

Análisis del Aseguramiento de Calidad en el Desarrollo Ágil de MarIAAna: Impacto de la Deuda Técnica y Propuestas de Mejora

Adolfo Espinosa Pérez
Profesor Guía:
Marcello Visconti Zamora
Profesor Correferente:
Ricardo Salas Letelier

Julio 2025

Resumen

El informe evalúa el aseguramiento de calidad en el desarrollo ágil de MarIAAna, plataforma SaaS para predicción y visualización geoespacial en emergencias marítimas. Detecta que la priorización de funcionalidades visibles generó deuda técnica que comprometió mantenibilidad, escalabilidad y confiabilidad. Mediante revisión de historias de usuario, criterios de aceptación y pruebas, se identificaron problemas en visualización, control de simulaciones, acoplamiento de módulos, validaciones temporales y espaciales e integración de fuentes. Propone un plan en dos fases: saneamiento técnico del prototipo y posterior integración modular de capacidades científicas, validación y automatización, para lograr una plataforma más robusta, mantenible, operativa, escalable y sostenible.

Palabras Clave: Metodologías ágiles, Calidad de software, Aseguramiento de calidad, Desarrollo de software

1. Introducción

El desarrollo de software en contextos académicos como la Feria de Software de la Universidad Técnica Federico Santa María representa una oportunidad única para aplicar conocimientos teóricos a problemas reales. En este entorno se gestó MarIAAna, una plataforma web desarrollada como Software as a Service (SaaS) orientada a la predicción y visualización geoespacial del movimiento de partículas en emergencias marítimas, tales como el rescate de personas o naves extraviadas en alta mar. Esta herramienta busca asistir en la toma de decisiones en situaciones críticas, donde el tiempo de reacción es fundamental.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Análisis del Aseguramiento de Calidad en el Desarrollo Ágil de MarIAna: Impacto de la Deuda Técnica y Propuestas de Mejora

Nombre del candidato(a): Adolfo Antonio Espinosa Pérez

Carrera / Grado: Ingeniería Civil Informatica

Campus: Santiago San Joaquin ; **Departamento:** Informatica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Marcello Visconti, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 18/08/2025

Firma: 

Estudiante o Candidato(a):

Fecha: 18/08/2025

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.



Sin embargo, a pesar del valor funcional que ofrece la plataforma, su desarrollo estuvo marcado por una fuerte presión por cumplir plazos, priorizando la entrega de funcionalidades visibles en detrimento de un proceso estructurado de aseguramiento de calidad (QA). Como consecuencia, se acumuló una cantidad significativa de deuda técnica, afectando la mantenibilidad, escalabilidad y confiabilidad del sistema. Esta situación no solo impactó negativamente en la calidad del software, sino que también comprometió la experiencia del usuario y la viabilidad a largo plazo del proyecto.

La presente memoria tiene como propósito analizar sistemáticamente el impacto de esta deuda técnica dentro del marco del desarrollo ágil de MarIAAna. A partir de la evaluación de los criterios de aceptación de las historias de usuario, se identifican los principales déficits de calidad técnica y funcional. Este diagnóstico se complementa con una propuesta de mejora que plantea un plan de trabajo dividido en dos etapas: una primera centrada en estabilizar y sanear el sistema, y una segunda orientada a la integración de capacidades técnicas avanzadas. Con ello, se busca sentar las bases para una segunda versión más robusta y sostenible del software.

2. Marco Teórico

2.1. Definición de MarIAAna e historias de usuario

La Feria de Software de la Universidad Técnica Federico Santa María es un evento donde los estudiantes del Departamento de Informática muestran sus habilidades en el desarrollo de software, presentando soluciones tecnológicas innovadoras para problemas reales de la sociedad. Esta feria destaca por fomentar el trabajo en equipo, el emprendimiento y el aprendizaje mediante la creación de productos que demuestran la capacidad de los estudiantes para enfrentar desafíos reales y entregar soluciones aplicables en el mundo profesional [5].

Dentro de la Feria de Software, en su edición XXXII, se presentó el proyecto de software MarIAAna, desarrollado por Swal en colaboración con RedOcean. Esta aplicación es un *SaaS* que responde a dos principales funcionalidades: el registro de accidentes marítimos, de toda índole, a lo largo del tiempo y la predicción, acompañada de una visualización georreferenciada y a través del tiempo, del movimiento de las partículas generadas por estos accidentes marítimos, esto con el objetivo de satisfacer 3 fines: rescate de personas, organismos peligrosos para el humano y/o fauna marina (fragata portuguesa, medusas, otras plagas marinas) y amenaza medioambiental (derrame de petróleo). En todas estas situaciones, el tiempo es un factor crucial, por lo que toda la información generada por el modelo es entregada en tiempo real al usuario.

Para poder construir este software, se definieron las siguientes historias de usuario:



Nombre Historia:	Visualización; Interacción del usuario con el mapa
Código Historia:	HU1.1
Descripción:	Como experto en altamar del sistema necesito poder interactuar con el mapa de la visualización para poder ajustarlo a mis necesidades y enfocar las secciones que estime de mayor relevancia.
Criterios de Aceptación:	Necesito controlar la velocidad y reproducción de la visualización de la predicción. Cambiar elementos visuales del mapa. Visualizar líneas vectoriales de las corrientes. Se debe mostrar, en todo punto de la predicción, toda la trayectoria que ha recorrido y recorrerá la partícula a la deriva. El reproductor de la predicción debe indicar hasta qué instante del tiempo se tienen los datos de la predicción de acuerdo a la información recibida vía streaming. Múltiples datos de viento almacenados en el caché para predecir distintos días. Interpolación de la posición si no existe la predicción para ese punto específico. Se ve la línea de costa, respetando las islas y el terreno costero.

Cuadro 1: Historia de Usuario 1.1

Nombre Historia:	Visualización; Proyección de la predicción
Código Historia:	HU1.2
Descripción:	Como experto en altamar quiero poder ver una predicción del movimiento de una partícula a la deriva para conocer su comportamiento
Criterios de Aceptación:	El usuario podrá visualizar una cantidad definida de partículas en el mapa. Las partículas se mueven en el tiempo. Se puede visualizar un área de error.

Cuadro 2: Historia de Usuario 1.2

Nombre Historia:	Predicción; Dominio y línea de Costas con IA
Código Historia:	HU2.1
Descripción:	Como experto en altamar del sistema necesito poder declarar un dominio de predicción para declarar el lugar y momento donde se desarrolla la emergencia.
Criterios de Aceptación:	Declarar manualmente el dominio geográfico. Declarar dominio temporal. Se genera una línea de costa dependiendo del dominio. La línea de costa es más fina en las partes más cercanas al inicio del dominio. La línea de costa es continua dentro del dominio. Todos los puntos de la línea de costa deben estar conectados al mar abierto. La línea de costa no pasa sobre terreno costero.

Cuadro 3: Historia de Usuario 2.1



Nombre Historia:	Predicción; MDP Modelo Local de alta precisión
Código Historia:	HU2.2
Descripción:	Como experto en altamar quiero realizar una predicción de alta precisión sobre el movimiento de un objeto o partícula en base a parámetros ingresados por mi, con el objetivo de obtener una posible ruta sobre el movimiento de dicha partícula.
Criterios de Aceptación:	La predicción se calcula a través de la declaración de un dominio geográfico y temporal definido por el usuario. Los datos necesarios para la predicción estarán dados por el dominio establecido. Los datos llegarán a la plataforma de visualización vía streaming. La predicción, al acercarse a costa, se va haciendo más precisa. La predicción se calcula a través de datos climáticos interpolados a partir de los modelos de fuentes públicas.

