Gestión de proyectos con Redmine

– – – – – – – – – – – – – – – – –

Project management with Redmine

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
CURSO 2022–2023

Lucas Iván Sánchez Correa

Directores
Luis Javier García Villalba
Ana Lucila Sandoval Orozco

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, Mayo de 2023

Agradecimientos

En primer lugar me gustaría agradecer el trabajo de mis directores, Luis Javier y Ana Lucila por todo el apoyo dado durante la realización de este trabajo.

Asimismo, me gustaría agradecer a Pablo y a Luis Miguel por sus comentarios y ayudas durante el desarrollo del gestor de proyectos.

Por último, gracias a Agustín, Daniel y Gonzalo por demostrarme que la confianza en los demás les hace más fuertes.

Índice General

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

CPM	*Critical Path Method*
CRUD	*Create, Read, Update, Delete*
FDD	*Feature Driven Development*
MVC	*Modelo-Vista-Controlador*
WIP	*Work in progress*
XP	*Extreme Programming*

Abstract

With the increasing complexity of bureaucracy, the National Police needs an efficient system to manage projects, tasks and resources effectively. This final degree project explores the importance of using Redmine, a powerful project management tool, to automate various project-related activities. Specifically, it focuses on the transition from a daily work to a centralized database in Redmine. The move to Redmine brings numerous benefits, including improved collaboration, increased data accuracy, real-time tracking, and extensive reporting capabilities. By leveraging the automation features offered by Redmine, project managers can streamline their workflows, allocate resources effectively and achieve greater project efficiency.

Keywords: Automation, Collaboration, Project Efficiency, Project management, Redmine, Task Automation.

Resumen

Con la creciente complejidad de la burocracia, la Policía Nacional necesita un sistema eficiente para gestionar proyectos, tareas y recursos de forma eficaz. Este Trabajo Fin de Grado explora la importancia de utilizar Redmine, una potente herramienta de gestión de proyectos, para automatizar diversas actividades relacionadas con los proyectos. En concreto, se centra en la transición de una base de trabajo diaria a una base de datos centralizada en Redmine. El cambio a Redmine aporta numerosas ventajas, como una mejor colaboración, una mayor precisión de los datos, un seguimiento en tiempo real y amplias capacidades de elaboración de informes. Al aprovechar las funciones de automatización que ofrece Redmine, los gestores de proyectos pueden optimizar sus flujos de trabajo, asignar recursos de forma eficaz y lograr una mayor eficiencia en los proyectos.

Keywords: Automatización, Automatización de tareas, Colaboración, Eficiencia de proyectos, Gestión de proyectos, Redmine.

Capítulo 1

Introducción

1.1. Motivación

Las herramientas cuya finalidad es la gestión de proyectos han demostrado ser activos indispensables para organizaciones de diversos sectores. Estas herramientas sirven como soluciones integrales para agilizar los flujos de trabajo de los proyectos, mejorar la colaboración y garantizar el éxito en la entrega de los proyectos [2]. La motivación que subyace al desarrollo y la utilización de herramientas de gestión de proyectos se deriva de varios factores clave que abordan los desafíos propuestos a los equipos de proyectos y las organizaciones. Esta sección explora las principales motivaciones que subyacen a la adopción de gestores de proyectos y su impacto en el correcto desarrollo de un proyecto.

Una de las principales motivaciones para utilizar herramientas gestoras de proyectos es lograr una gestión eficiente de las tareas. Estas herramientas proporcionan una plataforma centralizada en la que los equipos de proyecto pueden crear, asignar y realizar un seguimiento de las tareas a lo largo del desarrollo del proyecto. Gracias a las funciones de gestión de tareas, los miembros del equipo pueden priorizar las asignaciones, fijar plazos y supervisar el progreso. Esta motivación surge de la necesidad de poder distribuir de una manera óptima de los recursos disponibles, evitar retrasos y cumplir eficazmente los hitos del proyecto.

La colaboración eficiente resulta fundamental para lograr el éxito del proyecto, especialmente cuando se trata de equipos multifuncionales o entornos descentralizados. Las herramientas gestionadoras de proyectos ofrecen funciones de colaboración como calendarios compartidos, documentos compartidos, foros de debate y canales de comunicación en tiempo real. Al facilitar una colaboración fluida, estas herramientas promueven el intercambio de conocimientos, la transparencia y la toma de decisiones eficaz entre los integrantes del equipo de trabajo. La motivación para mejorar la colaboración es poder ser capaz de garantizar la comunicación eficiente, fomentar el trabajo realizado de manera grupal y, en última instancia, impulsar el éxito del proyecto.

