una mejor comprensión del problema. También se empleó un enfoque cuantitativo con métricas que comprobó la necesidad de un marco que permita gestionar la deuda técnica con actividades para prevenirla, identificarla, medirla y controlarla, para lo cual se plantea el marco de gestión basado en el valor. El marco demuestra la importancia de utilizar un enfoque sistemático para mitigar la deuda técnica con lo cual se logró una notable reducción de esta, obteniendo un código eficiente y mejorando el impacto en la calidad del software y la productividad del equipo.

Palabras clave: Deuda Técnica, Gestión de la Deuda Técnica, Modelos de Desarrollo de Software, Modelos Agiles, Marco de Gestión de la Deuda Técnica

Abstract

The main challenge for software developers is ensuring a quality product. However, the pressure to meet deadlines and adhere to agile principles leads to a series of problems that increase the risk of inefficient software. Producing code and documentation with errors creates work debt, resulting in delivery delays and a low-quality product. This generates extra work to correct the errors, known as technical debt, which increases over time and, in some cases, accrues unpayable interest. This study develops a framework for managing technical debt in agile development models, providing a conceptual framework for controlling and reducing it, thus enabling a better approach to understanding and managing it. A case study was conducted with the development team of a higher education institution, using data collected from a semi-structured interview in which a qualitative method was applied for a better understanding of the problem. A quantitative approach using metrics was also employed, confirming the need for a framework to manage technical debt through activities to prevent, identify, measure, and control it. To this end, a value-based management framework was proposed. This framework demonstrates the importance of using a systematic approach to mitigate technical debt, resulting in a significant reduction, efficient code, and improved impact on software quality and team productivity.

Keywords: Technical Debt, Technical Debt Management, Software Development Models, Agile Models, Technical Debt Management Framework

Dedicatorias

A mi esposa, hijos y nietas

por ser mi fuente constante de inspiración,

por su amor incondicional y su apoyo diario,

que me impulsan a alcanzar cada uno de mis objetivos,

por acompañarme en cada paso de este camino,

y por creer en mí incluso en los momentos más difíciles.

Agradecimientos

A Dios, por sus bendiciones y ser guía constante en cada paso de este camino,

por fortalecer mi espíritu y renovar mi esperanza en los momentos de dificultad.

A mis padres, por inculcarme principios sólidos,

por su ejemplo de integridad, esfuerzo y dedicación,

y por brindarme el amor y el apoyo que han sido fundamentales en esta etapa de mi vida.

Contenido

Resumen....ii
Dedicatorias....iv
Agradecimientos....v
Contenido....vi
Lista de tablas y figuras....vii
Tablas....vii
Figuras....viii
Introducción....1
Descripción del problema....1
Hipótesis....3
Justificación....3
Objetivos....5
1. Marco de referencia....7
Marco Teórico....7
Estado del arte....31
2. Metodología, Diseño, Software o Implementación propuesta....55
3. Pruebas y Resultados....93
4. Conclusiones y Recomendaciones....153
Referencias....156
Apéndices....177
Glosario....184

Lista de tablas y figuras

Tablas

Tabla 1. 15

Tabla 2. 41

Tabla 3. 57

Tabla 4. 57

Tabla 5. 58

Tabla 6. 58

Tabla 7. 59

Tabla 8. 60

Tabla 9. 100

Tabla 10. 100

Tabla 11. 102

Tabla 12. 110

Tabla 13. 132

Tabla 14. 133

Tabla 15. 136

Tabla 16. 137

Tabla 17. 138

Tabla 18. 139

Tabla 19. 143

Tabla 20. 145

Figuras

Figura 1. 75

Figura 2. 89

Figura 3. 144

Figura 4. 144

Figura 5. 145

Figura 6. 145

Introducción

Descripción del problema

La agilidad en el desarrollo de software se ha adaptado a aquellos proyectos con naturaleza cambiante y con clientes que requieren un retorno inmediato de su inversión, esto se deriva en una de sus características principales que es la entrega de productos de calidad de acuerdo con el tiempo establecido. Sin embargo, uno de los atributos importantes de los modelos ágiles que es la calidad por lo general no se cumple.

El principal desafío para los desarrolladores en un proyecto de software es garantizar una correcta implementación con agilidad, no obstante, la capacidad del equipo de desarrollo para entregar valor bajo presión y con rigor metodológico conlleva a una serie de problemas que aumentan el riesgo a una falta en los controles de calidad del producto (Traini, 2022).

El dar prioridad al proceso de desarrollo rápido se deriva en varios problemas como consecuencia de los atajos técnicos que toma el equipo para cumplir los plazos de entrega, siendo una de las consecuencias los errores y deficiencias en el código y en la documentación. Esto se agrava por una mala gestión de los errores ya que casi siempre los gestores del producto priorizan el crear nuevas funciones antes que corregirlos (Saeeda et al., 2022).

Al producir código y documentación con deficiencias se adquiere una deuda de trabajo que tiene como consecuencia que la entrega del producto se prolongue o que se implemente un producto de pobre calidad, esto genera un costo al corregirlo que es conocido como Deuda Técnica, que tarde o temprano se la debe pagar y que aumenta cuanto más tiempo pase generando intereses que se acarrean a medio y largo plazo (Besker et al., Technical Debt Cripples Software Developer Productivity: A Longitudinal Study on Developers' Daily Software Development Work, 2018).

Cuanto más se tarde la Deuda Técnica en controlarse, más difícil es pagarla, por lo que es necesario una adecuada gestión que permita identificarla, entenderla, exponerla y remediarla de forma transparente (Avgeriou et al., 2024).

Para esto se plantea diseñar un marco conceptual para un Modelo en el cual se definan técnicas que permitan el tratamiento adecuado de la deuda técnica en ambientes de desarrollo ágiles con la incorporación de tecnologías y herramientas para tratar la deuda técnica, lo que contribuirá a reducirla y gestionarla, teniendo como objeto de estudio analizar como la falta de gestión de la deuda técnica emerge en los contextos ágiles de desarrollo y como su acumulación degradan la calidad del producto. Con base a la literatura revisada se evidencia como la premura por las entregas en la agilidad, normaliza atajos técnicos lo que genera deuda técnica que se incrementa con el tiempo. Si bien, se encontraron estudios sobre gestión basada en el valor, por separado, de la deuda técnica y de los problemas de la agilidad en el desarrollo, no existen investigaciones que articulen estos tres ejes, por lo que falta evidencia científica y empírica que explique como gestionar la deuda técnica de una manera transparente y sostenible.

Además, el modelo propuesto proporcionará un enfoque con el cual se comprenda adecuadamente y se gestione la deuda técnica en la práctica, lo que servirá en la toma de las decisiones de los proyectos, lo que mejorará las prácticas de ingeniería en el desarrollo de software en proyectos ágiles al identificar entre otras cosas el esfuerzo que es necesario realizar hasta alcanzar una calidad ideal en el software al incorporar indicadores que minimicen la deuda técnica.

Con base a lo indicado la pregunta de investigación planteada es: ¿Cómo un marco conceptual basado en el valor usando técnicas, indicadores y herramientas de soporte permiten la identificación, gestión y reducción de la deuda técnica en entornos de desarrollo ágiles, de modo que se mejore la calidad del software en el desarrollo?

Hipótesis

El diseño de un marco para la gestión de la deuda técnica en modelos ágiles de desarrollo, que integre técnicas sistemáticas, indicadores de medición y herramientas de soporte, influye en la mejora de los procesos de identificación, priorización y remediación de la deuda técnica, lo que se traduce en una reducción de la deuda técnica y en una mejora de la calidad del software.

Justificación

Como se indicó en la descripción del problema, la deuda técnica es el término con el cual se conceptualiza el retraso, solución alternativa u omisión del trabajo para completar un proyecto de software, lo que provoca mayor cantidad de trabajos al final. La pertinencia del estudio aborda la relación entre deuda técnica, modelos ágiles y enfoques de gestión basados en valor de una manera crítica e integrada, que, al tener una relevancia fundamental en la práctica, no posee suficientes investigaciones sobre el tema. Si bien la deuda técnica afecta en la calidad del software, por factores de presión y entregas continuas de la agilidad, la mayoría de los trabajos investigativos lo tratan de forma aislada, lo que denota un vacío científico que afecta a las entregas con valor del software. Como aporte de esta investigación se propone un marco conceptual que contribuya en la comprensión epistemología de los problemas que surgen por la deuda técnica al ser un fenómeno socio-técnico permitiendo una gestión basada en el valor que sea ágil y sostenible permitiendo una eficiente identificación y remediación de la deuda técnica.

Las consecuencias derivadas en la entrega ágil de los productos de software ocasionan negligencias, errores o deficiencias ya sean deliberadas o involuntarias en relación con la documentación y principalmente con el código. Ante esta situación es necesario realizar correcciones y medidas de mantenimiento al software lo que entre otras cosas frenan el

rendimiento óptimo del equipo de desarrollo de software y generan elevados costes adicionales

En esta perspectiva, la agilidad requiere un constante nivel de calidad, lo que se contrasta con la deuda técnica, ya que, esta impacta directamente sobre los atributos internos de las aplicaciones, e indirectamente sobre sus atributos externos. Esto siempre es percibido por el equipo de trabajo, pero a menudo no es considerada y rara vez es comunicada.

Es por este motivo que es fundamental comprender y gestionar la deuda técnica por parte de todos los involucrados en desarrollo de software, tanto negocio, organizaciones y equipos de desarrollo de software, deben entender cómo se identifica la deuda técnica, cuál es la mejor estrategia para su gestión, cuando y en qué condiciones se la permite incrementar y cómo reducirla.

Lo que se trata con el estudio planteado es que la cultura del desarrollo de software establezca la realidad de la deuda técnica ya que se debe emplear tiempo y recursos como parte del flujo de trabajo habitual y no esperar a llegar al punto de ruptura. Se plantearán estrategia para pagar la deuda, estas estarán enfocadas a la reducción continua de la deuda y los intereses generados.

En función de lo planteado, el proyecto es viable tomando en cuenta que la deuda técnica es una realidad que por lo general se la quiere ocultar, pero es necesario tomar en cuenta que su gestión es una carrera de fondo, por lo que es importante la concienciación y la constancia para su tratamiento.

Aunque existen una gran cantidad de investigaciones relacionadas con el tema, no se han presentado soluciones puntuales enfocadas a la realidad actual del desarrollo de software abatida por una serie de problemas surgidos especialmente por la gran cantidad de metodologías de desarrollo de software existentes, por lo que se propone realizar un modelo

que permita contribuir de manera general para la solución de este problema en el desarrollo ágil.

Para la investigación se realiza un estudio bibliográfico y de campo, que permita obtener datos relevantes para alcanzar un conocimiento del estado actual del problema, lo que permitirá desarrollar el modelo. Dentro de este marco, para demostrar la efectividad del modelo, se aplicará un estudio cuasi experimental mediante un experimento, para el cual se trabajará con un equipo de desarrollo experimentados para obtener datos lo más reales posibles.

Los beneficiarios directos con la elaboración de esta investigación serán los clientes y equipos de desarrollo. La deuda técnica no se puede gestionar adecuadamente a menos que las compañías de desarrollo de software, organizaciones o equipos de desarrollo le dediquen el tiempo y la capacidad necesarios. Los miembros técnicos del equipo son los comprometidos en la calidad del software, es por esta razón que son los mayores conocedores de la deuda técnica existente y de su impacto a largo plazo.

Diseñar un modelo que puedan utilizar los equipos de desarrollo de software donde se definan diferentes estrategias para gestionar la deuda técnica, establecerá en las organizaciones una cultura de confianza, transparencia que permitirán asegurar niveles óptimos de calidad del software. Además, un modelo que permita una adecuada gestión de la deuda técnica permitirá generar acuerdos entre los involucrados de manera constante lo que permitirá que las organizaciones administren de manera adecuada los riesgos que generan la deuda.

Objetivos

Objetivo general.

Desarrollar un marco para la gestión y reducción de la deuda técnica en modelos de desarrollo de software ágiles que proporcione un enfoque conceptual para administrarla.

Objetivos específicos.

- Analizar los tipos de deuda técnica que emergen en el desarrollo ágil, identificando sus causas, el impacto en la calidad del software orientado a la entrega continua de valor y su relación en la toma de decisiones.
- Identificar el estado actual de la gestión de la deuda técnica en modelos de desarrollo de software ágiles con enfoque basado en el valor.
- Diseñar un marco en el cual se definan criterios, técnicas y herramientas con lineamientos conceptuales y operativos, para el tratamiento de la deuda técnica en ambientes de desarrollo ágiles, que permita evaluar su impacto en términos de valor.
- Evaluar el impacto de la deuda técnica no gestionada y los resultados de aplicar el modelo propuesto a través de un estudio de caso, analizando su viabilidad, coherencia y contribución para entregar a tiempo un producto de calidad.

1. Marco de referencia

Marco Teórico

El marco teórico se basa en la teoría crítica constructivista, en este sentido, el autor (Floridi, 2013) señala que el conocimiento se construye y transforma en entornos digitales. Su enfoque denominado epistemología centrada en la información es relevante para el presente estudio ya que considera al software como un artefacto informacional al mismo tiempo que señala que los sistemas digitales generan y transforman el conocimiento, también, reconoce que la calidad de la información es fundamental para la toma de decisiones.

Con este enfoque se puede entender a la deuda técnica como una degradación del ecosistema informacional en el desarrollo del software que afecta la documentación técnica y calidad del software distorsionando la sostenibilidad del conocimiento técnico y los entornos computacionales.

De ahí que, la deuda técnica es un problema actual que se ha vuelto común en todos los entornos que han adoptado los modelos ágiles como marco en el desarrollo de software, esta surge por la premura por cumplir con los requerimientos de los clientes y por obtener un producto funcional lo más pronto posible. Este fenómeno afecta a la ingeniería del software y sus buenas prácticas que son importantes en el desarrollo ya que, garantiza la toma de decisiones técnicas para una entrega oportuna del producto de acuerdo con los requerimientos y cronogramas planificados.

El marco teórico se ha estructurado con la finalidad de proporcionar los fundamentos teóricos y prácticos de los temas relacionados con la deuda técnica y las técnicas aplicadas para reducirla y controlarla.

Para esto se contextualiza y define la deuda técnica identificándola y clasificándola, analizando los factores para su aparecimiento y acumulación con lo que examinaran sus

consecuencias en el desarrollo de software, lo que permitirá proponer estrategias para mitigarla.

Antecedentes de la Investigación

La deuda técnica surgió como termino en 1992 cuando Ward Cunningham la comparó con la deuda financiera que generan intereses representados por el costo del trabajo extra por la toma de malas decisiones durante el proceso de desarrollo normal del software y la aplicación de soluciones que entregan un producto bajo de calidad por cumplir con los tiempos de entrega.

En el 2010 el concepto de deuda técnica ya se generalizó en los desarrolladores, sin embargo, gestionarla con técnicas que se acoplen a los diferentes contextos de desarrollo fue uno de los principales problemas, los autores (Brown et al., 2010) en su estudio proponen un modelo de gestión en entornos ágiles.

Medir la deuda técnica es otro aspecto fundamental para tomar en cuenta en este estudio, (Seaman y Guo, 2011) proponen procesos para identificar y medir la deuda formalizando la correspondencia de esta con el costo de su pago. Este trabajo contribuye con estrategias para mejorar la mantenibilidad y calidad del software con la aplicación de métodos cualitativos.

Dos décadas luego del surgimiento del término deuda técnica, los autores (Kruchten et al., 2012) describen los aspectos relacionados con este fenómeno, su influencia en la arquitectura de software y la necesidad de la refactorización del código.

En el estudio desarrollado por (Tom et al., 2013), se realiza una revisión crítica de la deuda técnica y su gestión, para esto aplicaron encuestas y entrevistas a desarrolladores de la industria con la finalidad de proponer un marco enmarcado en la colaboración con fines prácticos para la gestión de la deuda técnica.

Estos estudios contribuyen en el conocimiento y comprensión de la deuda técnica proporcionando indicios valiosos para entender el impacto de la deuda en el desarrollo de software con la utilización de modelos ágiles, además, suministran una base para el uso de estrategias y herramientas para la gestión de la deuda técnica.

Bases Teóricas

Modelos ágiles

El uso de los métodos ágiles se ha generalizado por su flexibilidad para aceptar cambios en los requisitos, mejorar la productividad del equipo y alinearse con las necesidades del cliente. Cada equipo de desarrollo utiliza métodos ágiles de manera diferente, encontrándose muchos enfoques que proporcionan a los desarrolladores directrices para ayudarles a encontrar el marco que mejor se adapte a su equipo. Para esto es necesario identificar los factores situacionales para la adopción de las prácticas ágiles analizando las ventajas y desventajas del uso del enfoque (Kiv et al., 2019).

Los modelos ágiles son un marco de desarrollo que se sustenta en la iteración y la colaboración centrándose en la mayoría de las modelos en la entrega de pequeñas porciones de software funcionales de forma regular, en lugar de seguir un plan fijo y rígido. La entrega continua, los métodos flexibles de ingeniería y las formas de trabajo que son parte del manifiesto ágil ha llevado a varias interpretaciones de agilidad creándose muchos métodos ágiles de desarrollo de software (Berntsson Svensson et al., 2019).

Dentro de este orden de ideas, la comunicación entre los desarrolladores es importante en el diseño ágil, siendo uno de los desafíos que se enfrenta el equipo al adoptar estos modelos, además no se debe olvidar que la agilidad se encuentra centrada en el usuario siendo fundamental la comunicación entre los profesionales involucrados (Garcia et al., 2019).

En efecto, las prácticas ágiles se han desarrollado por su adaptación a los requisitos cambiantes de los clientes, permitiendo que los proyectos ágiles respondan rápidamente a los cambios centrándose en el individuo y las iteraciones. Aunque al inicio lo utilizaban equipos pequeños actualmente se la utiliza en organizaciones más grandes planteando desafíos en todos los actores que requieren respuestas y una comunicación suficiente para adaptarse a las necesidades identificadas (Figalist et al., 2019).

En general, los modelos ágiles son una opción viable para el desarrollo de software, aunque son propensos a la acumulación de la deuda técnica al no implementarse de manera adecuada especialmente cuando los equipos se centran en la rapidez de entrega y no se preocupan por la calidad del código, además, los modelos ágiles pueden hacer que sea más difícil identificar y abordar la deuda técnica, ya que, al ser las iteraciones corta se dificulta la identificación de problemas de calidad a largo plazo.

Deuda Técnica

La Deuda Técnica describe la deuda de trabajo que se genera al construir código fuente bajo de calidad incumpliendo las buenas prácticas que se deben utilizar para un correcto desarrollo de software, esto genera interés que se tendrá que pagar, con la desventaja que mientras más pase el tiempo demandará una mayor cuantía. El resultado de la deuda técnica puede afectar en la documentación que es escasa, mal realizada, inconclusa o inservible, generar defectos, provocar un deficiente control de entregas y versiones, arquitectura que no permite escalar, problemas al actualizar a nuevas herramientas y tecnologías o usar nuevas plataformas, etc. (Rosser y Norton, 2021).

En efecto, la deuda técnica es aquella situación en la que los desarrolladores construyen software sin tomar en cuenta las buenas prácticas que se deben aplicar en el desarrollo de Software como: la arquitectura, la estructura del código, los patrones de diseño o los estándares para la creación de código (Besker et al., 2019).

En este sentido, en el trabajo presentado por (Berenguer et al., 2021), no solamente considera el código fuente al investigar la Deuda Técnica, sino que, toma en cuenta otras cuestiones del ciclo de vida del desarrollo de software, encontrando que por lo general los inconvenientes son afines a las causas, los efectos, la prevención, y la falta de reembolso de la Deuda Técnica

Por otro lado, los autores (Freire et al., 2020), resumen las prácticas identificadas y las razones para no pagar la deuda Técnica. El estudio investiga desde la óptica de los profesionales del software, si los elementos de la Deuda Técnica han sido pagados en los proyectos de software. Además, estudia las prácticas que se han utilizado para pagar la Deuda Técnica y las razones que dificultan la implementación de estas prácticas, encontrando que, a menudo, los ítems de la Deuda Técnica no han sido eliminados de los proyectos de software.

Otro aspecto, es el presentado en la investigación realizada por (Rios et al., 2018), donde identifica las causas que conducen a la ocurrencia de la Deuda Técnica, para esto estudia si estas causas ocurren de forma aislada o en combinación para así entender si la Deuda Técnica se puede prevenir. También investiga si es mejor prevenir la deuda, o contraerla y pagarla, en términos de esfuerzo, encontrando que la deuda se puede prevenir, y es mejor trabajar en actividades de prevención que pagar la deuda después.

Del mismo modo, (Freire S. , 2021), en su trabajo estudia qué prácticas de pago de la Deuda Técnica se han utilizado cuando se sienten ciertos efectos de la presencia de deuda en los proyectos de software desde el enfoque de los profesionales desarrolladores de software. También investiga las razones para no pagar la deuda a pesar de los efectos que causan al proyecto. El estudio arrojó como resultado que la práctica de refactorización de código utilizada para cancelar la deuda cuando los efectos retraso en la entrega y reelaboración se sienten en los proyectos de software.

La deuda técnica ha sido estudiada por muchos investigadores, pero en la mayoría de los trabajos no se aborda de manera apropiada el interés que se genera al incurrir en los costos de mantenimiento por la deuda, ya que al ser muy alto es insostenible. En la investigación desarrollada por (Digkas et al., 2020) presenta una métrica para abordar el interés que cuantifica el riesgo de generarlo, obteniendo como resultado que es necesario la priorización de su pago para lo cual proponen calcular la deuda técnica mediante el impacto de un posible cambio en un artefacto específico mediante una herramienta que cuantifica la medida de Importancia del riesgo de generación de intereses (IGRI).

Deuda Técnica Potencial

La deuda técnica potencial es aquella que pueden resolverse en las revisiones previo a la fusión del código, antes de que evolucione a deuda técnica. Si bien, la deuda técnica potencial de la documentación y código son las más comunes, también se detectan frecuentemente la deuda potencial de diseño, defecto, documentación, requisito, prueba y código. En el estudio realizado por (Fu et al., 2022) determinaron que la deuda técnica potencial de código es la que toma menos tiempo en resolverla, también, señalan que el 81% de la deuda potencial de código se resolvió inmediatamente por los desarrolladores y el 78% se resolvió en una semana.

Deuda Técnica de Procesos

La deuda técnica de procesos es el resultado de la flexibilidad y el desarrollo iterativo de los modelos ágiles que, dentro de sus principios indica que se lo debe utilizar para trabajar con proyectos pequeños, sin embargo, debido a su auge y características también es usado en grandes proyectos donde incorporar sus prácticas denotan grandes desafíos. Los flujos de trabajo que se deben desarrollar con agilidad derivan en procesos ineficientes que acumula deuda técnica conocida como deuda de procesos identificada en cinco tipos: deuda de documentación, deuda de roles y responsabilidades, deuda de sincronización, deuda de

ineficiencia e inadecuación y deuda de infraestructura. Prevenir el surgimiento de la deuda de procesos y gestionarla junto a la deuda técnica mitigará las consecuencias relacionadas a la acumulación de la deuda garantizando la calidad del software (Saeeda et al., 2024).

Deuda Técnica de Requisitos

Los requisitos en el desarrollo son cruciales para que el software cumpla las necesidades planteadas por el cliente. La definición adecuada de los requisitos y el cumplimento de estos en la funcionalidad del software son fundamentales para el éxito del producto. La deuda técnica de requisitos surge por los procesos subóptimos aplicados en la ingeniería de requisitos, esta se acumula en los diferentes ciclos del desarrollo desde la recopilación, definición, implantación y validación, por lo que cuantificarla con un modelo adecuado permite minimizarla y gestionarla generando beneficios en calidad y costos del producto (Perera et al., 2023).

Deuda Técnica Arquitectónica

Otro de los campos con pocos estudios sobre la deuda técnica es el relacionado a la deuda arquitectónica, en el estudio realizado por (Verdecchia et al., 2020) analiza las causas, consecuencias, síntomas, estrategias de manejo y otros fenómenos relacionados a la deuda técnica arquitectónica. Esta se refiere a la deuda que se genera en el diseño de la estructura, marcos, tecnologías, lenguajes, etc., para lo cual presentan un conjunto de categorías interrelacionadas con respecto a la deuda técnica arquitectónica fundamentada en los desafíos que enfrentan los entornos industriales.

Además, se debe percibir como los arquitectos de software entienden a la deuda técnica, de acuerdo con (Pérez et al., 2021) estos están pendientes de la deuda de código que de la deuda de diseño o de prueba. Además, de acuerdo con el estudio realizado a los arquitectos de software, el principal pago utilizado para la deuda técnica es la refactorización,

a continuación las mejoras de diseño y las mejoras de prueba. Sin embargo, la prevención de la deuda es la práctica más recomendada y menos costosa que realizar cualquier tipo de pago.

Deuda Técnica Organizacional

La deuda organizacional es un tipo de deuda técnica producto de los atajos en el desarrollo de software originados en los procesos organizacionales. Las causas para la deuda técnica organizacional son: los cambios organizacionales mal gestionados, la falta de cultura de colaboración, esfuerzos de cambio aislados, decisiones de liderazgo complejas. Las estrategias de mitigación para controlar la deuda técnica organizacional son las siguientes: diseñar flujos de trabajo para la creación de valor, permitir la creación de puestos de trabajo y un entorno de trabajo flexible, establecer una cultura de retroalimentación y generar confianza, practicar la gobernanza participativa continua y aplicar métricas de desempeño, comportamiento e innovación (Ovais Ahmad y Al-Baik, 2024).

El desarrollo de Software incluye varias actividades con tareas que involucran tanto tecnología como aspectos organizacionales con artefactos que al ser ineficientes generan riesgos que a la postre se convierte en deuda técnica. Las dimensiones sociales de esta deuda en la ingeniería de software generan un costo imprevisto que afecta a la comunidad de desarrollo lo que conlleva a una deuda social. La deuda sociotécnica contribuye en la comunicación de los elementos de la deuda y permite plantear las prácticas de mitigación y priorización de la deuda técnica (Riquet et al., 2024).

Deuda Técnica en el sector de la automatización industrial

El software para el ámbito industrial al ser un proceso multidisciplinario se ve afectado por la deuda técnica que, además de incidir en el desarrollo del software también influye en el área mecánica, eléctrica y electrónica. Los equipos mecatrónicos se encuentran compuestos de diferentes elementos electromecánicos y software embebido que, debido a subestimaciones, estimaciones incorrectas, requisitos inconsistentes o mal especificados,

pruebas nulas o incompletas, documentación, control en la fabricación y mantenimiento entre otros generan la deuda técnica. Esta deuda contamina todos los procesos involucrados por diferentes incidentes que requieren una gestión particular de la deuda técnica (Bi et al., 2023).

Deuda Técnica autoamitida SATD.

La deuda técnica autoadmitida (SATD) es aquella que describe la deuda detectada por los desarrolladores por medio de comentarios en el código fuente introducidos intencionalmente que sugiere cambios a realizar mediante lenguaje natural o las soluciones temporales realizadas en el software (OBrien et al., 2022). Los tipos de (SATD) se presentan en la Tabla 1:

Tabla 1.

Tipos de SATD

SATD Tipos	Descripción
Requerimientos	No se implementaron completamente los requerimientos. No cumplen con las especificaciones Necesitan soporte al implementar nuevas funciones.
Código	Código complejo y de baja calidad que se deriva en dificultad para comprenderlo y mantenerlo.
Pruebas	Involucran problemas en subcomponentes de prueba o monitoreo.
Defectos	Todo tipo de defecto identificado.
Diseño	Diseño de baja calidad
Documentación	Documentación inadecuada.

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: (OBrien et al., 2022)

Los desarrolladores reconocen y documentan la deuda técnica autoadmitida (SATD) usando comentarios explícitos o anotaciones en el código suministrando información de los riesgos potenciales por la aplicación de soluciones alternativas o atajos técnicos. Los artefactos utilizados para identificar el SATD son: sistemas de seguimiento de problemas, solicitudes de incorporación de cambios, código fuente y mensajes de confirmación (Sutoyo

y Capiluppi, 2024). Estas fuentes tienen en común elementos para identificar la SATD que se repiten al reportarlos en cada una de estas, como es el caso de que el mismo elemento sea reportado tanto en las solicitudes de cambios como en el código. Detectar estas relaciones entre las distintas fuentes contribuye en la gestión adecuada de la SATD (Li et al., 2023).

Para comprender la SATD es necesario analizar el código antes y después de eliminar los comentarios. El proceso de eliminar los comentarios se encuentra asociado al pago de la deuda técnica SATD, para lo cual se deben realizar las acciones necesarias para un reembolso formal de la deuda para lo cual se pueden utilizar diversas técnicas o herramientas automatizadas (Alhefdhi et al., 2024).

Del mismo modo los comentarios de las SATD tienen características específicas como es el caso de las anotaciones duplicadas y casi duplicadas, encontrándose un alto porcentaje de comentarios idénticos como resultado de la copia de fragmentos de código. De ahí la necesidad de abalizar los comentarios duplicados y no centrarse solamente en la identificación de los SATD (Yasmin et al., 2022).

Por otro lado, las anotaciones de la SATD pueden realizarse cuando se crea el código o durante las revisiones ligeras del código, para identificarla se utilizan términos como TODO/FIXME/XXX. Las revisiones de código se realizan con la finalidad de mejorar la calidad del código pudiéndose encontrar comentarios anteriores SATD y deuda técnica no detectada anteriormente que genera la introducción de nuevo SATD requiriendo la adopción de medidas para procesarla y gestionarla (Kashiwa et al., 2022).

En el mismo sentido, los desarrolladores utilizan un vocabulario contextualizado para comentar la SATD, aunque al tratar de identificarlo se encuentra un alto número de falsos positivos. Al analizar los comentarios SATD, se encontraron patrones concurrentes como: “hack, fixme, stupid, hope everything will work”, entre otros, por lo que se formuló un vocabulario contextualizado, con patrones y etiquetas para identificar la SATD con una

nomenclatura amplia, con lo cual se logra determinar tanto el código como el comentario asociado y así realizar el análisis respectivo de la SATD (de Freitas Farias et al., 2020).

De acuerdo con los comentarios de la SATD la deuda técnica ralentiza la evolución del código ya que dificulta la comprensión del código aumentando el costo y trabajo extra de mantenimiento, de ahí que los desarrolladores se enfoquen en el pago de la SATD para así reducir el interés, limpiar el código, familiarizarse con el código base, colaborar con otros trabajos relacionados y aumentar la cobertura de las pruebas (Xavier et al., 2020).

La SATD también tiene relación con la refactorización, la rectificación de errores y la añadidura de nuevas características. Con la comprensión de la relación entre la SATD y este tipo de actividades durante el desarrollo de software se obtendrá un conocimiento más amplio para prevenir la deuda técnica y mejorar el código mediante su eliminación y pago con la refactorización (Esfandiari y Sami, 2023).

Las aplicaciones móviles también generan deuda técnica autoamitida, sin embargo, la SATD es menor que en las aplicaciones no móviles siendo la deuda de código la más común, seguida de la deuda que se crea al construir la interfaz de usuario y generada por la implementación del código que usa las API. Los tipos de SATD que se ocasionan en el desarrollo de aplicaciones móviles son: deuda de código, deuda de diseño, deuda de requisitos, deuda de defectos, deuda de documentación, deuda de pruebas, todas como resultado de la implementación de soluciones alternativas y el uso de diseño subóptimo para cumplir con sus objetivos planteados por el equipo de desarrollo (Wilder et al., 2023).

Deuda Técnica y la calidad del Software

La productividad del desarrollo y la viabilidad del software se ven directamente afectados por la deuda técnica, ya que de acuerdo con el trabajo realizado por (Besker et al., 2019) el 23% del tiempo destinado al desarrollo del software se desperdicia ya que el equipo de desarrollo lo dedica a tratar de controlar la deuda al realizar actividades extras como

aplicar pruebas adicionales, análisis del código fuente y refactorización. También señalan que estos problemas generan contagio de la deuda técnica esto es introducir deuda adicional debido a la existente lo que genera aún más interés.

Por otro lado, la deuda técnica se encuentra directamente relacionada con los fallos de la calidad del software. De acuerdo con el trabajo realizado por (Arvanitou, et al., 2019) las estimaciones relacionadas con el esfuerzo resultan de gran utilidad para los desarrolladores ya que de esta manera pueden visualizar de mejor manera la deuda técnica lo que permitirá representarla de manera uniforme, hacerla visible a los stakeholders y monitorearla.

Con relación a los responsables de las TI, estos entienden las implicaciones de la deuda técnica, considerándola como preocupante, sin embargo, no han establecido acciones para su gestión, no obstante, se evidencia que han considerado comunicar y establecer un proceso para identificar y monitorear la deuda técnica, encontrando de esta manera a los desarrolladores que funcionan y a los que no funcionan finalmente señalan lo beneficioso que puede ser el invertir en un proceso de gestión de la deuda técnica (Wiese y Borowa, 2023).

Code Smells

Los "code smells" también conocidos como "hedor de código" o "malos olores de código" es una característica del código fuente que indican deficiencias en el diseño o la implementación del software. Como ejemplos del code smells se tiene: duplicación de código, métodos o funciones demasiado largos, clases o módulos con demasiadas responsabilidades, nombres de variables y funciones poco descriptivos, acoplamiento excesivo, clases o métodos inutilizados, exceso de comentarios o documentación redundante, entre otros.

La presencia de Code smells acrecienta la deuda técnica en el desarrollo de software (Sangeetha y Chandrasekar, 2019). Los code smells son síntomas de malas opciones de diseño e implementación que pueden originarse por malas prácticas realizadas por los

desarrolladores durante emergencias, diseño o soluciones de codificación deficientes o por tomar malas decisiones (Kokol et al., Code smells, 2018).

Los Code smells están relacionados con la gestión del mantenimiento del software, refactorización de software basada en la detección de olores, olores arquitectónicos, mejora de la calidad del software, deuda técnica y su instancia, defectos de diseño y olores de diseño (Kokol et al., 2020); lo que deriva en que podrían ser la consecuencia de la deuda técnica. Además, los Code Smells pueden ser resultado de una deficiente arquitectura, diseño y codificación en el desarrollo del software, lo que lleva a generar deuda técnica.

Asimismo, los autores (Cairo et al., 2018), en su trabajo proporcionan un análisis detallado sobre cómo los code smells influyen en la aparición de errores en los proyectos de software, también, describen las herramientas, recursos y técnicas que respaldan el análisis. Además, como parte de la investigación identifican herramientas que detectan los code smells, así como herramientas complementarias, técnicas y recursos que pueden ser utilizadas en los análisis para relacionar la codificación de los proyectos de software con los errores registrados.

De acuerdo con (Lewowski y Madeyski, 2022), en su estudio señala que el estado actual de los estudios sobre code smells deja mucho que desear, por lo que en su investigación se enfocan en la detección del code smells, exponiendo los detalles, así como la escala del problema en esa área en particular.

La deuda técnica es generalmente un resultado del defecto o mala práctica relacionado con la mantenibilidad técnicamente conocido como code smells, el porcentaje de deuda técnica es el resultado del tiempo necesario para solucionar esos code smells, entre el tiempo empleado total en el desarrollo del proyecto (Besker et al., 2019). Identificar y abordar los code smells es importante para generar código de calidad y prevenir la acumulación de deuda técnica.

Por otro lado, existen pocas herramientas para detectar los code smells, las investigaciones se han realizado en enfoques centrados a los datos y también en enfoques de aprendizaje automático. El uso de herramientas para identificar code smells es una técnica eficaz que beneficia al desarrollo con la reducción de costos y el ahorro de tiempo en el desarrollo lo que genera ventajas a largo plazo. Sin embargo, este campo no se encuentra aún explotado siendo necesario investigaciones de herramientas que identifiquen los code smells de manera automática para aportar a los proyectos actuales que cada vez aumentan su complejidad (Siksna et al., 2023).

Refactoring

La técnica de la refactorización implica reestructurar y mejorar el código fuente existente sin modificar su funcionalidad. Con esto se mejora la calidad del código, produciendo una codificación más limpia, legible, mantenible y eficiente, sin alterar el comportamiento del programa.

De hecho, el refactoring es el proceso por el cual se modifica un programa para mejorar la calidad estructural del código fuente sin alterar su funcionalidad. Hay diferentes maneras de realizar el refactoring que en varios casos depende del modelo de desarrollo de software con el cual se trabaja por ejemplo de manera aleatoria, no planificada o planificada (Abdullah AlOmar et al., 2021). La reestructuración o técnicamente conocida como refactorización del software es indispensable en el desarrollo para así aportar en la mejora de la calidad del software y la productividad, pero si no se la toma en serio puede ser compleja, costosa y arriesgada.

Es necesario definir el proceso de refactorización basado en los code smells, ya que los Code Smells y el refactoring tienen una fuerte relación con los atributos de calidad, como es el caso de la comprensibilidad, la mantenibilidad, la capacidad de prueba, la complejidad, la funcionalidad y la reutilización (Lacerda et al., 2020). El Refactoring y Code Smelles son

el resultado del uso de modelos de desarrollo de Software ágiles siendo necesario una adecuada gestión de la deuda técnica con lo que se conseguirá una mejora en la calidad del sistema y la optimización en la calidad del proceso de desarrollo.

Además, la refactorización puede estar presente en todas las fases del desarrollo de software que aceptan cambios en las funcionalidades y requisitos o que evolucionan de manera rápida, la mejora continua en la calidad requiere procesos de refactorización de manera continua para así mantener equilibrado tanto el rendimiento como la confiabilidad manteniendo niveles satisfactorios al mismo tiempo (Cortellessa et al., 2023).

Así mismo, durante el proceso de desarrollo de software, también la arquitectura es factible de refactorización ya que, al satisfacer nuevos requisitos funcionales o mejorar propiedades no funcionales se induce a aplicarla. La optimización para realizar la refactorización de manera automática y semiautomática devela una manera viable que permite aplicar soluciones con distintas configuraciones. Un proceso automatizado en el proceso de refactorización identifica de manera más adecuada los elementos críticos para aplicar las soluciones en el rendimiento en la arquitectura del software (Cortellessa y Di Pompeo, 2021).

Algo semejante ocurre al optimizar la arquitectura de software, para esto se puede utilizar la refactorización automatizada, lo que permitirá mejorar la calidad incluyendo tanto el rendimiento como la confiabilidad del software, que, aunque son tareas complicadas y que requieren mucho tiempo y recursos, producen alternativas de diseño en el proceso de la arquitectura obteniendo atributos de calidad (Di Pompeo y Tucci, 2023).

Por otra parte, el mantenimiento de software requiere de la refactorización como apoyo para cumplir las mejores prácticas de diseño e implementación, con esto no solo se apoya a esta tarea que es crítica, sino que también se hace frente a los defectos de diseño. Para esto, el autor (AlOmar et al., 2019) identifica modelos para optimizar la calidad de

desarrollo mediante la refactorización obteniendo como resultado que existen métricas para representar los atributos de calidad mejorándolos.

En el proceso de la ingeniería de software se pueden generar alternativas de diseño mediante el uso de la refactorización para perfeccionar el modelo de software, creando automáticamente diferentes técnicas que permitan mejorar la calidad con un eficiente rendimiento y confiabilidad optimizando los recursos y el tiempo consiguiendo mejores soluciones (Di Pompeo y Tucci, 2022).