Cuadro 4: Historia de Usuario 2.2

Nombre Historia:	Predicción; MDP Modelo Global
Código Historia:	HU2.3
Descripción:	Como experto de altamar quiero realizar una predicción utilizando datos climáticos de origen público con el objetivo de obtener una posible ruta sobre el movimiento de dicha partícula.
Criterios de Aceptación:	La predicción se calcula a través de un punto geográfico y ventana temporal establecidos por el usuario. La predicción se calcula utilizando los datos climáticos de fuentes públicas. Los datos llegarán a la plataforma de visualización vía streaming. Se va a poder elegir entre las distintas fuentes de datos públicos.

Cuadro 5: Historia de Usuario 2.3

Nombre Historia:	Predicción; Generación de Grillas
Código Historia:	HU2.4
Descripción:	Como experto de altamar, quiero que se genere una grilla alrededor del punto de inicio del problema, para tener una representación del espacio en el que se mueve la partícula.
Criterios de Aceptación:	Se genera una grilla alrededor del punto de inicio del problema. Los elementos de la grilla no se superponen entre sí. La grilla se vuelve más densa cerca de la costa. La grilla se vuelve más densa cerca de las islas. La grilla es completamente conexas. Se generan puntos de esponja en la grilla. La grilla toma en consideración los puntos de esponja y es conexas con estos. La grilla se ve limitada por los bordes costeros. La grilla se genera dentro del terreno en ríos y lagos.

Cuadro 6: Historia de Usuario 2.4



Nombre Historia:	Predicción; Línea de Costas con IA
Código Historia:	HU2.5
Descripción:	Como experto de altamar, quiero que la línea de costa que utilice el programa sea más fina en las secciones de relevancia, para reducir tiempo de cómputo y mejorar la calidad de la predicción.
Criterios de Aceptación:	Se genera una línea de costa dependiendo del dominio. La línea de costa es más fina en las partes más cercanas al inicio del dominio. La línea de costa es continua dentro del dominio. Todos los puntos de la línea de costa deben estar conectados al mar abierto. La línea de costa no pasa sobre terreno costero.

Cuadro 7: Historia de Usuario 2.5

Nombre Historia:	Rendimiento; Estadísticas
Código Historia:	HU3.1
Descripción:	Como experto de altamar, necesito un dashboard con el informe de métricas de los modelos y monitoreo de uso de la aplicación para poder evaluar el uso de la aplicación.
Criterios de Aceptación:	Tiempo de ejecución modelo según la o las bases de datos escogidas y el método para la predicción. Posición final de la predicción de la partícula. Cantidad de puntos generados. Área de búsqueda reducida.

Cuadro 8: Historia de Usuario 3.1

Nombre Historia:	Rendimiento; Historial de dominios y visualizaciones
Código Historia:	HU3.2
Descripción:	Como experto de alta mar, necesito un antecedente de predicciones pasadas, para poder acceder a ellas en otro momento u otro dispositivo.
Criterios de Aceptación:	La sección de historial muestra en orden cronológico las predicciones realizadas por el usuario. Cada visualización contiene información relevante de la predicción. Las visualizaciones tienen la opción de volver a verlas. Filtros (Tiempo, lugar, tipo de partícula). Se guardarán los dominios previamente generados. En caso de coincidir espacial y temporalmente, se podrá reutilizar un dominio previamente generado para una nueva visualización.

Cuadro 9: Historia de Usuario 3.2

2.2. Definición de metodologías ágiles, QA y deuda técnica en MarIAna

MarIAna fue desarrollado utilizando una metodología ágil, que se caracteriza por su enfoque en la flexibilidad, colaboración continua y entrega incremental de software [6]. Las

metodologías ágiles están diseñadas para adaptarse a los cambios constantes en los requisitos del cliente, lo cual es fundamental en un entorno dinámico como el de las emergencias marítimas. El objetivo es entregar software funcional en ciclos cortos, conocidos como iteraciones, que permiten ajustes rápidos basados en la retroalimentación recibida por parte de los usuarios y *stakeholders*, garantizando que el producto evolucione de acuerdo con las necesidades reales del proyecto.

Existen muchas formas de aplicar metodologías ágiles, como *XP (Extreme Programming)*, *Scrum*, *Kanban*, *Lean*, *Feature-Driven Development (FDD)*, *Dynamic Systems Development Method (DSDM)*, *Adaptive Software Development (ASD)*, *Crystal*, y *Rational Unified Process (RUP)* [1]. Sin embargo, en el contexto de la Feria de Software, la metodología ágil más utilizada es Scrum, debido a su enfoque empírico. Esto significa que las decisiones en Scrum se basan en el conocimiento adquirido durante el proceso de desarrollo y la experiencia práctica obtenida en cada iteración, lo que permite una toma de decisiones más informada y ajustada a la realidad del proyecto [9].

Una de las partes más importantes de las metodologías ágiles es la calidad del software. Algunos autores incluso afirman que la calidad es un aspecto inherente al desarrollo ágil [6]. Dentro del desarrollo ágil, la calidad del software puede definirse mediante varios conceptos fundamentales, como la corrección, solidez, extensibilidad, reutilización, compatibilidad, eficacia, portabilidad, integridad, verificabilidad y facilidad de uso [10]. Estos atributos permiten evaluar qué tan bien se desempeña un software no solo en su funcionalidad básica, sino también en su capacidad para adaptarse a nuevos contextos y mantener su fiabilidad a lo largo del tiempo.

En este sentido, existen distintos modelos o metodologías para trabajar el desarrollo ágil con un enfoque en QA; estos modelos pueden ser *Test-Driven Development*, *Behavior-Driven Development* y *Acceptance Test-Driven Development* [7].

Test-Driven Development es una metodología en la que se escriben pruebas antes de desarrollar el código. El proceso comienza con una prueba que inicialmente falla, luego se escribe el código mínimo para hacerla pasar, y finalmente se refactoriza el código para optimizarlo. Este ciclo se repite para cada nueva funcionalidad. TDD ofrece varias ventajas, como la prevención temprana de errores, la creación de código más modular y la automatización de pruebas de regresión, facilitando la refactorización segura. Sin embargo, presenta desafíos como su curva de aprendizaje, la dificultad de aplicarlo en sistemas heredados y el esfuerzo inicial que requiere [7].

Behavior-Driven Development es una metodología ágil que se enfoca en definir el comportamiento esperado del sistema en lugar de centrarse solo en la implementación. Su objetivo principal es alinear las pruebas con los requisitos de negocio mediante un lenguaje común que todos los involucrados (desarrolladores, testers y clientes) puedan entender. En BDD, los escenarios se escriben utilizando una estructura clara (Given-When-Then) para describir cómo debe comportarse el sistema en situaciones específicas. Estos escenarios se convierten en pruebas automatizadas que verifican que el sistema cumpla con las expectativas del cliente. El enfoque de BDD promueve la colaboración, mejorando la comunicación entre los equipos técnicos y de negocio. Además, fomenta un desarrollo orientado al comportamiento desde las primeras fases del proyecto, ayudando a garantizar que el producto final tenga valor para el



negocio [12].