La gestión eficiente de los recursos es crucial para el éxito del proyecto [3]. Las herramientas de gestión de proyectos ofrecen funciones para planificar, asignar y hacer un seguimiento eficaz de los recursos. Con funciones como calendarios de recursos, seguimiento de las asignaciones de tareas y visibilidad de los recursos disponibles, estas herramientas ayudan a los gestores de proyectos a optimizar la asignación de recursos, evitar la sobrecarga o infrautilización y garantizar que se están asignando los recursos adecuados

a las tareas correctas en el momento oportuno. La motivación detrás de la planificación y asignación de recursos es maximizar la utilización de los recursos, reducir los puntos de congestión y aumentar la eficiencia general del proyecto.

El seguimiento del progreso del proyecto y la generación de informes esclarecedores son esenciales para que las partes interesadas se mantengan informadas y tomen decisiones fundamentadas. Las herramientas gestoras de proyectos ofrecen sólidas capacidades de seguimiento e informes, lo que permite a las personas designadas para gestionar proyectos supervisar los hitos, realizar un seguimiento de las tareas y generar informes personalizados sobre el estado del proyecto, la utilización de los recursos, el presupuesto y otras métricas críticas. La motivación detrás de la supervisión continua del progreso y la presentación de informes es proporcionar a las partes interesadas información precisa y actualizada, identificar posibles riesgos o retrasos y permitir intervenciones oportunas para el éxito del proyecto.

Todo proyecto conlleva riesgos inherentes que pueden afectar a sus resultados. Las herramientas de gestión de proyectos permiten identificar, evaluar y mitigar los riesgos. Estas herramientas permiten a los equipos de proyecto documentar los riesgos, asignar propietarios, definir estrategias de mitigación y realizar un seguimiento del progreso de la mitigación de riesgos. La motivación de la administración de posibles riesgos es identificar y abordar proactivamente los riesgos potenciales, minimizar su impacto que puedan tener en la finalidad del proyecto y fortalecer la resiliencia de las partes implicadas en el desarrollo de cualquier actividad.

Es decir, la motivación detrás de las herramientas gestoras de proyectos radica en abordar los desafíos que enfrentan los equipos de proyecto, mejorar la colaboración, optimizar la asignación de recursos, permitir el seguimiento del progreso y gestionar los riesgos del proyecto. Al adoptar estas herramientas, las organizaciones pueden agilizar los flujos de trabajo de los proyectos, mejorar la comunicación y garantizar el éxito de la correcta finalización de los proyectos. El perfeccionamiento continuo de las herramientas de gestión de proyectos demuestran el compromiso permanente de poder cumplir con las necesidades cambiantes de los equipos de proyecto y las organizaciones en un panorama empresarial cada vez más complejo y dinámico.

1.2. Contexto

La gestión de numerosos proyectos complejos requieren una gestión eficaz de las tareas. Al principio, la Policía Nacional utilizaba sistemas dispares, principalmente Microsoft Excel para el seguimiento de tareas y Microsoft Access para administrar su base de datos. Sin embargo, la naturaleza fragmentada de estas herramientas dificultaba la gestión eficaz de los proyectos, lo que provocaba tediosos procesos en la comunicación y en la elaboración de informes. Para hacer frente a estos retos, la Policía Nacional buscó una solución integral que pudiera unificar sus necesidades de gestión de tareas y bases de datos. Redmine surgió como la herramienta de gestión de proyectos ideal, ya que ofrecía una plataforma centralizada que integraba a la perfección las funciones de seguimiento de tareas y gestión de bases de datos. Con la adopción de Redmine, la Policía Nacional pretendía consolidar sus procesos de gestión de proyectos, agilizar la comunicación y la colaboración, garantizar una gestión precisa de los datos y generar informes detallados para tomar decisiones con conocimiento de causa.

El presente Trabajo de Fin de Grado pretende satisfacer la necesidad de la Policía Nacional de gestionar todos aquellos proyectos propuestos por el Programa Horizonte Europa, sucesor de Horizonte 2020. El trabajo ha sido desarrollado en conjunto con el departamento de innovación y desarrollo de la Policía nacional.

1.3. Plan de Trabajo

Se ha decidido llevar a cabo el trabajo las siguientes fases:

1. **Investigación**: La fase de Investigación consistió en dar respuesta a porqué es necesario organizarse con metodologías de trabajo a la hora de desarrollar un proyecto software. Asimismo, se escoge una metodología de trabajo en concreto. Los resultados de esta fase sentaron las bases para las dos fases posteriores, al proporcionar una guía para el proceso de Desarrollo y de Experimentación.