También, se tiene el caso de la investigación realizada por (Alizadeh et al., 2019) donde analiza las métricas de calidad usadas en la refactorización para evaluar cambios en el código, ya que según estudios los desarrolladores dedican más del 50% de tiempo en comprender y reformar el código creado en lugar de generar un nuevo. Para esto pondera los objetivos de calidad ofreciendo a los desarrolladores soluciones de refactorización mediante un modelo que le permite seleccionar una única solución tomando en cuenta la recomendación de refactorización que le entrega el modelo para la reestructuración del código.

Por otra parte, los procesos de refactorización se pueden aplicar en diferentes etapas de la codificación por lo que es importante que el programador comprenda el código base existente para así poder tener mejores resultados durante la refactorización o el mantenimiento. Dentro de esta tarea se encuentra el nombrar adecuadamente los tipos de datos, ya que el no hacerlo disminuye la productividad de los desarrolladores al mismo tiempo que degrada el código fuente. Para esto es recomendable conocer cuándo se debe aplicar un cambio de nombre, cómo cambiar el nombre de un identificador y usar un modelo que permita a los desarrolladores asociar los nombres utilizando el conocimiento del dominio (Peruma et al., 2020).

La Refactorización en general ayuda a mitigar la deuda técnica, la implementación de una solución rápida a corto plazo plantea la necesidad de refactorización, considerándose este tiempo como el pago que se realiza a los intereses que se ha generado por la deuda existente, que se cancela en su totalidad solo una vez que el software se actualiza a la mejor solución posible (Mohan et al., 2016). El proceso de refactorización debe realizarse de manera cuidadosa y controlada, los desarrolladores utilizan pruebas unitarias y otras técnicas de verificación garantizando que no se altere el comportamiento del programa. La refactorización permite mantener la calidad del código a medida que evoluciona el proyecto siendo una práctica esencial en el desarrollo de software ágil.

Code Smells y Refactoring

Los code smells y el refactoring en el desarrollo de software están siempre presentes y se encuentran estrechamente relacionados. Los code smells indican que puede haber errores o problemas en el código fuente, como redundancia, complejidad excesiva, falta de cohesión, entre otros, mientras que, el refactoring, trata de mejorar el código sin alterar su funcionalidad externa.

El refactoring permite corregir los code smells mejorando así la calidad del código. En su proceso se aplican cambios controlados al código para eliminar los problemas subyacentes mediante diferentes técnicas y estrategias.

Los desarrolladores utilizan técnicas de refactorización para eliminar code smells y de esta manera contribuir para mejorar la calidad general del código, facilitando la comprensión y el mantenimiento del software, reduciendo de esta manera la probabilidad de incurrir en errores y facilitar el trabajo colaborativo en el proyecto.

Cabe considerar que, algunas razones comunes para realizar refactorización en el desarrollo de software incluyen: eliminar code smells, mejorar la legibilidad, facilitar el mantenimiento, aumentar la eficiencia, habilitar la reutilización, etc.

En su investigación, los autores (Lacerda et al., 2020), realizan un estudio sobre code smells y refactoring, donde analizan 40 estudios, mostrando que la mayoría de los estudios discuten los code smells y refactoring por separado, también, encontraron estudios que relacionan los code smells con la deuda técnica. Además, encontraron varios enfoques de detección de code smells, siendo los basados en métricas y estrategias los más citados.

En resumen, los code smells se generan por problemas en el código que podrían beneficiarse de mejoras, mientras que el refactoring aborda estos problemas, reestructurando y mejorando el código para hacerlo más limpio y eficiente manteniendo la funcionalidad y el comportamiento externo.

Refactoring y el mantenimiento de Software

El refactoring y el mantenimiento de software se complementan en el desarrollo de software, ya que al realizar el refactoring se mejora la mantenibilidad del software, además mediante la refactorización se puede reducir la deuda técnica al controlar los code smells que dificultan el mantenimiento. El refactoring permite que las modificaciones sean más seguras y eficientes debido a que se trabaja con un código más limpio reduciendo el tiempo y recursos necesarios para mantenerlo.

El mantenimiento del software se realiza cuando este entra en producción y posterior a la implementación ya sea por factores externos, internos, por mejoras del software, la corrección de problemas o errores, o la mejora del rendimiento se lo debe reestructurar. En la investigación realizada por (Mohan y Greer, 2018), señalan que existen tres métodos principales para usar la refactorización para de esta manera mejorar el mantenimiento del software, el primer método consiste en encontrar los defectos de diseño en el código y luego eliminarlo: el segundo señala que el software se puede refactorizar por adelantado para mejorar ciertas métricas de software o conjuntos de métricas; y el enfoque menos común

refactorizó el software hacia un diseño de software ideal previamente generado utilizando modelos UML.

Por otra parte, en la investigación realizada por (Conejero et al., 2018), presenta un estudio empírico para evaluar si las anomalías de modularidad a nivel de requisitos están directamente relacionadas con los atributos de mantenibilidad que afectan la calidad del software. La investigación se basa en un marco que permite la identificación de anomalías de modularidad y su cuantificación mediante el uso de métricas de modularidad. Las métricas de mantenibilidad también se utilizan para evaluar las propiedades de mantenibilidad dinámica. Los resultados obtenidos por ambos conjuntos de métricas se comparan por pares para comprobar si, cuantas más anomalías de modularidad presenta el sistema, menos estable y más difícil de mantener es.

En resumen, el refactoring es una práctica necesaria para un buen mantenimiento de software ya que al mantener un código coherente y bien estructurado se asegura un software funcional reduciendo los costos que se asocian al mantenimiento.

Métricas de la Deuda Técnica

La deuda técnica se puede cuantificar en diferentes aspectos como por ejemplo el code smell, el retorno de la inversión de la refactorización, la deuda por la arquitectura, el costo del nuevo código y el costo de reelaboración (Perera et al., 2023), sin embargo, los enfoques para cuantificar la deuda no lo miden bajo el mismo criterio. Para controlar esta brecha se utilizan herramientas automáticas para identificar la deuda técnica permitiendo identificar y cuantificar la deuda en aspectos como: la arquitectura, la compilación, el código, la documentación, el diseño, los requisitos, las personas y la deuda de prueba (Mendes et al., 2019).

En la investigación realizada por (Al Mamun et al., 2019) aplicó la métrica "tendencia de densidad de deuda técnica" donde analiza la acumulación y pago de la deuda técnica, en la

cual utilizó la herramienta SonarQube para el cálculo de la deuda. Encontró que la densidad de la deuda técnica es más baja conforme más líneas de código existen, contrariamente la "densidad de deuda técnica" es muy alta cuando el tamaño del código es menor, aunque es más fácil de leer y mantener posiblemente los desarrolladores no consideren escribir código de calidad al inicio.

En el trabajo realizado por (Guamán et al., 2017) compara las herramientas SonarQube, PMD y Kiuwan para identificar y analizar características del software desde el código fuente, obteniendo diferentes valores de la deuda técnica por cada una de las plataformas, esta variación en los resultados se da por el número de métricas que aplica cada una de estas al evaluar de manera distinta la ausencia de comentarios, la duplicación de código, el número de bucles, el acoplamiento entre clases entre otros.

En el mismo sentido en la investigación realizada por (Avgeriou et al., 2020) estudia el estado actual de las herramientas y plataformas que se utilizan en el mercado para medir la deuda técnica, estas poseen diferentes funcionalidades, niveles de análisis y cálculos de capital de la deuda, además promueven la comunicación, identifican, miden y monitorean la deuda técnica. Dentro de las herramientas analizadas se tiene a: SonarQube, NDepend, SonarGraph, CAST, DV8, CodeMRI y SQuORE.

De igual manera, en la investigación desarrollada por (Pavlic et al., 2022) utilizan las plataformas SonarQube y Vector Squore para medir la deuda técnica, comparada con la auto admisión como resultado del uso de una adecuada gestión de la deuda técnica para identificarla, denominado a esta actividad como deuda técnica admitida. El uso de estas dos técnicas arrojó una brecha al admitir la deuda técnica y medirla, concluyendo que se deben aplicar los dos enfoques para obtener una mejor visión de la deuda.

Además de las herramientas automatizadas descritas en la medición de la deuda técnica se han analizado otras como: Jira software, Hansoft, DebtFlag, TD-Tracker Tool,

Teamscale, Redmine, las cuales permiten en cierto grado la identificación de la deuda pero que no se pueden utilizar para una adecuada gestión de la deuda al no abarcar todas las actividades necesarias para controlarla y minimizarla (Pavlic y Hlis, 2019).

En este sentido, una de las herramientas más utilizada por los desarrolladores para el análisis de la deuda técnica es SonarQube ya que permite monitorear la evolución de la deuda con la gestión de código y una temprana alerta de los problemas seleccionados a los programadores (Lenarduzzi et al., 2020). También los investigadores señalan que, de acuerdo con los resultados obtenidos con SonarQube, los problemas generados por la deuda técnica están relacionados con aspectos de mantenibilidad, al mismo tiempo que señalan que esta herramienta ayuda a escribir código más limpio y reduciendo la propensión al cambio.

Sin embargo, estas herramientas utilizan sus propias definiciones, entregando resultados diferentes en las métricas que más se utilizan en la deuda técnica. De igual manera, las herramientas difieren significativamente al identificar los archivos que poseen problemas con amplios márgenes al identificar tamaño, complejidad y ciclos por lo que se debe estandarizar con modelos uniformes para la detección de la deuda técnica con datos comunes que aporten a la precisión de las herramientas (Lefever et al., 2021).

Estimación de la deuda técnica

La estimación de la deuda técnica es una actividad muy aplicada en los modelos de desarrollo ágil especialmente en scrum, para que la estimación sea lo más posible exacta es necesario que exista una comunicación efectiva con el cliente, mejorar las rutinas de desarrollo con una correcta estructuración de los componentes de software, flujos de datos, integraciones y otros, lo que permitirá un mejor análisis del impacto de la deuda en el proyecto de software (Klimczyk y Madeyski, 2020).

En este sentido, es importante estimar la deuda técnica en el desarrollo debido a que minimiza el impacto en el producto final, en esta etapa la estimación se promedia en 3,6

dólares las 300.000 líneas de código (LOC), aumentando el costo de acuerdo con el avance en el ciclo de vida del desarrollo y contrastando significativamente con el costo al descubrir la deuda técnica después del lanzamiento llegando a costar hasta 30 veces más (Paradkar, 2021).

Gestión de la Deuda Técnica

La gestión de la deuda técnica son las acciones para identificar, controlar y reducir la deuda técnica durante el desarrollo de software controlando de esta manera los retrasos en el desarrollo, costos adicionales y dificultades para adaptar el software al mantenimiento.

Es necesario que la deuda deba ser gestionada, para de esta manera organizar las acciones que permitan pagar o cancelar la deuda técnica mediante mecanismos de seguimiento y gestión adecuado. Las metodologías ágiles se basan en fases cortas de desarrollo, con entregas rápidas que se derivan en correcciones continuas generando la deuda técnica, es por lo que tener un control y claridad en el desarrollo del software permitirá evitar riesgos financieros y reducirla (Fernández-Sánchez y Garbajosa, 2017).

Existen varios desafíos en la gestión de la deuda técnica que los equipos de desarrollo de software deben comprender mediante la aplicación de un adecuado marco técnico de gestión de la deuda que describa los procesos de gestión, las partes interesadas y las responsabilidades y actividades que deben cumplir los miembros del equipo (Yli-Huumo et al., 2016).

Por otro lado, los autores (Freire et al.), presentan una investigación donde estudia qué prácticas de pago de la Deuda Técnica se han utilizado cuando se sienten ciertos efectos de la presencia de deuda en los proyectos de software desde la perspectiva de los profesionales desarrolladores. También investiga las razones para no pagar la deuda a pesar de los efectos que causan al proyecto. El estudio se realiza mediante una encuesta a profesionales en cuatro países encontrando entre las relaciones identificadas, la práctica de refactorización de código

utilizada para cancelar la deuda cuando los efectos retraso en la entrega y reelaboración se sienten en los proyectos de software.

Es fundamental la gestión de la deuda técnica ya que así se garantiza la calidad, la eficiencia y la sostenibilidad a largo plazo de un proyecto de desarrollo de software. Con esto se garantiza un código saludable y adaptable, contribuyendo con el éxito del proyecto. Al mismo tiempo se reducen costos, se mejora la productividad del equipo, la satisfacción del cliente y se fomenta la colaboración.

Mitigación de la Deuda Técnica

Para mantener el software funcional, sin defectos y con procesos eficientes se deben aplicar diferentes prácticas para mitigar la deuda técnica. Prevenir proactivamente los problemas que afectan la calidad del software permiten la estabilidad y mantenimiento garantizando el correcto funcionamiento de las aplicaciones.

La prevención y gestión de la deuda técnica se debe enfatizar con estrategias dinámicas que eviten problemas técnicos abordando sistemáticamente los procesos que generan la deuda, con estrategias claves que mitiguen el impacto en el proyecto y que impidan que la deuda se propague logrando la reducción de costos en la corrección de errores, satisfacción en el equipo de desarrollo y un software de calidad, confiable y mantenible.

Es así como, el desarrollo de nuevos artefactos para mitigar la deuda técnica con la generación de código limpio es una estrategia que permite obtener código de mejor calidad que aquel donde se ha realizado adiciones, modificaciones y eliminaciones de código para pagar la deuda. Al escribir nuevo código no solo se elimina la deuda técnica, sino que también se crea código sin deuda o con una mínima cantidad de esta lo que resulta en una estrategia de bajo costo para gestionar y reducir la deuda técnica (Digkas et al., 2022).

En el mismo sentido, controlar el impacto que ocasiona a largo plazo la deuda técnica en la planificación de un proyecto incide en el presupuesto y cronograma. Actualizar permanentemente el cronograma en función del impacto de la deuda en las tareas que sufren de esta y en las tareas sucesoras permite una estimación más precisa y la temprana determinación de los riesgos y las estrategias para mitigarlos y prevenir alargues de entregas y problemas futuros (Kleinwaks et al., 2023).

Pago de la Deuda Técnica

El pago de la deuda técnica se la debe asumir tarde o temprano siendo importante priorizar eficientemente los elementos que se deben pagar. Las estrategias de pago deben relacionarse con los efectos de los ítems afectados por la deuda técnica ya que esta información proporciona los componentes necesarios para la toma de decisiones de las prácticas de pago a aplicar. En la investigación realizada por (Freire et al., 2021) se identificaron las diez principales prácticas de pago siendo estas las siguientes en orden de importancia: “refactorización de código, invertir esfuerzo en actividades de pago de deuda técnica, refactorización de diseño, invertir esfuerzo en actividades de prueba, priorizar elementos de deuda técnica, seguimiento y control de las actividades del proyecto, negociación de ampliación del plazo, aumentar el presupuesto del proyecto, resolver problemas técnicos y actualizar la documentación del sistema”.

También, se identificaron diez razones para el no pago de la deuda técnica, listándose en orden de prioridad: “centrarse en objetivos a corto plazo, falta de interés organizacional, costo, falta de tiempo, decisión del cliente, falta de recursos, complejidad del ítem de deuda técnica, complejidad del proyecto, vista de gestión insuficiente sobre el pago de la deuda, el proyecto fue suspendido” (Freire et al., 2021).

La deuda técnica generada por la SATD es una de las primeras en ser pagadas, debido a que los desarrolladores son conscientes del perjuicio de esta deuda y al no ser una amenaza

crítica es la mayormente pagada y en menos tiempo. De acuerdo con el trabajo realizado por (Li et al., 2020), en su investigación señalan que entre 25 y 872 horas es el tiempo promedio para pagarla, además indican que el 71,7% de la SATD identificada es pagada.

Por otro lado, los desarrolladores pagan la deuda técnica utilizando la refactorización. Esta actividad la aplican al código base tanto para el reembolso de la deuda como para dar mantenimiento al código tomando en consideración los comentarios realizados en la codificación producto de la SATD pagando la deuda técnica y mejorando la calidad del producto (Peruma et al., 2022).

Estado del arte

Introducción

En el contexto del desarrollo actual, caracterizado por entregas rápidas debido a la agilidad de las metodologías aplicadas en el desarrollo de software, la gestión de la deuda técnica es un aspecto critico en las actividades de la ingeniería de software, siendo una alternativa necesaria para garantizar la calidad del producto.

En este sentido, surge la necesidad de un marco adecuado que permite, identificar, controlar y gestionar la deuda técnica. Si bien el campo científico ha aportado con varios estudios que abordan este tema, no existe un marco estandarizado o ampliamente adoptado por los desarrolladores para la gestión de la deuda técnica.

Por lo tanto, el objetivo del presente estado del arte es obtener mediante una revisión sistemática y exhaustiva de los estudios más relevantes la deuda técnica en el marco de los modelos ágiles y su gestión con relación a las estrategias de identificación, el impacto en el desarrollo y los métodos para controlar la deuda. Para cumplirlo se analizará el estado de las investigaciones que abordan las variables planteadas, las estrategias, herramientas y técnicas para gestionar la deuda técnica y los factores contextuales para establecer un marco para la gestión de la deuda.

Con esta revisión, se sintetizan los hallazgos fundamentales de los trabajos consultados, se identifican las herramientas y metodologías empleadas, al mismo tiempo que, se establece los vacíos existentes en las investigaciones realizadas con relación al tema. Además, se analizarán y discutirán los estudios de modelos ágiles, análisis de código, buenas prácticas y modelos de gestión en el contexto de la deuda técnica.

Alcance y limitaciones

Se abordarán investigaciones divulgadas en los idiomas español e inglés, publicadas en el lapso comprendido entre el 2017 al 2024 y que se encuentren indexadas en bases de datos académicas reconocidas como IEEE, Springer, Science Direct, Google Scholar, entre otras, para esto se considerarán estudios cuantitativos y cualitativos.

No obstante, la revisión no es exhaustiva debido a la cantidad considerable de estudios sobre la deuda técnica. Aunque, es importante destacar que mediante una selección rigurosa de las principales investigaciones se busca presentar los trabajos relevantes y de mayor calidad para ofrecer una visión actualizada y completa del estado del conocimiento de las variables estudiadas.

Se espera que el presente estado del arte contribuya significativamente a la comprensión del problema al ofrecer una visión actualizada y completa del estado del conocimiento en el área, identificando los factores que influyen en la disminución y control de la deuda técnica lo que permitirá proponer futuras líneas de investigación para comprender mejor las metodologías para la gestión de la deuda.

Antecedentes investigativos en el Ecuador

En el Ecuador existen algunos estudios relacionados a la deuda técnica y sus implicaciones. En 2017 la Universidad Particular de Loja publicó un estudio sobre la identificación de deuda técnica en aplicaciones desarrolladas con estilos y patrones arquitectónicos (Rosillo Reyes y Guamán Coronel, 2017). La investigación encontró que el

calcular manualmente la deuda resulta complicado, además de que se requiere una mayor cantidad de tiempo para realizar esta actividad, al mismo tiempo señalan que las características y métricas de calidad que se van a evaluar se las debe seleccionar de acuerdo con el ciclo de vida del proyecto.

Otra investigación, es la realizada en la Universidad de Guayaquil donde se publicó un estudio sobre la deuda técnica en proyectos de desarrollo de software en el Ecuador. El estudio encontró que la deuda técnica es por lo general un problema común en los proyectos de desarrollo en el Ecuador, además de impactar negativamente en la calidad del software (Ayala González y García Lizarazo, 2018).

También se puede mencionar el estudio sobre la gestión de la deuda técnica en metodologías ágiles en el Ecuador. La investigación encontró que la gestión de la deuda técnica es un tema que está ganando importancia en el país y que se están desarrollando nuevos métodos y técnicas para gestionar la deuda técnica en modelos de desarrollo de software ágiles (Jiménez-Reyes et al., 2020).

En el mismo sentido, en la Universidad de Loja se publicó un artículo sobre la gestión de la deuda técnica en modelos de desarrollo de software ágiles. La investigación se refiere a la presentación de un estudio para la gestión de la deuda técnica en modelos de desarrollo de software ágiles que se basa en los principios de la inspección y la adaptación (Naranjo-Vélez et al., 2021).

Los estudios, investigaciones y publicaciones sobre la gestión de la deuda técnica en el Ecuador son pocos, por lo que se requiere más investigación y desarrollo en esta área para comprender mejor la deuda técnica y desarrollar métodos y técnicas efectivos para gestionarla de acuerdo con la realidad del país.

Antecedentes investigativos en el ámbito mundial

A nivel mundial, la gestión de la deuda técnica es un tema que ha recibido una creciente investigación en los últimos años. Es así como se han efectuado numerosos estudios e investigaciones sobre el tema.

Uno de estos estudios encontró que la deuda técnica es un problema común en los proyectos de desarrollo de software, la investigación señala que la deuda técnica interfiere en la calidad del software negativamente, así como en el tiempo de desarrollo y el costo de mantenimiento (Tan et al., 2023). Este es un problema común que afecta a los equipos de desarrollo, siendo un reto para los desarrolladores obtener productos que cumplan las necesidades y requerimientos de los clientes de manera óptima.

Otro estudio publicado encontró que existen una variedad de métodos y técnicas para gestionar la deuda técnica. El estudio encontró que los enfoques más efectivos para gestionar la deuda técnica varían según el contexto específico del proyecto (Albuquerque et al., 2022). Estos trabajos aportan a la presente investigación con el desarrollo de un marco de gestión que se adapte a las necesidades de los equipos de desarrollo.

La gestión de la deuda técnica es un tema que frecuentemente se discuten en las conferencias de ingeniería de software. En 2023, se celebraron varias conferencias relacionadas con la gestión de la deuda técnica, incluyendo: IEEE Software Conference, ACM SIGSOFT Conference on Software Engineering, International Conference on Software Engineering

El estado de los estudios, investigaciones y publicaciones con relación al control y gestión de la deuda técnica en el mundo es positivo. Aunque se ha evidenciado un importante progreso en la comprensión de la deuda técnica no existe métodos y técnicas mayormente adoptados en la industria para gestionarla. Es por lo que, se requiere mayor investigación y

desarrollo en esta área para comprender mejor la deuda técnica y desarrollar métodos y técnicas aún más efectivos para gestionarla.

Contribuciones del Investigador

Para contribuir el tema se han desarrollado varias investigaciones como es el caso del estudio (Lozada Torres et al., 2025), en el cual se aborda la deuda técnica y los problemas de calidad del software, en la cual se analiza las herramientas de análisis estático de código y su aporte para identificar y mitigar la deuda técnica. También, se aporta con la investigación (Lozada Torres E. F. et al., 2024), en la cual se investiga como las decisiones rápidas en los entornos ágiles afectan en la calidad del software y a la larga se derivan en deuda técnica.

Otra de las contribuciones del autor es el artículo (Lozada Torres et al., 2024), donde se estudia el impacto de la deuda técnica en entornos académicos para los cual se investiga el conocimiento y la percepción de la deuda técnica por parte de los desarrolladores. Además, en el estudio (Lozada Torres et al., 2023), se analiza como los programadores noveles perciben la deuda técnica, encontrándose que se subestima el valor de la deuda técnica y que los programadores noveles no tienen el conocimiento del verdadero problema ni el impacto que se genera en la calidad del software.

También, se realizó la investigación (Lozada Torres et al., 2023), donde se estudia la influencia de la agilidad en el desarrollo de software para lo cual se plantea un análisis estadístico de la evaluación del conocimiento que se tiene en la educación superior con referencia a los modelos de desarrollo ágiles.

Además, se formó parte del equipo de investigación del trabajo (Pico Pico y Lozada Torres, 2023), que estudia a los modelos con principios ágiles que se usan en el desarrollo de aplicaciones móviles, realizándose una investigación bibliográfica y de campo para analizar las diversas metodologías de desarrollo ágil existentes y evaluar su uso en los programadores noveles que desarrollan aplicaciones móviles.

Tesis relacionadas a la Deuda Técnica

En el ámbito de la gestión de la deuda técnica se han desarrollado una variedad de tesis con diferentes propuestas que se caracterizan por enfoques únicos, aunque estos enfoques han sido soluciones específicas no se ajustan a la realidad del desarrollo de software en la industria siendo necesario un modelo estándar que se soluciones las necesidades comunes de la industria.

El uso de metodologías ágiles en el desarrollo de software a derivado en que la deuda técnica se acreciente afectando la calidad del código fuente y las funcionalidades del producto. Para controlar la deuda se han realizado un sinnúmero de investigaciones que aportan en su conocimiento y posible solución. En la tesis realizada por (Torres Ruiz, 2019), se plantea gestionar la deuda técnica mediante siete fases fundamentadas en buenas prácticas. Como resultado presenta un enfoque de gestión de la deuda y concluye que tener un conocimiento cabal por parte de los programadores sobre este fenómeno contribuirá al proceso de desarrollo.

En la tesis desarrollada por (Simone Tonin, 2018) presenta el resultado del estudio realizado a nueve equipos de desarrollo en un ambiente académico con el uso del modelo XP y cuatro equipos de dos compañías de desarrollo de software brasileñas con la finalidad de analizar la deuda técnica. Como resultado se obtiene un modelo conceptual para la gestión de la deuda técnica, en el cual se presenta el marco para identificar, monitorear, categorizar, priorizar y pagar la deuda.

Igualmente, en la tesis desarrollada por (Hoyos Rentería, 2021) analiza un método para identificar toggle debt y su deuda técnica, para este estudio utilizan 438 funciones de 72 proyectos de código abierto donde seleccionan aquellos que usan toggles con la finalidad de identificar la deuda técnica que se cruzan con los feature toggles que se identificaron

preliminarmente. El investigador concluye que, el usar herramientas automatizadas beneficia en la revisión de código.

En este sentido la tesis realizada por (Daniel Alejandro, 2021) desarrolla SIRA (Software Improvement in the Reconstruction of Aechitectures), una herramienta semi automática que, por medio de la recomendación de mejores prácticas, permite la reconstrucción de la arquitectura de software sostenible, para esto se han utilizado herramientas, técnicas y modelos en el marco de la deuda técnica relacionadas con el consumo de energía. Como resultado de la investigación se obtuvieron una herramienta y enfoques de medición que permitieron reconstruir la arquitectura considerando la deuda técnica en el contexto del consumo de energía.

Dentro de este orden, en la tesis realizada por (Preciado Orozco , 2022) estudia una plataforma corporativa que posee una gran cantidad de deuda técnica con la finalidad de reducirla y asegurar la calidad del software mediante la definición de una nueva arquitectura mediante la refactorización, nuevas funcionalidades y la creación de normas en el equipo de desarrollo para controlar la deuda.

Por otro lado, las aplicaciones de las ciencias de datos se encuentran en auge gracias a las herramientas, métodos y tecnología que a partir de los datos permiten generar significado, sin embargo, las metodologías que se utilizan para desarrollar este tipo de aplicaciones generan deuda técnica que, en la mayoría de las ocasiones llevan al fracaso de los proyectos. En la tesis realizada por (Tamayo Palomino, 2021) se integran los enfoques DataOps y MLOps en el desarrollo de aplicaciones Big Data y Machine Learning obteniendo como resultado la reducción de la deuda técnica generadas con el uso de las metodologías de desarrollo tradicionales.

También, en la tesis realizada por (Rebouças de Almeida, 2022) estudia los plazos ajustados y la priorización de las características sobre el pago de la deuda técnica, además

analizan un modelo para la toma de decisiones que considera los activos fijos de TI para priorizar la deuda técnica, para lo cual aplica su marco en dos empresas con varios estudios de caso, obteniendo como resultado que el enfoque propuesto permite priorizar la deuda técnica y la comunicación entre los stakeholders.

Ahora bien, con relación a la codificación en la tesis desarrollada por (González Núñez, 2021) utiliza las herramientas SonaeQube y GitLab para analizar de manera automática el código en la industria del software. SonarQube realiza el análisis estático del código detectando métricas con relación a la complejidad cognitiva, code smells, errores, test unitarios y de integración. De acuerdo con el estudio realizado se propone un enfoque basado en la gamificación para minimizar la deuda técnica y beneficiar a la calidad del producto.

Finamente, la tesis desarrollada por (Sánchez Montoya, 2021) estudia la deuda técnica en los requerimientos, para lo cual se enfoca en prevenir, identificar y gestionar la deuda en esta fase del desarrollo. Para la investigación desarrollan prácticas para hacer visible la deuda y controlarla con base a buenas prácticas de desarrollo y fundamentándose en los estándares ISO/IEC 29148, 15288 y 29110.

Mapeos Sistemáticos y revisión de la literatura

Los mapeos sistemáticos, estados del arte y revisión de la literatura aportan con la visión y comprensión lo que permite evaluar la actualidad de la deuda técnica en el campo científico. A continuación, se presenta los principales aportes de estas herramientas.

En el trabajo realizado por (Alfayez et al., 2020) presenta un análisis de la literatura para identificar enfoques de priorización de la deuda técnica, así como los métodos y técnicas de priorización que utilizan. Como resultado se encontró 24 enfoques que utilizan 10 técnicas de priorización, además se evidenció que estos enfoques no han sido evaluados en la industria.

Los autores (Lenarduzzi et al., 2021) realizan una revisión sistemática para evidenciar como se prioriza la deuda técnica en las organizaciones especialmente en la corrección de errores y desarrollo de nuevas funciones durante el proceso de refactorización, para esto seleccionaron 38 investigaciones con datos sobre modelos, factores, medidas y herramientas para priorizar la deuda. Se demostró que la deuda de Código y la deuda arquitectónica son los tipos más investigados con relación a la priorización, además se encontró que existen pocos estudios sobre la deuda de prueba y la deuda de requisitos.

En el mapeo sistemático realizado por (Murillo et al., 2023) se trabajó con investigaciones publicadas entre 2015 al 2022 en ocho bases de datos, obteniendo 117 artículos relacionados a la identificación y gestión de deuda técnica, producto de la investigación se encontraron tres nuevos términos relacionado con la deuda técnica: deuda de seguridad, elasticidad y variabilidad, así como también una gran cantidad de estudios sobre la identificación automática de la deuda. Sin embargo, no encontraron literatura abundante sobre la documentación, personas e infraestructura relacionadas a la deuda técnica, finalmente se determinó que la propuesta de nuevos tipos de deuda a sufrido una disminución desde 2016.

Así mismo, el estudio de (Melo et al., 2022) se realiza un estudio sistemático de la literatura donde se busca evidenciar la identificación, métricas y gestión de requisitos de deuda técnica, para esto selecciona 61 investigaciones consideradas entre el 2010 al 2020. En la investigación se identificaron varias métricas para la deuda técnica de requerimientos, sin embargo, se determinó que no existen herramientas automatizadas para la detección y gestión de deuda técnica especifica.

En relación con la industria, el trabajo realizado por (Avgeriou et al., 2024) investigan el estado del arte de la gestión de la deuda técnica a nivel industrial y en las comunidades de investigación con el fin de alcanzar una visión a cinco años de la gestión de la deuda. De

acuerdo con el análisis realizado determinaron que las aplicaciones catalogadas como grandes son las más afectadas en la calidad debido a la deuda técnica, especialmente aquellos que tienen una vida útil considerablemente larga, concluyen que es fundamental que tanto equipos de desarrollo como organizaciones incorporen la gestión de la deuda técnica como una actividad común en el desarrollo y que se adopten modelos financieros de gestión de la Deuda.

Por su parte, los autores (Ciancarini y Russo, 2020) estudian la dimensión social y técnica de la deuda técnica con base a un mapeo teórico donde se realiza entrevistas a 124 profesionales del desarrollo especialmente del sector financiero. Como resultado se propone un modelo estratégico para la gestión de la deuda técnica.

Por último, en la investigación desarrollada por (Murillo y Jenkins, 2021) presenta una revisión de la literatura para detectar y obtener métricas para la deuda técnica producida en la codificación mediante el análisis estático del código utilizando herramientas automatizadas como el Sonarqube, para la investigación trabajan con un estudio de caso concluyendo que, aunque la herramienta apoya en cierta medida a la gestión de la deuda, esta presenta limitaciones al entregar datos inexactos.

Sobre la deuda técnica existen varias publicaciones que recopilan y analizan la evidencia científica para identificar, analizar y sintetizar estudios sobre el tema, estos trabajos permiten tener una mejor comprensión para gestionar la deuda técnica controlándola y mejorando la calidad del software.

Clasificación de la Deuda Técnica

Si bien no existe una clasificación universalmente aceptada es importante que el equipo de desarrollo identifique la deuda técnica de una manera efectiva para poder abordarla de una manera estratégica permitiendo una adecuada gestión orientada a la calidad en el producto.

La falta de consenso para una taxonomía genérica se debe a varios factores como la naturaleza subjetiva de la deuda técnica ya que los tipos y magnitud varían según el contexto y el diseño del proyecto. Además, la evolución constante de la tecnología contribuye al aparecimiento de nuevos tipos de deuda dificultando implantar una clasificación fija. Finalmente, los desarrolladores poseen diferentes perspectivas de lo que realmente es la deuda técnica, dificultando una clasificación aceptada y estandarizada.

De ahí que, la deuda técnica se puede clasificar desde diferentes enfoques de acuerdo con características y repercusiones que ocasiona en el desarrollo del Software. Desde un punto de vista general la deuda técnica se puede clasificar en:

- Deuda Técnica no intencional, que es aquella que se produce cuando el equipo de desarrollo de manera no deliberada incurre en actividades que generan deuda por la mala planificación del proyecto, la inexperiencia y por cambios no planificados en el entorno.
- Deuda Técnica intencional, en la cual se incurre por diferentes aspectos como la presión para cumplir los plazos de entrega en la cual el equipo de desarrollo de software realiza actividades incompletas y sin control de calidad de manera voluntaria con la finalidad de beneficiarse a corto plazo con soluciones incompletas o no realizadas.

Otra categoría es la relacionada a las actividades y fases en el desarrollo de software en la cual varios autores la clasifican como se observa en la Tabla 2:

Tabla 2.

Tipos de Deuda Técnica

Tipo	**Definición**
Diseño	Este tipo se refiere a la deuda creada al no utilizar los principios y técnicas de las ciencias ingeniería del software en el diseño, lo que a futuro restringe la escalabilidad y actualización del software.

Tipo	Definición
Código	Se refiere a la implementación del código fuente sin la aplicación de reglas y buenas prácticas de programación afectando la legibilidad y mantenimiento del código.
Arquitectura	Este tipo incluye a los problemas que se generan por una mala planificación en el diseño de los modelos y patrones de la arquitectura del producto lo que afecta a la estructura y aspectos internos del software.
Prueba	Se refiere a los problemas por la ausencia o mala aplicación de los procesos de verificación y pruebas al software, afectando la prevención de errores y el rendimiento del producto.
Documentación	Se refiere a todos los problemas relacionados por la falta o ineficiente documentación del proyecto de software.
Defecto	En esta categoría se incluyen los defectos identificados del sistema o de componentes que no se corrigen por otras prioridades en el desarrollo, por los cuales no se ejecuta adecuadamente una función determinada.
Infraestructura	Se refiere a los problemas en el desarrollo derivados de la infraestructura, herramientas y entornos que se presentan en la organización del proyecto.
Requisitos	Este tipo se refiere a las deficiencias en el software producto de una mala definición, seguimiento y verificación de los requisitos del sistema.
Personas	Se refiere a los problemas del equipo de trabajo y personas involucradas en el desarrollo del software lo que entorpece el normal avance del proyecto.
Construcción	Este tipo abarca los problemas que dificultan la tarea de construcción del software con actividades que requieren más tiempo de lo normal innecesariamente.
Proceso	Se refiere a los procesos incompletos o ineficientes en las fases del desarrollo de software.
Usabilidad	Consiste en las malas decisiones de problemas detectados en la usabilidad que los posponen para solucionar más adelante.
Servicio	Este tipo se refiere a errores al comunicarse con el servidor o servicio web lo que se deriva en una incompatibilidad entre lo que necesita la aplicación y las características del servicio.

Tipo	Definición
Control de versiones	Se refiere a los problemas en los controles de las versiones que se generan al realizar modificaciones en el diseño o codificación.

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: (Melo et al., 2022)

A pesar de que no existe una clasificación estándar de la deuda técnica, para la presente investigación es fundamental comprender los tipos de deuda y clasificarla para que el equipo de desarrollo pueda comprender el impacto en el proyecto y de esta manera priorice las actividades para minimizarla. También, al conocer las causas de los tipos de deuda técnica se puede prevenir su efecto con medidas efectivas para evitarla. Asimismo, al utilizar un léxico común en los términos de la deuda técnica se posibilita la comunicación tanto interna como externa con las partes interesadas lo que contribuye a una gestión y toma de decisiones efectivas.

Identificación de la Deuda Técnica

Identificar la deuda técnica es fundamental en el ámbito del desarrollo de software para controlarla a tiempo y evitar problemas futuros, si bien los equipos de desarrollo no le dan la importancia adecuada, conocen de su existencia y no la toman en serio hasta que un mínimo cambio en el software provoca un gran caos.

De ahí que, las razones para identificar la deuda técnica son cruciales ya que si no se la realiza a tiempo puede afectar al software poco a poco hasta que en un momento puede producir una falla grave que incluso llegue a paralizar el proyecto, además, corregir la deuda a tiempo es más barato ya que al convertirse en un problema mayor los costos para pagarla son muy altos. También, identificar la deuda de manera técnica va a permitir la generación de un código bien estructurado y estable lo que contribuye a la calidad del producto.

En el mismo sentido, el no identificar la deuda técnica se manifiesta en el deterioro de la estabilidad del software con un código pobre, documentación ineficiente y una estructura de baja calidad. Como resultado de su acumulación el código se vuelve inentendible por lo que el costo en el mantenimiento aumenta de una manera exponencial al mismo tiempo que requiere de un trabajo extra de los desarrolladores.

Por otro lado, para detectar la deuda técnica se pueden utilizar varias técnicas y métodos, los principales se listan a continuación:

- Análisis del código identificando código duplicado, variables sin usar, funciones y métodos demasiados largos entre otros.
- Ejecución de pruebas para identificar errores y el comportamiento individual de las unidades de código.
- Reuniones continuas con el equipo de desarrollo para identificar zonas problemáticas.
- El uso de métricas para comprobar el estado del software e identificar las diferentes áreas de mejora.

La identificación precisa y solida de la deuda técnica minimiza el riesgo de pasar por alto instancias de la deuda lo que garantiza la recuperación a tiempo de los problemas detectados en el desarrollo, permitiendo que el equipo aborde los problemas más críticos de la deuda técnica manteniendo la calidad del producto (Shivashankar et al., 2024).

Por consiguiente, la identificación de la deuda técnica acarrea la toma de decisiones que en muchos casos sin un marco establecido no es la adecuada. Asumir las actividades que permitan el flujo normal de las tareas para el pago de la deuda es esencial para el ahorro de costos y recursos, para esto se debe considerar el instante para aplicar la remediación, por ejemplo, es momento de refactorizar el código, es más económico implementar un código nuevo o reelaborarlo, etc. Tomar la decisión adecuada conducirá a un pago controlado de la deuda y un control en el interés generado por la deuda técnica (Perera et al., 2023).

Deuda técnica en sistemas para Inteligencia Artificial

El desarrollo de sistemas de inteligencia artificial ha tomado auge en la actualidad ya que son una herramienta de apoyo para la toma de decisiones informadas que se encuentra en diferentes dominios, por lo que garantizar la calidad de los modelos de inteligencia artificial, componentes y algoritmos utilizados es uno de los principales retos en los desarrolladores. La deuda técnica de Inteligencia Artificial es el resultado de los procesos subóptimos que afectan tanto al código como a la arquitectura en los sistemas habilitados para Inteligencia Artificial, detectar esta deuda y mitigarla es el desafío de los desarrolladores ante la falta de herramientas automatizadas que permitan realizar esta tarea (Recupito et al., 2024).