Acceptance Test-Driven Development (ATDD) es una metodología ágil centrada en la colaboración entre desarrolladores, testers y usuarios para definir los criterios de aceptación de una funcionalidad antes de que comience la implementación. El propósito de ATDD es garantizar que todo el equipo tenga una comprensión clara de los requisitos del cliente y que las pruebas reflejen el comportamiento esperado del sistema desde la perspectiva del usuario. ATDD se enfoca en la calidad externa del software, asegurando que el producto final responda a las expectativas del cliente y facilite la entrega de valor desde el inicio. Además, fomenta la automatización de las pruebas de aceptación para integrarlas en los procesos de desarrollo continuo [7].

En este contexto, resulta pertinente examinar otro concepto fundamental: la deuda técnica. Si bien este concepto no tiene una definición específica, el término se origina por Ward Cunningham en el congreso de orientación a objetos OOPSLA en el año 1992[2], donde menciona que la deuda técnica surge cuando los miembros del equipo deciden no realizar ciertas tareas de la manera adecuada en el presente, lo que puede obstaculizar el desarrollo futuro si no se corrige. Por otro lado, los intereses son los costos diarios asociados con mantener esa deuda, los cuales se acumulan y aumentan el costo total de resolverla.

A lo largo de los años, distintos autores han complementado o cambiado la definición de deuda técnica propuesta por Cunningham, por ejemplo, Martin Fowler agrega una deuda financiera a la definición, al incluir formas de pago e intereses generados[3]. Además, define que existen 4 tipos de deuda técnica, la deuda imprudente, la deuda prudente, la deuda deliberada y la deuda involuntaria. Chris Sterling agrega a la definición la degradación de los componentes del software. No está de más notar que hay que diferenciar entre deuda técnica y malas prácticas, según Robert Martin, la deuda técnica está formada por decisiones tomadas de manera consciente en proyectos reales que aunque impliquen ciertos riesgos, pueden aportar beneficios en etapas específicas del proyecto[8]. Por otro lado, las malas prácticas o *smells* representan deficiencias en el desempeño del software y esfuerzos adicionales innecesarios para adecuaciones futuras y dificultades en el mantenimiento que pueden incurrir en la introducción de errores[13].

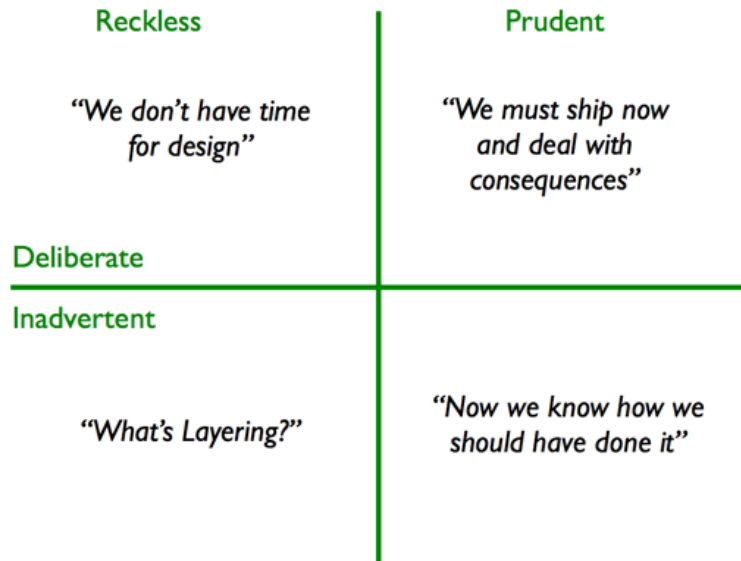


Figura 1: Cuadrante de tipos de deuda técnica. Fuente: Martin Fowler. Technical Debt[3]

Sin embargo, la definición que más se apega a la realidad del proyecto es la propuesta por M. Smit, B. Gergel, H. J. Hoover, y E. Stroulia, donde mencionan que la deuda técnica es una metáfora usada para describir la práctica de sacrificar las metas a largo plazo a cambio de un logro, barato y rápido, de metas a corto plazo, agregando situaciones en las cuales los desarrolladores aceptan sacrificios en una dimensión del desarrollo (por ejemplo, calidad de software) para optimizar otra dimensión (por ejemplo, características de implementación necesarias antes del tiempo límite)[11].

Ahora, las causas de tener deuda técnica en un proyecto de software varían, desde acelerar la producción para poder cumplir con los plazos estipulados por el cliente a pobre liderazgo, falta de colaboración, definición inicial insuficiente, falta de conocimiento, componentes acoplados, ausencia de estándares y ausencias de pruebas[4]. Todo esto puede llevar a que aumente el costo de desarrollo y/o soporte, junto a una atrofia del producto, la incapacidad de añadir nuevas funcionalidades o corregir defectos.

3. Definición del problema

3.1. Objetivo general

El principal problema que enfrentó MarIAAna durante su desarrollo fue la falta de un proceso sólido de aseguramiento de calidad (QA), lo que resultó en una deuda técnica significativa y en errores en producción. Esta situación es común en proyectos con plazos ajustados, donde el enfoque principal se centra en cumplir con las fechas de entrega de nuevas funcionalidades a expensas de un proceso de pruebas adecuado. El desarrollo de MarIAAna priorizó cumplir con los requisitos funcionales del sistema, sin dedicar suficiente tiempo a una revisión exhaustiva de la estabilidad, robustez y fiabilidad del software.

Esta falta de QA genera una situación problemática para varios actores involucrados en el proyecto. Para los desarrolladores, la acumulación de deuda técnica implica una mayor carga de trabajo post-lanzamiento al tener que corregir errores que podrían haberse evitado con pruebas más exhaustivas. Esto no solo retrasa el mantenimiento del software, sino que también afecta la capacidad del equipo para introducir nuevas funcionalidades de manera eficiente. Los usuarios finales se enfrentan a una experiencia menos confiable, con posibles fallos en el software que pueden surgir en situaciones críticas, como emergencias marítimas, lo que impacta directamente la capacidad de respuesta ante estas situaciones. Finalmente, los *stakeholders* deben dedicar más recursos y tiempo al mantenimiento y soporte de MarIAna, afectando la sostenibilidad del proyecto a largo plazo y generando una mayor inversión para solucionar problemas que podrían haberse mitigado desde el principio.

Por lo tanto, el objetivo general de este análisis es evaluar el impacto que la falta de QA tuvo sobre el desarrollo de MarIAna, identificando las principales causas de la deuda técnica y sus consecuencias. A partir de este análisis, se busca proponer un plan de mejora que permita optimizar la calidad, fiabilidad y sostenibilidad del sistema, buscando reducir la deuda técnica y mejorar la experiencia del usuario final en situaciones críticas.

Una buena forma de resumir los distintos problemas que enfrentó el desarrollo de MarIAna y las consecuencias que esto trajo es con el árbol del problema en la figura 2.



Figura 2: Árbol del problema. Fuente: Elaboración Propia



3.2. Objetivos específicos

Para lograr el objetivo general planteado, es necesario descomponer el análisis en varias áreas de investigación y acción concretas. En primer lugar, es crucial identificar las causas principales que llevaron a la acumulación de deuda técnica y a los errores en producción durante el desarrollo de MarIAana. Esto implica un análisis profundo de los plazos ajustados, la priorización de funcionalidades sobre el QA y las dificultades enfrentadas por los desarrolladores en la implementación de pruebas exhaustivas debido a la complejidad del sistema. Comprender estos factores será fundamental para trazar una línea de base desde donde se puedan proponer mejoras.