2. **Desarrollo**: La fase de desarrollo fue dividida en dos, la primera consistió en llevar a cabo una investigación y análisis exhaustivos para conocer en profundidad los objetivos del proyecto, los requisitos y las necesidades específicas de la Policía Nacional a la hora de gestionar proyectos en el marco del Programa Horizonte Europa. Esta fase implicó una revisión en profundidad de la metodología de trabajo utlizada actualmente por la Policía Nacional, además de llevar a cabo entrevistas para obtener la información requerida para el cumplimiento de los requisitos propuestos. Asimismo, se realizó una investigación de las herramientas gestoras existentes, comparándolas entre sí y seleccionando la que mejor se adecuara a las necesidades planteadas por los requisitos especificados en la anterior parte.

   La segunda parte se centra en la creación e implantación de la solución de gestión de proyectos para la Policía Nacional. Basándose en las conclusiones de la fase de Investigación, esta fase implicó la personalización de la herramienta gestora de proyectos seleccionada y la integración de las funcionalidades necesarias para cumplir los requisitos específicos del Programa Horizonte Europa. La fase de Desarrollo incluyó tareas como el diseño de la base de datos, el desarrollo de la interfaz de usuario, la implementación del flujo de trabajo y las pruebas del sistema para garantizar la funcionalidad, usabilidad y fiabilidad de la solución.

3. **Realización de pruebas**: La fase de pruebas tuvo como objetivo validar la eficacia y eficiencia de la solución gestora de proyectos desarrollada. Durante esta fase, la solución se desplegó en un entorno controlado dentro de la plataforma *fly.io*, y se simularon escenarios y proyectos reales del Programa Horizonte Europa. El objetivo fue evaluar el rendimiento de la solución, su capacidad para gestionar tareas y datos relacionados con los proyectos, y su compatibilidad con los flujos de trabajo y procesos existentes en la Policía Nacional. Los resultados de esta fase proporcionaron información valiosa para perfeccionar y optimizar la solución antes de su implementación final.

El Desarrollo y la Realización de pruebas se repitieron a lo largo del trabajo dada la naturaleza de la metodología de trabajo escogida en la Investigación.

A continuación se muestra un diagrama de Gantt mostrando las actividades desarrolladas durante el trabajo.

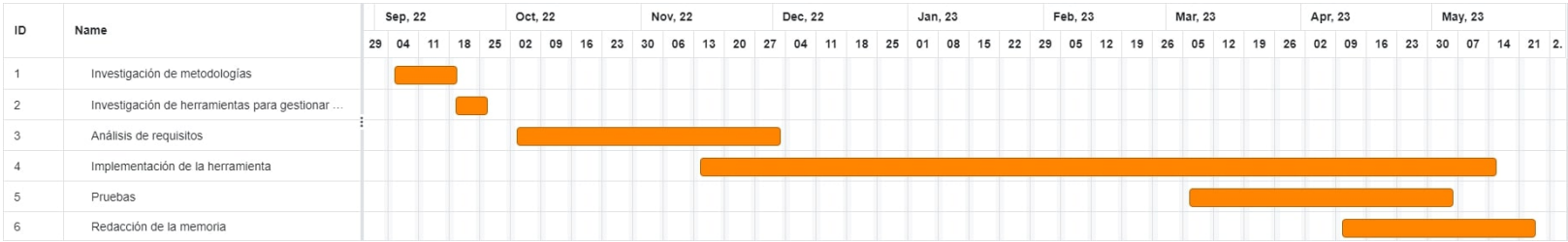

Figura 1.1: Diagrama de Gantt

1.4. Estructura del Trabajo

El trabajo se divide en 7 capítulos, siguiendo la siguiente estructura propuesta:

El Capítulo 2 presenta el marco teórico del trabajo, en el que se investiga qué son las metodologías de trabajo y porqué se deben emplear. Asimismo, se realiza una investigación de las metodologías de trabajo más utilizadas actualmente.

El Capítulo 3 presenta el Estado del Arte de los gestores de proyectos que existen actualmente, se listan los gestores de proyectos más utilizados y se hace una descripción de las características de cada uno. Finalmente se añade una tabla comparativa para juntar la información de cada gestor, permitiendo entender porqué se elige *Redmine* como gestor final.