En el mismo sentido, los aplicativos actuales han adoptado a los sistemas de aprendizaje automático basados en Inteligencia Artificial como parte fundamental en sus plataformas, para brindar asistencia a los usuarios en sus búsquedas en dependencia de sus preferencias. El desarrollar aplicaciones en los que se deba implementar este tipo de sistemas denota desafíos debido a la escalabilidad, las múltiples fuentes de datos con las que se debe trabajar, los procesos de retroalimentación, las consideraciones éticas entre otros. Estas características pueden generar deuda técnica que surge por procesos subóptimos, códigos y algoritmos incompletos y la ausencia de pruebas, lo que, a la larga resulta en un sistema no eficiente, difícil de mantener y de baja calidad (Moreschini et al., 2024).

La tecnología avanza de manera acelerada y la inteligencia artificial en la actualidad es parte de la mayoría de las aplicaciones, por lo que se debe analizar la implicación de la deuda técnica en este tipo de sistemas y las actividades para detectarla y gestionarla de manera óptima.

La Deuda Técnica y la experiencia de Usuario

La experiencia de usuario considera aspectos como la funcionalidad, interacción, habilidades del usuario y el contexto en el uso de las aplicaciones que deben cumplir las

especificaciones y requisitos en el marco del uso del software. La deuda técnica en la experiencia de usuario se produce por un diseño incorrecto de la interfaz de usuario al utilizar soluciones poco convencionales en las características de usabilidad que deben ajustarse a futuro, con un costo alto que terminan pagándolo tanto desarrolladores como usuarios. Tomar en cuenta aspectos técnicos y no técnicos para medir la experiencia de usuario permitirá aplicar actividades para priorizar y reembolsar la deuda de usabilidad (Rodriguez et al., 2023).

La calidad en el uso de una aplicación involucra al usuario y su experiencia en la usabilidad, para esto se debe tomar en cuenta los diferentes tipos de usuarios que tienen características diversas, preferencias personales, que usan la aplicación en diferentes contextos y costumbres del entorno. El tomar en cuenta estas particularidades generan deuda técnica, ya que, los desarrolladores dejan de lado la adaptación del software para aquel potencial usuario con muy diversos enfoques.

Impacto de la Deuda Técnica

El impacto o interés de la deuda técnica crece conforme al tiempo que se toma en pagarlo, es por lo que para priorizar las tareas que permitan eliminar la deuda, se debe estimar adecuadamente el tiempo para solucionar los problemas en las actividades afectadas. Para esto se debe medir el impacto de la deuda técnica en las diferentes actividades y priorizar su eliminación, además, conocer el impacto de la deuda en la calidad del producto mejorará el mantenimiento del software y los costos asociados a esta actividad, lo que requiere una estimación lo más precisa posible de las principales propiedades de la deuda técnica (Lenarduzzi et al., 2021).

No existen técnicas de prevención y pago de la deuda técnica que sean exclusivas para un modelo de desarrollo de software específico siendo su impacto similar, ya sea que se seleccione procesos agiles, tradicionales o híbridos, la percepción de la deuda es parecida. Sin

embargo, debido a las características de la agilidad, los equipos de desarrollo que trabajan con este modelo utilizan una mayor cantidad de actividades para el seguimiento y pago, debido al proceso iterativo que facilita la mitigación de la deuda mediante tareas que se pueden incluir en cada entregable (Rios et al., 2021).

De ahí que, el impacto se evidencia al priorizar determinadas funciones y características del software dejando en segundo plano la calidad, lo que aumenta la brecha que existe entre la empresa y las Tecnologías de la Información afectando los procesos empresariales y la toma de decisiones, por la falta de comunicación con el equipo técnico para anticiparse a la creación de la deuda mediante una planificación para estimar como gestionar de una maneta técnica la deuda técnica (Reboucas de Almeida et al., 2023).

Medición de la Deuda Técnica

El uso efectivo de las métricas contribuye a que las organizaciones tengan un sólido entendimiento de los objetivos a alcanzar a corto y largo plazo. Medir la deuda técnica es una actividad fundamental en los proyectos de desarrollo ya que permitirá obtener una mejor perspectiva tanto cuantitativa como cualitativa del estado real de la deuda con lo cual se proporciona datos valiosos para la toma de decisiones, planificación de tareas y la priorización de actividades enfocadas a gestionar adecuadamente la deuda.

La medición de la deuda técnica es una práctica necesaria para visualizar el estado de la deuda con lo que se logrará identificar las actividades que requieren mayor atención y mejora, al mismo tiempo que permite la evaluación del riesgo asociado a la deuda técnica para determinar las áreas que sufren un mayor impacto. Medir la deuda técnica de manera periódica proporciona datos objetivos para monitorear el alcance de la deuda y de esta manera priorizar las estrategias a implementar para mitigarla a tiempo.

Dentro de los métodos para las métricas de la deuda técnica se tienen las enfocadas al código que miden la complejidad ciclomática, el acoplamiento, la cohesión y la duplicación

de código. También se tienen herramientas para el análisis de código estático, de pruebas de cobertura de código y de pruebas de integración.

Los beneficios de medir la deuda técnica son la mejora en la fiabilidad y optimización del software, la reducción de costos en actividades correctivas, el aumento de productividad en el desarrollo y una mayor satisfacción del cliente garantizando la mantenibilidad del software y el éxito a largo plazo.

En este sentido, la medición de código es una de las practicas ya implantadas en los proyectos de desarrollo, esta actividad la realizan los equipos de manera manual, con plugin de los entornos de desarrollo o con herramientas automatizadas. Las reglas para encontrar defectos de código varías de una herramienta a otra obteniéndose en muchos casos falsos positivos al analizar la codificación, por lo que estas herramientas deben incluir la parametrización de los patrones para el análisis y la incorporación de funcionalidades que garanticen la valides de los datos obtenidos (Haertel et al., 2022).

Además, cumplir los estándares en la codificación determina la calidad del producto ya que de esta manera se crea un software libre de vulnerabilidades y errores. La precisión en los resultados que entregan las herramientas de análisis estático de código alerta a tiempo de posibles defectos en la construcción del software ahorrando tiempo y esfuerzo que puede ser aprovechado en otras tareas por los programadores. Sin embargo, estas herramientas no son cien por ciento efectivas y no cumplen con evaluar las principales métricas de desempeño que son la probabilidad de falsas alarmas, la precisión, la recuperación y el índice de detección (Desai y Jariwala, 2022).

Utilizar métricas basadas en las prácticas de la ingeniería de software de manera efectiva permitirá realizar un seguimiento de los tipos de deuda técnica que permita a los desarrolladores aplicar estrategias para mejorar la productividad del desarrollo de software.

Gestión de la deuda técnica

Con relación a la Gestión de la deuda técnica en la colección SEI Series in Software Engineering se encuentra la obra publicada por (Kruchten et al., 2019) en el cual se aborda la deuda técnica y su impacto por medio de ejemplos reales. Plantean controlar la deuda, así como analizar los costos y pagarla responsablemente mediante una adecuada gestión de esta.

En la investigación realizada por (Berenguer et al., 2023) se analiza seis elementos de la gestión de la deuda técnica: las causas, efectos, las prácticas preventivas, las razones para la no prevención, las prácticas de pago y los motivos de no pagarla. Para el estudio se encuesto a 653 profesionales encontrando que el principal problema relacionado a la deuda técnica es la codificación, pero los problemas relacionados con la planificación y gestión de proyectos, el diseño, la arquitectura, los factores humanos, la comprensión, la calidad, los procesos, los requerimientos, la comprobación, las pruebas y la organización son factores asociados a la deuda técnica.

Debe señalarse que, en la ingeniería en sistemas, muchos proyectos han fracasado por decisiones rápidas tomadas especialmente en la fase inicial del desarrollo. En el trabajo de investigación realizado por (Yang et al., 2023) realiza una analogía entre esta metáfora y la deuda técnica, para lo cual desarrolla una taxonomía de la deuda mediante sistemas complejos intensivos con la finalidad de concientizar en la identificación temprana de la deuda técnica en el proceso de adquisición y de esta manera garantizar la eficiencia, eficacia y responsabilidad de sistemas complejos.

En el campo de la programación, en la investigación realizada por (Cucalón-Gaibor y Sandoval Gutierrez, 2023) analiza la deuda técnica ocasionada por bugs y code smells que se genera en las actualizaciones de programas de tipo financiero por errores que el compilador no identifica. Para el estudio se definieron cinco métricas, tres para bugs que evitan interrupciones en los programas: Token Registry Metric, SSID Event Metric y TDE Object

ID Metric y dos para code smells: Class Prefix Metric y File Las Letter Metric. Se concluye que las métricas optimizan el mantenimiento del software y que el uso de comentarios en la codificación minimiza significativamente la deuda técnica.

Romper las barreras de una gestión de la deuda técnica que cumpla sus objetivos es el reto en los equipos de desarrollo, sin embargo, la falta de comunicación interna, los cambios repentimos y la dificultad para acceder a la información trascedente afecta esta tarea y la toma de decisiones. El análisis técnico y el seguimiento de la deuda técnica es importante para mitigarla para lo cual se deben incluir actividades como la identificación de las fuentes, las características de los problemas afectados, la medición y el impacto de la deuda, todo esto con el apoyo de herramientas automatizadas para obtener mejores resultados (Fungprasertkul, et al., 2024).

La gestión efectiva de la deuda técnica tiene un efecto directo en el equipo de desarrollo al aumentar la productividad, el ambiente de trabajo y el estado de ánimo de los desarrolladores. Las actividades desarrolladas en la gestión influyen de manera positiva en la moral del equipo dando como resultado que los desarrolladores adopten la cultura de prevención de la deuda y la aplicación de buenas prácticas de desarrollo lo que conlleva a entregar un producto de calidad (Besker et al., 2020).

Documentar y gestionar la deuda técnica de una manera formal es uno de los problemas en el desarrollo de software donde se utilizan metodologías ágiles. En la investigación realizada por (Xavier et al., 2023) presentan LTD, Less Technical Debt Framework que es un marco que, por medio del consenso, descubrimiento, planificación y pago de la deuda, aumenta la conciencia y apoya en la gestión de la deuda técnica. Las cuatro actividades aplicadas del marco lograron reducir la deuda técnica sin un gran esfuerzo adicional del equipo de desarrollo, fomentando la responsabilidad de incluir la deuda en las reuniones del equipo.

Muchos enfoques para la gestión de la deuda técnica utilizan como mecanismos para identificarla a los olores de código y la presencia de antipatrones. Una de las formas más utilizadas para detectar los code smells son con las herramientas automatizadas de análisis de código estático ya que estas se ejecutan en los entornos de desarrollo y advierten de código sospechoso a los programadores además del análisis de clases y funciones. Los antipatrones se refieren a los problemas en el diseño y las actividades subyacentes de las estructuras arquitectónicas. Si bien los code smells se detectan sin mucho esfuerzo, los antipatrones no son muy conocidos por lo que son difíciles de detectar, pero una comprensión adecuada de estos términos ayudará en el proceso de gestión de la deuda técnica (Ramirez Lahti et al., 2021).

El uso de técnicas inteligentes ha ganado terreno en distintas actividades de la gestión de la deuda técnica, en este contexto la exploración de datos para adquirir conocimiento, razonamiento, aprendizaje y la percepción han sido fundamentales para la toma adecuada de decisiones en el control de la deuda. Las técnicas inteligentes más utilizadas son el Machine Learning y Reasoning under doubt especialmente en la identificación, medición y monitoreo de la deuda técnica relacionada con el diseño, la codificación y la arquitectura. Sin embargo, estas técnicas son un apoyo para el equipo de desarrollo especialmente para la identificación de la deuda técnica ya que los enfoques existentes señalan que aún se requiere de una intervención humana para una adecuada gestión de la deuda técnica (Albuquerque et al., 2023).

Por consiguiente, la gestión de la deuda técnica es necesaria para controlar el riesgo de enfrentarse a un estado nefasto del desarrollo de software en el futuro, para lo cual los stakeholders deben trabajar de manera colaborativa para cambiar las prácticas en las distintas fases del desarrollo. Para una mejora en la gestión los modelos deben enfocarse en la generación de valor adecuándose al contexto de desarrollo propiciando un cambio en las

actividades de desarrollo en la cual el equipo desarrolle estrategias convenientes enfocadas a mitigar la deuda en su práctica diaria (Avgeriou et al., 2023).

Debe señalarse que gestionar determinadas partes en el desarrollo no genera mayores beneficios, ya que se debe considerar al sistema como un todo. Si un equipo se centra en la calidad del código puede tener problemas en el diseño y patrones arquitectónicos que incluso pueden afectar a toda la estructura del proyecto más que los problemas de codificación. Es por lo que el enfoque de gestión no puede aplicarse de forma parcial ya que un componente con deuda técnica no controlada puede afectar al resto de componentes provocando una falla en cascada, es por lo que el marco de gestión debe ser lo suficientemente sólido para abarcar todas las partes del sistema (Schütz y Gómez, 2020).

Enfoques para la Gestión de la Deuda Técnica

Reembolso de la SATD de manera automática

DLRepay es un modelo de aprendizaje que toma los comentarios SATD junto con el código y entrega una codificación sin deuda técnica de manera automatizada. Para esto presentan una taxonomía que se basa en la relación y beneficio de los comentarios SATD, esta nueva versión de código permite el rembolso completo y directo de la deuda técnica autoamitida (Alhefdhi et al., 2024).

De igual manera, pagar la SATD requiere de un esfuerzo extra de los desarrolladores, para esto se han planteado herramientas automatizadas con diferentes técnicas como es el caso del modelo generativo basado en neuronas propuesto por (Mastropaolo et al., 2023). Este modelo de aprendizaje profundo generativo corrige el código genérico, elimina el SATD y realiza el ajuste de los comentarios del código, obteniendo como resultado que el pago automático se lo realiza en una pequeña cantidad de casos, siendo necesario ampliar el campo de investigación.

GGSATD

La red neuronal de grafo cerrado con inductividad, para detectar SATD es un enfoque que utiliza la automatización para la detección de la deuda técnica autoamitida. Para la detección de comentarios generados por SATD se han utilizado mayormente técnicas manuales, introduciéndose poco a poco métodos automatizados, dando lugar a las técnicas hibridas sin obtener resultados óptimos. GGSATD es un enfoque que mediante gráficos para las instancias de comentarios identifica automáticamente las SATD y mediante redes neuronales actualiza los nodos del gráfico representada por coocurrencias, con lo que se genera una matriz, donde se obtienen los correspondientes vectores para identificar si, el comentario es un SATD (Yu et al., 2022).

Jitterbug

La gestión de la SATD es fundamental para mantener la calidad del software lo que requiere un alto costo y esfuerzo por parte del equipo de desarrollo. Jitterbug es una herramienta para detectar automáticamente la SATD que, con procesos como la identificación por reconocimiento de patrones y luego con herramientas de aprendizaje automático que posee datos de comentarios históricos, así como datos proporcionados por las nuevas decisiones de los desarrolladores actualiza el modelo de aprendizaje encontrando automáticamente los SATD más comunes o fáciles de encontrar, además, con toda esta información logra ser una base para que los desarrollares encuentren los SATD no fácilmente identificables o difíciles de identificar (Yu et al., 2022).

Dynamic Document Generation (DDG) and Automated Documentation Testing (ADT).

La documentación es uno de los principales problemas en el desarrollo del software, la falta de esta, y los documentos incompletos o desactualizados contribuye en la mala calidad del producto y el surgimiento de la deuda técnica. La deuda de documentación en la mayoría de los casos es indetectable debido a que los desarrolladores se enfocan en entregar

un producto funcional desestimando los recursos y fases para lograrlo. El enfoque de la Generación Dinámica de Documentos (DDG) utiliza los metadatos de la codificación para la generación de documentación y se complementa con las Pruebas de documentación automatizadas (ADT) que es un enfoque que considera a la documentación ya creada como fuente de información para la verificación de las pruebas (Silva et al., 2023).

SoHist

Esta herramienta permite evaluar y priorizar la deuda técnica mediante el análisis de código, para lo cual analiza las limitaciones de herramientas automatizadas existentes en el mercado y con base a la evolución de la deuda técnica en el tiempo proporciona un análisis de código extendidas. Sohist se adapta a las necesidades de cada proyecto y realiza un análisis histórico de la calidad del código utilizando una base de datos y una instancia de SonarQube para la activación del análisis de código. Esta herramienta se encuentra disponible para los usuarios ya que es una herramienta de código abierto que se encuentra en fase de prototipo por lo que tiene limitaciones de usabilidad, aunque presenta un renovado enfoque de visualización (Dornauer et al., 2023).

2. Metodología, Diseño, Software o Implementación propuesta

La investigación se realiza desde una postura crítica constructivista, en la cual el conocimiento es una construcción que converge contextos de la historia, cultura y tecnología, por lo que, no se considera una representación objetiva de la realidad. Desde esta perspectiva los actores del desarrollo de software aportan conocimiento con base a sus experiencias y los problemas que han encontrado en sus actividades relacionadas con la agilidad y la deuda técnica.

La perspectiva crítica permite analizar las aristas del desarrollo de software al interrogar los aspectos de la práctica desde un punto de vista técnico para eliminar supuestos. Con el constructivismo, se busca interpretar la deuda técnica y su gestión en función de los principios ágiles en el desarrollo de software.

En el diseño de la investigación se aplicó el paradigma mixto con base a (Pinto-Llorente y otros, 2021), con lo que se logró la integración real entre los paradigmas cualitativos y cuantitativos, logrando una mejor comprensión del problema, reducción de sesgos y generación de conocimiento, con lo cual se exploraron las perspectivas de los desarrolladores y sus necesidades con relación a la deuda técnica en el escenario donde se desarrolla el problema.

Este paradigma permite la generación de conocimiento detallado ya que en el enfoque cualitativo con el uso de técnicas como la entrevista se identifican las debilidades y oportunidades del marco propuesto en la práctica, además, se realiza una evaluación cualitativa al validar la pertinencia, usabilidad y aceptación del marco. La investigación cuantitativa permitió obtener datos obtenidos de una herramienta validada que permitió comparar métricas antes de aplicar el marco y luego de utilizar este enfoque en la gestión de la deuda técnica.

También, se utilizó la investigación bibliográfica y documental de los estudios relevantes, para profundizar en el entendimiento de las variables de la investigación y los temas relacionados a estas, lo que permitió obtener el conocimiento adecuado que permita entender y resolver el problema planteado.

Además, se utilizó la investigación cuasi experimental que permitió recopilar y analizar los datos cuantitativos al validar el marco planteado en la práctica, comparando los resultados de su aplicación frente al modelo tradicional al estudiar y analizar cualitativamente las experiencias, perspectivas y opiniones de los desarrolladores con el uso del nuevo modelo.

De ahí que, la justificación metodológica de la investigación combina estrategias mixtas al utilizar enfoques cualitativos, cuantitativos y cuasi experimentales, para estudiar el fenómeno desde una complejidad socio-técnica con lo que se aporta empíricamente al estudio del problema, ya que la deuda técnica no es solamente una anomalía técnica ya que, en el proceso de desarrollo es el resultado de malas decisiones, la agilidad y los criterios para determinar adecuadamente el valor.

La combinación de estos enfoques va a permitir una mejor comprensión del problema gracias al modelo cualitativo, una medición objetiva con el modelo cuantitativo y la evaluación de los efectos de la intervención en el estudio de caso con el modelo cuasi experimental de ahí que el diseño de investigación es metodológicamente consistente con lo que se asegura el rigor científico y la relevancia práctica.

Los pasos aplicados en la investigación son seleccionar el grupo de estudio, ejecutar el estudio, realizar el análisis de los resultados lo que permitió finalmente establecer las conclusiones.

Variables

Variable Independiente: Gestión de deuda técnica

Definición Operativa: (Berenguer et al., 2023) indican que el término deuda técnica se refiere a los trabajos pospuestos en el desarrollo de software que generan a corto plazo beneficios pero que a largo plazo impactan en el software negativamente con menor productividad y mayores costos. La gestión de la deuda técnica es un conjunto de actividades encaminadas a prevenir el impacto de su presencia en los proyectos de software, mediante su medición, análisis de la evolución, prevención y las acciones necesarias para pagar la deuda (Ampatzoglou et al., 2022).

Dimensiones.

En la tabla 3 se presente la operacionalización de la dimensión Deuda Técnica de Código

Tabla 3.

Operacionalización Variable Independiente. Dimensión Deuda Técnica de Código

Indicador	**Definición Operativa**	**Medición**
Complejidad del código.	Nivel de dificultad para entender, modificar o extender el código	Métricas de complejidad ciclomática
Código duplicado	Porcentaje de código repetido	% de duplicación
Code smells	Presencia de malas prácticas	Número de smells por módulo
Violaciones a estándares	Incumplimiento de normas de codificación	Número de reglas incumplidas

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Diseño de la investigación

En la tabla 4 se presente la operacionalización de la dimensión Deuda Técnica Arquitectónica

Tabla 4.

Operacionalización Variable Independiente. Dimensión Deuda Técnica Arquitectónica

Indicador	**Definición Operativa**	**Medición**
Acoplamiento excesivo	Dependencia fuerte entre componentes	Índices de acoplamiento
Falta de modularidad	Dificultad para aislar funcionalidades	Métricas de cohesión
Decisiones arquitectónicas	Ausencia de diseño estructural estable	Evidencia documental

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Diseño de la investigación

En la tabla 5 se presente la operacionalización de la dimensión Deuda Técnica Documental

Tabla 5.

Operacionalización Variable Independiente. Dimensión Deuda Técnica Documental

Indicador	**Definición Operativa**	**Medición**
Documentación incompleta	Falta de descripción técnica	% de módulos sin documentación
Documentación desactualizada	Desfase entre código y documentación	Análisis de consistencia
Ausencia de decisiones técnicas registradas	Falta de trazabilidad	Conteo de decisiones no documentadas

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Diseño de la investigación

En la tabla 6 se presente la operacionalización de la dimensión Deuda Técnica de Pruebas

Tabla 6.

Operacionalización Variable Independiente. Dimensión Deuda Técnica de Pruebas

Indicador	**Definición Operativa**	**Medición**
Cobertura de pruebas	Grado de validación automatizada	% de cobertura
Pruebas frágiles o ausentes	Dificultad para detectar defectos	Número de fallas no detectadas
Defectos recurrentes	Reaparición de errores	Tasa de defectos repetidos

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Diseño de la investigación

Con base a las definiciones descritas la variable independiente en el enfoque cuantitativo corresponde con la investigación del marco de gestión de deuda técnica propuesto.

Variable Dependiente: Modelos ágiles de desarrollo de software.

Definición Operativa: (Al-Baik et al., 2024) señalan que los modelos ágiles son aquellos que aplican los principios de agilidad en el desarrollo de software con prácticas como el desarrollo iterativo, la adaptabilidad, entregas continuas y la colaboración de los stakeholders. De acuerdo con (Behutiye et al., 2017) la capacidad para gestionar la deuda técnica se encuentra directamente relacionada con la efectividad de las metodologías ágiles en el

desarrollo, de estas actividades depende la calidad del producto y el cumplimiento de los principios ágiles.

Dimensiones.

En la tabla 7 se presente la operacionalización de la dimensión Toma de decisiones ágiles

Tabla 7.

Operacionalización Variable Independiente. Dimensión Toma de decisiones ágiles

Indicador	**Definición Operativa**	**Medición**
Velocidad	Cantidad promedio de Trabajo	Requisitos o Historias cumplidas
Entregas continuas	Tiempo Interacción	Cumplimiento de cronograma
Transparencia en entregas	Estimación de entregas	# de entregas a tiempo
Calidad del producto	Cumplimiento de atributos de calidad	# de defectos del producto
Priorización basada en valor	Alineación con objetivos estratégicos	Deuda Técnica Basado en el Valor

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Diseño de la investigación

Los modelos ágiles son utilizados en la industria del software en el desarrollo de aplicaciones, con lo que se mejorará la agilidad con la aplicación del marco de gestión de la deuda técnica.

Categorías de Análisis

Las categorías de análisis en el enfoque cualitativo describen el fenómeno estudiado como una estrategia metodológica que utiliza categorías de estudio, que, por medio de las dimensiones e indicadores, además, del marco conceptual y los objetivos del estudio, guían la investigación mediante conceptos que permiten un mejor entendimiento del tema (Oranga y Matere, 2023).

En la Tabla 8, se presenta las categorías de análisis identificadas en la tesis.

Tabla 8.

Categoría de Análisis

Categoría	Definición conceptual	Definición operacional	Indicadores
Conocimiento y Percepción	Cómo los desarrolladores identifican, entienden y conceptualizan la deuda técnica en el contexto del desarrollo de software.	Se evidencia mediante las respuestas de los desarrolladores que, describen qué es la deuda técnica y la forma en cómo la perciben en las actividades de desarrollo.	• Deuda técnica • Problemas en el desarrollo
Prácticas Actuales	Procesos y herramientas para identificar, abordar, controlar y reducir la deuda técnica	Se identifica en las actividades de los desarrolladores, las revisiones de código y el uso de herramientas específicas.	• Identificación de la deuda técnica • Forma de abordar la deuda técnica
Factores Contributivos	Razones que provocan la acumulación de deuda técnica en proyectos de software.	Se analiza los plazos ajustados, cambios de requisitos y la ausencia o falta de documentación en el desarrollo.	• Factores que contribuyen a la deuda técnica (Presión por plazos, falta de documentación, cambios en requerimientos) • Presión para liberar el producto.
Impacto y Consecuencias	Consecuencias de la deuda técnica que afecta la productividad, calidad y mantenimiento del software.	Se considera retrasos, errores recurrentes, aumento de costos, falta de calidad en el producto.	• Situaciones en que la deuda técnica impacta en el proyecto. • La deuda técnica y la eficiencia y calidad del desarrollo
Ideas y Soluciones	Estrategias para desarrollar un marco de gestión de la deuda técnica efectivo.	Se identifica las recomendaciones, ideas y conclusiones expresadas por	• Estrategias y practicas sugeridas.

Categoría	Definición conceptual	Definición operacional	Indicadores
		los entrevistados para gestionar la deuda técnica.	• Herramientas y procesos para la gestión de la deuda técnica.

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Enfoque Cualitativo de la Tesis

Población de estudio

La población para el presente estudio se encuentra compuesta por el equipo del departamento de desarrollo de la Universidad Regional Autónoma de Los Andes – UNIANDES. Esta población se encuentra compuesta por el jefe del equipo y cinco desarrolladores todos ingenieros en Sistemas e Informática con una dilatada experiencia desarrollando aplicaciones.

Para acceder a la población se solicitó permiso al director del área de Telemática de la Universidad y al jefe del equipo de desarrollo, posterior a la autorización obtenida para realizar la investigación se realizó el contacto con los miembros de la población objetivo primero de manera grupal y luego con entrevistas personales, con la finalidad de explicarles los objetivos y metodologías a utilizar en la investigación. Además, se les comunicó de los métodos para recolectar los datos, los aspectos éticos y la manera de validar los resultados en el estudio.

El departamento de desarrollo de la universidad se encuentra conformado por el jefe del equipo y cinco profesionales con experiencia en el desarrollo de aplicaciones de más de dos años.

El equipo ha buscado mantenerse a la vanguardia en su campo para lo cual se ha preocupado por el uso de nuevas herramientas, tecnologías y procesos para presentar proyectos eficientes.

Técnica de recolección de datos

Se utilizó la técnica de la entrevista con las que se obtuvieron datos detallados sobre las prácticas y experiencias de los desarrolladores con relación al problema de la deuda técnica lo que permitió comprender el fenómeno y generar conocimiento. Se trabajó con los profesionales desarrolladores donde se pretende tener un conocimiento de la realidad actual del problema, para luego recolectar los datos del caso de estudio. Esto permitió junto con la información relevante obtenida en la investigación bibliográfica y documental generar el nuevo modelo propuesto.

Para validar el modelo propuesto, se planteó experimentar con el mismo grupo, pero desde el enfoque de tratamiento y control mediante un caso de estudio donde se aplicó el enfoque, con la finalidad de validar la hipótesis propuesta, para esto se utilizó una herramienta que entregó datos cuantitativos para comparar las métricas antes y después de aplicar el marco.

La investigación requiere técnicas para medir objetivamente la deuda técnica y comprender las decisiones del equipo ágil debido a su enfoque mixto con predominio explicativo propositivo, por lo que se empleó en el enfoque cualitativo la entrevista semiestructurada y en el cuantitativo el análisis de la deuda técnica usando la herramienta SonarQube.

Relación con los objetivos y coherencia metodológica

La entrevista se enmarca en cumplir los objetivos planteados con relación a la comprensión de la deuda técnica, la toma de decisiones en la agilidad y la percepción del valor del negocio. Esta técnica aporta una evidencia interpretativa, siendo coherente con el enfoque mixto ya que aporta una evidencia interpretativa al contextualizar los resultados cuantitativos lo que fortalece la validez de la investigación.

La técnica cuantitativa aplica la herramienta SonarQube para obtener el análisis del código fuente de una manera automatizada. Esta herramienta se la utiliza para la detección y medición de la deuda técnica ampliamente ya que entrega datos con métricas para detectar la deuda técnica estimada, mantenibilidad, duplicación de código y complejidad, permitiendo cuantificar de manera sistemática el estado técnico del software.

Proceso de validación y confiabilidad

La validez de la herramienta SonarQube se respalda en el reconocimiento de esta en la industria y academia para la medición y calidad de la deuda técnica, garantizando la confiabilidad al ser un instrumento consistente asegurando la trazabilidad de los datos. Este instrumento se alinea de manera directa con los objetivos de medir la deuda existente, estableces una evidencia cuantitativa y evaluar su impacto técnico, lo que refuerza la validez de los resultados.

Validación de la técnica de recolección de datos

Para validar el contenido de la guía de entrevista se lo realizó mediante juicio de expertos, quienes evalúan la pertinencia, claridad y coherencia de las preguntas con los objetivos y el marco teórico. Para que los datos recopilados sean confiables se abordó por medio de un rigor cualitativo con la grabación y transcripción textual de las entrevistas lo que fortalece la credibilidad de los hallazgos.

Para garantizar la pertinencia y rigurosidad metodológica, la técnica utilizada fue validada por la Dra. Irene Becerril Arostegui y el Dr. Luis Llerena Ocaña, los dos expertos en la investigación, quienes luego de una revisión del diseño de la guía de entrevista, evaluaron la coherencia con los objetivos planteados en el estudio, la claridad de las preguntas y la capacidad de estas para generar información relevante, la validación asegura la calidad del proceso de recolección de datos y garantiza los resultados obtenidos (Apéndice).

La combinación de las entrevistas y el uso de la herramienta SonarQube permitió realizar una triangulación de métodos, indicadores y datos claves en estudios con enfoques mixtos. Con la técnica cualitativa se logró una amplia comprensión del fenómeno y de las decisiones humanas, mientras que con la técnica cuantitativa se obtuvieron mediciones objetivas y verificable del estado de la deuda técnica en el software, con lo que se logró coherencia en el enfoque metodológico aplicado logrando validez interna, externa y confiabilidad en la investigación respaldando la solidez de los datos obtenidos y la pertinencia del modelo propuesto.

Análisis de datos

Al ser una investigación con enfoque mixto los datos obtenidos en las diferentes fases del estudio, se los preparó en un formato accesible, identificando los temas claves de la técnica aplicada para la recolección de datos, lo que permitió generar teorías a partir de estos, identificando códigos, patrones y relaciones entre los datos. Para validarlos se verificó la credibilidad y confiabilidad utilizando herramientas como la auditoría y comprobación de miembros.

Resultados esperados

- Se identificaron patrones y temas recurrentes desde las perspectivas de los participantes para así comprender las experiencias desde su punto de vista.
- Se identificaron nuevas opiniones y conceptos que contribuyeron a comprender fenómenos relacionados a la deuda técnica.
- Se desarrolló un marco referencial apropiado, que permita la obtención de los conocimientos adecuados para conocer el estado actual del problema planteado, los resultados obtenidos y los trabajos futuros.
- Se trianguló los datos obtenidos con hallazgos de otras investigaciones obteniendo una mejor perspectiva de la investigación.

- Con el análisis, clasificación y síntesis de la información obtenida se desarrolló un marco conceptual para la gestión y reducción de la deuda técnica en modelos de desarrollo de software ágiles.
- Una vez desarrollado el modelo se aplicó un caso de estudio que permitió validarlo.

Implementación propuesta

Los objetivos propuestos y las preguntas específicas sirvieron como guía para la toma de decisiones durante la investigación y el desarrollo del marco propuesto.

Para el desarrollo del modelo se esbozó como punto inicial realizar un estudio de las nuevas metodologías de desarrollo ágil y la afectación de la deuda técnica en estos, adaptándolo a las necesidades de la investigación.

De igual manera, para el modelo de gestión de la deuda técnica se plantearon actividades que permitieron la identificación y seguimiento, la estabilización y la reducción de esta.

También, se analizaron las técnicas y herramientas para el tratamiento de la deuda técnica en ambientes de desarrollo ágiles. Finalmente se aplica un caso de estudio que permita evaluar la eficacia del modelo desarrollado en ambientes de desarrollo controlados.

De la aplicación del caso de estudio se obtuvieron datos que son documentados, clasificados, analizados transparentemente y auditados para que de esta manera sirvan de base para la toma de decisiones y la interpretación de la relación existente entre las variables y la validación de la propuesta de investigación.

Caso de Estudio

La investigación se realizó utilizando un enfoque mixto, en el cual el método cualitativo se encuentra relacionado con las características del caso estudiado y el propósito del estudio. El estudio de caso realizado se sustenta en las directrices de (Stol y Fitzgerald ,

2020) para lo cual se plantea el objetivo, caso de estudio, preguntas de investigación y método de acuerdo con lo planteado por (Robson, 2002)

El caso de estudio es una investigación empírica que analiza el fenómeno en el lugar donde se desarrollan los hechos (Yin, 2018), permitiendo descubrir lo que sucede en un contexto real para explorar nuevas ideas (Petersen, 2020).

De acuerdo con (Yin, 2018), el diseño de un caso de estudio debe tener los siguientes componentes: “las preguntas del estudio, la propuesta de estudio, la unidad de análisis, el enlace lógico de los datos al estudio propuesto y los criterios para interpretar los campos”.

El problema que resolvió

El desarrollo de Software aborda distintos tipos de modelos que se caracterizan por diferentes fases y actividades que se deben realizar para obtener un producto de calidad. Existe una gran variedad de modelos siendo las metodologías ágiles las más utilizadas por las bondades de sus principios que se caracterizan por la entrega de productos funcionales en un tiempo corto comprendido entre dos a cuatro semanas.

Esta premura por la entrega de un producto funcional deriva en que la mayoría de las ocasiones se deje de un lado la calidad del producto, sacrificando buenas prácticas de desarrollo, generación de documentación, controles básicos y pruebas que son necesarias en la construcción de aplicaciones.

Como resultado del mal diseño de software por cumplir con la agilidad surge la deuda técnica ya sea por tareas pendientes o errores técnicos en el desarrollo. Esta deuda debe ser pagada lo más pronto posible con la finalidad de que no se acreciente ni genere intereses que con el pasar del tiempo van a ser difíciles de pagar, generando costos excesivos y problemas de mantenibilidad en el software.

La deuda técnica abarca el costo que se va acumulando por no solucionar a tiempo los problemas voluntarios o involuntarios que se dan en el desarrollo de software especialmente al no realizar adecuadamente el mantenimiento o la refactorización en el proyecto.

El no gestionar la deuda técnica puede llevar a una serie de inconvenientes y desafíos que afectan la calidad del producto ya que generan problemas asociados como: arquitectura no escalable, código desorganizado y difícil de mantener, software con seguridad vulnerable, complicaciones en la eficiencia y funcionalidad, documentación deficiente, dificultad para aplicar pruebas, entre otras.

En consecuencia, se plantea la investigación para resolver el problema que produce la deuda técnica al utilizar modelos ágiles en el desarrollo de software mediante un marco para monitorearla y gestionarla con un conjunto de tareas y actividades encaminadas a garantizar la eficiencia y control efectivo de la deuda técnica asegurando el éxito del proyecto, así como la mantenibilidad y sostenibilidad del producto a mediano y largo plazo. Estas tareas garantizan las entregas planificadas en el contexto de la agilidad priorizando: la revisión continua del código, una documentación eficiente, pruebas automatizadas, mejora continua y la incorporación del marco para la gestión de la deuda técnica como parte del ciclo de vida de desarrollo.

Descripción general de la solución propuesta

La investigación propone la realización de un modelo innovador con la finalidad de gestionar la deuda técnica eficazmente en proyectos software que se desarrollan en contextos ágiles. El desarrollo del marco surge con base a los desafíos identificados en la investigación cualitativa que examina las experiencias y prácticas de los profesionales desarrolladores y la verificación del marco en un ambiente controlado.

El marco propuesto plantea un conjunto de actividades coordinadas a aplicar en la gestión de la deuda técnica compuesta inicialmente por la definición de una taxonomía que facilite la identificación y clasificación del contexto de la deuda técnica.

A continuación, se diseña la medición de la deuda mediante métricas específicas cuantitativas que permitan detectar la deuda técnica enfocándose en aspectos como la codificación, documentación, pruebas y arquitectura.

Los indicadores de las métricas serán monitoreados continuamente apoyándose en herramientas de análisis estático, inspección de artefactos y técnicas automatizadas para detectar errores en la codificación, que permitan una evaluación objetiva de los indicadores establecidos. Los resultados son analizados con la finalidad de priorizarlos y tratarlos para lo cual se integra un conjunto de buenas prácticas y estrategias que permitan controlar la deuda y pagar sistemáticamente los hallazgos procesados e identificados como deuda técnica.

El proceso se respalda en un conjunto de recomendaciones alineadas con los principios de la agilidad cumpliendo la integración y entregas que demandan estos modelos, acompañados de mecanismos de retroalimentación que permitan una mejora continua del marco. El marco propuesto se valida en un ambiente de desarrollo de software real con la participación de un equipo de profesionales que desarrollan aplicaciones en un contexto académico y administrativo de una institución de educación superior. El trabajar con un equipo que desarrollan para diferentes departamentos asegura que el marco sea adaptable y eficiente para todo tipo de proyectos y diferentes entornos.

La solución desarrollada brinda técnicas fundamentadas empíricamente que actúan de manera armonizada para abordar la deuda técnica con un enfoque metódico y medible que contribuye tanto a la academia como a la industria. Para esto, se la aplicará con un equipo de desarrollo de software que se desempeñan en un ambiente de trabajo dinámico y colaborativo donde los profesionales modelan, diseñan y construyen aplicaciones para una institución de

educación superior, para lo cual utilizan diferentes tipos de herramientas y framenworks innovadores priorizando el uso de buenas prácticas. La retroalimentación en el equipo es constante mediante una comunicación fluida y reuniones constantes para monitorear las dificultades y el avance de los proyectos asignados a los miembros del equipo de desarrollo.