En segundo lugar, se debe analizar las consecuencias que esta falta de QA y acumulación de deuda técnica ha generado. Ejemplos de esto podrían ser: saber si incrementó la dificultad para implementar nuevas funcionalidades de manera eficiente, qué impacto se tuvo sobre la experiencia del usuario final y qué tanto afectó el no desarrollo de pruebas al software.

A partir del análisis de las causas y consecuencias, se busca proponer un plan de acción que permita mejorar el proceso de QA en MarIAana. Este plan de mejora debe enfocarse en implementar procesos más robustos de aseguramiento de calidad, incluyendo la automatización de pruebas y técnicas ágiles que refuercen la estabilidad del software. Además, se pondrá especial atención en cómo reducir la deuda técnica y optimizar los recursos, garantizando que el software sea más confiable y estable en el futuro.

Por último, el análisis también debe evaluar el impacto potencial que estas mejoras tendrían en MarIAana. Se espera que la implementación del plan de acción no solo reduzca la carga de trabajo de los desarrolladores y minimice los costos de mantenimiento, sino que también mejore significativamente la experiencia del usuario final al ofrecer un software más preciso y confiable en escenarios críticos. Este impacto positivo fortalecería la viabilidad y sostenibilidad de MarIAana como herramienta clave en el ámbito de las emergencias marítimas y contribuiría a la confianza de los *stakeholders* en el sistema.

4. Análisis de Deuda Técnica y Calidad en las Historias de Usuario

Para analizar las historias de usuario, se procedió a evaluar detalladamente sus respectivos criterios de aceptación, lo que permitió identificar y clasificar la deuda técnica asociada a cada una. Para llevar a cabo esta evaluación, se realizó un proceso de testing manual exhaustivo, que permitió verificar el cumplimiento funcional de cada criterio en condiciones reales de uso, detectar fallos, limitaciones y comportamientos inesperados. Esta metodología de prueba manual fue fundamental para comprender no solo la funcionalidad visible para el usuario, sino también las posibles dificultades y riesgos técnicos subyacentes, facilitando una identificación más precisa de la deuda técnica.

Además, se complementó esta labor con un análisis profundo del diseño y la arquitectura del software. Este análisis estructural permitió detectar problemas de modularidad, acoplamiento, escalabilidad y mantenibilidad, aspectos críticos para la sostenibilidad del sistema a

largo plazo. La combinación del testing manual con el análisis arquitectónico proporcionó una visión integral, que permitió extraer de forma sistemática la deuda técnica correspondiente a cada historia de usuario.

Como ejemplo representativo de este proceso, se presenta la tabla 11 (las tablas restantes se incluyen en el anexo) y en la tabla 10 se muestra un resumen consolidado de la deuda técnica identificada para cada historia de usuario.

Historia de Usuario	Deuda Técnica Asociada
H1.1: Visualización; Interacción del usuario con el mapa	Código acoplado directamente al frontend, sin separación de lógica de visualización ni componentes reutilizables.
H1.2: Visualización; Proyección de la predicción	Reproducción sin control de velocidad ni barra de progreso; ausencia de optimización para animaciones largas.
H2.1: Predicción; Dominio y línea de Costas con IA	Falta validación de entrada (fechas inconsistentes, dominio sobre tierra) y feedback de errores al usuario.
H2.2: Predicción; MDP Modelo Local de alta precisión	Lógica de predicción monolítica, sin tests de unidad ni separación de responsabilidades (ingreso, simulación, visualización).
H2.3: Predicción; MDP Modelo Global	Ausencia de automatización en la descarga/preprocesamiento de datos climáticos; integración manual y propensa a errores.
H2.4: Predicción; Generación de Grillas	Generación de grilla acoplada al código de simulación, sin posibilidad de reutilización o prueba independiente.
H2.5: Predicción; Línea de Costas con IA	Línea de costa procesada sin diferenciación por zona, generando uso innecesario de recursos computacionales.
H3.1: Rendimiento; Estadísticas	Métricas incompletas o desactualizadas; no hay monitoreo activo del rendimiento ni logging estructurado.
H3.2: Rendimiento; Historial de dominios y visualizaciones	Predicciones previas no almacenadas o accesibles de forma limitada, sin filtros ni recuperación eficiente.

Cuadro 10: Resumen de deuda técnica por historia de usuario

Entrando en detalle, en el análisis de las historias de usuario relacionadas con la visualización (HU1.1 y HU1.2), se observa que la implementación cumple parcialmente con los criterios de aceptación, pero arroja déficits significativos en la experiencia de usuario y en la flexibilidad de los componentes gráficos. El control de velocidad y reproducción de la animación es funcional, pero restringido a dos escalas extremas ($\times 1$ y $\times 60$), lo que genera tiempos de espera inaceptables (una predicción de nueve días tarda nueve horas en reproducirse) y obstaculiza la evaluación rápida de resultados. Asimismo, la imposibilidad de personalizar



dinámicamente el número de partículas en pantalla y la carencia de una capa vectorial para representar corrientes limitan la utilidad analítica de la herramienta. Esta acumulación de deudas técnicas en los módulos de renderizado y de interfaz sugiere la necesidad de modularizar y abstraer dichos componentes para facilitar futuras mejoras en el rendimiento y en la interactividad.

En cuanto a las historias centradas en predicción y dominio geográfico (HU2.1–HU2.5), la mayor parte de los criterios de aceptación quedó satisfecho a nivel conceptual, pero sin un soporte robusto en el código. La definición manual de dominios espacial y temporal funciona, aunque sin validaciones suficientes, lo que puede derivar en errores de entrada y fallos en la descarga de datos. La generación y refinamiento de la línea de costa se comporta según lo esperado, pero adolece de puntos mal clasificados por el modelo de IA y de métodos estáticos que no discriminan entre tierra y mar. Además, la arquitectura monolítica del código de predicción (agrupando múltiples funcionalidades en un mismo archivo) dificulta la trazabilidad y el mantenimiento, mientras que la falta de integración completa de las etapas de preprocesamiento de mallas y de datos hidrodinámicos evidencia una deuda técnica de alcance medio que compromete la precisión y escalabilidad del sistema.

Finalmente, las historias de rendimiento e historial (HU3.1 y HU3.2) muestran un cumplimiento básico de métricas, como tiempos de ejecución y almacenamiento cronológico de visualizaciones, pero con funcionalidades deshabilitadas en la interfaz o implementadas de forma parcial. El cálculo del área de búsqueda reducida existe, pero no se promociona adecuadamente en la bandeja del usuario; los filtros y la reutilización de dominios en el historial no llegaron a materializarse, lo que genera un desajuste entre lo planificado y lo desplegado. De cara a la sostenibilidad del proyecto, resulta prioritario externalizar los procesos de monitoreo y registro en módulos especializados, activar los componentes de filtrado y reactivar las vistas deshabilitadas, con el fin de cerrar brechas de calidad, reducir el coste de mantenimiento y mejorar la percepción de fiabilidad por parte de los usuarios.