En el Capítulo 4, primero se lleva a cabo una investigación para obtener los requisitos específicos que debe cumplir el gestor de proyectos pedido por la Policía Nacional. Se hace una propuesta formal de requisitos. Asimismo, se obtiene el flujo de trabajo que deberían seguir las propuestas recibidas. Después se desglosa la implementación de las aplicaciones basadas en *Ruby on Rails*. Seguida de la explicación de la implementación de *Redmine* y sus posibles extensiones, junto con la implementación del flujo de trabajo descrito. Por último, se describe qué proceso se ha llevado a cabo para asegurar la efectividad de la implementación realizada.

El Capítulo 5 se exponen las conclusiones principales de este estudio, así como las áreas de investigación futuras que se plantean.

Por último, los capítulos 6 y 7 son las traducciones al inglés de la Introducción y de las Conclusiones.

Capítulo 2

Marco teórico

2.1. ¿Qué es una metodología?

La metodología hace referencia al conjunto de ideas, enfoques, principios y prácticas que se utilizan para llevar a cabo un proceso de trabajo o investigación de manera sistemática y organizada. Proporciona una guía estructurada para poder llevar a cabo un objetivo específico o resolver un problema de manera eficiente y efectiva [4].

En el contexto de gestionar proyectos, una metodología de trabajo es un marco de referencia que establece los pasos y las actividades a seguir para completar un proyecto con éxito. Define las reglas, los procesos y las herramientas necesarias para planificar, ejecutar y controlar todas las etapas que se realizan en un proyecto, desde las meras ideas hasta la implementación final.

Una metodología de trabajo efectiva proporciona una estructura clara para el equipo de proyecto y permite una gestión eficiente de los recursos, el tiempo y los riesgos involucrados. También establece roles y responsabilidades, define los entregables y los hitos importantes, y promueve la comunicación y el desarrollo de actividades de manera colaborativa entre los integrantes del equipo.

Cada metodología de trabajo puede tener características únicas y adaptarse a diferentes tipos de proyectos o entornos organizativos. Algunas metodologías, siguen un enfoque secuencial y rígido, mientras que otras, se centran en la adaptabilidad, la iteración y la respuesta rápida a los cambios.

La elección de la metodología adecuada depende de factores como el tamaño del proyecto, la naturaleza del trabajo, los requisitos del cliente, el nivel de incertidumbre y la cultura organizativa. Las metodologías también pueden combinarse o adaptarse según las necesidades específicas del proyecto.

2.2. Importancia de organizarse con metodologías

En el entorno empresarial actual, gestionar proyectos se ha convertido en una disciplina fundamental para lograr el éxito en la ejecución de iniciativas y alcanzar los objetivos establecidos. La gestión efectiva de proyectos requiere una estructura y un enfoque organizado, y es ahí donde entran en juego las metodologías. Estas metodologías proporcionan un marco de trabajo sistemático que guía a los equipos en cada etapa del

proyecto, desde la planificación hasta la finalización [5].

Asimismo, las metodologías pensadas para gestionar proyectos proporcionan un conjunto de pautas y procesos estandarizados que permiten a los equipos tener una estructura clara y definida para seguir durante todo el proyecto. Estas metodologías establecen pasos y fases específicas, desde la definición de requisitos hasta la entrega final, lo que facilita la planificación, la asignación de recursos y la ejecución de tareas. Al tener un marco de trabajo establecido, los equipos pueden evitar la confusión, la duplicación de esfuerzos y los retrasos, lo que conduce a una mayor eficiencia y productividad.

Además, las metodologías de gestión de proyectos permiten una mejor gestión del tiempo y los recursos disponibles. Establecen plazos, hitos y entregables, lo que ayuda a los equipos a priorizar tareas, asignar recursos de manera adecuada y realizar un seguimiento del progreso del proyecto. Al contar con un enfoque estructurado, los equipos pueden identificar posibles desviaciones o retrasos y tomar medidas correctivas de manera oportuna. Esto se traduce en una mayor capacidad para cumplir con los plazos establecidos y optimizar el uso de los recursos disponibles, lo que a su vez mejora la eficiencia y reduce los costos asociados al proyecto.

Las metodologías de gestión de proyectos fomentan una comunicación clara y efectiva entre los miembros del equipo y las partes interesadas. Estas metodologías establecen canales de comunicación formales, reuniones regulares y herramientas de seguimiento, lo que facilita la colaboración, el intercambio de información y la toma de decisiones. Al tener roles y responsabilidades bien definidos, los equipos pueden trabajar de manera más cohesionada y eficiente, minimizando la posibilidad de malentendidos o conflictos. Además, las metodologías también fomentan la participación activa de las partes interesadas y les brindan la oportunidad de dar retroalimentación y contribuir al proceso de toma de decisiones.