Las aplicaciones realizadas giran en torno a las actividades educativas y del negocio de la educación superior particular, las mismas que se encuentran destinadas a solucionar necesidades administrativas y académicas específicas para la institución. El departamento de desarrollo de la universidad viene trabajando por cinco años en la creación de aplicaciones y sistemas que atienden diversas necesidades académicas y administrativas. Estas aplicaciones son originales y específicas para la institución las que se describen a continuación.

- En el área académica: matriculas en línea, consulta de calificaciones, distributivos docentes, planificación de horarios, evaluaciones docentes.
- En el área administrativa: gestión de recursos humanos, aplicaciones financieras y de compras.
- En el área de vinculación e investigación: aplicaciones para gestión de proyectos de vinculación e investigación.

El departamento de desarrollo de la Universidad recibe continuamente requerimientos de varios departamentos con necesidades de procesos y proyectos por ser automatizados creando soluciones para la comunidad universitaria, a la vez que se encarga de dar mantenimiento a las aplicaciones que se encuentran en funcionamiento.

Descripción detallada de cada fase, etapa, módulo.

Fase 1: Fundamentación Teórica

Etapa 1: Revisión de la Literatura

Se realiza una exploración de la literatura relacionada con el tema planteado y la identificación de enfoques, teorías y modelos que aporten a la gestión de la deuda técnica.

Etapa 2: Marco Teórico

Se diseña el marco teórico y estudio del arte que sirven para estructurar la investigación y para el desarrollo del marco propuesto lo que proporciona una base consistente al estudio.

Fase 2. Descubrimiento y Definición

Etapa 3: Diseño de la Investigación

Con el diseño de la metodología cualitativa, se selecciona los participantes, las técnicas de recolección de datos y los protocolos de entrevista. También se definen los criterios de inclusión y exclusión de los participantes.

Etapa 4: Introducción Contextual

En esta etapa se realiza un estudio etnográfico, investigación de campo, entrevistas. Se construye una taxonomía con base a los hallazgos, esta permite definir y clasificar los temas relacionados a la deuda técnica.

Etapa 5: Recopilación de Datos

Se realiza la recolección de datos para lo cual se utiliza la entrevista y análisis de documentos. Las entrevistas se aplican a todos los miembros del departamento de desarrollo de software de la universidad para indagar sobre sus experiencias y prácticas con relación a la deuda técnica.

Etapa 6: Análisis de Datos

En esta etapa, se realiza el tratamiento de los datos obtenidos, registrándolos y validándolos. Se analizan los datos utilizando técnicas de análisis de contenido identificando patrones y relaciones en los datos.

II. Fase de Visualización y Reporte

Etapa 7: Procesamiento de Hallazgos

Los datos recolectados son procesados, correlacionados y priorizados mediante diferentes técnicas de análisis.

Vistas Personalizadas: Se diseñan e implementan vistas personalizadas con información comprensible de acuerdo con los roles del equipo de desarrollo

III. Fase de Desarrollo del Marco

Etapa 8: Diseño del Marco

Se diseña el marco para la gestión de la deuda técnica con base a los hallazgos de la investigación cualitativa y el marco teórico desarrollado en las etapas anteriores. Se definen las estrategias, principios y técnicas para identificar, medir, priorizar y abordar la deuda técnica.

Etapa 9: Validación del Marco

Se valida la solución propuesta mediante un estudio de caso y encuestas a profesionales de desarrollo de software. También, se evalúa la utilidad y aplicabilidad del marco en diferentes contextos para realizar ajustes en el caso que sea necesario.

IV. Fase de Institucionalización

Etapa 10: Integración en Procesos

El marco se acopla de una manera planificada y gradualmente en los flujos de trabajo de desarrollo, pruebas, implementación y operaciones. Se proporciona capacitación y apoyo al equipo de desarrollo para la adopción y utilización del marco.

Gobernanza: Se definen roles, políticas y prácticas con la finalidad de promover una cultura de concientización y acciones proactivas en el equipo de desarrollo frente a la deuda técnica.

Etapa 11: Evaluación del Impacto

En esta etapa, se evalúa el impacto del uso del marco en la práctica real. Se comparan métricas de calidad, productividad y satisfacción del equipo antes y después de la implementación del marco de gestión de la deuda técnica.

Etapa 12: Documentación y Diseminación de Resultados

Se documentan los resultados de la investigación y el proceso de desarrollo del marco en un informe final de la tesis doctoral. Se presentan los hallazgos con la comunidad académica y profesional.

Con base a la descripción detallada, en la Fase III como parte del desarrollo del marco, se plantea el marco propuesto para la presente tesis.

Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Las actividades de ingeniería de software basada en el Valor se enfocan en aquellas funcionalidades donde el software aporta más valor a la empresa, para la gestión de la Deuda Técnica en modelos de desarrollo ágiles se propone el marco Deuda Técnica Basado en el Valor (DTV), este enfoque no ha sido explorado en investigaciones sobre la deuda técnica, no obstante se planteó inicialmente por (Stochel et al., 2012), donde propone visibilizar la deuda por medio de modelos para evaluar la calidad del producto con base al diseño, código arquitectura y productividad en la ingeniería de software, sin embargo, no la explora a profundidad ni plantea un marco para la gestión de la deuda técnica.

El marco propuesto se enfoca en la gestión de la deuda técnica en función del impacto en el desarrollo de software ágil basado en el valor. En este contexto se proporciona mayor prioridad al desarrollo de los componentes de software que se consideran que entrega un mayor valor a la empresa, para determinar esta característica se analiza el contexto en que se desenvuelve el negocio y el software en desarrollo de acuerdo con los principios ágiles y las propuestas de valor alineadas a los objetivos estratégicos de la organización (Rodríguez et al., 2022).

En este sentido, se plantea un método para la resolución de la deuda técnica tanto desde un enfoque técnico, así como del valor que cada componente aporta al software. Con esta propuesta se gestiona la deuda técnica tomando en cuenta el aspecto técnico de los marcos ágiles en el desarrollo y el valor estratégico que aporta al software cada componente. Los principios del desarrollo basados en el valor consideran la satisfacción de los stakeholders, con métricas orientadas al valor enfocadas al rendimiento, la calidad del producto y la eficiencia de las pruebas con lo que se determina en qué fases de desarrollo se debe invertir mayor esfuerzo para que el software aporte más valor (Haindl y Plösch, 2021).

Descripción del marco

El enfoque DTV es una metodología para gestionar la deuda técnica en proyectos de desarrollo de software ágiles. Este marco permite tomar decisiones informadas para abordar la deuda técnica con base al impacto en el valor de los componentes de software desarrollados en las organizaciones. Las fases del modelo se sustentan en el enfoque basado en el valor y en los estudios presentados en los últimos cuatro años por diferentes autores, además, del aporte del autor de la tesis que gracias a la investigación realizada en el área de la deuda técnica ha obtenido la experticia necesaria, la misma que se respalda en las distintas publicaciones relacionadas al tema de la deuda técnica realizadas.

Estructura y componentes del modelo: El modelo de gestión basado en el valor tiene una estructura por capas que se relacionan entre si para abordar la deuda técnica sistemáticamente cumpliendo los principios ágiles y el enfoque basado en el valor. Los componentes se centran en identificar y registrar la deuda técnica, evaluar el valor técnico de los módulos, priorización de acuerdo con los resultados obtenidos, remediar y monitorear la deuda técnica.

Procesos y mecanismos de aplicación: El marco abraca procesos como la identificación, evaluación, priorización, control y mitigación de la deuda técnica. Estos procesos se aplican

de manera continua, junto con los mecanismos que incluyen métricas, toma de decisiones y herramientas que garanticen la remediación de la deuda técnica.

Integración de estructura, componentes y procesos: La integración va a permitir que la gestión de la deuda técnica sea parte del proceso de desarrollo de software convirtiéndose en parte inherente del flujo de trabajo ágil. El registro y monitoreo permiten evaluar y priorizar la deuda técnica asegurando la articulación entre la gestión de la deuda técnica y el desarrollo de software.

Mecanismos de aplicación y sostenibilidad del modelo: La aplicación del modelo se apoya en las prácticas que las organizaciones deben adoptar operativamente, como es el caso de asignación de políticas, buenas prácticas, seguimiento, retroalimentación y ajuste continuo con lo que el marco va a ser medible, adaptable y sostenible en el tiempo.

El enfoque se encuentra estructurada por las siguientes fases:

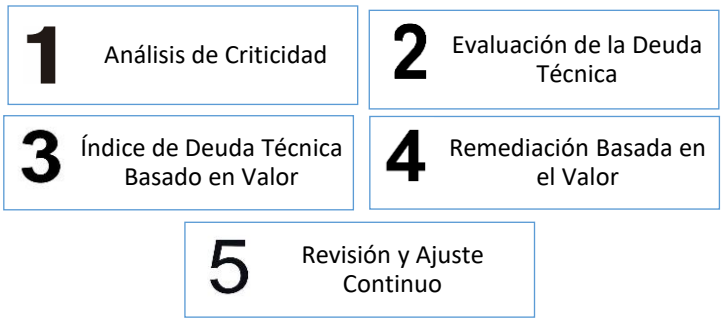
- Análisis de Criticidad: Como fase inicial se debe identificar los componentes o módulos que se cataloguen como aquellos que generen más valor para el software dentro de la agilidad y el negocio.
- Evaluación de la Deuda Técnica: Los componentes o módulos críticos son evaluados con métricas para determinar la cantidad de deuda técnica que poseen.
- Índice de Deuda Técnica Basado en Valor: Se calcula el índice de deuda técnica basado en valor con base a la criticidad del componente o módulo para el software y/o negocio y la deuda técnica que se encuentra asociada
- Remediación Basada en el Valor: Con base al índice de deuda técnica basado en valor, se optimizan los recursos con la priorización de los módulos críticos, con la finalidad de reducir la deuda técnica de acuerdo con el valor que se recuperará en el software o negocio.

- Revisión y Ajuste Continuo: Con la evolución del negocio y las prioridades, se deben realizar evaluaciones de los componentes críticos y aplicar estrategias para controlar y remediar la deuda técnica con la finalidad de asegurar una dinámica de la deuda técnica.

  En la figura 1 se describe el marco:

Figura 1.

Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)



Elaborado por: Lozada Torres Edwin Fabricio

Características del marco DTV

- Al ser un marco basado en el valor se prioriza la deuda técnica en función del impacto en el software y/o negocio, asegurando la optimización del uso de los recursos.
- Este marco es flexible y adaptable a cualquier organización o institución debido al enfoque que se ajusta a las propiedades de valor de la organización.
- El marco involucra al equipo de desarrollo y a los stakeholders de la organización fomentando la comunicación y una visión similar de la deuda técnica.
- El marco es simple de implementar ya que no demanda de inversiones extra adaptándose a cualquier equipo de desarrollo en diferentes contextos y con cualquier modelo de agilidad.

Fase 1. Análisis de Criticidad

En la Fase 1 se clasifican los componentes o módulos de acuerdo con el valor para la organización, debido a que no todos los componentes son relevantes, se los debe identificar y abordar la deuda técnica en función del área crítica en la cual se desempeña.

Identificación del Valor del Negocio

a) Definición del Valor

La definición del Valor del Negocio depende del contexto en que realiza las actividades la organización, así como de la naturaleza del software y las metas del negocio. Se definen los Indicadores Clave de Valor (KVI) alineando los objetivos técnicos con los del negocio, estas métricas identifican los componentes críticos del software que aportan en mayor medida a la organización.

Para determinar el valor de los componentes en la empresa se pueden tomar en cuenta las siguientes características:

Ingresos a la Organización: Los módulos o componentes que se encuentran relacionados con la generación o gestión de ingresos para la organización, son catalogados como de alta criticidad, en esta categoría se clasifican los componentes relacionados con la gestión financiera, pagos y las interfases con los clientes.

Satisfacción de usuarios: Módulos o componentes que dan servicio directamente al usuario o clientes, donde son fundamentales el rendimiento y la interfaz de usuario, se consideran como críticos. Para esta actividad se pueden utilizar métricas como Net Promoter Score (NPS) o encuestas de satisfacción.

Cumplimiento de normas: Los módulos o componentes relacionados al cumplimiento normativo son críticos ya que pueden ser objeto de algún tipo de sanción en caso de incumplimiento.

Diferenciadores del producto: Componentes del software con características especiales que lo diferencian de similares de los competidores por entregar valor añadido a los usuarios.

Para la definición del valor se pueden utilizar como técnicas matrices de decisión multicriterio (Taherdoost y Madanchian, 2023) o análisis de impacto del software.

Evaluación Técnica Concurrente: Se combinan los KVI y el índice de deuda técnica para los cual se pueden utilizar herramientas como SonarQube o Code Climate que sirven para cuantificar la deuda técnica con base al análisis del código, en términos de complejidad, duplicación de código, y otras métricas (Tahaei et al., 2021). Con estos datos se determinan los componentes prioritarios.

Asignación de Peso del Valor de Software: Con el cálculo de los KVI y la deuda técnica se asignan pesos relativos a estos indicadores, para lo cual se puede utilizar las matrices de decisión multicriterio, que permiten la ponderación de los componentes del sistema en relación con que tan crítico es para el negocio.

b) Mapeo de Componentes del Sistema

El siguiente paso en esta fase es el mapeo de los componentes o módulos, para lo cual se deben realizar las siguientes actividades:

Análisis del sistema: Se debe estructurar jerárquicamente al sistema donde se encuentren identificados los componentes principales y secundarios.

Mapeo de dependencias: Analizar e identificar las interdependencias entre módulos, lo que permitirá determinar el impacto o afectación que tendrán los componentes afectados por la deuda técnica de aquellos módulos que tienen algún tipo de dependencia.

c) Asignación de Valor a cada Componente

Se asigna el valor a los componentes del Software, con la utilización de los criterios previamente definidos. Para la valoración se puede utilizar métricas que permitan realizar la priorización de los componentes, las que se describen a continuación:

Evaluación cuantitativa: Identificar el impacto en función de términos económicos o de rendimiento que se pueden medir y cuantificar. Se pueden utilizar métricas para ingresos generados por el uso del módulo, costos evitados por inactividad, costos de soporte, tiempo ahorrado, entre otros.

Evaluación cualitativa: Identificar la percepción de los interesados en función de su experiencia con el sistema, lo que permitirá clasificar componentes con criterios como: crítico, importante y bajo impacto. Como métricas se pueden utilizar encuestas a los usuarios y entrevistas a los stakeholders para determinar la experiencia del usuario, eficiencia del software, etc. (Awad Elneel et al., 2022).

Análisis histórico: El análisis de los datos históricos para determinar incidentes relacionados a errores y rendimiento en los componentes.

d) Priorización de Componentes Críticos

Con la evaluación de los componentes se categoriza los módulos de acuerdo con la prioridad de impacto en el software. Los módulos de mayor impacto y alta deuda técnica serán los inicialmente ingresados en las estrategias de remediación.

Para la priorización se puede utilizar la matriz de impacto que es una herramienta para medir y visualizar la contribución al valor del software en relación con la deuda técnica de cada componente identificado como crítico para priorizarlos en la remediación.

Herramientas de Soporte: El equipo de desarrollo elegirá las herramientas tomando en cuenta los Framework, lenguajes de programación, Bases de Datos, etc., que utilizan en el desarrollo.

Análisis de arquitectura: Se utilizan herramientas automatizadas de análisis estático como es el caso de SonarQube o NDepend que permiten determinar los componentes con problemas en la arquitectura.

Seguimiento y Documentación: Herramientas de gestión como Trello, Jira o Confluence aportan al seguimiento de los módulos críticos.

Matrices de Impacto: Matrices de decisión para clasificar los componentes en función del valor y la deuda técnica.

Fase 2. Evaluación de la Deuda Técnica

A medida que la deuda técnica aumenta, se vuelve más difícil y costoso realizar cambios en el software. Esto incluye agregar nuevas funcionalidades, mejorar las existentes y adaptarse a nuevas necesidades del mercado.

Valor de Negocio: La agilidad del desarrollo disminuye, lo que significa que la empresa no puede responder rápidamente a las demandas del mercado. Esto puede resultar en una pérdida de oportunidades de negocio y una disminución de la competitividad.

Con los componentes críticos identificados se evalúa la Deuda Técnica. La medición de la Deuda Técnica permite la toma de decisiones con base a la afectación a futuro del desarrollo y el valor de la organización, los datos de estas métricas son insumos con los cuales se establecen prioridades para intervenir en los módulos críticos del sistema, para lo cual se utilizan métricas cuantitativas y cualitativas.

Evaluación Cuantitativa

Para la evaluación cuantitativa de la Deuda Técnica se utilizan métricas relacionadas al código fuente y la mantenibilidad, con lo cual se estima la deuda acumulada del sistema, estas métricas se explican a continuación:

- Complejidad ciclomática: Este enfoque sirve para medir la cantidad de posibles rutas en el código fuente, con esta métrica se determina la dificultad para mantener o

modificar los componentes. De acuerdo con los valores obtenidos, los valores de la complejidad ciclomática están directamente relacionados con los niveles de deuda técnica (Ebert et al., 2016).

- Duplicación de código: Se utilizan herramientas para determinar el código repetido que a la larga se deriva en riesgos de defectos y aumentan los esfuerzos y costos en el mantenimiento (Abdullah Alomar et al., 2023).
- Número de líneas de código: Se aplican técnicas para medir las líneas de código, debido a que mientras más extenso es un código es más propenso a generar deuda técnica.
- Cobertura de pruebas: Cuando las pruebas son mínimas o tienen un bajo nivel de cobertura, se aumenta el riesgo de defectos en el sistema.

La deuda técnica se cuantifica con base a los costos estimados de corrección o en el esfuerzo estimado para para remediarla. Los datos obtenidos permiten al equipo de desarrollo analizar los problemas técnicos y presentarlos en términos comprensibles a los stakeholders para la toma conjunta de decisiones, además del cálculo monetario de la deuda técnica en cada componente.

Evaluación Cualitativa

La evaluación cualitativa se enfoca en el impacto en el valor del Software, estas métricas evalúan la experiencia y percepción del equipo de desarrollo para determinar los módulos que a largo plazo pueden representar riesgo de presencia de deuda técnica, aunque no presenten problemas en las métricas de código. Los factores que se evalúan en esta evaluación se listan a continuación:

Desfase con los objetivos del Software: Un módulo que no se encuentre alineado con los objetivos del software y las metas de la organización puede causar deuda técnica.

También, en este factor se evalúan aquellos componentes que presenten tecnología obsoleta (Alves et al., 2020).

Complejidad para añadir nuevas funcionalidades: Si un módulo presenta dificultad para añadir nuevos componentes generan trabajo extra y aumento de costos tanto en el desarrollo como en el mantenimiento.

La evaluación cualitativa identifica factores de riesgo que no son detectados en las métricas cuantitativas, pero que impactan en el sistema al generar valor. El uso de las métricas cuantitativas y cualitativas permite identificar de manera más precisa a la deuda técnica.

Deuda Técnica Intencional y No Intencional

Determinar si la deuda técnica es intencional o no intencional permite establecer el tipo de estrategias a aplicar para la gestión de la deuda, sin embargo, la deuda técnica intencional se puede gestionar de manera más fácil, debido al conocimiento que tiene el equipo de su existencia, por lo que la remediación puede ser planificada. Por otro lado, la deuda técnica no intencional requiere un mayor esfuerzo debido a que inicialmente debe ser identificada de acuerdo con el tipo y de acuerdo con ello se debe determinar las estrategias para remediarla.

Costo de Oportunidad y Riesgo Asociado

Además de los costos directos para su gestión, la deuda técnica genera el llamado costo de oportunidad, este surge al no implementar mejoras o nuevas funcionalidades en el sistema, debido a que los desarrolladores deben trabajar en la remediación de la deuda técnica. Otro costo indirecto para tomar en cuenta es el que se genera por el riesgo de mantener el sistema técnicamente endeudado, este costo es producto de la acumulación de la deuda técnica por los riesgos en el sistema que se pueden dar por factores como: defectos,

rendimiento y vulnerabilidades de seguridad, que a la larga afectan en el valor del software (Shamuratov y Hamadan, 2022).

Herramientas para Evaluar la Deuda Técnica

Se detallan las herramientas y procesos cuantitativos y cualitativos para evaluar la deuda técnica:

- SonarQube: Herramienta automatizada que genera informes del estado del código con relación a deuda técnica, además de la estimación del esfuerzo para remediarla (Ernst et al., 2020).
- CodeScene: Esta herramienta presenta un análisis de las áreas del código donde se ha incrementado la deuda técnica proporcionando métricas de complejidad y frecuencia en los cambios.

Fase 3. Índice de Deuda Técnica Basado en Valor

El Índice de Deuda Técnica Basado en Valor (IDTV) es una métrica compuesta con la cual se puede priorizar los esfuerzos de remediación, esta evaluación toma en cuenta la deuda técnica y el impacto en el valor del software. El índice proporciona los datos para determinar aquellos componentes críticos que requieren remediación de manera inmediata, se calcula con la siguiente fórmula:

$$IDTV = \text{Deuda Técnica} \times \text{Peso Relativo del Valor del Software}$$

La *Deuda Técnica* se calcula a partir de las métricas para deuda técnica descritas en la Fase 2.

El *peso relativo del valor del Software* se basa en los KVI descritos en la Fase 1.

Se deben considerar los siguientes factores para calcular el peso relativo del valor del Software:

Mediante el análisis con los stakeholders, los análisis cuantitativos y las revisiones de los KVI, se estima el peso relativo del valor de software por cada módulo considerado como crítico.

Ventajas del Cálculo del IDTV

Las ventajas como métrica para priorizar los componentes se listan a continuación:

Equilibrio entre factores técnicos y de negocio: El IDTV toma en cuenta el valor del software en el negocio con la organización y puede tomar decisiones que se encuentran a la par de sus objetivos. Esta métrica difiere de las evaluaciones tradicionales ya que, en la mayoría de los casos se centran en medir inconsistencias en el código.

Prioridades basadas en datos: El IDTV evita la subjetividad al priorizar la remediación de la deuda técnica, debido a que no solamente toma en cuenta problemas técnicos, sino que también se enfoca en aquellos factores que generan impacto al negocio.

Mejor comunicación con stakeholders: La comunicación con los interesados es efectiva debido a la utilización de términos enfocados en la organización. Al utilizar un lenguaje común se aporta en conjunto con la asignación de recursos para remediar la deuda técnica generando un impacto positivo en la organización, además, de controlar y reducir el riesgo asociado a la deuda técnica.

Fase 4. Remediación Basada en el Valor.

La remediación basada en el valor es un enfoque estratégico que permite planificar el cómo gestionar la deuda técnica de acuerdo con los objetivos de la organización y las actividades de remediación priorizada. Para aplicar este enfoque se debe identificar, priorizar y remediar la deuda técnica basada en el valor, con lo cual se logra gestionar los recursos de manera óptima logrando aportar a la calidad del software.

Identificación de Prioridades

Para planificar la remediación la actividad inicial es la identificación de las prioridades en la deuda técnica, como insumos para este paso se utiliza el Índice de Deuda Técnica Basado en Valor (IDTV), como se mencionó previamente para este cálculo se utilizan herramientas automatizadas y la aplicación de entrevistas y encuestas a las partes interesadas que ayudan a evaluar el impacto de cada componente en el negocio.

Objetivos de Remediación

Con la identificación de los componentes priorizados, el siguiente paso es determinar los objetivos para la remediación de la deuda técnica. Estos objetivos deben ser factibles en su cumplimiento, además, deben ser medibles, concretos, asequibles, relevantes y limitados en el tiempo (SMART) (Ogbeiwi, 2017). Así también, los objetivos deben cumplir con el enfoque, el cual establece que debe existir siempre la relación entre la deuda técnica y el valor del negocio, como característica estos objetivos deben señalar el porcentaje de reducción de la deuda en un tiempo especificado.

Asignación de Recursos

La remediación de la deuda técnica es una actividad estratégica que implica la asignación de recursos como presupuesto, personal, tiempo, herramientas y la técnica y formación necesaria para remediar la deuda de aquellos componentes previamente identificados como prioritarios ya que presentan un alto IDTV. Para realizar esta tarea debe estar comprometido el equipo de desarrollo de software y alineados con el objetivo común que es reducir la deuda técnica.

Técnicas de Remediación

En la industria existen técnicas y metodologías probadas que se utilizan para remediar la deuda técnica, las principales se describen a continuación:

- Refactorización: Esta técnica consiste en reescribir el código existente optimizándolo, pero manteniendo la funcionalidad del sistema. Para determinar las partes del código que requieren una restructuración se utilizan herramientas automatizadas y esta actividad debe ser implementada como una práctica diaria en el desarrollo.
- Reescritura de componentes: Cuando la deuda técnica acumulada es demasiada alta es muy difícil corregirla, por lo que una de las soluciones efectivas es reescribir completamente el componente. Para aplicar esta técnica es necesario realizar un análisis costo-beneficio, además, se debe analizar el impacto en la organización que tendrá realizar esta actividad.
- Desarrollo de pruebas: La aplicación de pruebas es una técnica que asegura la reducción de la deuda técnica a largo plazo, además de asegurar que cualquier acción de remediación realizada en el código no cree nuevos problemas.

Seguimiento y Evaluación

Para asegurar que las soluciones implementadas en la remediación de la deuda técnica sean efectivas, es necesario realizar un proceso de seguimiento y evaluación. Para esto se debe establecer una revisión y evaluación continua de la deuda técnica y el valor de la organización, estas actividades deben incluir una re-evaluación del IDTV, con lo cual se puede determinar si las prioridades en los componentes han cambiado. Para evaluar la efectividad de las técnicas aplicadas en la remediación se utilizan métricas para el rendimiento del sistema, donde se mide la calidad y el tiempo de entrega del producto (Ernst et al., 2020).

Esta fase es un enfoque integral para gestionar de manera efectiva la deuda técnica en las organizaciones, las mismas que, obtienen varios beneficios como: la alineación estratégica de las acciones de remediación en el valor de la organización, la optimización de recursos y la

mejora continua. Estas acciones logran la reducción optima de la deuda técnica alineada al valor de la organización, aportando a la evolución efectiva empresarial.

Fase 5. Revisión y Ajuste Continuo.

La revisión y ajuste continuo es una fase crítica en el marco propuesto, ya que permite desarrollar un software de calidad al mismo tiempo que aporta a que las actividades para gestionar la deuda técnica se encuentren de acuerdo con los objetivos de la organización. La revisión continua detecta el posible surgimiento de la deuda técnica determinando nuevos componentes o módulos donde el equipo de desarrollo debería preocuparse, a la vez que, de evalúan las medidas aplicadas en la remediación de la deuda.

Establecimiento de un Ciclo de Revisión

Para un ciclo de revisión efectivo, se deben realizar las siguientes actividades:

- Monitoreo continuo: Un monitoreo efectivo debe entregar visibilidad continua del estado de la deuda técnica, para esto el equipo de desarrollo debe utilizar herramientas que permitan monitorear y medir el estado del código como es el caso de los plugins que integran la mayoría de IDEs y las herramientas automatizadas de análisis estático de código.
- Evaluación de Impacto: La revisión regular no debe enfocarse en la deuda técnica acumulada, también, debe evaluar el impacto de la deuda en el valor del negocio, para esto se debe considerar el IDTV y analizar cómo los cambios empresariales inciden en las prioridades de remediación.
- Reuniones de Revisión: Se deben planificar reuniones continuas y programadas con los stakeholders para analizar y discutir sobre los hallazgos producto de las revisiones realizadas, estas reuniones además de servir para la comunicación del estado de las prioridades de la organización van a permitir tomar las decisiones adecuadas para realizar los ajustes según sea necesario.

- Documentación de Cambios: Los cambios realizados en las actividades de remediación deben ser debidamente documentados, esta información servirá de insumo para revisiones a futuro, al mismo tiempo, que se puede utilizar como soporte en el caso de divergencias en las decisiones adoptadas para la remediación de la deuda técnica.

Ajuste de Estrategias de Remediación

Con la revisión continua se puede identificar el incremento de la deuda técnica en determinados componentes, para lo cual se deben modificar las estrategias de remediación de la deuda:

- Ajuste en la priorización de Componentes: Como resultado del monitoreo surgen componentes que se vuelven críticos, aunque inicialmente no fueron catalogados como prioritarios. De acuerdo con los datos obtenidos en el seguimiento, puede ser necesario efectuar reajustes del IDTV, modificando la planificación de la remediación.
- Adaptación de Recursos: Para que el reajuste en la remediación sea efectivo, es necesario reasignar el tiempo, personal y presupuesto a aquellas áreas que han sido identificadas como críticas, priorizando los esfuerzos para gestionar de manera efectiva la deuda técnica.
- Iteración de Metodologías: El equipo de desarrollo junto con la organización, deben dar apertura a la interacción con otras metodologías de remediación. Con esta estrategia se pueden incluir prácticas y técnicas que, si bien no son parte del marco propuesto, aportan para una gestión óptima de la deuda técnica.

La implementación de un enfoque de revisión y ajuste continuo proporciona varios beneficios, como son: agilidad y adaptabilidad, así como una mejora continua y compromiso de los stakeholders. Esta fase en fundamental para la gestión de la deuda técnica ya que

proporciona los procesos para adaptarse a las realidades de la agilidad en el desarrollo de software sin alejarse de las necesidades y objetivos de la empresa.

Implementación de la Propuesta

La implementación y validación del Marco de Gestión de Deuda Técnica Basado en el Valor (DTV) se lo realizó con un caso de estudio en un entorno real, con el equipo del departamento de desarrollo de software de una institución de educación superior, donde se analizó el impacto y la eficiencia del marco.

Para la implementación se desarrolló un cronograma, donde se planificaron los tiempos y responsabilidades para el proceso.

También, se informó a los miembros del equipo de desarrollo sobre los beneficios del marco a implantar y la afectación a sus actividades diarias. Como parte de la comunicación también se familiarizó al equipo con los conceptos, técnicas y fases del marco.

Si bien, el marco se estructuró de acuerdo con el contexto en que se desenvuelve el equipo de desarrollo en la institución, sin embargo, se realizaron ajuste para que el equipo se adapte a las características del enfoque propuesto.

La implementación fue progresiva, tomando en consideración ciertos módulos para evaluar la efectividad del marco, para luego implementarlo en otros proyectos en forma gradual.

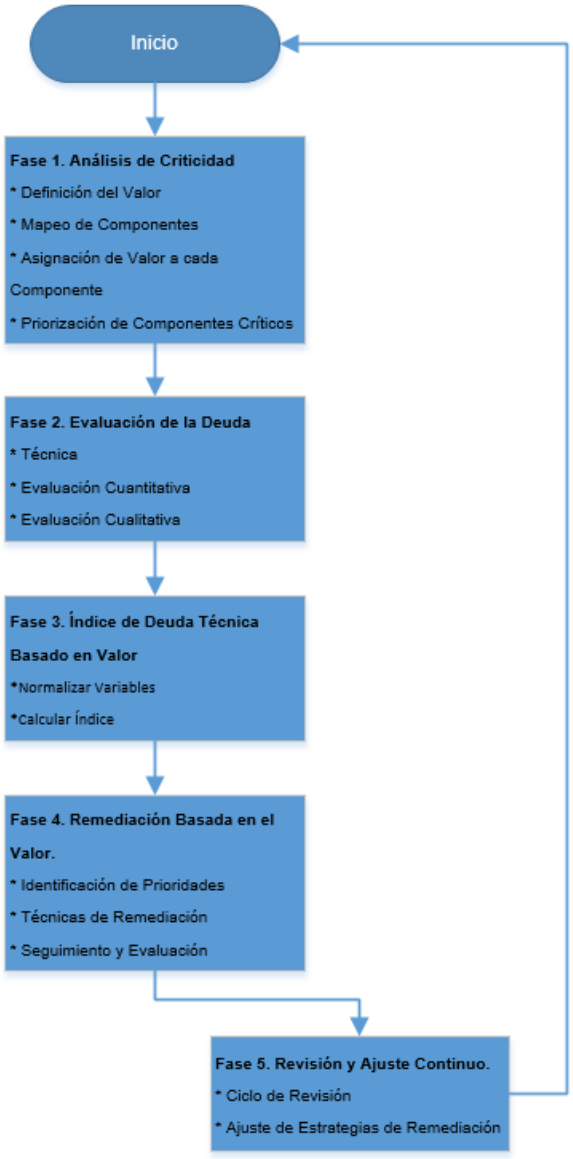
Como parte de la implementación se realizó una evaluación inicial del estado de la deuda técnica en los módulos seleccionados, para lo cual se realizó un análisis histórico de los problemas recurrentes que se presentan en el desarrollo, así como, del análisis del código con la utilización de las herramientas automatizadas de análisis estático descritos anteriormente.

El procedimiento para que los equipos ágiles gestionen la deuda técnica se lo detalla en el siguiente diagrama de flujo, en el cual se muestra el proceso completo que debe seguir

un equipo de desarrollo ágil para identificar, evaluar, priorizar y remediar la deuda técnica utilizando el enfoque propuesto. Además, se describe cómo se revisa y ajusta continuamente la estrategia para asegurar que la remediación aporte el máximo valor al equipo.

Figura 2.

Diagrama de flujo de procesos del Marco DTV



Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

A continuación, se detalla las fases para la implantación del Marco de Gestión de Deuda Técnica Basado en el Valor (DTV):

Fase 1: Definición y Planeación

Definir las bases para la implementación del marco.

Actividad 1.1: Análisis inicial del entorno.

- Tarea 1.1.1: Recopilar datos sobre los procesos actuales de gestión de la deuda técnica del equipo.
- Tarea 1.1.2: Identificar la deuda técnica acumulada.
- Tarea 1.1.3: Realizar entrevistas con stakeholders clave.

Actividad 1.2: Definición de objetivos y alcance.

- Tarea 1.2.1: Determinar metas específicas del marco a implementar.
- Tarea 1.2.2: Identificar métricas clave de éxito (KPIs).

Actividad 1.3: Planificación estratégica.

- Tarea 1.3.1: Crear un cronograma con hitos y entregables.
- Tarea 1.3.2: Asignar roles y responsabilidades para la implementación.
- Tarea 1.3.3: Determinar herramientas y recursos necesarios.

Fase 2: Adaptación del Marco

Actividad 2.1: Adaptación del marco.

- Tarea 2.1.1: Definir roles
- Tarea 2.1.2: Diseñar flujos de trabajo con base a las fases del marco.

Fase 3: Implementación Piloto

Implementar el Marco de Gestión de Deuda Técnica Basado en el Valor (DTV) en un equipo de desarrollo de software de una institución de educación superior, con un alcance controlado para evaluar su efectividad inicial.

Actividad 3.1: Implementación inicial.

- Tarea 3.1.1: Configurar herramientas específicas del marco.

- Tarea 3.1.2: Ejecutar reuniones de planeación y seguimiento con base a las actividades del marco.

Actividad 3.2: Monitoreo de la implementación.

- Tarea 3.2.1: Aplicar las métricas descritas en el marco.
- Tarea 3.2.2: Reuniones regulares para identificar problemas y evaluar la implementación.

Fase 4: Evaluación y Refinamiento

Analizar los resultados de la implementación y realizar ajustes al marco.

Actividad 4.1: Análisis Cuantitativo. Revisión de métricas y resultados.

- Tarea 4.1.1: Evaluar el nivel de adopción y satisfacción del equipo con el marco.

Actividad 4.2: Análisis cualitativo.

- Tarea 4.2.1: Recolectar feedback mediante encuestas y entrevistas.
- Tarea 4.2.2: Identificar prácticas efectivas y áreas que requieren mejora.

Fase 5: Escalamiento y Consolidación

Aplicar el marco en módulos o proyectos adicionales y consolidarlo como una práctica estándar en el equipo de trabajo.

Actividad 5.1: Escalamiento.

- Tarea 5.1.1: Introducir el marco en otros módulos o proyectos del equipo.

Actividad 5.2: Documentación

- Tarea 5.2.1: Redactar un informe final con los resultados del caso de estudio.
- Tarea 5.2.2: Preparar una guía para facilitar la adopción del marco en otros contextos.

Fase 6: Validación Académica y Profesional

Consolidar el marco como una contribución válida para la comunidad académica y profesional en el desarrollo del software.

Actividad 6.1: Difusión académica.

- Tarea 6.1.1: Presentar los hallazgos.

Actividad 6.2: Validación profesional.

- Tarea 6.2.1: Promover el uso del marco en el campo académico y en la industria del software.

3. Pruebas y Resultados

Diseño de estudio

El estudio se realizó con un enfoque mixto, en el cual se recolectaron datos para la validación del marco en el enfoque cuantitativo, mientras que se aplicaron entrevistas en el enfoque cualitativo. En el diseño de la investigación cualitativa, el investigador es parte activa del fenómeno que se estudia (Hudgikar, 2021), en esta tesis se aplicó un estudio de caso para evaluar el modelo propuesto, esta metodología utiliza un enfoque investigativo riguroso con la solución práctica en entornos reales.

Como resultado de los objetivos específicos uno y dos se planteó como población objeto de estudio un equipo de desarrollo de Software de una institución de educación superior de la ciudad de Ambato en Ecuador, que lo integran un jefe y cinco desarrolladores miembros del equipo de desarrollo.

A partir de la descripción general de la propuesta en la fase dos, descubrimiento y definición se procede con la etapa cinco y seis, recopilación y análisis de datos. Así mismo, se cumple con las tareas de la actividad análisis inicial del entorno, planteadas en la fase uno de la implementación de la propuesta.

Debido a que el software es un artefacto sociotécnico, se diseñaron entrevistas semiestructuradas (Seidman, 2019), para de esta manera obtener las experiencias del equipo en el desarrollo.

Los criterios para desarrollar la guía de entrevista se alinearon con las preguntas de investigación y los objetivos planteados, con lo cual se plantearon preguntas que resulten relevantes para la recolección de datos.

Las preguntas se organizaron de lo general a lo específico con cuestionamientos que van de lo menos a lo más sensible, donde se utilizaron cuestionamientos de tipo abierto, de

sondeo y de seguimiento utilizando un lenguaje natural y claro enfocándose en el área técnica de la deuda técnica.

Se trató en lo posible de evitar todo tipo de sesgos y preguntas directivas formulando cuestionamientos neutrales con lo que se logró no tener ningún tipo de influencias en las respuestas, logrando que el entrevistado agregue información en base a su reflexión sobre el problema y la entrevista.

Su utilizaron narrativas y ejemplos con lo que se aplicaron preguntas que el entrevistado pueda responder con base a sus experiencias permitiendo que la entrevista evolucione con base a las respuestas obtenidas del participante y respetando los limites personales.

La entrevista se dividió en preguntas relacionadas con la introducción, el conocimiento y percepción, las prácticas actuales, los factores contributivos, el impacto y consecuencias, las ideas y soluciones y cierre, en el orden descrito.

Debido a que las entrevistas se realizaron en el horario de trabajo del equipo y tomando en cuenta las agendas y tareas asignadas a cada miembro se consideró el tiempo y lugar adecuado para que los entrevistados puedan hablar sin ningún tipo de interrupción ni distracción.

Validación y análisis de los datos

Se utilizaron medios tecnológicos para grabar las entrevistas aplicadas a los miembros del equipo de desarrollo a continuación, se transcribieron las entrevistas de manera textual.

Entrevista aplicada al jefe del equipo

Nombre del entrevistado: Ing. Henry Baño

Cargo del entrevistado: Jefe del equipo de desarrollo (J)

Título: Ingeniero en Sistemas e Informática

Experiencia en el desarrollo: 15 años

P1. ¿Puedes describir tu rol y tus principales responsabilidades en el equipo?