HU2.3 Predicción; MDP Modelo Global	
Criterio de aceptación	Análisis de calidad
La predicción se calcula a través de un punto geográfico y ventana temporal establecidos por el usuario.	La predicción se calcula a través de los dominios establecidos, pero estos tienen que estar cargados en la nube, en caso de que no lo estén, el modelo de predicción se cae ya que no logra descargar los datos correspondientes.
La predicción se calcula utilizando los datos climáticos de fuentes públicas.	Sí, los datos provienen de fuentes públicas.
Los datos llegarán a la plataforma de visualización vía streaming.	Sí, los datos llegan vía streaming mediante un modelo de datos personalizados para el contexto del software.
Se va a poder elegir entre las distintas fuentes de datos públicos.	No se puede elegir entre distintas fuentes de datos públicos para realizar la predicción, solo es posible con la que está por defecto en el software.
Deuda Técnica.	Gran parte de la deuda técnica se va en poder elegir las distintas fuentes de datos. Todo el proyecto se confiere alrededor de solo una fuente de datos, con tal de acelerar el desarrollo. Además, la parte de descarga de datos no llega a funcionar, lanzando un error en caso de seleccionar un dominio temporal fuera de los ya almacenados en la nube. Por otro lado, todas las funciones que componen al modelo de predicción están definidas en un mismo archivo, lo que puede llevar a dificultades a la hora de mantener, probar y escalar el proyecto. Además, internamente se puede elegir si ocupar solo las fuerzas de marea o agregar las fuerzas del viento al modelo predictivo, pero esto también no se llegó a implementar en la interfaz de usuario.

Cuadro 11: Análisis de Calidad y Deuda Técnica H2.3

5. Propuesta de mejora

Con el objetivo de abordar de manera eficiente la deuda técnica acumulada y avanzar hacia una segunda versión más robusta del sistema, se propone dividir el desarrollo en dos etapas principales. Esta división responde a la necesidad de organizar el trabajo de forma progresiva, priorizando inicialmente la estabilización del sistema y la entrega de un prototipo funcional que permita realizar demostraciones y validaciones tempranas, antes de abordar los desafíos técnicos de mayor complejidad. Al separar las tareas según su impacto funcional y carga técnica, se facilita la planificación, la asignación de recursos y la gestión del riesgo, permitiendo avances sostenidos sin comprometer la calidad del producto final.

5.1. Etapa 1: Prototipo funcional y saneamiento técnico

En esta primera fase, se trabajará prioritariamente sobre las historias HU1.1, HU1.2, HU2.3, HU3.1 y HU3.2. Estas historias están mayormente relacionadas con aspectos visibles y funcionales del sistema, como la interfaz de usuario, la visualización de resultados y el acceso a datos climáticos globales provenientes de fuentes públicas. La elección de estas historias responde a su alta relevancia para la experiencia del usuario y su potencial para mostrar el valor central del sistema, incluso si aún no se han incorporado todas las capacidades de modelado avanzado.

El objetivo principal de esta etapa es generar un prototipo funcional pulido y coherente, capaz de demostrar el flujo general del sistema de manera realista, incluyendo la simulación visual de predicciones, la navegación entre vistas, y el historial de resultados obtenidos. Esto permite no solo validar decisiones de diseño y arquitectura tempranas, sino también recoger retroalimentación de usuarios y stakeholders clave sin necesidad de contar aún con la totalidad del sistema implementado. Asimismo, trabajar sobre estas historias facilita la identificación temprana de errores estructurales, permitiendo su corrección antes de que afecten el desarrollo de componentes más complejos. De esta manera, se establece una base sólida sobre la cual se podrá construir la segunda etapa del sistema con mayor seguridad y eficiencia.

Para realizar esto, se tienen las siguientes acciones a realizar sobre las historias de usuario mencionadas:

Historia	Acciones concretas para reducir deuda técnica
HU1.1	Agregar controles precisos de tiempo y velocidad; mejorar visualización de líneas vectoriales (animación fluida, capas); refactorizar código de visualización para que soporte ajustes dinámicos por parte del usuario; mejorar la visualización de los distintos puntos de la predicción.
HU1.2	Incorporar control sobre número de partículas y selección de sus posiciones iniciales; rediseñar la visualización del área de error.
HU2.3	Habilitar selección de distintas fuentes de datos climáticos; implementar fallback en caso de que no haya datos disponibles; mejorar logging de errores y modularidad del código de integración con APIs externas.
HU3.1	Reactivar y rediseñar el dashboard de métricas; definir un set mínimo de indicadores clave (ej. tiempo de ejecución, precisión, error de predicción); agregar visualizaciones interactivas.
HU3.2	Refinar estructura del historial para incluir metadatos clave (modelo, parámetros, resultado); añadir filtros funcionales; permitir reutilización directa del dominio desde el historial; validar almacenamiento solo de predicciones exitosas.

Cuadro 12: Acciones concretas para reducir la deuda técnica en la etapa 1

Se busca aquí reducir la deuda técnica más superficial y corregir fallas estructurales o de integración ligera, aprovechando que estas historias están más avanzadas o requieren menor esfuerzo técnico que otras.

5.2. Etapa 2: Integración de funcionalidades científicas

Una vez estabilizado el prototipo funcional desarrollado en la primera etapa, se dará paso al abordaje de las historias HU2.1, HU2.2, HU2.4 y HU2.5. Estas historias se caracterizan por implicar una mayor carga técnica y científica, debido a la naturaleza compleja de los algoritmos y procesos involucrados. Entre estos se incluyen la validación geoespacial del dominio de simulación, la ejecución de modelos hidrodinámicos, el uso de inteligencia artificial para el delineamiento de líneas de costa y la generación automatizada de grillas espaciales. La implementación de estas funcionalidades requiere una comprensión profunda de los aspectos matemáticos y físicos del sistema, así como una integración estrecha entre componentes del frontend, backend y servicios externos especializados.

Dado el nivel de dificultad y la interdependencia entre tareas, esta etapa se enfocará en una planificación detallada y adaptable. Se propone, si es necesario, reformular y subdividir las historias de usuario en tareas más pequeñas y manejables, lo que permitirá una implementación progresiva y modular. Esta estrategia facilita la identificación y resolución temprana de cuellos de botella técnicos, además de permitir pruebas incrementales que garanticen la estabilidad del sistema a medida que se incorporan nuevas capacidades. Asimismo, al tratarse de componentes de alto impacto en la precisión y utilidad del sistema, se priorizará el uso de buenas prácticas de diseño de software, validación de resultados y aseguramiento de calidad, con el fin de evitar la acumulación de nueva deuda técnica durante el desarrollo.

De manera específica, las siguientes acciones se pueden realizar para reducir la deuda técnica.

Historia	Acciones concretas para reducir deuda técnica
HU2.1	Separar los componentes de validación espacial y temporal del dominio; mejorar interfaz para evitar selección inválida (p. ej. tierra); usar herramientas geoespaciales para asegurar continuidad y conexión al mar de la línea de costa; automatizar refinamiento de la línea costera.
HU2.2	Integrar proceso completo desde UI a ejecución del modelo; diseñar una interfaz para elegir modelo predictivo; refactorizar backend para manejar el flujo de datos de forma automática y transparente al usuario; automatizar el proceso de generación de datos para modelos de alta precisión.
HU2.4	Incorporar algoritmo de generación de grillas directamente en el sistema; permitir visualización previa de la grilla generada; asegurar densificación adaptativa cerca de islas y costas mediante técnicas de suavizado espacial.
HU2.5	Agregar post-procesamiento a la salida de la IA para limpiar puntos erróneos; refinar filtros para asegurar continuidad y fidelidad geográfica de la línea costera; crear herramientas visuales para ajustar manualmente el resultado si es necesario.