Respuesta: Mi designación es de Director del Departamento de Desarrollo de la Universidad, mi rol como desarrollador es de jefe del equipo de desarrollo de software. Como jefe de equipo de desarrollo de software en la universidad, mi rol es fundamental para el éxito de los proyectos de software y el bienestar del equipo. Mis principales responsabilidades abarcan: liderar al equipo, asignar tareas, delegar responsabilidades y monitorear el progreso individual y del equipo, crear un ambiente de colaboración, comunicación y apoyo mutuo entre los miembros del equipo. También me encargo de la planificar de los proyectos, el cronograma, los recursos del proyecto, supervisar el desarrollo del proyecto, gestionar los riesgos y asegurar el cumplimiento del plan. Finalmente, yo me encargo de mantener a los usuarios informados sobre el progreso de los proyectos y las decisiones tomadas.

P2. ¿Cuánto tiempo llevas liderando este equipo?

Respuesta: Desde que asume la dirección del Departamento de Desarrollo, esto es hace 15 años atrás.

P3. ¿Cómo defines la deuda técnica en el contexto de tu equipo?

Respuesta: Defino la deuda técnica como un riesgo para la entrega pronta. Al principio, parece que avanzamos rápido, pero con el tiempo, las tareas simples se vuelven complejas. Los desarrolladores se frustran porque pasan más tiempo lidiando con problemas antiguos que construyendo cosas nuevas. Es un problema evidente entregar valor rápidamente sin comprometer la calidad a largo plazo

P4. ¿Consideras que la deuda técnica es un problema significativo en tus proyectos actuales? ¿Por qué?

Respuesta: Sí, y no solo técnicamente. La deuda técnica está desgastando a nuestro equipo. En nuestras reuniones, siempre surge el tema: 'No pudimos entregar la función X porque tuvimos que arreglar Y'. Los plazos se alargan, el equipo se estresa, y los diferentes

departamentos de la universidad piensa que somos ineficientes. Además, es difícil tener motivado al equipo cuando los desarrolladores se dan cuenta de que pasarán más tiempo apagando incendios que innovando.

P5. ¿Qué procesos y metodologías sigue el equipo para el desarrollo de software (e.g., Agile, Scrum, Kanban)?

Respuesta: En general, procuramos aplicar metodologías ágiles no una específica más bien una combinación de ellas, dividimos el trabajo en módulos cortos con objetivos definidos, realizamos reuniones diarias para evaluar el progreso y adaptar las prioridades. Sin embargo, los continuos problemas en el desarrollo nos obligan a adaptar estas metodologías, cambiando continuamente la prioridad a las tareas asignadas a los miembros ajustando las estimaciones de tiempo considerando el avance y tipo de tareas.

P6. ¿Existe alguna práctica establecida para identificar y gestionar la deuda técnica?

Respuesta: Cuando se estima incorrectamente, o cuando las supuestas tareas sencillas se convierten en complejas, eso pienso que es deuda técnica. También la noto en la motivación del equipo por las diferentes quejas de sobre trabajo. Para manejar este problema, se trata de crear espacios en los diferentes procesos. También facilito discusiones con los stakeholders sobre la priorización de las entregas y los riesgos de apresurarse

P7. ¿Cómo se toman las decisiones relacionadas con el manejo de la deuda técnica?

Respuesta: La toma de decisiones relacionadas con la deuda técnica es un proceso complejo en nuestro entorno. Por lo general dependemos de otros por las decisiones de los diferentes departamentos de la Universidad sobre las diferentes aplicaciones solicitadas, esto nos induce a no poder priorizar o implementar soluciones a la deuda técnica de forma independiente básicamente porque la asignación de tiempo y recursos para abordar la deuda técnica a menudo compite con otras prioridades, como el desarrollo de nuevas funcionalidades.

P8. ¿Qué factores internos (e.g., plazos de entrega, recursos limitados) crees que contribuyen a la deuda técnica en tu equipo?

Respuesta: Son varios factores ya que los plazos en el ámbito académicos son severos, por ejemplo, si el sistema de matrícula no está listo para el inicio del semestre, es un desastre. Pero los requisitos cambian porque cada departamento quiere su supuesto pequeño cambio o adición. También está el desconocimiento sobre los procesos de desarrollo y la falta de una verdadera participación de los usuarios finales. A menudo, construimos basándonos en lo que un decano o administrador cree que los estudiantes o profesores necesitan, no en investigación de usuarios real.

P9. ¿Y qué factores externos (ej., cambios en los requisitos del cliente, nuevas tecnologías)?

Respuesta: Los continuos cambios en los requisitos lo que genera retrabajo y sensación de que el proyecto no avanza. La presión por parte de los usuarios para la entrega pronta de los productos solicitados. También se tiene la necesidad de aprender y adaptarse a nuevas tecnologías rápidamente lo que aumenta el estrés y la carga de trabajo en el equipo.

P10. ¿Puedes compartir algunos ejemplos de cómo la deuda técnica ha afectado los proyectos en términos de tiempo, costos o calidad?

Respuesta: Para mí, el impacto más claro ha sido en el equipo mismo. Nos pasamos tratando de resolver los mismos problemas una y otra vez, esto afecta la moral del equipo limitándonos en el lanzamiento de nuevas funciones. Estos problemas no solo afectan al código, sino que también está afectando la pasión y creatividad del equipo.

P11. ¿Cómo afecta la deuda técnica a la moral y productividad del equipo?

Respuesta: Nos frustra tener que lidiar con código de baja calidad y soluciones temporales que retrasan el desarrollo. Nos desmotiva sentir que nuestro trabajo no es valorado por la complejidad del código heredado. El estrés aumenta por la presión de cumplir

plazos con problemas técnicos constantes. Todo esto afecta nuestra productividad. Dedicamos tiempo a corregir errores en lugar de innovar, y los cambios en el código son más lentos y complejos. Necesitamos asumir la deuda técnica como un problema serio, implementar una estrategia para gestionarla, buscar las herramientas para afrontarla, y así crear un ambiente de trabajo donde se valore la calidad del código.

P12. ¿Qué medidas crees que podrían implementarse para gestionar mejor la deuda técnica en tu equipo?

Respuesta: Adecuadas estimaciones tomando en cuenta la complejidad y el costo de los continuos cambios, esto pienso que haría que la deuda sea parte de cada decisión. También, creo que se necesitan aplicar mediciones y una guía para controlar adecuadamente todo este tipo de problemas.

P13. ¿Hay herramientas o prácticas que te gustaría ver adoptadas para reducir la deuda técnica?

Respuesta: Desde la perspectiva de gestión, necesitamos mejor visibilidad, en las reuniones identificar los problemas en los retrasos por la mala codificación, lo que haría que la deuda ya no se ignore. También, una integración entre nuestras herramientas de desarrollo y un software que permita gestionar proyectos. Además, un informe mensual que sume todo esto en términos de costo y riesgo.

P14. ¿Cómo evalúas el impacto de la deuda técnica en la calidad, mantenibilidad y escalabilidad del software desarrollado por el equipo?

Respuesta: La deuda técnica es un problema latente en nuestros procesos ya que la eficiencia no es solo velocidad, es la capacidad de entregar valor consistentemente. Con la deuda técnica, nuestras entregas se vuelven impredecibles. Las estimaciones no se cumplen. Eso destroza nuestra planificación y afecta a la calidad. Pensando bien con tanta deuda, nos preocupamos en como sobrevivir antes que en cómo mejorar. Estamos en modo de

supervivencia, no de calidad y a lo largo plazo, eso es más trabajo y mala calidad del producto.

P15. ¿Qué métricas o indicadores consideras relevantes para monitorear y medir la deuda técnica?

Respuesta: No utilizamos métricas ese es nuestro problema, esto se debe a que no tenemos un equipo numeroso y siempre nuestro enemigo es el tiempo. Nos preocupamos siempre por tiempo de entrega, tratando de cierta manera en medir cuánto tardamos en desarrollar. Algunas ocasiones hemos realizado encuestas de satisfacción, pero últimamente ya no las hemos aplicado.

P16. ¿Cómo priorizas el pago de la deuda técnica en relación con la entrega de nuevas funcionalidades?

Respuesta: Eso es un desafío constante, se evalúa la gravedad de los problemas, considerando su impacto en la calidad del software. Aquí se analiza los riesgos de que cause errores, bloqueos o retrasos en el desarrollo, todo esto siempre tomando en cuenta el impacto potencial de estos problemas en las operaciones de la universidad y la satisfacción de los usuarios.

P17. ¿Tienes algún otro comentario o reflexión sobre la deuda técnica que te gustaría compartir?

Respuesta: Definitivamente no tenemos una adecuada gestión para remediar este problema. La deuda técnica prospera en el silencio y aunque los desarrolladores ven el problema no lo expresan. También desarrollamos en un medio donde valoran las soluciones rápidas sobre la calidad.

Entrevista aplicada a los miembros del equipo de desarrollo

En la Tabla 9, se describe a los miembros del equipo:

Tabla 9.

Miembros del equipo de Desarrollo

Id	Nombre	Cargo	Título	Experiencia
M1	Camacho Byron	Analista desarrollador	Ing. en Sistemas e Informática	12 años
M2	Lozada Juan	Analista desarrollador	Ing. en Sistemas e Informática	8 años
M3	Núñez Guillermo	Analista desarrollador	Ing. en Sistemas e Informática	15 años
M4	Rodríguez Carlos	Analista desarrollador	Ing. en Sistemas e Informática	8 años
M5	Calero Roberto	Analista desarrollador	Ing. en Software	3 años

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Departamento de Desarrollo de la IES

Las preguntas planteadas en la entrevista son las que se indican en la Tabla 10, y las respuestas de la entrevista en la Tabla 11:

Tabla 10.

Preguntas al equipo de desarrollo de Software

	Introducción
P1	¿Puedes describir tu rol y tus principales responsabilidades en el equipo?
P2	¿Cuánto tiempo llevas trabajando en este equipo?
	Conocimiento y Percepción
P3	¿Qué entiendes por deuda técnica en el desarrollo de software?
P4	¿Consideras que la deuda técnica es un problema en los proyectos en los que trabajas? ¿Por qué?
	Prácticas Actuales
P5	¿Cómo identificas y manejas la deuda técnica en tu trabajo diario?
P6	¿Sientes que hay un proceso claro para abordar la deuda técnica?
	Factores Contributivos
P7	¿Qué factores en tu día a día crees que contribuyen a la deuda técnica (e.g., plazos de entrega, cambios de requisitos, falta de recursos)?
P8	¿Cómo manejas la presión de entregar rápido versus mantener la calidad del código?
	Impacto y Consecuencias
P9	¿Puedes dar ejemplos de situaciones en las que la deuda técnica ha impactado tu trabajo o el proyecto?

	Introducción
P10	¿Cómo sientes que la deuda técnica afecta la eficiencia y calidad del desarrollo?
	Ideas y Soluciones
P11	¿Qué prácticas crees que podrían implementarse para reducir la deuda técnica en tu equipo?
P12	¿Hay herramientas o procesos que crees que podrían ayudar a gestionar la deuda técnica mejor?
	Cierre
P13	¿Está dispuesto a dedicar tiempo y esfuerzo a abordar la deuda técnica?
P14	¿Tienes algún otro comentario o reflexión sobre la deuda técnica que te gustaría compartir?

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Guía de Entrevista

Tabla 11.

Entrevista aplicada a los miembros del equipo de desarrollo

Preg.	M1	M2	M3	M4	M5
P1	Soy pate del equipo del departamento de Desarrollo, realizo las tareas encomendadas a mi persona de acuerdo con la distribución de actividades que realiza el director del departamento.	Me desempeño como desarrollador. Mis responsabilidades son desarrollar, implementar y capacitar a los usuarios en las aplicaciones realizadas de acuerdo con la distribución realizada por nuestro jefe.	Actualmente me encargo de brindar soporte en los programas que utilizan en los diferentes departamentos de la universidad y dar mantenimiento a las aplicaciones de acuerdo con las necesidades solicitadas por los usuarios.	Soy desarrollador y me dedico a crear aplicaciones con base a la designación realizada por el jefe del equipo.	Me desempeño como desarrollador en el equipo. Mi principal responsabilidad es desarrollar software de acuerdo a módulos designados por el jefe del equipo
P2	Llevo 8 años trabajando en la institución	Llevo trabajando en el equipo alrededor de 5 años	Ya son 12 años que trabajo como parte del equipo	5 años trabajo en el departamento de desarrollo.	Soy el último en integrarme al equipo, hace un año empecé a trabajar en la Institución.
P3	De lo que tengo entendido la deuda técnica es como pedir un préstamo en el desarrollo. Para cumplir con los plazos, tomamos atajos o implementamos soluciones rápidas que sabemos que no son las mejores. Es como si	Honestamente, no estoy muy seguro. He oído hablar de deuda técnica, pero para mí suena como algo relacionado con las finanzas. Supongo que tiene que ver con cosas que hacemos mal o a medias porque estamos apurados. Como	Desde mi perspectiva en calidad, la deuda técnica es todo ese código espagueti y parches rápidos que hacen que las pruebas sean un dolor de cabeza. Es como programar en el aire,	Para mí la deuda técnica puede surgir de diversas fuentes: requisitos mal definidos, presión de tiempo, falta de pruebas, o desconocimiento.	La codificación no esencial que no se realizó para dar prioridad a la entrega del producto

Preg.	M1	M2	M3	M4	M5
	dijéramos que lo arreglaremos más tarde. Esto retrasa el trabajo y no permite realizar nuevas funcionalidades	cuando copio y pego código para ahorrarme tiempo.	cuanto más construimos, más inestable se vuelve todo		
P4	Definitivamente, la deuda técnica es un gran problema en nuestros proyectos. Trabajamos para una universidad, y los requisitos cambian constantemente con cada nuevo semestre o política. No se refactoriza adecuadamente, a menudo agregamos código parche. Hay proyectos, donde un cambio simple tomó semanas porque el código era tan frágil. Es frustrante porque sé que podríamos trabajar más eficientemente si tuviéramos tiempo.	Sí, creo que sí. A veces, cuando intento agregar una nueva función, termino rompiendo algo en otra parte del sistema. Paso tiempo depurando errores extraños que codificando. No estoy aprendiendo las mejores prácticas y estoy creando código complicado.	Por supuesto que es un problema. Cada proyecto es un problema. Un simple cambio en un módulo causa errores en otro modulo diferente. Todo esto está afectando la calidad y la confiabilidad de nuestro software.	Cada proyecto nuevo es una oportunidad perdida para mejorar. En lugar de eso, tomamos malas decisiones porque así es como siempre lo hemos hecho. Es técnicamente insostenible y un riesgo significativo para la universidad.	Definitivamente si, por la presión de los usuarios para que se les entregue el producto
P5	Identifico la deuda técnica cuando me complico. Por ejemplo, si tardo	Honestamente, me cuesta identificarla por mi cuenta. A	Para mí, la deuda técnica se nota cuando las pruebas empiezan a	Me gusta revisar el código varias veces, también reviso el tiempo de	No se la identifica y cuando debo seguir trabajando en el

Preg.	M1	M2	M3	M4	M5
	mucho tiempo en una tarea sencilla. Si siempre estamos parcheando el mismo módulo, probablemente necesita una reescritura. Trato de mejorar en mis tareas diarias. Si estoy tocando un área problemática, trato de dejarla un poco mejor de lo que la encontré.	veces, cuando me atasco, le pido al equipo que me ayuden. Ellos suelen señalar cosas como este método es demasiado largo o hay error en estas clases. En cuanto a manejarla, hago lo que puedo: escribo pruebas unitarias, intento no duplicar código. Pero a menudo, la presión por entregar hace que caiga en lo mismo de siempre.	fallar o cuando arreglar un error en un lugar causa problemas en otro. En cuanto al manejo, mi enfoque es tratar de documentar. Anoto cada problema, al menos así, cuando algo falla, tengo un rastro para seguir. También me gustaría más pruebas, pero es una batalla cuesta arriba.	compilación, que ha veces aumenta. Para manejarla, uso refactorización y en las reuniones hago énfasis en la calidad. pero sin el compromiso de todo el equipo, mis sugerencias quedan en el papel.	software por alguna modificación cuando me acuerdo o estoy pendiente de que algo no está bien trabajo en ese código, caso contrario no lo hago
P6	No hay proceso claro, todos sabemos que la deuda está ahí, pero no hay acciones para abordarla. En las reuniones se menciona estos problemas, pero casi siempre se pospone por prioridades más urgentes. Además, la universidad cree que el software puede crecer indefinidamente sin	Para ser honesto, no estoy seguro de que exista un proceso. Hay problemas donde todos se quejan, pero nadie realmente aborda. Cuando estoy consciente que algo está mal y trato de arreglarlo me asignan una nueva tarea urgente y lo peor sé que ese código que no arreglé está mal y se queda así.	Nuestro proceso es a veces arreglar errores, no hay una estrategia para prevenir o reducir estos problemas. Se ha sugerido y pruebas exhaustivas, pero por falta de tiempo no podemos hacerlo. El tiempo que perdemos lidiando con errores recurrentes se puede utilizar para crear nuevas funcionalidades	No se han implementado prácticas para revisiones o mejoras. Es problema interno y externo, la universidad ve el software como un gasto, no como un activo que requiere inversión. Nuestro proceso es abordar la deuda cuando causa un problema visible.	No existe ningún tipo de proceso para abordar la deuda técnica

Preg.	M1	M2	M3	M4	M5
	mantenimiento. La codificación se llena de parches.				
P7	Varios factores juegan un papel, pero diría que el principal culpable es el desconocimiento de lo que es desarrollar por parte de la administración universitaria. Nos tratan como si fuéramos un departamento de TI genérico, no un equipo de desarrollo. No entienden que cada vez que cambian un requisito a mitad del semestre es sobre trabajo al equipo, esto nos obliga a tomar atajos, sabiendo que nos complicarán más tarde	Para mí, es la falta de tiempo para hacer las cosas bien. Cuando tengo una tarea, la prioridad es que funcione, no que sea bien diseñado o codificado. A veces uso variables globales porque es más rápido y no se tiene tiempo para revisiones de código detalladas. Así que mis malos hábitos se quedan y se acumulan.	Definitivamente, los cambios constantes de requisitos. Un día estamos probando un sistema x, al día siguiente nos dicen que ahora debe integrarse con un nuevo sistema y las pruebas anteriores pues a la basura. También diría que la falta de documentación es un gran factor. Nadie tiene tiempo de documentar los cambios o el diseño.	Estamos atados a decisiones tomadas hace años, cuando la universidad tenía la mitad de los estudiantes y un tercio de los programas. Ahora, cada nuevo requisito es un parche sobre un sistema que debería haberse replanteado desde cero. Añade a eso la rotación del personal - hemos perdido desarrolladores clave que se llevaron conocimiento con ellos.	Los plazos para la entrega y los cambios frecuentes de los requisitos
P8	He aprendido a usar estrategias. En lugar de decir necesitamos dos semanas para refactorizar, digo si invertimos tres días ahora, ahorraremos dos semanas el	Siendo honesto, cedo a la presión, cuando el jefe del departamento dice que necesita algo ahora, no puedo decir que no. Así que hago que funcione como sea. Pero	La presión es pasar las pruebas, no tener pruebas exhaustivas. A veces, me salto casos o sólo pruebo el camino feliz porque no hay tiempo. Después llegan los	Entiendo la necesidad de entregar valor rápidamente, pero también sé que la velocidad sin una correcta dirección lleva al caos. Mi enfoque es priorizar ciertas partes	Realizo pruebas de funcionalidad de manera individual de acuerdo a como sigo codificando.

Preg.	M1	M2	M3	M4	M5
	próximo mes. Se nota la deuda técnica por retrasos, sobrecostos, riesgos de seguridad. A veces funciona, a veces no.	últimamente, destino más tiempo a las pruebas. Cuando estoy realmente presionado, si comento mi código para luego recordarme lo que debo arreglar.	problemas, especialmente cuando esos bugs aparecen en producción. Últimamente documento los riesgos por escrito, al menos así, cuando algo explota, puedo decir les advertí.	del sistema, como la seguridad de los datos. También utilizo soluciones temporales.	
P9	Por ejemplo, el proyecto del sistema de becas. Lo construimos rápido porque así nos solicitaron. Usamos un montón de código duro, asumiendo que los criterios de becas no cambiarían. Un año después, se cambió a la directora del departamento de bienestar estudiantil que tarjo un nuevo programa de becas con reglas totalmente diferentes. Lo que debería haber sido un cambio de una semana se convirtió en un trabajo de dos meses. Se trabajo	En una de las aplicaciones que me asignaron, se debía añadir notificaciones a los estudiantes, pero por trabajar en un código creado por otra persona y que nadie del equipo entendía completamente cómo funcionaba, se obtuvo un error que envió notificaciones de exámenes a los estudiantes equivocados.	Durante la pandemia, cuando todos los cursos se volvieron online, se debía sincronizar el distributivo con el EVA, el primer día de clases, todo se cayó. Los profesores estaban enojados, los estudiantes perdidos todo por falta de tiempo para realizar las pruebas adecuadamente, aunque el verdadero culpable era codificación obsoleta.	Mi ejemplo es nuestro sistema de datos estudiantiles. Originalmente, era una simple base de datos. Luego, cada departamento comenzó a añadir sus propias tablas, se convirtió en un monstruo. Cuando intentamos migrar a la nube para reducir costos, fue un desastre. Tuvimos que abortar la migración a mitad de camino. Todo por no invertir en una nueva aplicación limpia desde el principio	Cuando se entrega el producto y regresa con algún tipo de error en el almacenamiento en la base de datos y nos toca modificar la estructura de las tablas.

Preg.	M1	M2	M3	M4	M5
	horas extras, varios estudiantes casi pierden sus becas por retrasos.				
P10	La deuda técnica al principio no se la nota, pero con el tiempo, se desencadena en varios problemas. Cuando se tiene que actualizar ciertas funcionalidades en un sistema bien diseñado, habría sido una tarea de una semana, pero por como se encuentra estructurado nuestro código tardamos un mes. La eficiencia se pierde porque pasamos más tiempo parchando código que construyendo cosas nuevas. Por todo esto baja la calidad, las pruebas y las revisiones de código son parciales.	La deuda técnica hace que todo sea más lento y estresante. Cuando se piensa que una tarea es fácil, por los problemas acarrados y no solucionados con el código se vuelve algo muy difícil. Se pasa más tiempo entendiendo el código existente que realmente codificando.	La deuda técnica afecta la calidad de manera directa. En un proyecto limpio, se puede rastrear un error hasta su fuente, en cambio en nuestra realidad es muy difícil hacerlo. La confiabilidad del software no es buena y con ella, la confianza de los usuarios.	Un código bien diseñado es escalable, mantenible. Pero con la deuda técnica, la eficiencia se desploma porque los desarrolladores pasan más tiempo solucionando problemas antes que creando. De la misma manera, en un sistema sobrecargado de deuda técnica, no puedes asegurar que cada componente esté adecuadamente diseñado, probado y protegido.	El producto no sale al cien por ciento probado y toca a cada momento colocar parches para que funcione adecuadamente o cumpla con los nuevos requisitos solicitados
P11	Necesitamos revisiones de código más rigurosas y pruebas exhaustivas. También, necesitamos educar a los	Me gustaría que cada vez que encontremos algo que se necesite refactorizar, lo tomemos en cuenta. Así, los desarrolladores,	Documentar, hacer seguimiento, por ejemplo, tener claro donde cada cambio de código esté vinculado a un requisito y a	Aplicar pruebas, usando diferentes tipos de herramientas. Si el código no pasa, no se lo debe fusionar. Además, se debe refactorizar	Utilizar algún tipo de manual o instructivo para solucionar este tipo de problema.

Preg.	M1	M2	M3	M4	M5
	stakeholders y usuarios sobre sus requerimientos y necesidades con base a la realidad de como se encuentra actualmente el sistema.	pueden ver cuánto se acumula y tal vez, nos demos tiempo para hacerlo.	pruebas específicas. Así, cuando algo cambie, sabremos exactamente qué probar.	colocando esta tarea como prioridad.	
P12	Necesitamos un analizador de código para identificar code smells o vulnerabilidades. También, necesitamos priorizar tareas junto con las funcionales y automatizar las pruebas	Una herramienta de control de código y que detecte código duplicado automáticamente, ya que a veces copio y pego sin pensar. Además, que me alerte cuando estoy escribiendo algo que podría contribuir a la deuda técnica, como métodos demasiado largos o muchos parámetros.	Un framework de pruebas que se adapte al código. También necesitamos un sistema de monitoreo en producción que nos alerte de patrones de error en tiempo real, detectar problemas antes de que los usuarios los reporten sería genial.	Herramientas de modelado que validen la conformidad del código. Herramientas de seguimiento de código.	He leído algo sobre el tema y me enteré que hay varias herramientas para detectar problemas en la codificación.
P13	Totalmente dispuesto, creo que es mi responsabilidad como parte del equipo. La deuda técnica hace nuestro trabajo más difícil. Estoy dispuesto a quedarme después de horas, para implementar alguna solución al problema.	Sí, aunque admito que me da un poco de miedo. Veo cómo la deuda técnica nos está frenando, ya que tareas que deberían ser simples se convierten en un dolor de cabeza. Así que sí, estoy dispuesto, es una buena oportunidad para aprender y así	Claro que sí, arreglar los mismos problemas es agotador y aburrido. Se debe pasar más tiempo diseñando pruebas robustas y no arreglando errores. Abordar esta deuda técnica no es solo eficiencia, es garantizar nuestro trabajo y el servicio que	Considero que no abordar la deuda técnica es una negligencia profesional de mi parte, ya que se debe garantizar a largo plazo nuestros sistemas. Si no actuamos pronto, llegará un punto en que ninguna cantidad de tiempo o esfuerzo será suficiente, los	Si siempre y cuando no interfiera con mis tareas asignadas y permita mejorar nuestros productos.

Preg.	M1	M2	M3	M4	M5
		crear un código limpio y bien estructurado.	brindamos a la comunidad universitaria.	sistemas colapsarán y esto traerá graves consecuencias para todos.	
P14	La deuda técnica no es solo un problema técnico, es un problema ético. Cuando entregamos software lleno de deuda técnica, estamos poniendo en riesgo la privacidad de los estudiantes, la integridad de las calificaciones, la eficiencia de los procesos administrativos. Un error en el sistema de ayuda financiera podría significar que un estudiante no pueda pagar la matrícula. Un error en los datos podría retrasar la graduación de alguien. Como desarrolladores, tenemos una responsabilidad ética de hacer las cosas bien.	Me preocupa que esté aplicando malos hábitos en la programación al usar frecuentemente parches en código malo. La deuda técnica no solo ralentiza nuestro software, también podría estar frenando nuestras carreras.	Algo que no mencionamos mucho es el costo emocional de la deuda técnica. Arreglo un error, y dos más aparecen, es desmoralizante y estresante. La deuda técnica no solo hace nuestro software frágil, hace que nosotros también lo seamos	Las universidades son generadoras de conocimiento, pero aquí estamos, usando prácticas que van en contra de todo lo que enseñamos en nuestros cursos. La deuda técnica es como la erosión, lenta y si se deja sin control, puede hacer que toda la estructura se derrumbe.	Tener un mayor conocimiento para no caer en ese tipo de errores.

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: La Entrevista

La codificación de los datos se presenta en la Tabla 12:

Tabla 12.

Codificación de los datos de la entrevista.

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
J	P3	Defino la deuda técnica como un riesgo para la entrega pronta. Al principio, parece que avanzamos rápido, pero con el tiempo, las tareas simples se vuelven complejas. Los desarrolladores se frustran porque pasan más tiempo lidiando con problemas antiguos que construyendo cosas nuevas. Es un problema evidente entregar valor rápidamente sin comprometer la calidad a largo plazo	MET-Ágil DT-Riesgo DT-Complejidad DT-Comprometer la calidad	Metodologías ágiles Definición	Desarrollo de Software Deuda Técnica
J	P4	Sí, y no solo técnicamente. La deuda técnica está desgastando a nuestro equipo. En nuestras reuniones, siempre surge el tema: 'No pudimos entregar la función X porque tuvimos que arreglar Y'. Los plazos se alargan, el equipo se estresa, y los diferentes departamentos de la universidad piensa que somos ineficientes. Además, es difícil tener motivado al equipo cuando los desarrolladores se dan cuenta de que pasarán más tiempo apagando incendios que innovando	DT-Impacto DT-Estrés DT-Motivación DT-Ineficiencia	Impacto	Deuda Técnica
J	P5	En general, procuramos aplicar metodologías ágiles no una específica más bien una combinación de ellas, dividimos el trabajo en módulos cortos con objetivos definidos, realizamos reuniones diarias para evaluar el progreso y adaptar las prioridades. Sin embargo, los continuos problemas en el desarrollo nos obligan a adaptar estas metodologías, cambiando continuamente la prioridad a las tareas asignadas a los miembros ajustando las estimaciones de tiempo considerando el avance y tipo de tareas	MET-Ágil MET-Adaptación MET-Priorización	Metodologías ágiles	Desarrollo de Software

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
J	P6	Cuando se estima incorrectamente, o cuando las supuestas tareas sencillas se convierten en complejas, eso pienso que es deuda técnica. También la noto en la motivación del equipo por las diferentes quejas de sobre trabajo. Para manejar este problema, se trata de crear espacios en los diferentes procesos. También facilito discusiones con los stakeholders sobre la priorización de las entregas y los riesgos de apresurarse	DT-Identificación DT-Gestión DT-Estimación DT-Priorización	Gestión	Deuda Técnica
J	P7	La toma de decisiones relacionadas con la deuda técnica es un proceso complejo en nuestro entorno. Por lo general dependemos de otros por las decisiones de los diferentes departamentos de la Universidad sobre las diferentes aplicaciones solicitadas, esto nos induce a no poder priorizar o implementar soluciones a la deuda técnica de forma independiente básicamente porque la asignación de tiempo y recursos para abordar la deuda técnica a menudo compite con otras prioridades, como el desarrollo de nuevas funcionalidades	DT-Decisiones DT-Dependencia DT-Priorización DT-Recursos	Gestión	Deuda Técnica
J	P8	Son varios factores ya que los plazos en el ámbito académicos son severos, por ejemplo, si el sistema de matrícula no está listo para el inicio del semestre, es un desastre. Pero los requisitos cambian porque cada departamento quiere su supuesto pequeño cambio o adición. También está el desconocimiento sobre los procesos de desarrollo y la falta de una verdadera participación de los usuarios finales. A menudo, construimos basándonos en lo que un decano o administrador cree que los estudiantes o profesores necesitan, no en investigación de usuarios real	DT-Factores-Internos DT-Plazos DT-Cambios-requisitos DT-Desconocimiento DT-Participación-Usuarios	Causas	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
J	P9	Los continuos cambios en los requisitos lo que genera retrabajo y sensación de que el proyecto no avanza. La presión por parte de los usuarios para la entrega pronta de los productos solicitados. También se tiene la necesidad de aprender y adaptarse a nuevas tecnologías rápidamente lo que aumenta el estrés y la carga de trabajo en el equipo	DT-Factores-Externos DT-Cambios-Requisitos DT-Presión-Usuarios DT-Nuevas-tecnologías	Causas	Deuda Técnica
J	P10	Para mí, el impacto más claro ha sido en el equipo mismo. Nos pasamos tratando de resolver los mismos problemas una y otra vez, esto afecta la moral del equipo limitándonos en el lanzamiento de nuevas funciones. Estos problemas no solo afectan al código, sino que también está afectando la pasión y creatividad del equipo	DT-Impacto DT-Moral-Equipo DT-Creatividad DT-Repetición-Problemas	Impacto	Deuda Técnica
J	P11	Nos frustra tener que lidiar con código de baja calidad y soluciones temporales que retrasan el desarrollo. Nos desmotiva sentir que nuestro trabajo no es valorado por la complejidad del código heredado. El estrés aumenta por la presión de cumplir plazos con problemas técnicos constantes. Todo esto afecta nuestra productividad. Dedicamos tiempo a corregir errores en lugar de innovar, y los cambios en el código son más lentos y complejos. Necesitamos asumir la deuda técnica como un problema serio, implementar una estrategia para gestionarla, buscar las herramientas para afrontarla, y así crear un ambiente de trabajo donde se valore la calidad del código	DT-Moral DT-Productividad DT-Frustración DT-Desmotivación DT-Estrés DT-Innovación	Impacto	Deuda Técnica
J	P12	Adecuadas estimaciones tomando en cuenta la complejidad y el costo de los continuos cambios, esto pienso que haría que la deuda sea parte de cada decisión. También, creo que se necesitan aplicar mediciones y una guía para controlar adecuadamente todo este tipo de problemas	DT-Medidas DT-Estimaciones DT-Mediciones DT-Guía	Soluciones	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
J	P13	Desde la perspectiva de gestión, necesitamos mejor visibilidad, en las reuniones identificar los problemas en los retrasos por la mala codificación, lo que haría que la deuda ya no se ignore. También, una integración entre nuestras herramientas de desarrollo y un software que permita gestionar proyectos. Además, un informe mensual que sume todo esto en términos de costo y riesgo	DT-Herramientas DT-Visibilidad DT-Integración DT-Informes	Soluciones	Deuda Técnica
J	P14	La deuda técnica es un problema latente en nuestros procesos ya que la eficiencia no es solo velocidad, es la capacidad de entregar valor consistentemente. Con la deuda técnica, nuestras entregas se vuelven impredecibles. Las estimaciones no se cumplen. Eso destroza nuestra planificación y afecta a la calidad. Pensando bien con tanta deuda, nos preocupamos en como sobrevivir antes que en cómo mejorar. Estamos en modo de supervivencia, no de calidad y a lo largo plazo, eso es más trabajo y mala calidad del producto	DT-Impacto-Calidad DT-Impacto-Mantenibilidad DT-Impacto-Escalabilidad DT-Impredecibilidad DT-Planificación	Impacto	Deuda Técnica
J	P15	No utilizamos métricas ese es nuestro problema, esto se debe a que no tenemos un equipo numeroso y siempre nuestro enemigo es el tiempo. Nos preocupamos siempre por tiempo de entrega, tratando de cierta manera en medir cuánto tardamos en desarrollar. Algunas ocasiones hemos realizado encuestas de satisfacción, pero últimamente ya no las hemos aplicado	DT-Métricas DT-Tiempo-Entrega DT-Satisfacción-Usuarios	Medición	Deuda Técnica
J	P16	Eso es un desafío constante, se evalúa la gravedad de los problemas, considerando su impacto en la calidad del software. Aquí se analiza los riesgos de que cause errores, bloqueos o retrasos en el desarrollo, todo esto siempre tomando en cuenta el impacto potencial de estos problemas en las operaciones de la universidad y la satisfacción de los usuarios	DT-Priorización DT-Evaluación-Riesgos DT-Impacto-Operaciones DT-Satisfacción-Usuarios	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M1	P3	De lo que tengo entendido la deuda técnica es como pedir un préstamo en el desarrollo. Para cumplir con los plazos, tomamos atajos o implementamos soluciones rápidas que sabemos que no son las mejores. Es como si dijéramos que lo arreglaremos más tarde. Esto retrasa el trabajo y no permite realizar nuevas funcionalidades	MET-Plazos entrega DT-Impredecibilidad DT-Planificación	Metodologías Ágiles Impacto	Desarrollo de Software Deuda Técnica
M2	P3	Honestamente, no estoy muy seguro. He oído hablar de deuda técnica, pero para mí suena como algo relacionado con las finanzas. Supongo que tiene que ver con cosas que hacemos mal o a medias porque estamos apurados. Como cuando copio y pego código para ahorrarme tiempo.	DT-Identificación	Gestión	Deuda Técnica
M3	P3	Desde mi perspectiva en calidad, la deuda técnica es todo ese código espagueti y parches rápidos que hacen que las pruebas sean un dolor de cabeza. Es como programar en el aire, cuanto más construimos, más inestable se vuelve todo	MET-Calidad DT-Impacto	Metodologías Ágiles Impacto	Desarrollo de Software Deuda Técnica
M4	P3	Para mí la deuda técnica puede surgir de diversas fuentes: requisitos mal definidos, presión de tiempo, falta de pruebas, o desconocimiento.	MET-Requisitos DT-Impacto DT-Pruebas	Metodologías ágiles Causas	Desarrollo de Software Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M5	P3	La codificación no esencial que no se realizó para dar prioridad a la entrega del producto	DT-Codificación	Causas	Deuda Técnica
M1	P4	Definitivamente, la deuda técnica es un gran problema en nuestros proyectos. Trabajamos para una universidad, y los requisitos cambian constantemente con cada nuevo semestre o política. No se refactoriza adecuadamente, a menudo agregamos código parche. Hay proyectos, donde un cambio simple tomó semanas porque el código era tan frágil. Es frustrante porque sé que podríamos trabajar más eficientemente si tuviéramos tiempo.	DT-Factores-Internos DT-Plazos DT-Cambios-requisitos DT-Refactorización-No adecuada	Causas	Deuda Técnica
M2	P4	Sí, creo que sí. A veces, cuando intento agregar una nueva función, termino rompiendo algo en otra parte del sistema. Paso tiempo depurando errores extraños que codificando. No estoy aprendiendo las mejores prácticas y estoy creando código complicado.	DT-Impacto-Calidad DT-Impacto-Codificación	Impacto	Deuda Técnica
M3	P4	Por supuesto que es un problema. Cada proyecto es un problema. Un simple cambio en un módulo causa errores en otro modulo diferente. Todo esto está afectando la calidad y la confiabilidad de nuestro software.	DT-Impacto-Calidad DT-Impacto-Codificación	Impacto	Deuda Técnica
M4	P4	Cada proyecto nuevo es una oportunidad perdida para mejorar. En lugar de eso, tomamos malas decisiones porque así es como siempre lo hemos hecho. Es técnicamente insostenible y un riesgo significativo para la universidad.	DT-Evaluación-Riesgos	Gestión	Deuda Técnica
M5	P4	Definitivamente si, por la presión de los usuarios para que se les entregue el producto	DT-Satisfacción-Usuarios	Gestión	Deuda Técnica
M1	P5	Identifico la deuda técnica cuando me complico. Por ejemplo, si tardo mucho tiempo en una tarea sencilla. Si siempre estamos parcheando el mismo módulo, probablemente necesita una	DT-Identificación	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
		reescritura. Trato de mejorar en mis tareas diarias. Si estoy tocando un área problemática, trato de dejarla un poco mejor de lo que la encontré. .			
M2	P5	Honestamente, me cuesta identificarla por mi cuenta. A veces, cuando me atasco, le pido al equipo que me ayuden. Ellos suelen señalar cosas como este método es demasiado largo o hay error en estas clases. En cuanto a manejarla, hago lo que puedo: escribo pruebas unitarias, intento no duplicar código. Pero a menudo, la presión por entregar hace que caiga en lo mismo de siempre.	DT-Identificación DT-Control	Gestión	Deuda Técnica
M3	P5	Para mí, la deuda técnica se nota cuando las pruebas empiezan a fallar o cuando arreglar un error en un lugar causa problemas en otro. En cuanto al manejo, mi enfoque es tratar de documentar. Anoto cada problema, al menos así, cuando algo falla, tengo un rastro para seguir. También me gustaría más pruebas, pero es una batalla cuesta arriba	DT-Documentación DT-Pruebas	Soluciones	Deuda Técnica
M4	P5	Me gusta revisar el código varias veces, también reviso el tiempo de compilación, que a veces aumenta. Para manejarla, uso refactorización y en las reuniones hago énfasis en la calidad. pero sin el compromiso de todo el equipo, mis sugerencias quedan en el papel.	DT-Documentación DT-Pruebas DT-Refactorización	Soluciones	Deuda Técnica
M5	P5	No se la identifica y cuando debo seguir trabajando en el software por alguna modificación cuando me acuerdo o estoy pendiente de que algo no está bien trabajo en ese código, caso contrario no lo hago	DT-Identificación DT-Control	Gestión	Deuda Técnica
M1	P6	No hay proceso claro, todos sabemos que la deuda está ahí, pero no hay acciones para abordarla. En las reuniones se menciona estos problemas, pero casi siempre se pospone por prioridades más urgentes. Además, la universidad cree que el software puede crecer	DT-Identificación DT-Control	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
		indefinidamente sin mantenimiento. La codificación se llena de parches.			
M2	P6	Para ser honesto, no estoy seguro de que exista un proceso. Hay problemas donde todos se quejan, pero nadie realmente aborda. Cuando estoy consciente que algo está mal y trato de arreglarlo me asignan una nueva tarea urgente y lo peor sé que ese código que no arreglé está mal y se queda así.	DT-Identificación DT-Control	Gestión	Deuda Técnica
M3	P6	Nuestro proceso es a veces arreglar errores, no hay una estrategia para prevenir o reducir estos problemas. Se ha sugerido y pruebas exhaustivas, pero por falta de tiempo no podemos hacerlo. El tiempo que perdemos lidiando con errores recurrentes se puede utilizar para crear nuevas funcionalidades	DT-Identificación DT-Control	Gestión	Deuda Técnica
M4	P6	No se han implementado prácticas para revisiones o mejoras. Es problema interno y externo, la universidad ve el software como un gasto, no como un activo que requiere inversión. Nuestro proceso es abordar la deuda cuando causa un problema visible.	DT-Identificación DT-Control	Gestión	Deuda Técnica
M5	P6	No existe ningún tipo de proceso para abordar la deuda técnica	DT-Identificación DT-Control	Gestión	Deuda Técnica
M1	P7	Varios factores juegan un papel, pero diría que el principal culpable es el desconocimiento de lo que es desarrollar por parte de la administración universitaria. Nos tratan como si fuéramos un departamento de TI genérico, no un equipo de desarrollo. No entienden que cada vez que cambian un requisito a mitad del semestre es sobre trabajo al equipo, esto nos obliga a tomar atajos, sabiendo que nos complicarán más tarde.	DT-Factores-Internos DT-Cambios-requisitos	Causas	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M2	P7	Para mí, es la falta de tiempo para hacer las cosas bien. Cuando tengo una tarea, la prioridad es que funcione, no que sea bien diseñado o codificado. A veces uso variables globales porque es más rápido y no se tiene tiempo para revisiones de código detalladas. Así que mis malos hábitos se quedan y se acumulan.	DT-Identificación DT-Control	Gestión	Deuda Técnica
M3	P7	Definitivamente, los cambios constantes de requisitos. Un día estamos probando un sistema x, al día siguiente nos dicen que ahora debe integrarse con un nuevo sistema y las pruebas anteriores pues a la basura. También diría que la falta de documentación es un gran factor. Nadie tiene tiempo de documentar los cambios o el diseño.	DT-Factores-Internos DT-Cambios-requisitos DT-Falta-Documentación	Causas	Deuda Técnica
M4	P7	Estamos atados a decisiones tomadas hace años, cuando la universidad tenía la mitad de los estudiantes y un tercio de los programas. Ahora, cada nuevo requisito es un parche sobre un sistema que debería haberse replanteado desde cero. Añade a eso la rotación del personal - hemos perdido desarrolladores clave que se llevaron conocimiento con ellos.	DT-Factores-Internos DT-Cambios-requisitos	Causas	Deuda Técnica
M5	P7	Los plazos para la entrega y los cambios frecuentes de los requisitos	DT-Cambios-requisitos	Causas	Deuda Técnica
M1	P8	He aprendido a usar estrategias. En lugar de decir necesitamos dos semanas para refactorizar, digo si invertimos tres días ahora, ahorraremos dos semanas el próximo mes. Se nota la deuda técnica por retrasos, sobrecostos, riesgos de seguridad. A veces funciona, a veces no.	DT-Identificación DT-Control	Gestión	Deuda Técnica
M2	P8	Siendo honesto, cedo a la presión, cuando el jefe del departamento dice que necesita algo ahora, no puedo decir que no. Así que hago que funcione como sea. Pero últimamente,	DT-Control	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
		destino más tiempo a las pruebas. Cuando estoy realmente presionado, si comento mi código para luego recordarme lo que debo arreglar.			
M3	P8	La presión es pasar las pruebas, no tener pruebas exhaustivas. A veces, me salto casos o sólo pruebo el camino feliz porque no hay tiempo. Después llegan los problemas, especialmente cuando esos bugs aparecen en producción. Últimamente documento los riesgos por escrito, al menos así, cuando algo explota, puedo decir les advertí.	DT-Pruebas-exhaustivas DT-Falta-Documentación	Causas	Deuda Técnica
M4	P8	Entiendo la necesidad de entregar valor rápidamente, pero también sé que la velocidad sin una correcta dirección lleva al caos. Mi enfoque es priorizar ciertas partes del sistema, como la seguridad de los datos. También utilizo soluciones temporales.	DT-Control	Gestión	Deuda Técnica
M5	P8	Realizo pruebas de funcionalidad de manera individual de acuerdo a como sigo codificando.	DT-Pruebas-exhaustivas	Causas	Deuda Técnica
M1	P9	Por ejemplo, el proyecto del sistema de becas. Lo construimos rápido porque así nos solicitaron. Usamos un montón de código duro, asumiendo que los criterios de becas no cambiarían. Un año después, se cambió a la directora del departamento de bienestar estudiantil que tarjo un nuevo programa de becas con reglas totalmente diferentes. Lo que debería haber sido un cambio de una semana se convirtió en un trabajo de dos meses. Se trabajo horas extras, varios estudiantes casi pierden sus becas por retrasos. .	DT-Factores-Internos DT-Cambios-requisitos	Causas	Deuda Técnica
M2	P9	En una de las aplicaciones que me asignaron, se debía añadir notificaciones a los estudiantes, pero por trabajar en un código creado por otra persona y que nadie del equipo entendía	DT-Codificación-Obsoleta	Causas	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
		completamente cómo funcionaba, se obtuvo un error que envió notificaciones de exámenes a los estudiantes equivocados			
M3	P9	Durante la pandemia, cuando todos los cursos se volvieron online, se debía sincronizar el distributivo con el EVA, el primer día de clases, todo se cayó. Los profesores estaban enojados, los estudiantes perdidos todo por falta de tiempo para realizar las pruebas adecuadamente, aunque el verdadero culpable era codificación obsoleta.	DT-Falta-Pruebas DT-Codificación-Obsoleta	Causas	Deuda Técnica
M4	P9	Mi ejemplo es nuestro sistema de datos estudiantiles. Originalmente, era una simple base de datos. Luego, cada departamento comenzó a añadir sus propias tablas, se convirtió en un monstruo. Cuando intentamos migrar a la nube para reducir costos, fue un desastre. Tuvimos que abortar la migración a mitad de camino. Todo por no invertir en una nueva aplicación limpia desde el principio	DT-Factores-Internos DT-Cambios-requisitos	Causas	Deuda Técnica
M5	P9	Cuando se entrega el producto y regresa con algún tipo de error en el almacenamiento en la base de datos y nos toca modificar la estructura de las tablas.	DT-Codificación-Obsoleta	Causas	Deuda Técnica
M1	P10	La deuda técnica al principio no se la nota, pero con el tiempo, se desencadena en varios problemas. Cuando se tiene que actualizar ciertas funcionalidades en un sistema bien diseñado, habría sido una tarea de una semana, pero por como se encuentra estructurado nuestro código tardamos un mes. La eficiencia se pierde porque pasamos más tiempo parchando código que construyendo cosas nuevas. Por todo esto baja la calidad, las pruebas y las revisiones de código son parciales.	DT-Codificación-Obsoleta	Causas	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M2	P10	La deuda técnica hace que todo sea más lento y estresante. Cuando se piensa que una tarea es fácil, por los problemas acarrados y no solucionados con el código se vuelve algo muy difícil. Se pasa más tiempo entendiendo el código existente que realmente codificando.	DT-Codificación-Obsoleta	Causas	Deuda Técnica
M3	P10	La deuda técnica afecta la calidad de manera directa. En un proyecto limpio, se puede rastrear un error hasta su fuente, en cambio en nuestra realidad es muy difícil hacerlo. La confiabilidad del software no es buena y con ella, la confianza de los usuarios.	DT-Impacto-Calidad	Impacto	Deuda Técnica
M4	P10	Un código bien diseñado es escalable, mantenible. Pero con la deuda técnica, la eficiencia se desploma porque los desarrolladores pasan más tiempo solucionando problemas antes que creando. De la misma manera, en un sistema sobrecargado de deuda técnica, no puedes asegurar que cada componente esté adecuadamente diseñado, probado y protegido.	DT-Impacto-Eficiencia	Impacto	Deuda Técnica
M5	P10	El producto no sale al cien por ciento probado y toca a cada momento colocar parches para que funcione adecuadamente o cumpla con los nuevos requisitos solicitados	DT-Falta-Pruebas	Causas	Deuda Técnica
M1	P11	Necesitamos revisiones de código más rigurosas y pruebas exhaustivas. También, necesitamos educar a los stakeholders y usuarios sobre sus requerimientos y necesidades con base a la realidad de como se encuentra actualmente el sistema.	DT-Aplicar-Pruebas	Gestión	Deuda Técnica
M2	P11	Me gustaría que cada vez que encontremos algo que se necesite refactorizar, lo tomemos en cuenta. Así, los desarrolladores, pueden ver cuánto se acumula y tal vez, nos demos tiempo para hacerlo.	DT-Aplicar-Refactorización	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M3	P11	Documentar, hacer seguimiento, por ejemplo, tener claro donde cada cambio de código esté vinculado a un requisito y a pruebas específicas. Así, cuando algo cambie, sabremos exactamente qué probar.	DT-Aplicar-Pruebas DT-Documentar	Gestión	Deuda Técnica
M4	P11	Aplicar pruebas, usando diferentes tipos de herramientas. Si el código no pasa, no se lo debe fusionar. Además, se debe refactorizar colocando esta tarea como prioridad.	DT-Aplicar-Refactorización DT-Aplicar-Pruebas	Gestión	Deuda Técnica
M5	P11	Utilizar algún tipo de manual o instructivo para solucionar este tipo de problema.	DT-Marco-Gestión	Gestión	Deuda Técnica
M1	P12	Necesitamos un analizador de código para identificar code smells o vulnerabilidades. También, necesitamos priorizar tareas junto con las funcionales y automatizar las pruebas.	DT-Aplicar-Pruebas DT-Análisis-Estático-Código	Gestión	Deuda Técnica
M2	P12	Una herramienta de control de código y que detecte código duplicado automáticamente, ya que a veces copio y pego sin pensar. Además, que me alerte cuando estoy escribiendo algo que podría contribuir a la deuda técnica, como métodos demasiado largos o muchos parámetros	DT-Análisis-Estático-Código	Gestión	Deuda Técnica
M3	P12	Un framework de pruebas que se adapte al código. También necesitamos un sistema de monitoreo en producción que nos alerte de patrones de error en tiempo real, detectar problemas antes de que los usuarios los reporten sería genial.	DT-Marco-Gestión DT-Análisis-Estático-Código	Gestión	Deuda Técnica
M4	P12	Herramientas de modelado que validen la conformidad del código. Herramientas de seguimiento de código.	DT-Marco-Gestión DT-Análisis-Estático-Código	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M5	P12	He leído algo sobre el tema y me enteré que hay varias herramientas para detectar problemas en la codificación.	DT-Análisis-Estático-Código	Gestión	Deuda Técnica
M1	P13	Totalmente dispuesto, creo que es mi responsabilidad como parte del equipo. La deuda técnica hace nuestro trabajo más difícil. Estoy dispuesto a quedarme después de horas, para implementar alguna solución al problema.	DT-Marco-Gestión	Gestión	Deuda Técnica
M2	P13	Sí, aunque admito que me da un poco de miedo. Veo cómo la deuda técnica nos esta frenando, ya que tareas que deberían ser simples se convierten en un dolor de cabeza. Así que sí, estoy dispuesto, es una buena oportunidad para aprender y así crear un código limpio y bien estructurado.	DT-Marco-Gestión	Gestión	Deuda Técnica
M3	P13	Claro que sí, arreglar los mismos problemas es agotador y aburrido. Se debe pasar más tiempo diseñando pruebas robustas y no arreglando errores. Abordar esta deuda técnica no es solo eficiencia, es garantizar nuestro trabajo y el servicio que brindamos a la comunidad universitaria.	DT-Marco-Gestión	Gestión	Deuda Técnica
M4	P13	Considero que no abordar la deuda técnica es una negligencia profesional de mi parte, ya que se debe garantizar a largo plazo nuestros sistemas. Si no actuamos pronto, llegará un punto en que ninguna cantidad de tiempo o esfuerzo será suficiente, los sistemas colapsarán y esto traerá graves consecuencias para todos.	DT-Marco-Gestión	Gestión	Deuda Técnica
M5	P13	Si siempre y cuando no interfiera con mis tareas asignadas y permita mejorar nuestros productos.	DT-Marco-Gestión	Gestión	Deuda Técnica

Id. Entre.	Preg.	Respuesta Textual	Código	Categoría	Tema
M1	P13	La deuda técnica no es solo un problema técnico, es un problema ético. Cuando entregamos software lleno de deuda técnica, estamos poniendo en riesgo la privacidad de los estudiantes, la integridad de las calificaciones, la eficiencia de los procesos administrativos. Un error en el sistema de ayuda financiera podría significar que un estudiante no pueda pagar la matrícula. Un error en los datos podría retrasar la graduación de alguien. Como desarrolladores, tenemos una responsabilidad ética de hacer las cosas bien.	DT-Marco-Gestión	Gestión	Deuda Técnica
M2	P13	Me preocupa que esté aplicando malos hábitos en la programación al usar frecuentemente parches en código malo. La deuda técnica no solo ralentiza nuestro software, también podría estar frenando nuestras carreras.	DT-Marco-Gestión	Gestión	Deuda Técnica
M3	P13	Algo que no mencionamos mucho es el costo emocional de la deuda técnica. Arreglo un error, y dos más aparecen, es desmoralizante y estresante. La deuda técnica no solo hace nuestro software frágil, hace que nosotros también lo seamos	DT-Marco-Gestión	Gestión	Deuda Técnica
M4	P13	Las universidades son generadoras de conocimiento, pero aquí estamos, usando prácticas que van en contra de todo lo que enseñamos en nuestros cursos. La deuda técnica es como la erosión, lenta y si se deja sin control, puede hacer que toda la estructura se derrumbe.	DT-Marco-Gestión	Gestión	Deuda Técnica
M5	P13	Tener un mayor conocimiento para no caer en ese tipo de errores.	DT-Marco-Gestión	Gestión	Deuda Técnica

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: La Entrevista

Presentación de los resultados

Impacto de la Deuda Técnica

- La deuda técnica afecta el software frenando el desarrollo planificado debido a que la mayoría de las tareas simples las convierte en complejas (DT-Impacto-Calidad, DT-Impacto-Eficiencia).
- La deuda técnica afecta en la calidad del software, siendo la responsable de errores recurrentes y pruebas inexistentes o inestables (DT-Impacto-Calidad, DT-Impacto-Codificación, DT-Falta-Pruebas)
- La deuda técnica es responsable de la baja moral del equipo causando frustración en los integrantes (DT-Factores-Externos, DT-Presión-Usuarios, DT-Moral-Equipo, DT-Creatividad, DT-desmotivación, DT-estrés)

Causas de la Deuda Técnica

- La premura por entregas prontas conlleva a que el equipo utilice atajos (DT-Plazos, DT-Complejidad, MET-Plazos-Entrega).
- La deuda técnica no se puede obviar en ciertos casos debido a información incompleta y a solicitudes de último momento (DT-Complejidad, DT-Impredecibilidad, DT-Planificación, DT-Factores-Internos, DT-Cambios-requisitos).

Estrategias para Gestionar la Deuda Técnica

- Ejecutar medidas mediante la implementación de métricas acordes a las actividades de desarrollo realizadas (DT-Estimaciones, DT-Mediciones, DT-Marco-Gestión).
- Efectuar una revisión rigurosa de la codificación (DT-Cambios-requisitos, DT-Codificación-Obsoleta, DT-Aplicar-Pruebas, DT-Análisis-Estático-Código, DT-Marco-Gestión)

- Implementar pruebas frecuentes (DT-Documentar, DT-Aplicar-Refactorización, DT-Aplicar-Pruebas, DT-Análisis-Estático-Código, DT-Marco-Gestión)
- Socializar a todos los stakeholders sobre el problema, impacto y costos de la deuda técnica (DT-Impacto, DT-Presión-Usuarios)

Herramientas y Procesos

- Analizadores de código estático y herramientas de visualización (DT-Análisis-Estático-Código, DT-Marco-Gestión).
- Gestión adecuada para identificar y priorizar la deuda técnica (DT-Guía, DT-Marco-Gestión)
- Documentar adecuadamente el diseño, modelado y proceso de desarrollo (DT-Falta-Documentación, DT-Control, DT-Marco-Gestión)

Roles y Responsabilidades

- Jefe de equipo: ser el líder en la búsqueda de soluciones, guiar al equipo, proteger al equipo, facilitar cultura de responsabilidad
- Equipo de desarrollo: aplicar buenas prácticas, contribuir con la gestión de la deuda.
- Testers: aplicar pruebas, documentar fallos, hacer visible la deuda técnica
- Arquitectos: establecer estándares, gestionar y comunicar riesgos

Disposición y Reflexiones

- Apoyo total y un alto compromiso para abordar la deuda técnica sacrificando horario de trabajo y tiempo extra.
- Concientización y reconocimiento del impacto de la deuda técnica en el ámbito ético y emocional.
- Comprometimiento con la institución y aporte para el desarrollo profesional.

Conclusiones de la investigación cualitativa

Los resultados de la entrevista revelan que existe una clara percepción de la deuda técnica y su preocupación por una adecuada gestión por parte de los miembros del equipo. El análisis de los datos obtenidos señala varios puntos que se destacan a continuación:

- Aunque algunos entrevistados no se encuentran familiarizados con el término de la deuda técnica, sin embargo, se encuentran conscientes del riesgo de la deuda para la entrega a tiempo del producto. Los desarrolladores consideran que la deuda técnica complica las tareas generando frustración en el equipo.
- La deuda técnica la identifican como un problema que además de afectar las actividades técnicas también incurren en el desempeño del equipo. Los desarrolladores se sienten frustrados y estresados al tener que lidiar con problemas acumulados en lugar de innovar.
- En relación con el desarrollo de software el equipo utiliza como metodología un modelo ágil adaptado, debido a la flexibilidad con las peticiones de la universidad a pesar de los diferentes desafíos técnicos derivando en una inestabilidad en los procesos.
- No existe una identificación y gestión de la deuda técnica, se deben establecer prácticas solidas con un marco formal. Las decisiones no las toma el equipo ya que estas dependen de los requerimientos de los diversos departamentos universitarios.
- Los factores internos por la falta de una adecuada organización institucional son uno de los problemas principales que contribuyen en la acumulación de la deuda técnica. Solicitud de entregas inmediatas, cumplimiento de regulaciones estrictas de organismos de control, requisitos cambiantes y la poca participación de los usuarios acrecientan el problema de la deuda.

- Se limita la implantación de nuevas funcionalidades por la acumulación de problemas repetidos. Esto compromete en la calidad del software a corto y largo plazo repercutiendo en la creatividad del equipo.
- No se utilizan métricas para identificar y monitorear la deuda técnica. Se deben implantar estimaciones, mediciones y marcos de control, para esto se deben integrar diferentes técnicas y herramientas que aporten en la gestión de los proyectos.
- Es fundamental la aplicación de prácticas formales para la gestión de la deuda técnica, se deben establecer estrategias para identificar, medir, monitorear y controlar la deuda, mediante una combinación de herramientas técnicas y procesos adaptados que abarque a todos los interesados. Estas acciones contribuirán en mejorar la calidad del Software aportando en la eficiencia del equipo, la satisfacción de los usuarios, y el desarrollo de la institución.

Análisis de datos

Además de aplicar entrevistas semiestructuradas para identificar las principales fuentes de deuda técnica y comprender cómo se priorizan las tareas en el equipo de desarrollo, se realizó un análisis del módulo de Sistema de Gestión utilizando la herramienta de análisis estático de código SonarQube.

Nombre del Proyecto: Módulo SG (Sistema de Gestión).

Fecha del Informe: 06 de febrero de 2022

Versión del Análisis: 1.0

1. Debt Ratio (Relación de Deuda Técnica)

Debt Ratio: 35%

Deuda Técnica Total: 110 horas

Los resultados señalan que la relación de deuda técnica es alta, indicando que un 35% del esfuerzo total del proyecto estaría dedicado a resolver la deuda técnica existente.

2. Bugs (Errores)

Total de Bugs: 120

Críticos: 25

Mayores: 45

Menores: 50

De acuerdo con el informe la cantidad de errores críticos es alta por lo que se requiere atención inmediata para evitar fallos graves en el módulo.

3. Vulnerabilidades de Seguridad

Total de Vulnerabilidades: 80

Críticas: 15

Mayores: 30

Menores: 35

Los resultados indican una cantidad considerable de vulnerabilidades, siendo las críticas las que representan un riesgo significativo para la seguridad del sistema.

4. Code Smells

Total de Code Smells: 300

Altos: 100

Medios: 120

Bajos: 80

La cantidad de "code smells" es alta, lo que indica que el código tiene problemas de mantenibilidad y legibilidad, siendo la refactorización una solución que se debe aplicar.

5. Cobertura de Pruebas

Cobertura Total: 40%

Cobertura de Líneas: 35%

Cobertura de Ramas: 20%

Existe una baja cobertura de pruebas, este resultado es preocupante ya que una gran parte del código no está siendo testeado adecuadamente.

6. Duplicación de Código

Líneas Duplicadas: 1500 (30%)

Bloques Duplicados: 200

Los resultados arrojados de la duplicación de código son demasiado altos, lo que señala que el código se repite innecesariamente, lo que a futuro afectará en el mantenimiento, al mismo tiempo que contribuye al aumento de la deuda técnica.

Conclusiones del análisis de datos

El análisis de los resultados obtenidos por SonarQube demuestra que el módulo evaluado tiene problemas en la calidad del código. Los valores altos de la deuda técnica, el número de bugs encontrados, las vulnerabilidades y la baja cobertura de pruebas necesitan abordarse inmediatamente por ser críticos. Estas áreas requieren atención para lograr un sistema eficiente y que asegure la mantenibilidad del módulo. Se recomienda aplicar una gestión de la deuda técnica para mejorar la calidad del código y las siguientes acciones:

- Bugs: Minimizar los bugs y priorizar la corrección de los errores críticos, con lo que se logrará la estabilidad del sistema.
- Vulnerabilidades: Se debe solucionar las vulnerabilidades críticas de seguridad para evitar que el sistema sufra de posibles ataques.

- Code Smells: Aplicar la refactorización planificada para asegurar la mantenibilidad y legibilidad del código.
- Cobertura de Pruebas: Implementar un programa de cobertura de pruebas, para reducir el riesgo de errores no detectados.
- Duplicación de Código: Refactorizar el código con la finalidad de eliminar duplicaciones y garantizar la mantenibilidad del proyecto.

Además, el análisis realizado con la aplicación de las entrevistas y el análisis de datos reveló las principales causas de la deuda técnica y su impacto en el desarrollo de software, también, se identificó el estado actual de la deuda técnica en modelos de desarrollo ágil, cumpliendo los objetivos uno y dos propuestos. Tanto a nivel del equipo de desarrollo como en el módulo evaluado se presenta un alto índice de deuda técnica, la aplicación de una adecuada gestión de la deuda técnica mejorará la calidad del software, además, de disminuir los tiempos y costos de mantenimiento.

Desarrollo y pruebas

De acuerdo con lo planteado en la etapa nueve, validación del marco, de la fase tres de la descripción general de la solución propuesta y las actividades de las fases tres y cuatro de la implementación de la propuesta, se realizan las tareas trazadas.

Fase 1. Análisis de Criticidad

Se elaboró la matriz de Impacto, considerando los siguientes módulos:

- Evidencias: Este módulo gestiona la carga de evidencias de las actividades efectuadas por los docentes mensualmente, la creación de planes de clase semanal y su seguimiento, la carga de calificaciones por parte de los docentes.

- SEDD (Sistema de Evaluación al Desempeño Docente): Este módulo gestiona la evaluación a los docentes mediante la autoevaluación, la coevaluación y la heteroevaluación de los docentes, además genera todos los informes de desempeño. Además, este módulo controla la carga de los sílabos.
- SG (Sistema de Gestión): Este módulo gestiona la carga de distributivos y horarios semestrales de los docentes.
- Portal Estudiantil (UNIMATIC): Portal donde se gestiona los expedientes académicos de los estudiantes, en este módulo los estudiantes pueden consultar las notas y registro de inasistencias de todas las asignaturas.
- Inscripciones y matrículas en línea: Este módulo gestiona las inscripciones y matrícula de los estudiantes.

Se seleccionan los criterios de impacto, a los cuales se les asigna una puntuación comprendida en una escala que va del 1 al 5, estos criterios se describen a continuación y los resultados se observan en la Tabla 13:

- Complejidad técnica.
- Importancia para los usuarios (autoridades, docentes, estudiantes).
- Repercusiones en la calidad del servicio educativo.

Tabla 13.

Matriz de Impacto.

Módulo	Complejidad Técnica	Importancia para Usuarios	Repercusiones en Calidad	Total
Evidencias	3	4	4	11
SEDD	4	4	4	12
SG	5	5	5	15

Módulo	Complejidad Técnica	Importancia para Usuarios	Repercusiones en Calidad	Total
Portal estudiantil	3	5	4	12
Inscripciones y matrículas	3	5	4	12

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Para identificar los módulos críticos se utiliza la matriz de decisión en la cual, se aplican los siguientes criterios de decisión y asignación de pesos:

- Valor (impacto + esfuerzo), 60% (importancia para usuarios) + 40% (repercusiones en calidad).
- Deuda técnica, 100% (complejidad técnica).

Los resultados se presentan en la Tabla 14:

Tabla 14.

Matriz de Decisión.

Módulo	Valor	Deuda Técnica	Prioridad
Evidencias	4	3	7
SEDD	4	4	8
SG	5	5	10
Portal estudiantil	4,6	3	7,6
Inscripciones y matrículas	4,6	3	7,6

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

De acuerdo con los resultados de la matriz de decisión se consideran como prioridad Alta al sistema SG, prioridad Media al Portal Estudiantil y el módulo de Inscripciones y matrículas,

prioridad Baja al sistema de Evidencias. Con base a los datos obtenidos se presenta el análisis de resultados a continuación:

- Sistema de Gestión (SG): Este módulo es fundamental para el inicio del período académico semestral e intersemestral, presenta un valor de criticidad crucial y una deuda técnica moderada. Los resultados señalan que es prioritario invertir en mejorar este módulo y en el mantenimiento.
- Portal estudiantil: El valor de negocio de este módulo es intermedio, sin embargo, presenta un valor bajo de deuda técnica, por lo que las mejoras a aplicar en este módulo pueden ser incrementales.
- Inscripciones y matrículas: Este módulo tiene resultados semejantes al módulo anterior por lo que es candidato a mejoras incrementales.
- Evidencias: Los resultados del análisis de este módulo indica que el valor de negocio es bajo, no por esto deja de ser un módulo útil para los procesos académicos en la institución. También la deuda técnica es baja por lo que la prioridad para trabajar en este módulo es mínima.

Fase 2. Evaluación de la Deuda Técnica

Con base a los resultados obtenidos en la investigación cualitativa a través de las entrevistas aplicadas a los miembros del equipo de desarrollo, el análisis cuantitativo utilizando la herramienta SonarQube, y la identificación del valor del negocio de los módulos desarrollados se realiza la evaluación de la deuda técnica.

Como resultado de las entrevistas la mayoría de los miembros perciben la deuda técnica, ya que reconocen su existencia y el impacto en la eficiencia y calidad del software. Entre las principales causas de la deuda técnica los miembros del equipo de desarrollo de software señalan

a la presión por cumplir con los plazos de entrega, la falta de documentación, falta de pruebas y la falta de revisión del código. También, los desarrolladores indicaron que el impacto de la deuda técnica se da en el desarrollo de sus actividades ya que dificulta implementar nuevas funcionalidades y aumenta el tiempo en el desarrollo.

Así también, como resultado del análisis estático del código, se utilizó la herramienta SonarQube con la cual se evaluó el sistema de gestión donde, se obtuvo un alto valor en el número de bugs, en las vulnerabilidades y la necesidad de aumentar la cobertura de las pruebas.

De igual manera, se realizó la identificación del valor de negocio de los módulos desarrollados con lo cual se logró una visión completa de la situación real de la institución, donde se determinó la necesidad de la implementación de un marco de gestión de la deuda técnica que se adapte a las necesidades del equipo con lo cual se asegure la calidad del software y la eficiencia de los miembros del equipo en el entorno ágil de desarrollo.

Fase 3. Índice de Deuda Técnica Basado en Valor

Para el cálculo del Índice de Deuda Técnica Basado en Valor (IDTV), se necesita el peso relativo del valor del software para priorizar los módulos críticos en los cuales se debe tomar atención en el marco de gestión de la deuda técnica.

Para el cálculo del peso relativo de los módulos se toman en cuenta los siguientes factores clave:

- Frecuencia de uso: Frecuencia con la que se utiliza cada módulo.
- Impacto en el usuario: El número de usuarios que tienen algún tipo de afectación en cada módulo en la eficiencia y calidad de los procesos.
- Funcionalidad y complejidad: Importancia de las funcionalidades y nivel de complejidad técnica de cada módulo.

- Riesgo: El riesgo que tiene asociado cada módulo a algún tipo de fallas o problemas.
- Valor Estratégico: Medida en que el módulo se encuentra alineado a los objetivos estratégicos de la institución.

Los criterios se han ponderado para poder realizar el cálculo con base a la evaluación del equipo de desarrollo, de acuerdo con los siguientes porcentajes:

- Frecuencia de uso: 15%.
- Impacto en el usuario: 25%.
- Funcionalidad y complejidad: 25%.
- Riesgo: 15%.
- Valor Estratégico: 20%.

Para evaluar los módulos se asigna una puntuación de 1 al 50, para cada criterio por cada módulo, los resultados se observan en la Tabla 15:

Tabla 15.

Asignación de puntuación.

Módulo	**Frecuencia de uso**	**Impacto en el usuario**	**Funcionalidad y complejidad**	**Riesgo**	**Valor Estratégico**
Evidencias	45	50	40	35	40
SEDD	35	35	30	25	35
SG	40	50	45	45	50
Portal estudiantil	45	50	25	25	40
Inscripciones y matrículas	35	50	30	25	40

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

El cálculo del valor relativo se observa en la tabla 16.

Tabla 16.

Cálculo del Valor relativo.

Módulo	Frecuencia de uso (15%)	Impacto en el usuario (25%)	Funcionalidad y complejidad (25%)	Riesgo (15%)	Valor Estratégico (20%)	Peso Relativo
Evidencias	6,75	12,50	10,00	5,25	8,00	42,50
SEDD	5,25	8,75	7,50	5,00	7,00	33,50
SG	6,00	12,50	11.25	9,00	10,00	48,75
Portal estudiantil	6,75	12,50	6,25	5,00	8,00	38,50
Inscripciones y matrículas	5,25	12,50	7,50	5,00	8,00	38,25

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Los resultados del cálculo señalan que el módulo SG tiene el mayor valor relativo (48,75), lo que indica que genera el mayor impacto y valor para la institución. Con los valores obtenidos en el análisis de datos se procede con el cálculo del índice IDTV, para este caso el módulo SG posee una deuda técnica de 110 horas de trabajo y un peso relativo de valor del software de 0,49%, la fórmula se aplicaría de la siguiente manera:

IDTV=Deuda Técnica × Peso Relativo del Valor del Software

IDTV = 110 horas × 0,49

IDTV = 53,9

Este resultado indica que este componente se cataloga como crítico por lo que la prioridad para su remediación es alta, ya que no solamente la deuda técnica es elevada, además, el valor del peso relativo por su importancia para el software lo justifica, por lo que reducir la

deuda técnica en este módulo considerado como clave mediante una adecuada gestión, asegurará la eficiencia y calidad de todo el sistema.

Fase 4. Remediación Basada en el Valor.

La remediación de la deuda técnica basada en el valor se la aplica al módulo SG, con base a los datos obtenidos hasta el momento en las diferentes fases el marco propuesto, las acciones planteadas deben cumplir los criterios SMART y se deben enfocar en abordar los elementos de la deuda técnica más críticos, para lo que se debe asignar los recursos necesarios, además, de la revisión periódica para evaluar el progreso de la remediación. Las tareas se plantean a continuación en la Tabla 17:

Tabla 17.

Remediación Basada en el Valor.

Tarea	Objetivo	Medición	Relevancia	Tiempo
Reducción de la Relación de Deuda Técnica (Debt Ratio)	Reducir la Relación de Deuda Técnica del 35% al 20%	Reducción porcentual de la deuda técnica.	Mejora la mantenibilidad del módulo.	6 meses
Disminución del número de Bugs Críticos y Mayores	Reducir los Bugs críticos de 25 a 10 y los mayores de 45 a 20.	Número de Bugs críticos y mayores.	Mejora de la estabilidad y confiabilidad del módulo	4 meses
Mitigar Vulnerabilidades de Seguridad Críticas y Mayores	Reducir las vulnerabilidades de seguridad críticas de 15 a 5 y las mayores de 30 a 10.	Número de vulnerabilidades críticas y mayores corregidas.	Incremento en la seguridad del sistema.	3 meses
Reducción de Code Smells Altos y Medios	Disminuir los Code Smells altos de 100 a 30 y los medios de 120 a 50.	Número de Code Smells altos y medios resueltos.	Mejorará la calidad del código, facilita su comprensión y mantenimiento.	5 meses

Tarea	Objetivo	Medición	Relevancia	Tiempo
Reducción de la Duplicación de Código	Reducir las líneas duplicadas del 30% al 15% y los bloques duplicados de 200 a 80.	Porcentaje de código duplicado eliminado.	Incrementa la claridad y la eficiencia del código, disminuye el riesgo de errores y facilita la implementación de cambios.	4 meses
Documentación y capacitación	Crear y distribuir la documentación actualizada del módulo y realizar reuniones de capacitación para el equipo	Documentación entregada y número de sesiones de capacitación realizadas	Facilitará el mantenimiento y la mejora continua del sistema	2 meses
Mejorar la Cobertura de Pruebas	Aumentar la cobertura total de pruebas que cubran al menos el 80% de las funcionalidades críticas del módulo.	Porcentaje de cobertura de pruebas.	Aumentará la calidad y reducirá el tiempo de implementación de nuevas funcionalidades.	5 meses

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Fase 5. Revisión y Ajuste Continuo.

Para la revisión de la remediación de la deuda técnica se debe establecer un ciclo continuo de monitoreo que incluya reuniones de revisión donde se evalúe la documentación, el impacto, y se realicen ajustes en caso de ser necesario. Las actividades por realizar en esta fase se las describe en la Tabla 18:

Tabla 18.

Actividades de Revisión y Ajuste Continuo.

Actividad	Objetivo	Medición	Tareas	Relevancia	Tiempo
Monitoreo Continuo	Implementar un sistema de monitoreo continuo.	Porcentaje del avance de la implementación del sistema de monitoreo	Implementar herramientas de monitoreo con alertas y reportes automáticos	Detectar problemas y avances en tiempo real.	2 meses
Evaluación de Impacto	Realizar evaluaciones continuas del impacto de las tareas de remediación.	Cantidad de evaluaciones realizadas y resultados documentados.	Definir métricas clave de evaluación. Realizar análisis luego de la remediación.	Entender el efecto de las acciones de remediación y ajustar estrategias en el caso de ser necesario.	Cada 3 meses
Reuniones de Revisión	Coordinar reuniones mensuales para revisar el progreso de la remediación de la deuda técnica.	Cantidad de reuniones realizadas y acciones tomadas.	Programar reuniones. Revisar y discutir el progreso y las actividades.	Facilitar la comunicación y coordinación del equipo de desarrollo.	Cada 1 mes
Documentar los cambios	Crear documentación de los cambios y mejoras realizadas en la fase.	Cantidad de cambios documentados.	Actualizar continuamente la documentación.	Proporcionar un registro de los cambios para referencias y auditorías.	Cada 1 mes
Ajuste de Estrategias	Realizar el ajuste las estrategias de remediación con base a los resultados de las evaluaciones de impacto y las reuniones de revisión.	Número de ajustes realizados y su efectividad.	Analizar los resultados de las evaluaciones y reuniones. Proponer y aplicar ajustes estratégicos.	Asegurar que las estrategias de remediación sean efectivas y adaptables a las necesidades del equipo.	Cada 3 meses

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Evaluación del modelo propuesto

Como parte de la fase 4 de la implementación de la propuesta y de acuerdo con lo planteado en el objetivo cuatro de la investigación, se analizan los resultados de la aplicación del marco con la evaluación de métricas y el nivel de adopción y satisfacción del equipo con el marco.

El diseño del marco se lo ha planteado con la finalidad de gestionar la deuda técnica en entornos de desarrollo ágiles, lo que proporcionará un software estable y de calidad mediante la aplicación de mejores procesos y técnicas para desarrollar el software y administrar la deuda técnica.

Para la validación del marco de gestión diseñado para entornos ágiles se aplicarán las mismas herramientas usadas inicialmente en el equipo de desarrollo que proporcionó las métricas y datos antes de aplicar el marco. Se seleccionó el mismo equipo por la comprensión y familiaridad de la investigación, además, de su conocimiento inicial del estado de la deuda técnica y los problemas específicos relacionados a esta.

Con la validación se demuestra que el marco propuesto es replicable para los equipos que trabajan con modelos ágiles ya que además de mitigar la deuda técnica optimizan la eficiencia en los procesos de desarrollo.

Para validar los datos se utilizó la herramienta de análisis estático SonarQube la que se aplicó en el módulo Sistema de Gestión de distributivos (SG), la cual arrojó los siguientes resultados:

Nombre del Proyecto: Módulo SG (Sistema de Gestión).

Fecha del Informe: 13 de mayo de 2024

Versión del Análisis: 2.0

1. Debt Ratio (Relación de Deuda Técnica)

Debt Ratio: 5%

Deuda Técnica Total: 50 horas

La relación de deuda técnica señala que un 5% del esfuerzo total del módulo estaría dedicado a resolver la deuda técnica existente, disminuyendo significativamente con relación al valor inicial.

2. Bugs (Errores)

Total de Bugs: 35

Críticos: 5

Mayores: 13

Menores: 17

Los errores han disminuido significativamente con pocos errores críticos y un bajo número de errores mayores y menores que necesitan atención.