Cuadro 13: Acciones concretas para reducir la deuda técnica en la etapa 2

La aplicación de este plan de mejora permitirá evolucionar hacia una versión del sistema

mucho más robusta y mantenible. En la figura 3, se presenta una imagen referencial que ilustra un posible resultado final luego de implementar las acciones propuestas, en donde se eliminó la deuda técnica correspondiente a las historias de usuario relacionadas con la visualización.

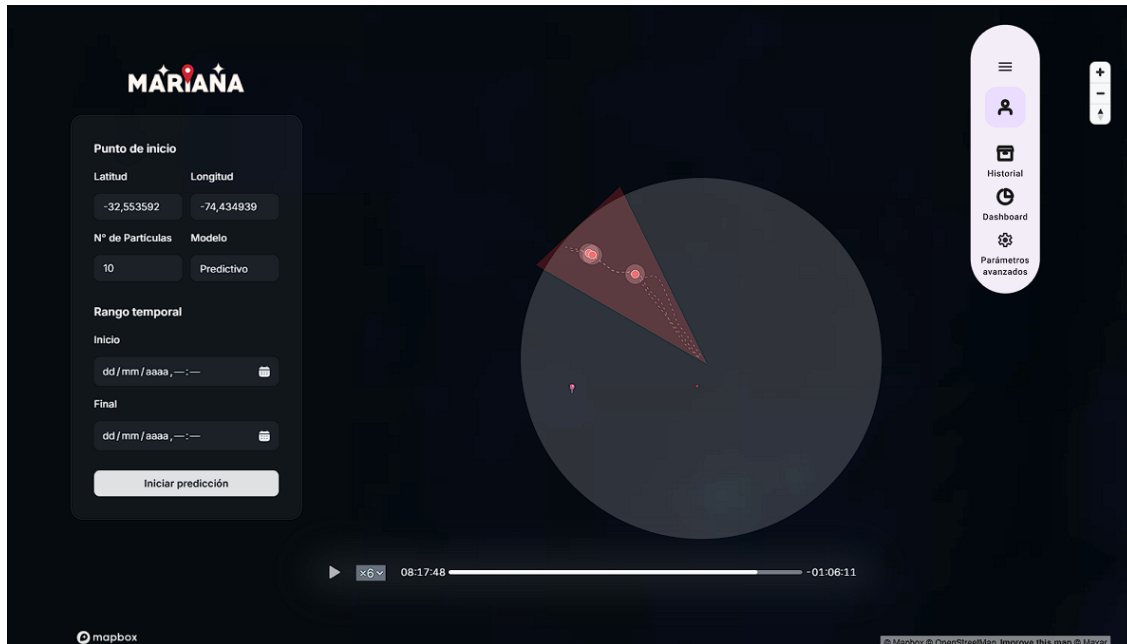


Figura 3: Vista conceptual del sistema mejorado luego de aplicar el plan de reducción de deuda técnica.

6. Conclusión

El desarrollo de MarIAna representó un desafío significativo en términos técnicos, funcionales y organizativos, especialmente al estar enmarcado en un entorno académico exigente como la Feria de Software. Si bien se logró construir un sistema funcional con un alto potencial de impacto en la gestión de emergencias marítimas, el proceso de desarrollo priorizó la entrega rápida de resultados visibles, dejando en segundo plano prácticas fundamentales de aseguramiento de calidad. Esto se tradujo en una acumulación considerable de deuda técnica, evidenciada tanto en fallas de diseño arquitectónico como en la implementación parcial o deficiente de varios criterios de aceptación.

El análisis detallado de las historias de usuario permitió identificar con precisión los puntos críticos que afectan la mantenibilidad, escalabilidad y usabilidad del sistema. A través de testing manual y revisión estructural del software, se documentaron diversas carencias técnicas relevantes. Entre ellas, destaca la ausencia de validaciones para dominios temporales y geográficos (por ejemplo, permitir fechas invertidas o seleccionar zonas terrestres para predicciones marítimas), la falta de control granular sobre la reproducción de la simulación, y el acoplamiento excesivo de funcionalidades clave en archivos únicos de gran tamaño, lo que



dificulta el mantenimiento y la extensión del sistema. Asimismo, se detectaron implementaciones parciales de criterios de aceptación, como en la visualización de líneas vectoriales, la representación del área de error de las predicciones o la imposibilidad de seleccionar distintas fuentes de datos climáticos.

Junto con estos déficits, se identificaron también importantes oportunidades de mejora. Por ejemplo, modularizar el reproductor de la visualización permitiría incorporar controles más precisos y reutilizar el componente en distintas vistas; integrar mecanismos automáticos de validación de entradas reduciría errores de predicción desde la interfaz; y dividir funcionalidades complejas como la generación de grillas o la línea de costa en submódulos independientes facilitarían su implementación incremental. Esta evaluación motivó la elaboración de una propuesta de solución estructurada en dos etapas: una primera orientada a estabilizar el sistema mediante el saneamiento técnico de sus componentes fundamentales: mejorando la interfaz, el historial y la visualización básica. Y una segunda destinada a integrar capacidades avanzadas, como la predicción de alta precisión y la configuración dinámica de modelos, de manera modular y escalable.

En definitiva, esta memoria no solo ofrece un diagnóstico claro sobre el estado actual del sistema, sino que también sienta las bases para su evolución futura. La implementación del plan de mejora propuesto permitirá reducir la deuda técnica, optimizar la calidad del software y entregar una segunda versión de MarIAna más robusta, confiable y alineada con las exigencias de su contexto de uso. Así, se contribuye no solo al fortalecimiento del proyecto en sí, sino también al aprendizaje y aplicación efectiva de buenas prácticas en el desarrollo ágil de software.

Referencias

- [1] Amadeu Silveira Campanelli y Fernando Silva Parreiras. “Agile methods tailoring – A systematic literature review”. En: *Journal of Systems and Software* 110 (2015), págs. 85-100. ISSN: 0164-1212. DOI: <https://doi.org/10.1016/j.jss.2015.08.035>. URL: <https://www.sciencedirect.com/science/article/pii/S0164121215001843>.
- [2] Ward Cunningham. “The WyCash portfolio management system”. En: *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum)*. OOPSLA '92. Vancouver, British Columbia, Canada: Association for Computing Machinery, 1992, págs. 29-30. ISBN: 0897916107. DOI: 10.1145/157709.157715. URL: <https://doi.org/10.1145/157709.157715>.
- [3] Martin Fowler. *Technical Debt*. Disponible en <https://martinfowler.com/bliki/TechnicalDebt.html>. Última vez visitado el 18 de Marzo de 2025.
- [4] Jesús José Lárez Mata. “Calidad de Software-Deuda Técnica”. En: *Repositorio Institucional Saber UCAB* (2020).
- [5] Universidad Técnica Federico Santa María. *Historia Feria de Software*. Disponible en <https://www.feriadesoftware.cl/historia>. Última vez visitado el 10 de Septiembre de 2024.



- [6] E. Mnkandla y B. Dwolatzky. “Defining Agile Software Quality Assurance”. En: *2006 International Conference on Software Engineering Advances (ICSEA '06)*. 2006, págs. 36-36. DOI: 10.1109/ICSEA.2006.261292.
- [7] Myint Myint Moe. “Comparative study of test-driven development tdd, behavior-driven development bdd and acceptance test-driven development atdd”. En: *International Journal of Trend in Scientific Research and Development* 3.4 (2019), págs. 231-234.
- [8] M R. *A Mess is not a Technical Debt*. Disponible en <https://sites.google.com/site/unclebobconsultingllc/a-mess-is-not-a-technical-debt>. Última vez visitado el 18 de Marzo de 2025.
- [9] Sakshi Sachdeva. “Scrum methodology”. En: *Int. J. Eng. Comput. Sci* 5.16792 (2016), págs. 16792-16800.
- [10] Mehreen Sirshar. “Evaluation of Quality Assurance Factors in Agile Methodologies.” En: *International Journal of Advanced Computer Science* 2 (mar. de 2012), págs. 73-78.
- [11] Michael Smit et al. “Code convention adherence in evolving software”. En: *2011 27th IEEE International Conference on Software Maintenance*. Sep. de 2011, págs. 504-507. DOI: 10.1109/ICSM.2011.6080819.
- [12] Carlos Solis y Xiaofeng Wang. “A Study of the Characteristics of Behaviour Driven Development”. En: *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. 2011, págs. 383-387. DOI: 10.1109/SEAA.2011.76.
- [13] Carlos Zapata, María Alvarez y Juan Palencia. “SMELLWARE: un juego para la enseñanza de buenas prácticas en el proceso de desarrollo de software”. En: *Ingeniare. Revista chilena de ingeniería* 28 (dic. de 2020), págs. 645-653. DOI: 10.4067/S0718-33052020000400645.

7. Anexo

En las siguientes tablas se realizó el análisis de calidad y deuda técnica de las HUs de MarIAna.



HU1.1 Visualización; Interacción del usuario con el mapa	
Criterio de aceptación	Análisis de calidad
Necesito controlar la velocidad y reproducción de la visualización de la predicción. Cambiar elementos visuales del mapa.	Se puede controlar la velocidad y reproducción de la visualización, la velocidad va solo a x1 o x60, la reproducción es muy lenta, en x60 cada minuto es un segundo, por tanto, una predicción de 9 días se demora 9 horas en reproducirse por completo, Se podría mejorar la experiencia de usuario de esto. Los elementos visuales no se pueden cambiar.
Visualizar líneas vectoriales de las corrientes.	No se muestran las líneas vectoriales de las corrientes, si bien se trató de implementar, no se podía hacer dinámico para distintas predicciones, solo para un momento en el tiempo en específico.
Se debe mostrar, en todo punto de la predicción, toda la trayectoria que ha recorrido y recorrerá la partícula a la deriva.	Se visualiza la trayectoria de la partícula, pero es posible que el punto de inicio y el punto final de la trayectoria estén conectados por la línea de la predicción.
El reproductor de la predicción debe indicar hasta qué instante del tiempo se tienen los datos de la predicción de acuerdo con la información recibida vía streaming.	Se cumple en el sentido de que se muestra una barra en el menú de reproducción que se va cargando mientras se van recibiendo los datos.
Múltiples datos de viento almacenados en el caché para predecir distintos días.	Se tienen datos almacenados en la nube con datos de viento y marea para poder hacer la predicción, con un margen de 30 días.
Interpolación de la posición si no existe la predicción para ese punto específico	Se tiene una interpolación entre los distintos puntos
Se ve la línea de costa, respetando las islas y el terreno costero.	Se ve la línea de costa, pero al momento de graficarla hay problemas con cómo se representan la unión de los puntos, con líneas que no corresponden a la costa.
Deuda Técnica	Acá la mayor deuda técnica se lo lleva la parte de la visualización de las líneas vectoriales de fuerzas de marea, ya que se logró implementar a nivel de funcionalidad, pero no se pudo adaptar a que funcione con distintas predicciones, que ocupan distintos datos de mareas, y se dejó en pendiente para poder resolverlo en un futuro. Lo mismo ocurre con el CA de líneas de predicción, que se sabía del problema de que se conectan los puntos iniciales y finales de la predicción, pero no se corrigió por seguir desarrollando las otras funcionalidades.

Cuadro 14: Análisis de Calidad y Deuda Técnica H1.1



HU1.2 Visualización; Proyección de la predicción	
Criterio de aceptación	Análisis de calidad
El usuario podrá visualizar una cantidad definida de partículas en el mapa.	Hay una cantidad definida de partículas en el mapa, sin embargo, no se puede elegir cuantas partículas se quieren ver.
Las partículas se mueven en el tiempo.	Las partículas se mueven en el tiempo, pero por las escalas se mueven muy lento, se puede acelerar, pero no se llega a corregir el problema del todo.
Se puede visualizar un área de error.	Esta parte se puede interpretar de dos maneras, un área de error en la posición de las partículas, en este caso, un pulso, pero que depende más del zoom que se le tenga aplicado al mapa para saber si el error es muy grande o pequeño. La otra opción es el área de error total de la predicción, es decir, el cono de posibles ubicaciones de la partícula, en este caso, esto no fue desarrollado.
Deuda Técnica	Con respecto al CA de visualizar cierta cantidad de partículas, se puede internamente cambiar la cantidad, pero no se llegó a implementar en la interfaz de usuario para cambiar esa cantidad.

Cuadro 15: Análisis de Calidad y Deuda Técnica H1.2



HU2.1 Predicción; Dominio y línea de Costas con IA	
Criterio de aceptación	Análisis de calidad
Declarar manualmente el dominio geográfico.	El dominio geográfico sí se puede declarar manualmente, tanto con las coordenadas de latitud y longitud como haciendo click en el mapa. Sin embargo, no se hace distinción entre tierra y mar, lo que puede generar problemas al predecir.
Declarar dominio temporal.	El rango temporal se declara con una fecha de inicio y fin, pero no hay validación (por ejemplo, fecha fin menor que inicio). Si la fecha no está previamente en la nube, falla la descarga de datos y el frontend queda esperando.
Se genera una línea de costa dependiendo del dominio.	Se genera la línea de costa según el dominio, pero incluye puntos erróneos del modelo o bordes del dominio. Si el dominio está lejos del mar, solo aparece un punto pequeño.
La línea de costa es más fina en las partes más cercanas al inicio del dominio.	Esto no se implementó, ya que el modelo de IA no fue entrenado para esta funcionalidad.
La línea de costa es continua dentro del dominio.	Técnicamente no es continua; el modelo entrega una serie de puntos.
Todos los puntos de la línea de costa deben estar conectados al mar abierto	Estos dos requisitos dependen de qué tan bueno es el modelo de inteligencia artificial, así que se escapan de lo que busca evaluar esta memoria.
La línea de costa no pasa sobre terreno costero.	
Deuda Técnica.	El archivo en donde están definidas todas las funciones para hacer el modelo de línea de costa es muy grande, debería estar separado para una mejor organización del modelo.