3. Vulnerabilidades de Seguridad

Total de Vulnerabilidades: 27

Críticas: 5

Mayores: 10

Menores: 12

Se observa la disminución de las vulnerabilidades de seguridad críticas, así como también, de la vulnerabilidades mayores y menores que deben abordarse.

4. Code Smells

Total de Code Smells: 40

Altos: 5

Medios: 15

Bajos: 20

La cantidad de code smells es mucho menor a la inicial, con lo que se mejora la mantenibilidad y legibilidad del código.

5. Cobertura de Pruebas

Cobertura Total: 85%

Cobertura de Líneas: 80%

Cobertura de Ramas: 70%

Se observa un aumento considerable en la cobertura de pruebas, con un alto porcentaje de líneas y ramas del código cubiertas por la aplicación de pruebas unitarias.

6. Duplicación de Código

Líneas Duplicadas: 200 (5%)

Bloques Duplicados: 15

La duplicación de código se ha reducido, lo que señala que en la actualidad se posee un código más limpio y fácil de mantener.

Los datos obtenidos demuestran una mejora sustancial en la calidad del código con lo cual se ha disminuido la complejidad técnica, al mismo tiempo que, se redujeron las duplicaciones, lo que beneficia en el aumento de la productividad, la satisfacción del equipo y la reducción de la deuda técnica. Los datos comparativos de la deuda técnica antes y después de la implementación del marco se muestran en la Tabla 19.

Tabla 19.

Datos comparativos de la Deuda Técnica.

Estado	**Debt Ratio (%)**	**Deuda Técnica (horas)**
Antes de la Gestión	35	110

Estado	Debt Ratio (%)	Deuda Técnica (horas)
Después de la Gestión	5	50

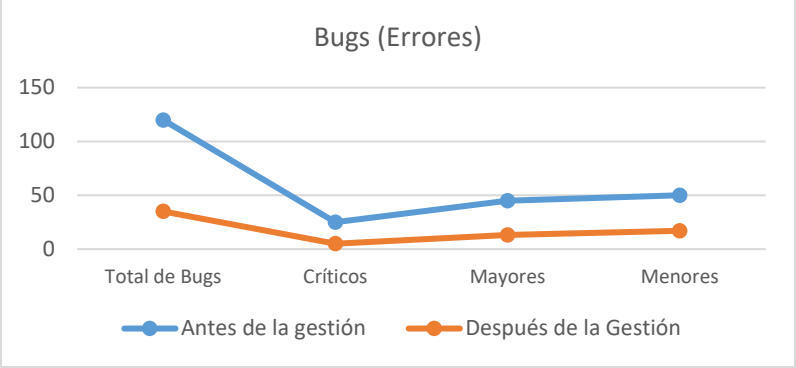
Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

La comparación de errores se observa en la siguiente Figura.

Figura 3.

Comparación de Bugs



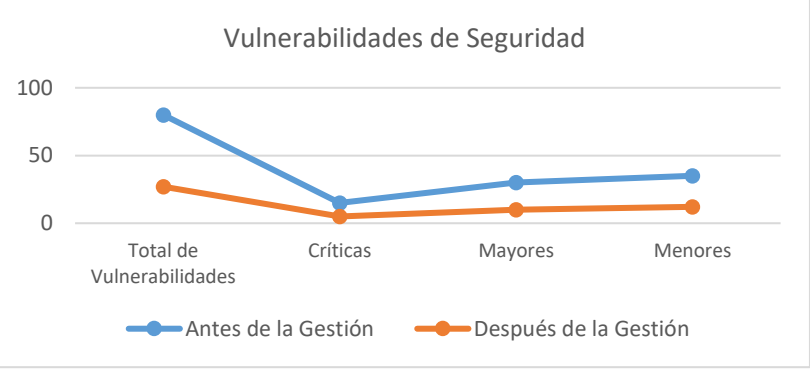
Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

En la siguiente Figura se observa la comparación de las vulnerabilidades de seguridad.

Figura 4.

Vulnerabilidades de Seguridad



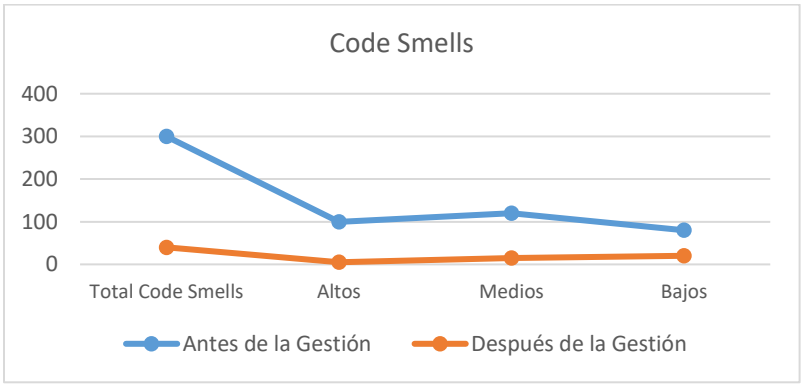
Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

La Figura comparativa de los Code Smells se presenta a continuación.

Figura 5.

Code Smells



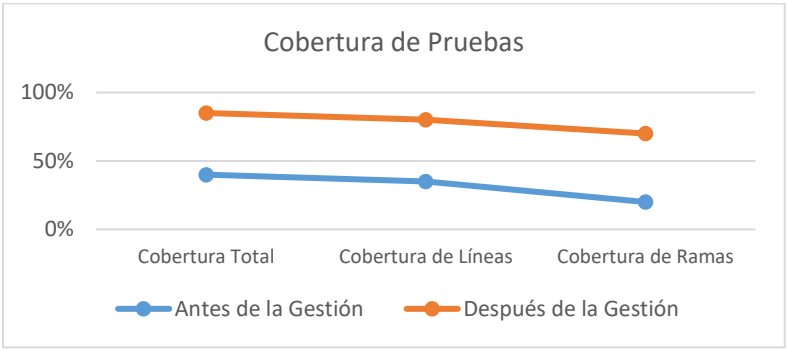
Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Los datos comparativos de la cobertura de pruebas antes y después de la gestión de la deuda técnica se observan en la siguiente Figura.

Figura 6.

Cobertura de Pruebas



Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

A continuación, se presentan, los datos de la duplicación de código antes y después de la implementación del marco se muestran en la Tabla 20.

Tabla 20.

Duplicación de Código.

Estado	Líneas Duplicadas (%)	Bloques Duplicados
Antes de la Gestión	30	200
Después de la Gestión	5	15

Elaborado por: Lozada Torres Edwin Fabricio

Fuente: Marco de Gestión de Deuda Técnica Basado en el Valor (DTV)

Para complementar la validación luego de la implementación del marco propuesto se aplicó una entrevista al director del departamento de software que también cumple las funciones de jefe del equipo de desarrollo, esta información se presenta a continuación.

Entrevista aplicada al jefe del equipo

El objetivo de la entrevista es validar la efectividad de la aplicación del marco de gestión de deuda técnica basado en el valor el que se aplicó en el módulo Sistema de Gestión, al mismo tiempo se analiza el impacto de las métricas aplicadas y se identifican posibles mejoras del proceso.

Nombre del entrevistado: Ing. Henry Baño

Cargo del entrevistado: Jefe del equipo de desarrollo

Título: Ingeniero en Sistemas e Informática

Experiencia en el desarrollo: 15 años

P1. ¿Puedes describir el contexto del módulo en el que se trabajó para obtener los datos?

Respuesta: El sistema para la gestión de Distributivo Docente SG es un módulo crítico en la universidad debido a que es el que permite gestionar la asignación de las asignaturas y las horas a los docentes al inicio de cada período académico. La criticidad de esta aplicación es que gestiona los distributivos en todas las carreras y se debe tomar en cuenta que actualmente la institución cuenta con carreras en diferentes modalidades como son presencial, semi presencial, híbrida y en línea por lo que la asignación de la carga horaria en lo que tiene que ver con las

horas en contacto con el docente, experimentales y autónomas son diferentes para cada modalidad, tomando en cuenta que hay modalidades en las cuales se deben tomar en cuenta horas síncronas y asíncronas. Uno de los principales problemas de este módulo es que se encuentra en constante mantenimiento ya que cada inicio de período existe cambios solicitados por la Dirección Académica, así como por la Secretaría General.

P2. ¿Cuáles eran las expectativas iniciales al implementar el marco de gestión de deuda técnica basado en el valor en el módulo SG?

Respuesta: Las expectativas fueron muy altas en vista a que en el equipo de desarrollo estábamos conscientes de los problemas que hemos tenido siempre en el desarrollo, si bien no la catalogábamos como deuda técnica sabíamos que afectaba en la calidad del producto y que tarde o temprano teníamos que abordarlas.

P3. ¿Me puedes describir la forma en la que se realizó la implementación del marco?

Respuesta: En ocasiones anteriores no se ha documentado las actividades que se desarrolla en el equipo, tampoco hemos utilizado ningún tipo de herramientas para analizar el código creado. Con los resultados obtenidos con la herramienta SonarQube y el conocimiento de la deuda técnica y sus consecuencias, nos dimos cuenta de la necesidad de abordar estos problemas, por lo que se dio todas las facilidades para la implementación del marco, siempre pensando en que gestionar adecuadamente estos problemas va a traer muchos beneficios para el equipo.

P4. ¿Tuvieron algún tipo de dificultad durante la implementación del marco?

Respuesta: la principal dificultad encontrada fue el tiempo, debido a que todos los miembros del equipo tienen asignadas tareas a desarrollar, que las deben cumplir de acuerdo con un cronograma previamente establecido. El utilizar herramientas de análisis de código, analizar

los resultados y la implementación de las actividades del marco ocupaba tiempo que podría utilizarse en el desarrollo de las tareas asignadas a cada desarrollador.

P5. ¿Cómo se resolvieron estos problemas?

Respuesta: Fue clave en la implementación del marco el trabajar con un único modulo, si bien este se encuentra en constante mantenimiento por los requerimientos cambiantes de los departamentos de la universidad, no interfirió con el resto de las tareas asignadas, por lo que este problema fue resuelto de manera planificada.

P6. ¿Qué aspectos del marco considera que fueron más relevantes para este proyecto?

Respuesta: Lo principal que el equipo comprenda lo que es la deuda técnica y que de aquí en adelante se preocupe por su control, en segundo lugar, el trabajar con base a un marco que nos permite identificar problemas y abordarlas a tiempo y lo más importante el obtener productos libres de errores, con un código más eficiente y controlando las vulnerabilidades.

P7. ¿Cómo se integró el uso de herramientas de análisis estático de código en el proceso de gestión de deuda técnica?

Respuesta: La integración de la herramienta no tuvo ningún tipo de dificultad, más bien las mejoras obtenidas en el código se observaron inmediatamente por lo que dentro del equipo se decidió que de ahora en adelante se va a utilizar esta herramienta para evaluar el código de todos los módulos, además, el impacto en la productividad y moral del desarrollador del módulo se notó inmediatamente por lo que se superaron las expectativas del uso de esta herramienta.

P8. ¿Qué lecciones se aprendieron durante el proceso de implementación de la gestión de la deuda técnica?

Respuesta: Varias lecciones, una de ellas el dimensionar los problemas a futuro si no se gestiona adecuadamente los problemas relacionados a la deuda técnica, la segunda el hecho de

que destinar tiempo a detectar la deuda técnica no es una pérdida de tiempo, ya que, los resultados son notorios en favor del equipo y el software desarrollado. Una de las principales lecciones es que la mayoría de las ocasiones el equipo se hizo de la vista gorda ante este tipo de problemas, aunque sabíamos que existían no queríamos tomarlo en cuenta, en cambio de ahora en adelante vamos a poner en práctica las recomendaciones del marco. Finalmente, el uso de pruebas documentadas antes de entregar el producto es una lección aprendida que obligatoriamente la vamos a utilizar de aquí en adelante.

P9. ¿Qué recomendaciones haría para aquellos proyectos que van a implementar el marco de gestión de deuda técnica basado en el valor?

Respuesta: Primero aceptar la existencia de la deuda técnica, segundo estar conscientes de que se la debe controlar, una vez que se admite su problema la recomendación principal es confiar en las tareas de las distintas fases del marco y pensar que implementarlas no es una pérdida de tiempo, al contrario va a contribuir en obtener un software estable y los usuarios que son los beneficiarios van a notar inmediatamente el cambio en el producto, además del ahorro de recursos en trabajo extra que se debe realizar para solucionar los problemas.

P10. ¿Cómo planea seguir utilizando el marco de gestión de la deuda técnica en los proyectos a largo plazo?

Respuesta: Utilizar las actividades del marco para realizar un monitoreo continuo de la deuda técnica, además de la aplicación de las fases cada vez que se detecte algún problema que tenga relación con la deuda técnica. Estas actividades las vamos a implementar en todos los módulos en los que nos encontramos trabajando actualmente y en aquellos proyectos que tengamos que realizar a futuro.

Análisis e interpretación

En la investigación inicial se determinó la existencia de la deuda técnica y las implicaciones en la planificación del desarrollo, lo que se derivó en la adopción de un marco que permita gestionar de una manera planificada la deuda técnica.

En la entrevista realizada una vez que se implementó el marco de gestión de deuda técnica basado en el valor se reveló que los procesos internos mejoraron significativamente lo que se evidenció en la reducción de errores y vulnerabilidades. También, se logró obtener un código más limpio y la eficiencia en la cobertura de pruebas, con lo que aseguro un producto de calidad y la optimización en la agilidad del equipo de desarrollo lo que consolida la efectividad del marco de gestión implementado.

En conclusión, con la implementación del marco de gestión de deuda técnica basado en el valor se obtuvieron los siguientes beneficios:

- El código desarrollado es de mejor calidad, controlando las duplicaciones, utilizando comentarios entendibles y logrando una estructura más organizada en la codificación.
- Se aumento la cobertura de las pruebas de una manera significativa con lo que se detectan errores a tiempo.
- Se redujo considerablemente el número de bugs y el tiempo de resolución cuando se los detecta.
- La agilidad en el desarrollo aumentó, gracias a la eficiencia de la calidad del código y la facilidad del mantenimiento.
- Los módulos son más estables y eficientes optimizando la mantenibilidad.
- El equipo de desarrollo se siente más satisfecho y eficiente para solucionar problemas y puede desarrollar nuevas funcionalidades.

Los beneficios obtenidos al implementar el marco de gestión de la deuda técnica basado en el valor son positivos, ya que su impacto se nota en la calidad del software y la agilidad en el equipo de desarrollo.

Comparación del estado inicial y estado posterior

Se presenta una comparación estructurada entre el estado inicial y el estado posterior a la aplicación del Marco de Gestión de la Deuda Técnica Basado en el Valor, para lo que se utiliza los datos obtenidos de la herramienta SonarQube y los resultados de la entrevista aplicada al jefe del equipo de desarrollo con lo que se demuestra empíricamente la efectividad del modelo propuesto y se fortalece la validez de los resultados del estudio doctoral.

Para la comparación se evaluó el impacto del marco de gestión de la deuda técnica basado en el valor con la aplicación de métricas objetivas obtenidas por la herramienta SonarQube con la que se realizó inicialmente un diagnóstico previo a la implementación del marco y la medición realizada luego de la aplicación sistemática del modelo después de un período definido.

Con la finalidad de garantizar el rigor metodológico las dos mediciones cumplen las condiciones siguientes:

- Mismo repositorio de código fuente.
- Misma versión del sistema analizado.
- Misma configuración de SonarQube.
- Mismo lenguaje de programación.

Con estas acciones se aseguró que no exista factores externos que influyan en los resultados, con lo que se verificó empíricamente el impacto del marco propuesto, demostrando mejoras alineadas a los objetivos planteados, además. Las métricas obtenidas de la herramienta

SonarQube constituyen una evidencia sólida de la efectividad del modelo aplicado que al integrarse con el resultado cualitativo de las entrevistas fortalece la coherencia, el rigor y la contribución de la investigación.

Interpretación de los hallazgos respecto al problema de investigación

Los resultados obtenidos al realizar la comparación de los datos antes y después de aplicar el modelo determina que el problema de la gestión de la deuda técnica en modelos ágiles no es solo es observable, sino que también es controlable con la aplicación de un marco específico de gestión basado en el valor. Se encontró una reducción considerable de la deuda técnica de acuerdo con los resultados cualitativos y los datos obtenidos con la herramienta SonarQube, disminuyendo los problemas críticos

Los resultados se relacionan con los objetivos específicos con lo que se validad empíricamente la efectividad del marco propuesto al identificar, evaluar y priorizar la deuda técnica, además, se logró una gestión estructurada de la deuda mejorando la capacidad del equipo de desarrollo ágil para entregar valor sostenible.

4. Conclusiones y Recomendaciones

Conclusiones

La Deuda Técnica a presentado desafíos constantes en los desarrolladores de software ágiles, por lo que se propuso una investigación con enfoque mixto para comprender la percepción de los estos y los factores que contribuyen para su surgimiento. Los hallazgos obtenidos en este trabajo contribuyen al avance del conocimiento de la deuda técnica y destacan aspectos que aportan al desarrollo de un enfoque con recomendaciones prácticas para una adecuada gestión. Las conclusiones resultado de la investigación presentan un significativo aporte para los equipos de desarrollo que se encuentran en constante búsqueda de mejorar la calidad en sus productos.

La principal causa de la deuda técnica en el desarrollo ágil es la presión por la entrega del producto por parte del cliente, esta premura en el desarrollo se deriva en que el equipo no ponga en práctica todos los principios y buenas prácticas de la ingeniería del software cumpliendo a medias fases del ciclo de desarrollo establecidos y enfocándose únicamente en la obtención del producto dejando de lado actividades de modelamiento de software, arquitectura y pruebas que finalmente afectan en la calidad del software.

El estado actual de la gestión de la deuda en entornos ágiles gira en torno a la generación de la deuda técnica de manera involuntaria o deliberada por parte del equipo de desarrollo, con la finalidad de cumplir con los objetivos propuestos inicialmente en los proyectos. Si bien tarde o temprano se dan cuenta de la presencia de la deuda técnica, la mayoría de las veces no se preocupan por pagarla a tiempo acrecentando los problemas derivados de la deuda incurriendo en sobre trabajo y costos extras que afectan a los proyectos.

Se desarrollo un marco de gestión de deuda técnica basado en el valor con base a un caso de estudio realizado en un equipo de desarrollo de una institución de educación superior, para lo cual, se utilizó un enfoque cualitativo que, junto al análisis cuantitativo de los datos obtenidos en las métricas aplicadas y la identificación del valor del negocio, reveló información clave que contribuyó al desarrollo del marco con el aporte de prácticas sistemáticas y estructuradas para la gestión de la deuda. El análisis realizado de acuerdo con las fases del marco identifico los módulos que requieren mayor atención por su alto valor relativo y cantidad de deuda técnica, priorizando los componentes críticos debido a los altos índices de deuda técnica. Los resultados obtenidos demuestran que se alcanzó el objetivo planteado, ya que con la aplicación del marco se evidenció mejoras técnicas y un mayor control de la deuda técnica de acuerdo con los resultados obtenidos cualitativamente y de los datos de la medición aplicada.

La implementación del marco ha demostrado la importancia de utilizar un enfoque sistemático para identificar y mitigar la deuda técnica con lo cual se logró una notable reducción de esta, obteniendo un código eficiente y mejorando el impacto en la calidad del software y la productividad del equipo. Para la remediación de la deuda técnica se definieron objetivos con los cuales se obtuvo un monitoreo eficaz de la deuda, se aseguró la estabilidad y el rendimiento del software a largo plazo y se contribuyó con la mejora en la satisfacción de los usuarios. Además, se establecieron ciclos de revisiones continuos mediante reuniones de revisión y evaluaciones que miden el impacto de la remediación planteada en el marco con la finalidad de realizar ajustes en la remediación de la deuda técnica en caso de ser necesario. Los resultados cualitativos obtenidos demuestran que el equipo logró percibir y priorizar la deuda técnica pasando de prácticas implícitas a un enfoque explicito, que se apoya con métricas técnicas en base a los criterios de valor del negocio.

Si bien, la investigación se centró en aclarar los temas relacionados a la gestión de la deuda técnica, esta es un área dinámica y en evolución constante, es por lo que desde una perspectiva integradora el Modelo de Gestión de la Deuda Técnica Basado en el Valor se presenta conceptualmente consistente con una metodología sólida y aplicable en contextos reales. La investigación es un gran aporte para la teoría y la práctica en el campo de la ingeniería de Software ya que, responde a los objetivos planteados aportando a los procesos de agilidad, la toma de decisiones y la entrega de valor sostenible, poniendo de manifiesto la efectividad del marco.

Recomendaciones

Incorporar el proceso de detección de la deuda técnica como un elemento explicito en el proceso de desarrollo de software, utilizando los criterios técnicos descritos en el marco y aplicando el enfoque basado en el valor.

Aplicar herramientas para el análisis estático de código no solamente para diagnosticar el estado de la codificación, sino como soporte para adoptar estrategias que permitan mitigar la deuda técnica, junto con políticas de asignación de roles y recursos para su adecuada gestión.

Utilizar el marco propuesto como una guía que se adapte a los diferentes contextos del desarrollo y que permita fomentar una cultura de decisiones al combinar métricas técnicas, agilidad y el enfoque basado en el valor, de acuerdo con el modelo organizacional.

El estudio abre oportunidades para futuras investigaciones, por lo que se sugiere que se evalúen la aplicación del marco de gestión de deuda técnica basado en el valor en diferentes contextos y ambientes de desarrollo. También, se recomienda analizar el marco con estudios aplicados en diferentes organizaciones donde se pueda evaluar los factores humanos e influencia cultural que puedan afectar en la efectividad del enfoque planteado.

Referencias

Abdullah Alomar, E., Ivanov, A., Kurbatova, Z., Golubev, Y., Wiem Mkaouer, M., Ouni, A., . . . Thakur, A. (2023). Anti Copy Paster: Extracting Code Duplicates As Soon As They Are Introduced in the IDE. *37th IEEE/ACM International Conference on Automated Software Engineering*. https://doi.org/10.1145/3551349.3559537

Abdullah AlOmar, E., Wiem Mkaouer, M., Newman, C., y Ouni, A. (2021). On preserving the behavior in software refactoring: A systematic mapping study. *Information and Software Technology, 140*. https://doi.org/10.1016/j.infsof.2021.106675

Al Mamun, M., Martini, A., y Staron, M. (2019). Evolution of technical debt: An exploratory study. *CEUR Workshop Proceedings*, 87-102.

Al-Baik, O., Abu Alhija, M., Abdeljaber, H., y Ovais Ahmad, M. (2024). Organizational debt—Roadblock to agility in software engineering: Exploring an emerging concept and future research for software excellence. *PLOS ONE, 19*(11). https://doi.org/10.1371/journal.pone.0308183

Albuquerque, D., Guimaraes, E., Simone Tonin, G., y Perkusich, A. (2022). Perceptions of Technical Debt and its Management Activities - A Survey of Software Practitioners. *XXXVI Brazilian Symposium on Software Engineering*. https://doi.org/10.1145/3555228.3555237

Albuquerque, D., Guimarães, E., Tonin, G., Rodriguez, P., Perkusich, M., Almeida, H., . . . Chagas, F. (2023). Managing Technical Debt Using Intelligent Techniques - A Systematic Mapping Study. *IEEE Transactions on Software Engineering, 49*(4), 2202-2220. https://doi.org/10.1109/TSE.2022.3214764

Alfayez, R., Alwehaibi, W., Winn, R., Venson, E., y Boehm, B. (2020). A Systematic Literature Review of Technical Debt Prioritization. *International Conference on Technical Debt*, 1-10. https://doi.org/10.1145/3387906.3388630

Alhefdhi, A., Khanh Dam, H., y Ghose, A. (2024). Towards automating self-admitted technical debt repayment. *Information and Software Technology, 167*. https://doi.org/10.1016/j.infsof.2023.107376

Alizadeh, V., Fehri, H., y Kessentini, M. (2019). Less is More: From Multi-objective to Mono-objective Refactoring via Developer's Knowledge Extraction. *IEEE 19th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 181-192. https://doi.org/10.1109/SCAM.2019.00029

AlOmar, E., Mkaouer, M., Ouni, A., y Kessentini, M. (2019). On the Impact of Refactoring on the Relationship between Quality Attributes and Design Metrics. *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1-11. https://doi.org/10.1109/ESEM.2019.8870177

Ampatzoglou, A., Chatzigeorgiou, A., Arvanitou, E. M., y Bibi, S. (2022). SDK4ED: A platform for technical debt management. *Software: Practice and Experience, 52*(8), 1879-1902. https://doi.org/10.1002/spe.3093

Arvanitou,, E.-M., Ampatzoglou, A., Bibi, S., Chatzigeorgiou,, A., y Stamelos, I. (2019). Monitoring Technical Debt in an Industrial Setting. *EASE '19*, 15-17.

Avgeriou, P., Ozkaya, I., Chatzigeorgiou, A., Ciolkowski, M., Ernst, N., Koontz, R., . . . Shull, F. (2023). Technical Debt Management: The Road Ahead for Successful Software Delivery. *IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, 15-30. https://doi.org/10.1109/ICSE-FoSE59343.2023.00007

Avgeriou, P., Ozkaya, I., Chatzigeorgiou, A., Ciolkowski, M., Ernst, N., Koontz, R., . . . Shull, F. (2024). Technical Debt Management: The Road Ahead for Successful Software Delivery. *International Conference on Software Engineering: Future of Software Engineering*. https://doi.org/10.1109/ICSE-FoSE59343.2023.00007

Avgeriou, P., Taibi, D., Ampatzoglou, A., Arcelli Fontana, F., y Besker, T. (2020). An Overview and Comparison of Technical Debt Measurement Tools. *IEEE Software*, 61-71.

Awad Elneel, D., Sahli Fakharudin, A., Musa Ahmed, E., Kahtan, H., y Abdulgabber Abdulhak, M. (2022). Stakeholder Identification Overview and Challenges in Requirements Engineering Prospective. *2nd International Conference on Computing and Information Technology (ICCIT)*. https://doi.org/10.1109/ICCIT52419.2022.9711653

Ayala González, L., y García Lizarazo, D. (2018). La deuda técnica en proyectos de desarrollo de software en el Ecuador. *Revista de Investigación y Desarrollo Científico" de la Universidad de Guayaquil*.

Behutiye , W. N., Rodríguez, P., Oivo , M., y Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology, 82*, 139-158. https://doi.org/10.1016/j.infsof.2016.10.004

Berenguer, C., Borges, A., Freire, S., Rios, N., Ramač, R., Taušan, N., . . . Castellanos, C. (2023). Investigating the Relationship between Technical Debt Management and Software Development Issues. *Journal of Software Engineering Research and Development, 11*(3). https://doi.org/10.5753/jserd.2023.2581

Berenguer, C., Borges, A., Freire, S., Rios, N., y Tausan, N. (2021). Technical Debt is not Only about Code and We Need to be Aware about It. *Conference: 20th Brazilian Symposium on Software Quality.*, 1-12. https://doi.org/10.1145/3493244.3493285

Berntsson Svensson, R., Gorschek, T., Bengtsson, P.-O., y Widerberg, J. (2019). BAM - Backlog Assessment Method. *20th International Conference, XP 2019 Montréal, QC, Canada*, 53-68. https://doi.org/10.1007/978-3-030-19034-7_4

Besker, T., Ghanbari, H., Martini, A., y Bosch, J. (2020). The influence of Technical Debt on software developer morale. *The Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2020.110586

Besker, T., Martini, A., y Bosch, J. (2019). Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. *The Journal of Systems and Software*(156), 41–61. https://doi.org/10.1016/j.jss.2019.06.004

Besker, T., Martini, A., y Bosch, J. (2019). Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. *The Journal of Systems and Software*, 41-61.

Bi, F., Vogel-Heuser, B., Huang, Z., y Ocker, F. (2023). Characteristics, causes, and consequences of technical debt in the automation domain. *The Journal of Systems & Software, 204*. https://doi.org/10.1016/j.jss.2023.111725

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., . . . Zazworka, N. (2010). Managing technical debt in software-reliant systems. *Conference: Proceedings of the Workshop on Future of Software Engineering Research, FoSER 2010, at the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. https://doi.org/10.1145/1882362.1882373

Cairo, A., Carneiro, G., y Monteiro, M. (2018). The Impact of Code Smells on Software Bugs: A Systematic Literature Review. *Information, 9*(273). https://doi.org/doi:10.3390/info9110273

Ciancarini, P., y Russo, D. (s.f.). The Strategic Technical Debt Management Model: An Empirical Proposal. *IFIP Advances in Information and Communication Technology, 582*. https://doi.org/10.1007/978-3-030-47240-5_13

Conejero, J., Rodríguez, R., Hernández, J., Clemente, P., Ortíz, C., Jurado, E., y Sánchez, F. (2018). Early evaluation of technical debt impact on maintainability. *Journal of Systems and Software, 142*. https://doi.org/10.1016/j.jss.2018.04.035

Cortellessa, V., Di Pompeo, D., Stoico, V., y Tucci, M. (2023). Many-objective optimization of non-functional attributes based on refactoring of software models. *Information and Software Technology*. https://doi.org/10.1016/j.infsof.2023.107159

Cortellessa, V., y Di Pompeo, D. (2021). Analyzing the sensitivity of multi-objective software architecture refactoring to configuration characteristics. *Information and Software Technology*. https://doi.org/10.1016/j.infsof.2021.106568

Cucalón-Gaibor, J., y Sandoval Gutierrez, J. (2023). Deudas técnicas en el sistema transaccional. *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica, 12*(2). https://doi.org/10000-0002-5578-4389

Daniel Alejandro, G. (2021). *SIRA: An energy-efficiency and tecnical debt oriented process for software architecture reconstruction.* Universidad Politécnica de Madrid.

de Freitas Farias, M., de Mendonça Neto, M., Kalinowski, M., y Oliveira Spínola, R. (2020). Identifying Self-Admitted Technical Debt through Code Comment Analysis with a

Contextualized Vocabulary. *Information and Software Technology*. https://doi.org/10.1016/j.infsof.2020.106270

Desai, V., y Jariwala, V. (2022). Comprehensive Empirical Study of Static Code Analysis Tools for C Language. *International Journal of Intelligent Systems and Applications in Engineering, 10*(4), 695–700. https://ijisae.org/index.php/IJISAE/article/view/2342

Di Pompeo, D., y Tucci, M. (2022). Search Budget in Multi-Objective Refactoring optimization: a Model-Based Empirical Study. *48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 406-413. https://doi.org/10.1109/SEAA56994.2022.00070

Di Pompeo, D., y Tucci, M. (2023). Multi-objective Software Architecture Refactoring driven by Quality Attributes. *IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, 175-178. https://doi.org/10.1109/ICSA-C57050.2023.00046

Digkas, G., Ampatzoglou, A., Chatzigeorgiou, A., Avgeriou, P., Matei, O., y Heb, R. (2020). The Risk of Generating Technical Debt Interest: A Case Study. *SN Computer Science*, 1-12.

Digkas, G., Chatzigeorgiou, A., Apostolos, A., y Avgeriou, P. (2022). Can Clean New Code Reduce Technical Debt Density? *IEEE Transactions on Software Engineering*, 1705-1721. https://doi.org/10.1109/TSE.2020.3032557

Dornauer, B., Felderer, M., Weinzerl, J., Racasan, M.-C., y Hess, M. (2023). SoHist: A Tool for Managing Technical Debt through Retro Perspective Code Analysis. *EASE '23: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, 184-187. https://doi.org/10.1145/3593434.3593460

Ebert, C., Cain, J., Antoniol, G., Counsell, S., y Laplante, P. (2016). Cyclomatic Complexity. *IEEE Software, 33*(6), 27-29. https://doi.org/10.1109/MS.2016.147

Esfandiari, S., y Sami, A. (2023). An Exploratory Study of the Relationship between SATD and Other Software Development Activities. *13th International Conference on Computer and Knowledge Engineering*. https://doi.org/10.1109/ICCKE60553.2023.10326279

Fernández-Sánchez, C., y Garbajosa, J. (2017). Identification and analysis of the elements required to manage technical debt by means of a systematic mapping study. *The Journal of Systems and Software*(124), 22–38. https://doi.org/10.1016/j.jss.2016.10.018

Figalist, I., Elsner, C., Bosch, J., y Holmstrom Olsson, H. (2019). Scaling Agile Beyond Organizational Boundaries: Coordination Challenges in Software Ecosystems. *20th International Conference, XP 2019 Montréal, QC, Canada*, 189-206. https://doi.org/10.1007/978-3-030-19034-7_12

Floridi, L. (2013). Información y Filosofía. *Revista de Filosofía, 49*, 127-142. https://doi.org/10.12795/themata.2014.i49.07

Freire, S. (2021). Cita formato APA: Freire et al., (2021). How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects?. *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).* , 605-609. https://doi.org/10.1109/SANER50967.2021.00074

Freire, S., Rios, N., Mendonça, M., y Izurieta, C. (2020). Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items. *Evaluation and Assessment in Software Engineering.*, 605-609. https://doi.org/10.1109/SANER50967.2021.00074.

Freire, S., Rios, N., Pérez, B., Torres, D., Mendonça, M., Izurieta, C., . . . Spínola, R. (2021). How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects? *IEEE International Conference on Software Analysis, Evolution and Reengineering* , 605-609. https://doi.org/10.1109/SANER50967.2021.00074

Freire, S., Rios, N., Pérez, B., Torres, D., Mendonça, M., Izurieta, C., . . . Spínola, R. (s.f.). How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects?. *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).*, 605-609. https://doi.org/10.1109/SANER50967.2021.00074

Fu, L., Liang, P., Rasheed, Z., Li, Z., Tahir, A., y Han, X. (2022). Potential Technical Debt and Its Resolution in Code Reviews: An Exploratory Study of the OpenStack and Qt Communities. *International Symposium on Empirical Software Engineering and Measurement*, 216-226. https://doi.org/10.1145/3544902.3546253

Fungprasertkul,, S., Bahsoon, R., y Kazman, R. (2024). Technical Debt Monitoring Decision Making with Skin in the Game. *ACM Transactions on Software Engineering and Methodology, 33*(7), 1-27. https://doi.org/10.1145/3664805

Garcia, A., Silva da Silva, T., y Selbach Silveira, M. (2019). Artifact-Facilitated Communication in Agile User-Centered Design. *20th International Conference, XP 2019 Montréal, QC, Canada*, 102-118. https://doi.org/10.1007/978-3-030-19034-7_7

González Núñez, V. (2021). *Gamificación en entornos de producción de la industria del software para favorecer los indicadores de calidad y reducir la deuda técnica.* Universidad de Valladolid.

Guamán, D., Quezada-Sarmiento, P., Barba-Guaman, L., y Enciso, L. (2017). Use of SQALE and tools for analysis and identification of code technical debt through static analysis. *Iberian Conference on Information Systems and Technologies (CISTI)*, 1-7.

Haertel, C., Pohl, M., Staegemann, D., y Turowski, K. (2022). A Method for Comparing and Selecting Static Code Analysis Tools in Software Development Projects. *Springer Nature*. https://doi.org/10.21203/rs.3.rs-1546855/v1

Hoyos Rentería, J. (2021). *A method to identify toggle debt in a project from its source code.* Universidad Nacional de Colombia.

Hudgikar, S. (2021). *Research methods for engineers.* Kripa-Drishti Publications. https://doi.org/978-93-90847-07-5

Jiménez-Reyes, J., Naranjo-Vélez, C., y Muñoz-Vásconez, D. (2020). Gestión de la deuda técnica en modelos de desarrollo de software ágiles: un estudio en Ecuador. *Revista de Ingeniería" de la Universidad de Cuenca*.

Kashiwa, Y., Nishikawa, R., Kamei, Y., Kondo, M., Shihab, E., Sato, R., y Ubayashi, N. (2022). An Empirical Study on Self-Admitted Technical Debt in Modern Code Review. *Information and Software Technology, 146*. https://doi.org/10.1016/j.infsof.2022.106855

Kiv, S., Heng, S., Kolp, M., y Wautelet, Y. (2019). Agile Methods Knowledge Representation for Systematic Practices Adoption. *20th International Conference, XP 2019*, 19-34. https://doi.org/10.1007/978-3-030-19034-7_2

Kleinwaks, H., Batchelor, A., y Bradley, T. (2023). Predicting the Dynamics of Earned Value Creation in the Presence of Technical Debt. *IEEE Access*, 1-16. https://doi.org/10.1109/ACCESS.2023.3331112

Klimczyk, P., y Madeyski, L. (2020). Technical Debt Aware Estimations in Software Engineering: A Systematic Mapping Study. *Informatica Software Engineering Journal*, 61-76.

Kokol, P., Kokol, M., y Zagoranski, S. (2020). Code smells: A Synthetic Narrative Review. *Library Philosophy and Practice (e-journal)*. https://doi.org/10.48550/arXiv.2103.01088

Kokol, P., Zorman, M., y Žlahtič, B. (2018). Code smells. *arXiv preprint arXiv*. https://doi.org/10.13140/RG.2.2.15941.14560

Kruchten, P., Nord , R., y Ozkaya , I. (2019). *Managing Technical Debt: Reducing Friction in Software Development.* Addison-Wesley Professional.

Kruchten, P., Nord, R., y Ozkaya, I. (2012). Technical Debt: From Metaphor to Theory and Practice. *IEEE Software, 29*(6). https://doi.org/10.1109/MS.2012.167

Lacerda, G., Petrillo, F., Pimenta, M., y Guéhéneuc, Y. (2020). Code Smells and Refactoring: A Tertiary Systematic Review of Challenges and Observations. *Journal of Systems and Software, 167*. https://doi.org/10.1016/j.jss.2020.110610

Lefever, J., Cai, Y., Cervantes, H., Kazman, R., y Fang, H. (2021). On the Lack of Consensus Among Technical Debt Detection Tools. *International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 121-130. https://doi.org/10.1109/ICSE-SEIP52600.2021.00021

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., y Arcelli Fontana, F. (2021). A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools. *The Journal of Systems & Software, 171*. https://doi.org/10.1016/j.jss.2020.110827

Lenarduzzi, V., Martini, A., Saarimaki, N., y Tamburri, D. (2021). Technical Debt Impacting Lead-Times: An Exploratory Study. *Euromicro Conference on Software Engineering and*

Advanced Applications (SEAA), 188-195. https://doi.org/10.1109/SEAA53835.2021.00032

Lenarduzzi, V., Saarimäki, N., y Taibi, D. (2020). Some SonarQube issues have a significant but small effect on faults and changes. A large-scale empirical study. *The Journal of Systems & Software*, 1-15.

Lewowski, T., y Madeyski, L. (2022). How far are we from reproducible research on code smell detection? A systematic literature review. *Information and Software Technology*(144). https://doi.org/10.1016/j.infsof.2021.106783

Li, Y., Soliman, M., y Avgeriou, P. (2020). Identification and Remediation of Self-Admitted Technical Debt in Issue Trackers. *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 495-503. https://doi.org/10.1109/SEAA51224.2020.00083

Li, Y., Soliman, M., y Avgeriou, P. (2023). Automatically Identifying Relations Between Self-Admitted Technical Debt Across Different Sources. *ACM/IEEE International Conference on Technical Debt (TechDebt)*, 11-21. https://doi.org/10.1109/TechDebt59074.2023.00008

Lozada Torres, E. F., Cadena Martínez, R., y Pico Pico, M. A. (2025). Análisis estático de código como estrategia para gestionar la deuda técnica: Enfoque multicriterio y tendencias emergentes. *Revista Científica Cultura, Comunicación y Desarrollo, 10*. https://rccd.ucf.edu.cu/index.php/aes/article/view/711

Lozada Torres, E. F., Pico Pico, M. A., Llerena Ocaña, L. A., y Viscaino Naranjo, F. A. (2024). Influencia de las elecciones aceleradas en la producción de software: repercusiones en la

excelencia y rendimiento del producto. *Revista Dilemas Contemporáneos*(3). https://doi.org/10.46377/dilemas.v11i3.4205

Lozada Torres, E., Cadena Martínez, R., y Pico Pico, M. (2024). Gestión de la deuda técnica en entornos académicos: perspectivas y desafíos de un equipo de desarrollo de software. *Data and Metadata, 3*, 1-8. https://doi.org/10.56294/dm2024.233

Lozada Torres, E., Pico Pico, M., Llerena Ocaña, L., y Viscaino Naranjo, F. (2024). Influencia de las elecciones aceleradas en la producción de software: repercusiones en la excelencia y rendimiento del producto. *Revista Dilemas Contemporáneos: Educación, Política y Valores*(3), 1-16. https://doi.org/10.46377/dilemas.v11i3.4205

Lozada Torres, E., Pico Pico, M., y Llerena Ocaña, L. (2023). Neutrosophic statistical analysis of knowledge assessment of agile development models in Higher Education. *Neutrosophic Computing and Machine Learning, 29*, 313-321. https://doi.org/2574-1101

Lozada Torres, E., Pico Pico, M., y Viteri Moya, J. (2023). Análisis de la deuda técnica en desarrolladores noveles de software: Enfoque desde la perspectiva académica. *Universidad y Sociedad Revista Científica de la Universidad de Cienfuegos*.

Mastropaolo, A., Di Penta, M., y Bavota, G. (2023). Towards Automatically Addressing Self-Admitted Technical Debt: How Far Are We? *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 585-597. https://doi.org/10.1109/ASE56229.2023.00103

Melo, A., Fagundes, R., Lenarduzzi, V., y Santos, W. (2022). Identification and Measurement of Technical Debt Requirements in Software Development: a Systematic Literature Review. *Journal of Systems and Software, 194*. https://doi.org/10.1016/j.jss.2022.111483

Mendes, T., Gomes, F., Gonçalves, D., Mendonça, M., Novais, R., y Spínola, R. (2019). VisminerTD: a tool for automatic identification and interactive monitoring of the evolution of technical debt items. *Journal of the Brazilian Computer Society*, 1-28.

Mohan, M., Greer, D., y McMullan, P. (2016). Technical debt reduction using search based automated refactoring. *The Journal of Systems and Software*(120), 183–194. https://doi.org/10.1016/j.jss.2016.05.019

Mohan, M., y Greer, D. (2018). A Survey of Search-Based Refactoring for Software Maintenance. *Journal of Software Engineering Research and Development, 6*(3). https://doi.org/10.1186/s40411-018-0046-4

Moreschini, S., Coba, L., y Lenarduzzi, V. (2024). Towards a Technical Debt for AI-based Recommender System. *ACM/IEEE International Conference on Technical Debt (TechDebt)*, 36-39. https://doi.org/10.1145/3644384.3648574

Murillo, M., López, G., Spínola, R., Guzmán, J., Rios, N., y Pacheco, A. (2023). Identification and Management of Technical Debt: A Systematic Mapping Study Update. *Journal of Software Engineering Research and Development, 11*(8). https://doi.org/10.5753/jserd.2023.2671

Murillo, M., y Jenkins, M. (2021). Technical Debt Measurement during Software Development using Sonarqube: Literature Review and a Case Study. *V Jornadas Costarricenses de Investigación en Computación e Informática* , 1-6. https://doi.org/10.1109/JoCICI54528.2021.9794341

Naranjo-Vélez, M., Naranjo-Vélez, J., y Naranjo-Vélez, C. (2021). Un modelo para la gestión de la deuda técnica en modelos de desarrollo de software ágiles. *"Revista de Ingeniería" de la Universidad de Loja*.

OBrien, D., Biswas, S., Imtiaz, S., Abdalkareem, R., Shihab, E., y Rajan, H. (2022). 23 Shades of Self-Admitted Technical Debt: An Empirical Study on Machine Learning Software. *ESEC/FSE 2022: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 734-746. https://doi.org/10.5281/zenodo.7033365

Ogbeiwi, O. (2017). Why written objectives need to be really SMART. *British Journal of Healthcare Management, 23*(7), 324-336. https://doi.org/10.12968/bjhc.2017.23.7.324

Oranga, J., y Matere, A. (2023). Qualitative Research: Essence, Types and Advantages. *Open Access Library, 10*. https://doi.org/10.4236/oalib.1111001

Ovais Ahmad, M., y Al-Baik, O. (2024). Beyond Technical Debt Unravelling Organisational Debt Concept. *39th ACM/SIGAPP Symposium on Applied Computing*, 802-809. https://doi.org/10.1145/3605098.363591

Paradkar, S. (2021). Framework and Techniques for Managing Technical Debt in Software Development Lifecycle. *Global Journal of Enterprise Information System*, 71-77.

Pavlic, L., Hliš, T., Hericko, M., y Beranic, T. (2022). The Gap between the Admitted and the Measured Technical Debt: An Empirical Study. *Applied Sciences*, 1-21.

Pavlic, L., y Hlis, T. (2019). The Technical Debt Management Tools Comparison. *SQAMIA*, 1-9.

Perera, J., Tempero, E., Tu, Y.-C., y Blincoe, K. (2023). Quantifying Requirements Technical Debt: A Systematic Mapping Study and a Conceptual Model. *International Requirements Engineering Conference (RE)*, 123-133. https://doi.org/10.1109/RE57278.2023.00021

Perera, J., Tempero, E., Tu, Y.-C., y Blincoe, K. (2023). Understanding the relationship between Technical Debt, New Code Cost and Rework Cost in Open-Source Software Projects: An Empirical Study. *EASE '23: Proceedings of the 27th International Conference on*

Evaluation and Assessment in Software Engineering, 247-252. https://doi.org/10.1145/3593434.3593490

Pérez, B., Castellanos, C., Correal, D., Rios, N., Freire, S., Spínola, R., . . . Izurieta, C. (2021). Technical debt payment and prevention through the lenses of software architects. *Information and Software Technology*, 1-16.

Peruma, A., AlOmar, E., y Newman, C. (2022). Refactoring Debt: Myth or Reality? An Exploratory Study on the Relationship Between Technical Debt and Refactoring. *IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, 127-131. https://doi.org/10.1145/3524842.3528510

Peruma, A., Wiem Mkaouer, M., Decker, M., y Newman, C. (2020). Contextualizing rename decisions using refactorings, commit messages, and data types. *Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2020.110704

Petersen, K. (2020). Guidelines for Case Survey Research in Software Engineering. *Contemporary Empirical Methods in Software Engineering*, 63-92. https://doi.org/10.1007/978-3-030-32489-6_3

Pico Pico, M., y Lozada Torres, E. (2023). Implementación de metodologías ágiles en el desarrollo de aplicaciones móviles para programadores inexpertos. *UNIVERSIDAD Y SOCIEDAD | Revista Científica de la Universidad de Cienfuegos*.

Pinto-Llorente, A., Sánchez-Gómez, M., y Costa, A. (2021). Mixed Methods Research: From complementarity to integration. *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality*, 1-4. https://doi.org/10.1145/3486011.3486420

Preciado Orozco , S. (2022). *Technical Debt Analysis and Project Architecturization of a Jenkins Platform based on Groovy* . Universidad Politecnica de Cataluña .

Ramirez Lahti, J., Tuovinen, A.-P., y Mikkonen, T. (2021). Experiences on Managing Technical Debt with Code Smells and AntiPatterns. *IEEE/ACM International Conference on Technical Debt (TechDebt)*, 36-44. https://doi.org/10.1109/TechDebt52882.2021.00013

Rebouças de Almeida, R. (2022). *Business-driven Technical Debt Prioritization.* Universidade Federal do Rio Grande do Norte.

Reboucas de Almeida, R., Treude, C., y Kulesza, U. (2023). What's behind tight deadlines? Business causes of technical debt. *IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, 25-30. https://doi.org/10.1109/CHASE58964.2023.00011

Recupito, G., Pecorelli, F., Catolino, G., Lenarduzzi, V., Taibi, D., Di Nucci, D., y Palomba, F. (2024). Technical debt in AI-enabled systems: On the prevalence, severity, impact, andmanagement strategies for code and architecture. *The Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2024.112151

Rios, N., Freire, S., Perez, B., Castellanos, C., Correal, D., y Mendonca, M. (2021). On the Relationship between Technical Debt Management and Process Models. *IEEE Software, 38*(5), 56-64. https://doi.org/10.1109/MS.2021.3058652

Rios, N., Mendonca, M., y Seaman, C. (2018). A Study of Factors that Lead Development Teams to Incur Technical Debt in Software Projects. *Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* https://doi.org/10.1109/SEAA.2018.00076

Riquet, N., Devroey, X., y Vanderose, B. (2024). Debt Stories: Capturing Social and Technical Debt in the Industry. *International Conference on Technical Debt (TechDebt '24)*. https://doi.org/10.1145/3644384.3644473

Robson, C. (2002). *Real world research : a resource for social scientists and practitioner-researchers.* Blackwell Publishers.

Rodriguez, A., Cruz Gardey, J., Grigera, J., Rossi, G., y Garrido, A. (2023). UX Debt in an Agile Development Process: Evidence and Characterization. *Software Quality Journal, 31*, 1467–1498. https://doi.org/10.1007/s11219-023-09652-2

Rosillo Reyes, J., y Guamán Coronel, D. (2017). *Identificación de Technical debt (TD) en aplicaciones de software construidas bajo estilos y patrones arquitectónicos, a partir del análisis del código fuente o ejecutable.* Loja: Universidad Técnica Particular de Loja.

Rosser, L., y Norton, J. (2021). A Systems Perspective on Technical Debt. *IEEE Aerospace Conference*, 1-10. https://doi.org/10.1109/AERO50100.2021.9438359

Saeeda, H., Ahmad, M., y Gustavsson, T. (2024). Exploring Process Debt in Large-Scale Agile Software Development For Secure Telecom Solutions. *ACM/IEEE International Conference on Technical Debt (TechDebt)*, 11-20. https://doi.org/10.1145/3644384.3644470

Sánchez Montoya, F. A. (2021). *Deuda Técnica: Método de Gestión de la Deuda Técnica en los Requerimientos.* UNiversidad Nacional Autónoma de México.

Sangeetha, M., y Chandrasekar, C. (2019). An empirical investigation into code smells rectifications through. *Ain Shams Engineering Journal*, 549-553. https://doi.org/10.1016/j.asej.2018.10.005

Schütz, J., y Gómez, J. (2020). Towards Collaborative Technical Debt Management in Systems of Systems. *IEEE/ACM International Conference on Technical Debt (TechDebt)*, 87-91. https://doi.org/10.1145/3387906.3388620

Seaman, C., y Guo, Y. (2011). Measuring and Monitoring Technical Debt. *Advances in Computers, 82*, 25-46. https://doi.org/10.1016/B978-0-12-385512-1.00002-5

Seidman, I. (2019). *Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences.* Teachers College Press. https://doi.org/978-0-8077-6148-9

Shamuratov, B., y Hamadan, M. (2022). Comparative Analysis of Modern Methods of Software Risk Management for IT/IS Projects. *7th International Conference on Digital Technologies in Education, Science and Industry (DTESI 2022), 3382*. https://doi.org/1613-0073

Shivashankar, K., Orucevic, M., Maritsdatter Kruke, M., y Martini, A. (2024). Identifying Technical Debt and Its Types Across Diverse Software Projects Issues. *Software Engineering*. https://doi.org/10.48550/arXiv.2408.09128

Siksna, M., Berzina, I., y Romanovs, A. (2023). Machine Learning Powered Code Smell Detection as a Business Improvement Tool. *International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, 1-6. https://doi.org/10.1109/ITMS59786.2023.10317724

Silva, L., Unterkalmsteiner, M., y Wnuk, K. (2023). Towards identifying and minimizing customer-facing documentation debt. *ACM/IEEE International Conference on Technical Debt*, 72-81. https://doi.org/10.1109/TechDebt59074.2023.00015

Simone Tonin, G. (2018). *Technical debt management in the context of agile methods in software development* . Institute of mathematics and statistic of the university of Sao Paulo .

Stol, K.-J., y Fitzgerald , B. (2020). Guidelines for Conducting Software Engineering Research. *Contemporary Empirical Methods in Software Engineering*, 27-62. https://doi.org/10.1007/978-3-030-32489-6_2

Sutoyo, E., y Capiluppi, A. (2024). SATDAUG - A Balanced and Augmented Dataset for Detecting Self-Admitted Technical Debt. *IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, 298-293. https://doi.org/10.1145/3643991.3644880

Tahaei, M., Vaniea, K., Beznosov, K., y Wolters, M. (2021). Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. *CHI Conference on Human Factors in Computing Systems*, 1-17. https://doi.org/10.1145/3411764.3445616

Taherdoost, H., y Madanchian, M. (2023). Multi-Criteria Decision Making (MCDM) Methods and Concepts. *Encyclopedia, 3*(1), 77-87. https://doi.org/10.3390/encyclopedia3010006

Tamayo Palomino, A. (2021). *Aproximación metodológica para la integración de las metodologías DataOps y MLOps aplicadas al trading automático.* Universidad Nacional de Colombia.

Tan, J., Feitosa, D., y Avgeriou, P. (2023). The lifecycle of Technical Debt that manifests in both source code and issue trackers. *Information and Software Technology*. https://doi.org/10.1016/j.infsof.2023.107216

Tom, E., Aurum , A., , y Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software, 86*(6), 1498-1516.

Torres Ruiz, D. (2019). *Modelo para el tratamiento de la deuda técnica orientado a la evolución de los componentes para que las aplicaciones sean sostenibles a largo plazo.* Universidad de Medellín.

Verdecchia, R., Kruchten, P., y Lago, P. (2020). Architectural Technical Debt: A Grounded Theory. *European Conference on Software Architecture*, 202-219.

Wiese, M., y Borowa, K. (2023). IT managers perspective on Technical Debt Management. *Journal of Systems and Software, 202*. https://doi.org/10.1016/j.jss.2023.111700

Wilder, G., Miyamoto, R., Watson, S., Kazman, R., y Peruma, A. (2023). An Exploratory Study on the Occurrence of Self-Admitted Technical Debt in Android Apps. *ACM/IEEE International Conference on Technical Debt (TechDebt)*, 1-10. https://doi.org/10.1109/TechDebt59074.2023.00007

Xavier, L., Dos Santos, R., Bessa, S., y Valente, M. (2023). Agile Technical Debt Management using the LTD Framework. *ACM SIGSOFT Software Engineering Notes, 49*(1), 13-23. https://doi.org/10.1145/3635439.36354

Xavier, L., Ferreira, F., Brito, R., y Valente, M. (2020). Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. *17th International Conference on Mining Software Repositories (MSR)*, 137-146. https://doi.org/10.1145/3379597.3387459

Yang, Y., Verma, D., y Anton, P. (2023). Technical debt in the engineering of complex systems. *System Engineering, 26*, 590-603. https://doi.org/https://doi.org/10.1002/sys.21677

Yasmin, J., Sadegh Sheikhaei, M., y Tian, Y. (2022). A First Look at Duplicate and Near-duplicate Self-admitted Technical Debt Comments. *30th International Conference on Program Comprehension (ICPC)*, 614-618. https://doi.org/10.1145/3524610.3528387

Yin, R. (2018). *Case Study Research and Applications: Design and Methods.* SAGE.

Yli-Huumo, J., Maglyas, A., y Smolander, K. (2016). How do software development teams manage technical debt? –An empirical study. *The Journal of Systems and Software*(120), 195-218. https://doi.org/10.1016/j.jss.2016.05.018

Yu, J., Zhao, K., Liu, J., Liu, X., Xu, Z., y Wang, X. (2022). Exploiting gated graph neural network for detecting and explaining self-admitted technical debts. *The Journal of Systems & Software*. https://doi.org/10.1016/j.jss.2022.111219

Yu, Z., Fahid, F., Tu, H., y Menzies, T. (2022). Identifying Self-Admitted Technical Debts with Jitterbug: A Two-Step Approach. *IEEE Transactions on Software Engineering, 48*(5), 1676 - 1691. https://doi.org/10.1109/TSE.2020.3031401

Apéndices

Apéndice A. Validación de la técnica de recolección de datos

Universidad Americana de Europa – UNADE
Doctorado en Informática
Validación de expertos de Instrumento de Recolección de datos

Alumno:

Edwin Fabricio Lozada Torres

Tema:

Gestión de deuda técnica en modelos ágiles de desarrollo de software bajo enfoque basado en valor

Objetivos:

Objetivo general.

Desarrollar un marco para la gestión y reducción de la deuda técnica en modelos de desarrollo de software ágiles que proporcione un enfoque conceptual para administrarla.

Objetivos específicos.

- Analizar las principales causas de la deuda técnica en proyectos de desarrollo ágil y su impacto en la calidad del software.
- Identificar el estado actual de la gestión de la deuda técnica en modelos de desarrollo de software ágiles.
- Desarrollar un marco en el cual se definan técnicas y herramientas para el tratamiento de la deuda técnica en ambientes de desarrollo ágiles.
- Evaluar el modelo propuesto a través de un estudio de caso.

Hipótesis:

Con el diseño de un marco para la gestión de la deuda técnica en modelos de desarrollo de software ágiles se esperarán mejores procesos y herramientas para administrar y optimizar la deuda técnica lo que contribuirá a reducirla y gestionarla

Instrumento:

Entrevista semisestructurada

Las preguntas planteadas en la entrevista son las que se indican en la tabla:

Introducción	
P1	¿Puedes describir tu rol y tus principales responsabilidades en el equipo?
P2	¿Cuánto tiempo llevas trabajando en este equipo?
Conocimiento y Percepción	
P3	¿Qué entiendes por deuda técnica en el desarrollo de software?
P4	¿Consideras que la deuda técnica es un problema en los proyectos en los que trabajas? ¿Por qué?
Prácticas Actuales	
P5	¿Cómo identificas y manejas la deuda técnica en tu trabajo diario?
P6	¿Sientes que hay un proceso claro para abordar la deuda técnica?
Factores Contributivos	
P7	¿Qué factores en tu día a día crees que contribuyen a la deuda técnica (e.g., plazos de entrega, cambios de requisitos, falta de recursos)?
P8	¿Cómo manejas la presión de entregar rápido versus mantener la calidad del código?
Impacto y Consecuencias	
P9	¿Puedes dar ejemplos de situaciones en las que la deuda técnica ha impactado tu trabajo o el proyecto?
P10	¿Cómo sientes que la deuda técnica afecta la eficiencia y calidad del desarrollo?
Ideas y Soluciones	
P11	¿Qué prácticas crees que podrían implementarse para reducir la deuda técnica en tu equipo?
P12	¿Hay herramientas o procesos que crees que podrían ayudar a gestionar la deuda técnica mejor?
Cierre	
P13	¿Está dispuesto a dedicar tiempo y esfuerzo a abordar la deuda técnica?
P14	¿Tienes algún otro comentario o reflexión sobre la deuda técnica que te gustaría compartir?

Tabla de Revisión de Instrumentos Metodológicos

Criterio	Descripción	Evaluación (Sí/No)	Observaciones
Claridad	¿Las preguntas están redactadas de forma clara y comprensible para los encuestados?	Sí	
Relevancia	¿Cada ítem está alineado con los objetivos de la investigación?	Sí	
Validez de contenido	¿Los ítems representan adecuadamente la variable que se quiere medir?	Sí	
Consistencia interna	¿Los ítems relacionados con una misma dimensión presentan coherencia en sus respuestas?	Sí	
Neutralidad de los ítems	¿Las preguntas están formuladas de manera objetiva, sin inducir respuestas?	Sí	
Comprensibilidad del lenguaje	¿El vocabulario utilizado es accesible y acorde al perfil de los encuestados?	Es una entrevista, no una encuesta.	
Tiempo de aplicación	¿El tiempo estimado para responder la encuesta es adecuado y no genera fatiga?	No es una encuesta y no se ha estipulado tiempo de respuesta	
Adaptabilidad	¿El instrumento puede ajustarse si se identifican mejoras necesarias?	Sí	

Observaciones: En general, la entrevista aborda todas las preguntas necesarias para obtener datos relevantes para la investigación.

Dra. Irene Becerril Arostegui

.....

Nombre y Firma del experto

Universidad Americana de Europa – UNADE
Doctorado en Informática
Validación de expertos de Instrumento de Recolección de datos

Alumno:

Edwin Fabricio Lozada Torres

Tema:

Gestión de deuda técnica en modelos ágiles de desarrollo de software bajo enfoque basado en valor

Objetivos:

Objetivo general.

Desarrollar un marco para la gestión y reducción de la deuda técnica en modelos de desarrollo de software ágiles que proporcione un enfoque conceptual para administrarla.

Objetivos específicos.

- Analizar las principales causas de la deuda técnica en proyectos de desarrollo ágil y su impacto en la calidad del software.
- Identificar el estado actual de la gestión de la deuda técnica en modelos de desarrollo de software ágiles.
- Desarrollar un marco en el cual se definan técnicas y herramientas para el tratamiento de la deuda técnica en ambientes de desarrollo ágiles.
- Evaluar el modelo propuesto a través de un estudio de caso.

Hipótesis:

Con el diseño de un marco para la gestión de la deuda técnica en modelos de desarrollo de software ágiles se esperarán mejores procesos y herramientas para administrar y optimizar la deuda técnica lo que contribuirá a reducirla y gestionarla

Instrumento:

Entrevista semisestructurada

Las preguntas planteadas en la entrevista son las que se indican en la tabla:

	Introducción
P1	¿Puedes describir tu rol y tus principales responsabilidades en el equipo?
P2	¿Cuánto tiempo llevas trabajando en este equipo?
	Conocimiento y Percepción
P3	¿Qué entiendes por deuda técnica en el desarrollo de software?
P4	¿Consideras que la deuda técnica es un problema en los proyectos en los que trabajas? ¿Por qué?
	Prácticas Actuales
P5	¿Cómo identificas y manejas la deuda técnica en tu trabajo diario?
P6	¿Sientes que hay un proceso claro para abordar la deuda técnica?
	Factores Contributivos
P7	¿Qué factores en tu día a día crees que contribuyen a la deuda técnica (e.g., plazos de entrega, cambios de requisitos, falta de recursos)?
P8	¿Cómo manejas la presión de entregar rápido versus mantener la calidad del código?
	Impacto y Consecuencias
P9	¿Puedes dar ejemplos de situaciones en las que la deuda técnica ha impactado tu trabajo o el proyecto?
P10	¿Cómo sientes que la deuda técnica afecta la eficiencia y calidad del desarrollo?
	Ideas y Soluciones
P11	¿Qué prácticas crees que podrían implementarse para reducir la deuda técnica en tu equipo?
P12	¿Hay herramientas o procesos que crees que podrían ayudar a gestionar la deuda técnica mejor?
	Cierre
P13	¿Está dispuesto a dedicar tiempo y esfuerzo a abordar la deuda técnica?
P14	¿Tienes algún otro comentario o reflexión sobre la deuda técnica que te gustaría compartir?

Tabla de Revisión de Instrumentos Metodológicos

Criterio	Descripción	Evaluación (Sí/No)	Observaciones
Claridad	¿Las preguntas están redactadas de forma clara y comprensible para los encuestados?	**Sí**	Las preguntas son directas, con lenguaje comprensible y adecuado para profesionales de desarrollo de software. Se entiende el propósito de cada una sin ambigüedad.
Relevancia	¿Cada ítem está alineado con los objetivos de la investigación?	**Sí**	Todas las preguntas se relacionan con los objetivos específicos: identificación, gestión, impacto y estrategias de reducción de la deuda técnica.
Validez de contenido	¿Los ítems representan adecuadamente la variable que se quiere medir?	**Sí**	Las preguntas abordan dimensiones conceptuales, prácticas y actitudinales sobre la deuda técnica, cubriendo el fenómeno desde distintos ángulos.
Consistencia interna	¿Los ítems relacionados con una misma dimensión presentan coherencia en sus respuestas?	**Sí**	Existe coherencia interna: las preguntas P3–P4 (percepción), P5–P6 (prácticas), P7–P8 (factores), y P9–P10 (impacto) están bien agrupadas temáticamente.
Neutralidad de los ítems	¿Las preguntas están formuladas de manera objetiva, sin inducir respuestas?	**Sí**	Las preguntas no sugieren respuestas deseadas. Son abiertas, permitiendo libre expresión de opiniones y experiencias.
Comprensibilidad del lenguaje	¿El vocabulario utilizado es accesible y acorde al perfil de los encuestados?	**Sí**	El vocabulario es técnico pero pertinente al perfil del encuestado (profesionales de desarrollo de software). Se sugiere definir brevemente "deuda técnica" al inicio de la entrevista.
Tiempo de aplicación	¿El tiempo estimado para responder la entrevista es adecuado y no genera fatiga?	**Sí**	La entrevista contiene 14 preguntas abiertas. Se estima un tiempo de 20 a 30 minutos, lo cual es razonable para entrevistas semiestructuradas.
Adaptabilidad	¿El instrumento puede ajustarse si se identifican mejoras necesarias?	**Sí**	El formato permite incorporar nuevas preguntas o ajustar la redacción según los resultados de las pruebas piloto.

Observaciones: En general, la entrevista aborda todas las preguntas necesarias para obtener datos relevantes para la investigación.

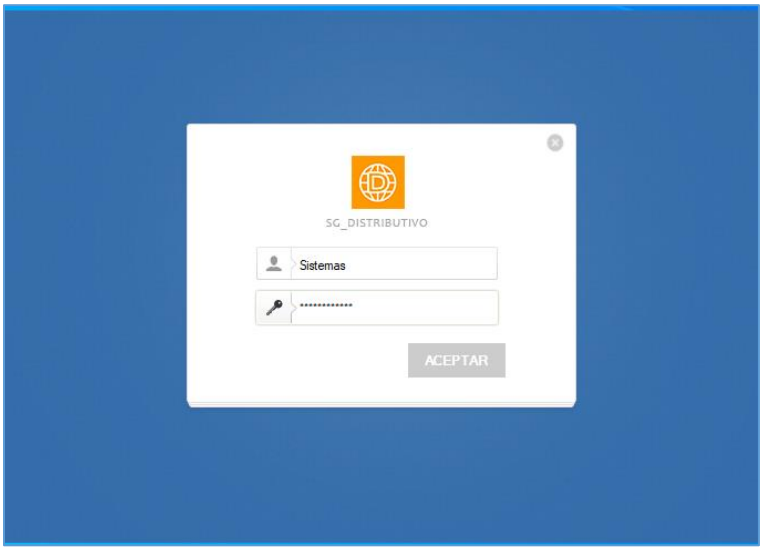
Dr. Luis Antonio Llerena Ocaña, PhD

……

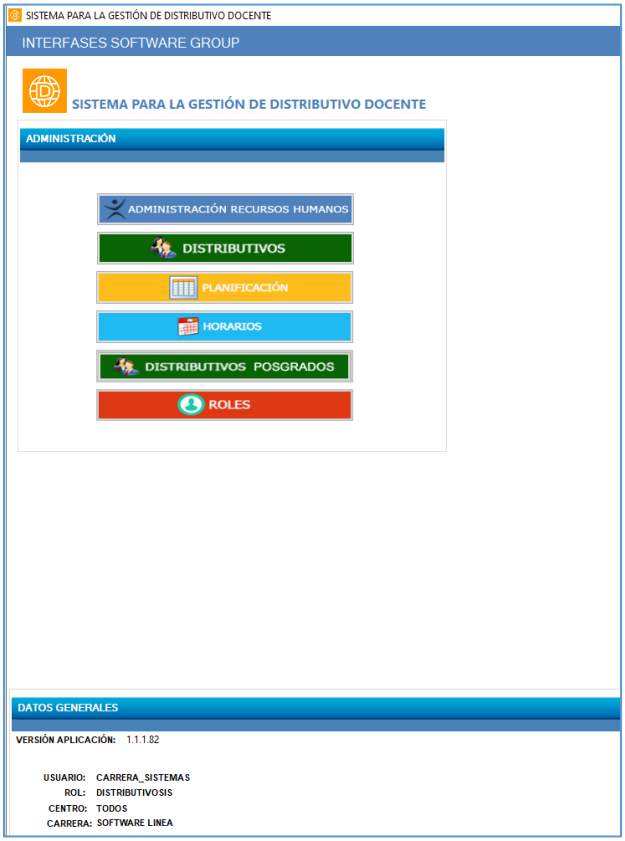
Nombre y Firma del experto

Apéndice B. Módulo Sistema de Gestión - SG

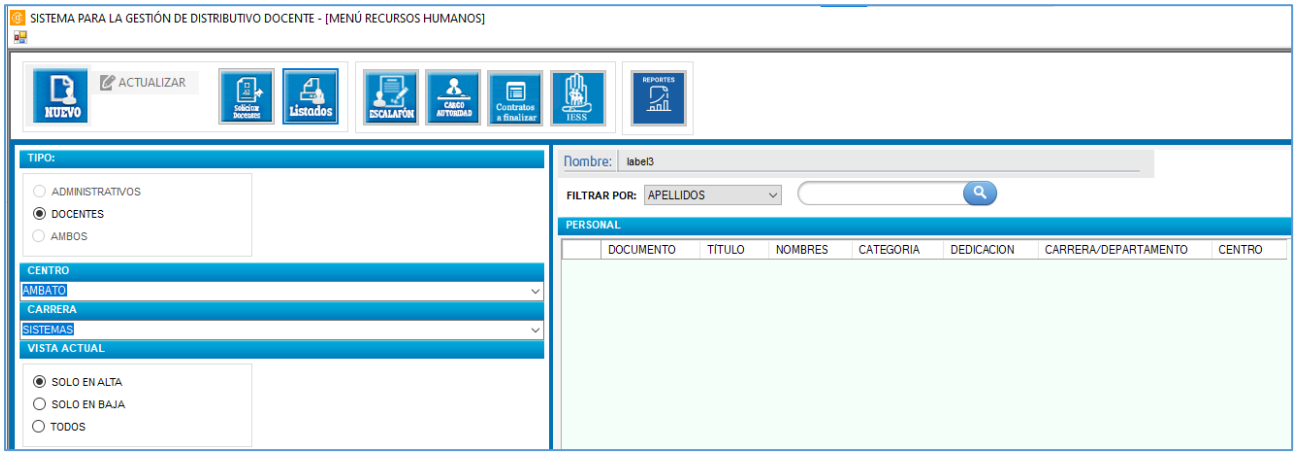
Ingreso a la aplicación: Formulario principal para ingresar al SG



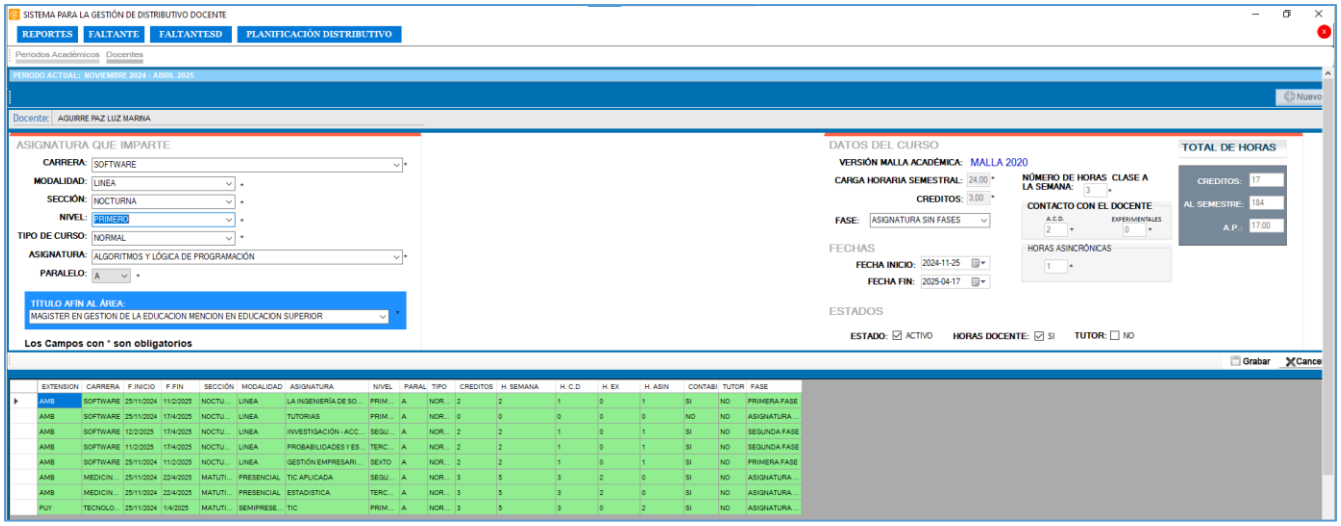
Pantalla principal: Menú principal de opciones



Administración de Recursos Humanos: Formulario de Recurso Humanos



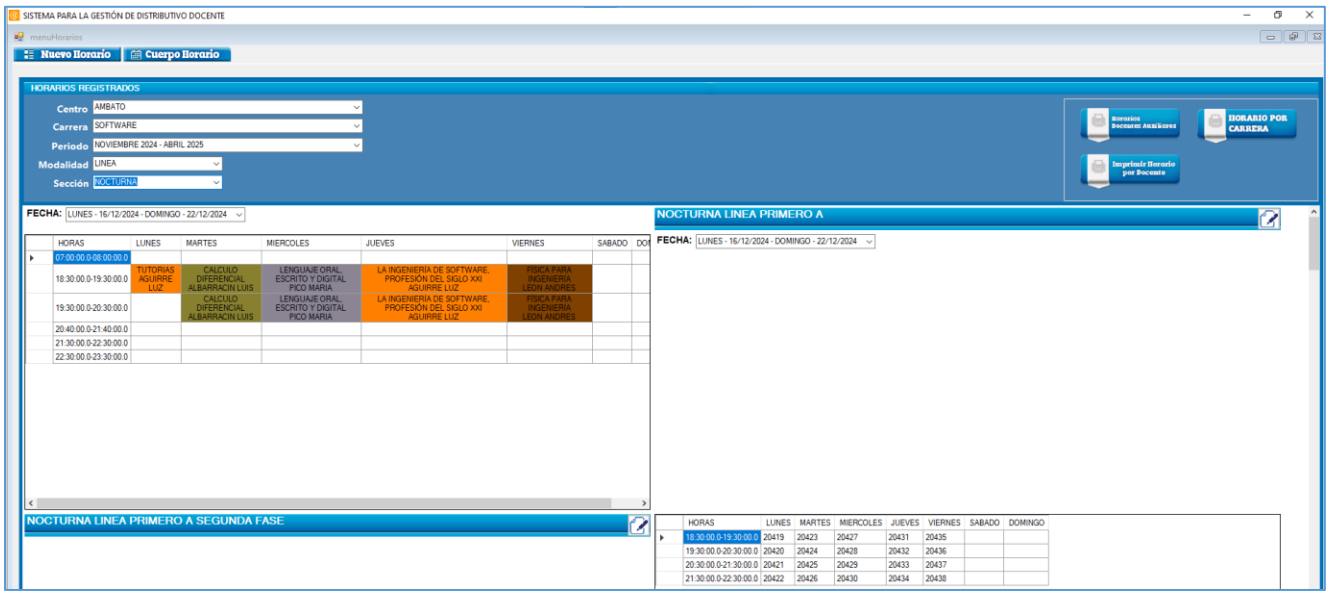
Distributivos: Formulario para la gestión de Distributivos



Planificación: Formulario para la gestión de horas de planificación



Horarios: Formulario para el ingreso de horarios



Glosario

Arquitectura de Software: Estructura fundamental del sistema, que define componentes y sus interacciones.

Calidad del Software: Conjunto de características que determinan la capacidad de un producto de software para satisfacer requisitos funcionales y no funcionales, incluyendo atributos como mantenibilidad, confiabilidad, eficiencia, usabilidad y seguridad.

Clean Code: Principios y prácticas para escribir código legible, mantenible y eficiente.

Código Espagueti: Código desordenado y difícil de mantener, que incrementa la deuda técnica.

Code Smell: Indicios en el código que sugieren problemas de diseño o mantenimiento.

Continuous Delivery (CD): Extensión que asegura que el software esté siempre listo para ser desplegado.

Ciclo de Vida del Software: Etapas por las que pasa un sistema desde su concepción hasta su retiro.

Deuda Técnica (Technical Debt): Metáfora que describe el costo futuro asociado a decisiones de diseño o implementación subóptimas.

Design Patterns: Soluciones reutilizables para problemas comunes en diseño de software.

Desarrollo de Software: Proceso de concepción, diseño, implementación y mantenimiento de aplicaciones informáticas.

DevOps: Cultura y conjunto de prácticas que integran desarrollo y operaciones para acelerar entregas.

Extreme Programming (XP): Metodología ágil centrada en la mejora continua, la simplicidad y la calidad del código mediante prácticas como pair programming.

Gestión de la Deuda Técnica: Conjunto de prácticas para identificar, medir, priorizar y reducir la deuda técnica en proyectos de software.

Interés en la Deuda Técnica: Costo adicional que se genera por mantener deuda técnica en el tiempo. Este interés se manifiesta en forma de mayor esfuerzo para realizar cambios, incremento en defectos, disminución de productividad y riesgo de fallos, afectando la evolución del sistema.

Kanban: Método visual para gestionar el flujo de trabajo, basado en tableros y tarjetas que representan tareas.

Lean Software Development: Enfoque ágil que busca eliminar desperdicios y optimizar el flujo de valor.

Marco de Trabajo (Framework): Estructura conceptual y práctica que define un conjunto de reglas, roles, artefactos y procesos para organizar y ejecutar actividades en el desarrollo de software.

Metodologías Ágiles: Conjunto de prácticas y principios para el desarrollo iterativo e incremental, priorizando la colaboración y la adaptabilidad.

Métricas de Calidad: Indicadores que evalúan atributos como mantenibilidad, complejidad y cobertura de pruebas.

Pair Programming: Técnica en la que dos desarrolladores trabajan juntos en el mismo código, mejorando calidad y conocimiento compartido.

Product Backlog: Lista priorizada de funcionalidades, mejoras y correcciones que guían el desarrollo en Scrum.

Refactor Debt: Deuda técnica acumulada por falta de refactorización en el tiempo.

Refactorización: Proceso de mejorar la estructura interna del código sin alterar su comportamiento externo.

Refinamiento del Backlog: Proceso de revisión y actualización del backlog para mantenerlo claro y priorizado.

Scrum: Marco ágil que organiza el trabajo en ciclos cortos llamados sprints, con roles definidos y reuniones periódicas.

SonarQube: Herramienta para análisis estático del código y detección de deuda técnica.

Sprint: Iteración corta (generalmente de 1 a 4 semanas) en la que se desarrolla un conjunto de funcionalidades.

Technical Debt Ratio: Métrica que relaciona el costo de reparar la deuda técnica con el costo total del sistema.

Testing Automatizado: Uso de herramientas para ejecutar pruebas de manera automática, reduciendo errores humanos.

User Story: Descripción breve de una funcionalidad desde la perspectiva del usuario, utilizada para definir requisitos ágiles.