Cuadro 16: Análisis de Calidad y Deuda Técnica H2.1



HU2.2 Predicción; MDP Modelo Local de alta precisión	
Criterio de aceptación	Análisis de calidad
La predicción se calcula a través de la declaración de un dominio geográfico y temporal definido por el usuario.	Si bien se establecen los dominios, para poder hacer el cálculo es necesario hacer los archivos de datos, estos a través de modelos hidrodinámicos para saber el movimiento de las mareas en el tiempo dado y esto no llega a implementarse correctamente en el software, pero en caso de tener los archivos necesarios, la predicción se calcula a través de los dominios establecidos.
Los datos necesarios para la predicción estarán dados por el dominio establecido.	Ocurre lo mismo que con el criterio de aceptación anterior, las etapas para realizar los datos de archivos están definidas, pero no se llegaron a implementar.
Los datos llegarán a la plataforma de visualización vía streaming.	Los datos se envían a la plataforma vía streaming a través de un modelo de datos personalizado al contexto del software.
La predicción, al acercarse a costa, se va haciendo más precisa.	Esto no se llegó a comprobar, por tanto, no se puede afirmar que esto llega a suceder.
La predicción se calcula a través de datos climáticos interpolados a partir de los modelos de fuentes públicas.	Este criterio de aceptación nunca se llegó a implementar en el software como tal, solo se tenían las herramientas necesarias para hacerlo.
Deuda Técnica.	En primer lugar, de toda esta historia solo se llegó a implementar el modelo de predicción, pero para poder hacer la predicción completa hay que pasar por varias etapas, primero, generar una grilla triangular no estructurada según el dominio establecido; segundo, pasar esta grilla a un formato de dato necesario para el modelo hidrodinámico de fuerzas de marea; tercero, con el formato de dato de la grilla correcto y los datos de fuerzas de mareas de fuentes públicas se ejecuta el modelo hidrodinámico, con el que obtenemos los datos más finos para poder hacer la predicción, ya por último, tenemos la ejecución de la predicción del movimiento de las partículas como tal. La cuestión es, que todas estas etapas se encontraron con problemas al momento de unir las, al menos para esta historia de usuario. Además, el usuario no puede seleccionar el modelo de predicción en la plataforma, es necesario cambiar el código desde el backend para poder realizar la predicción con este modelo junto a los archivos de datos necesarios.

Cuadro 17: Análisis de Calidad y Deuda Técnica H2.2



HU2.4 Predicción; Generación de Grillas	
Criterio de aceptación	Análisis de calidad
Se genera una grilla alrededor del punto de inicio del problema.	Todos estos CA son cumplidos, la mayoría por el simple hecho de que el algoritmo para generar la malla genera esas características.
Los elementos de la grilla no se superponen entre sí.	
La grilla se vuelve más densa cerca de la costa.	
La grilla se vuelve más densa cerca de las islas.	
La grilla es completamente conexa.	
La grilla se ve limitada por los bordes costeros	
La grilla se genera dentro del terreno en ríos y lagos	Estos CA no se lograron realizar por errores en los módulos que realizaban la transformación al formato de dato necesario para el modelo hidrodinámico y generaban los demás datos que se necesitaban para ejecutar el modelo hidrodinámico.
Se generan puntos de esponja en la grilla	
La grilla toma en consideración los puntos de esponja y es conexa con estos	
Deuda Técnica.	Toda esta funcionalidad del software no se llegó a implementar como tal, se desarrolló a parte del software con tal de que después se agregara, pero esto generó dificultades a la hora de realizarlo, al punto de que no se logró. Por otro lado, cuando se trató de implementar las funcionalidades de generar los puntos de esponja y los demás datos necesarios para poder ejecutar el modelo hidrodinámico, se encontraron problemas con el módulo que realizaba estas funciones, y si bien se trataron de arreglar, al final no se pudieron corregir estos problemas.

Cuadro 18: Análisis de Calidad y Deuda Técnica H2.4



HU2.5 Predicción; Línea de Costas con IA	
Criterio de aceptación	Análisis de calidad
Se genera una línea de costa dependiendo del dominio.	En general, se cumplen estos criterios, porque la inteligencia artificial desarrollada realiza todo esto, pero el analizar la calidad de esta IA se escapa del alcance de esta memoria.
La línea de costa es más fina en las partes más cercanas al inicio del dominio.	
La línea de costa es continua dentro del dominio.	
Todos los puntos de la línea de costa deben estar conectados al mar abierto.	
La línea de costa no pasa sobre terreno costero	
Deuda Técnica.	Un problema que ocurre es que se agregan puntos que no son necesarios al momento de la visualización de la línea de costa en la interfaz de usuario, como lo pueden ser los límites que se tienen en cuenta para generar la línea de costa, que se agregan como puntos a visualizar, o errores en la misma visualización, como que se conecten el primer punto de la visualización con el último punto o puntos que no pertenecen a la línea de costa, pero que son entregados por la IA, donde no se realiza la diferenciación entre unos u otros.

Cuadro 19: Análisis de Calidad y Deuda Técnica H2.5



HU3.1 Rendimiento; Estadísticas	
Criterio de aceptación	Análisis de calidad
Tiempo de ejecución modelo según la o las bases de datos escogidas y el método para la predicción.	Para este criterio de aceptación, si se calcula el tiempo de ejecución del modelo, pero solo para un modelo en específico y con solo una fuente de datos públicos.
Posición final de la predicción de la partícula.	Para estos criterios en general no hubo mayor problema y se cumplen perfectamente
Cantidad de puntos generados.	
Área de búsqueda reducida.	Si bien se calcula el área de búsqueda reducida una vez terminada la predicción, no se le dio la importancia que se le tenía que dar, ya que parte de la propuesta de valor del software cae en esta parte.
Deuda Técnica.	Esta historia fue desarrollada e implementada, pero después fue desactivada en la interfaz de usuario con tal de desarrollarla mejor en un futuro, con una mejor vista, cosa que no se terminó realizando y por tanto quedo al debe.

Cuadro 20: Análisis de Calidad y Deuda Técnica H3.1



HU3.2 Rendimiento; Historial de dominios y visualizaciones	
Criterio de aceptación	Análisis de calidad
La sección de historial muestra en orden cronológico las predicciones realizadas por el usuario.	Se muestran todas las predicciones que realiza el usuario, incluso las que generaron algún error en el sistema.
Cada visualización contiene información relevante de la predicción.	Cada visualización realizada muestra el rango temporal en el que fue realizada y el punto de inicio de la predicción, junto con hace cuanto tiempo fue realizada.
Las visualizaciones tienen la opción de volver a verlas.	Sí, las visualizaciones se pueden volver a ver, incluso las que no tienen datos almacenados.
Filtros (Tiempo, lugar, tipo de partícula.	Ningún filtro fue implementado en el historial.
Se guardarán los dominios previamente generados.	No se guarda ningún dominio geográfico generado.
En caso de coincidir espacial y temporalmente, se podrá reutilizar un dominio previamente generado para una nueva visualización.	No se llegó a implementar, si coincide un dominio, cual sea, se va a volver a calcular todo nuevamente.
Deuda Técnica.	La gran mayoría del historial cumple con lo mínimo, que es guardar las visualizaciones del usuario y la información importante de la misma. Además, si se genera algún error al momento de hacer la predicción, esta "visualización" igual quedara guardada en el historial.

Cuadro 21: Análisis de Calidad y Deuda Técnica H3.2