Por otro lado, las metodologías de gestión de proyectos incluyen en su enfoque la identificación y el manejo de los riesgos. Estas metodologías promueven la realización de análisis de riesgos, la identificación de posibles problemas y la implementación de planes de mitigación adecuados. Al gestionar de manera proactiva los riesgos, los equipos pueden anticiparse a posibles obstáculos, tomar medidas preventivas y reducir la probabilidad de que los problemas afecten negativamente el proyecto. Esto permite un mayor control sobre los factores que podrían afectar el éxito del proyecto y aumenta la capacidad de respuesta frente a desafíos inesperados.

Por último, las metodologías de gestión de proyectos promueven la mejora continua y el aprendizaje a través de la retroalimentación y la evaluación de cada etapa del proyecto. Estas metodologías fomentan la revisión y la reflexión constante sobre las lecciones aprendidas, los éxitos y fracasos, y las áreas de mejora. Al aplicar una metodología, los equipos pueden identificar qué prácticas funcionaron bien y cuáles pueden mejorarse en futuros proyectos. Esto lleva a una evolución constante de las habilidades y capacidades del equipo de proyecto, lo que contribuye a un mayor nivel de eficiencia y efectividad en la gestión de proyectos a largo plazo.

En definitiva, gestionar proyectos con metodologías es fundamental para alcanzar el éxito en la ejecución de iniciativas empresariales. Estas metodologías proporcionan estructura, estandarización, gestión eficiente de tiempo y recursos, comunicación efectiva, control de riesgos y promueven la mejora continua [6]. Al adoptar y aplicar adecuadamente una metodología de gestión de proyectos, los equipos pueden aumentar su productividad, minimizar los riesgos y maximizar los resultados finales del proyecto. En un entorno

empresarial cada vez más competitivo y en constante cambio, la gestión de proyectos con metodologías se convierte en un elemento diferenciador para las organizaciones que buscan lograr sus objetivos de manera eficaz y eficiente.

2.3. Metodologías existentes

Explicado los conceptos de metodología y por qué es necesario organizarse así, se presentan a continuación las metodologías más usadas en la industria.

2.3.1. Metodologías Clásicas

2.3.1.1. Cascada

La metodología Cascada, también conocida como modelo *Waterfall*, es un enfoque clásico y secuencial para el desarrollo de software. Fue propuesto por Winston W. Royce en 1970 y se ha utilizado ampliamente en la industria durante décadas. Esta metodología sigue una estructura lineal y rígida, donde cada etapa se lleva a cabo de forma secuencial y se considera una fase distinta del proceso de desarrollo [7].

El enfoque de Cascada se compone de las siguientes etapas principales [8]:

- **Requisitos**: En esta etapa inicial, se presentan e identifican los requisitos del software. Esto implica tener una clara idea de las necesidades del cliente y definir claramente qué funcionalidades y características debe tener el producto final.
- **Análisis**: Una vez que los requisitos se han establecido, se realiza un análisis detallado para descomponer el sistema en componentes más pequeños y comprensibles. Esto implica definir la estructura general del software y las relaciones entre los diferentes módulos o componentes.
- **Diseño**: En esta fase, se desarrolla el diseño detallado del sistema, definiendo la arquitectura, las interfaces, los algoritmos y las estructuras de datos necesarias. El objetivo es establecer una guía clara para la implementación del software.
- **Implementación**: En esta etapa, se realiza la implementación del código siguiendo el diseño definido. Los programadores escriben el código en el lenguaje de programación adecuado, utilizando las mejores prácticas y estándares establecidos.
- **Verficación**: Una vez completada la implementación, se realizan pruebas exhaustivas para detectar errores y garantizar que el software funcione según lo esperado. Se prueban tanto los componentes individuales como el sistema en su conjunto para validar su correcto funcionamiento.
- **Despliegue**: Después de que el software ha pasado las pruebas y se considera estable, se realiza la fase de despliegue. El producto final se entrega al cliente o se implementa en el entorno de producción, listo para su uso.
- **Mantenimiento**: Esta etapa implica el seguimiento y la corrección de errores o problemas que puedan surgir después del despliegue. También puede incluir la implementación de mejoras o actualizaciones en respuesta a las necesidades del usuario.

Una de las principales características del método en Cascada es su enfoque secuencial y lineal, lo que significa que cada etapa debe completarse antes de pasar a la siguiente. Esto puede ser beneficioso para proyectos con requisitos estables y bien definidos, pero también puede ser una limitación si surgen cambios o problemas durante el proceso de desarrollo. La Figura 2.1 muestra el modelo explicado anteriormente.

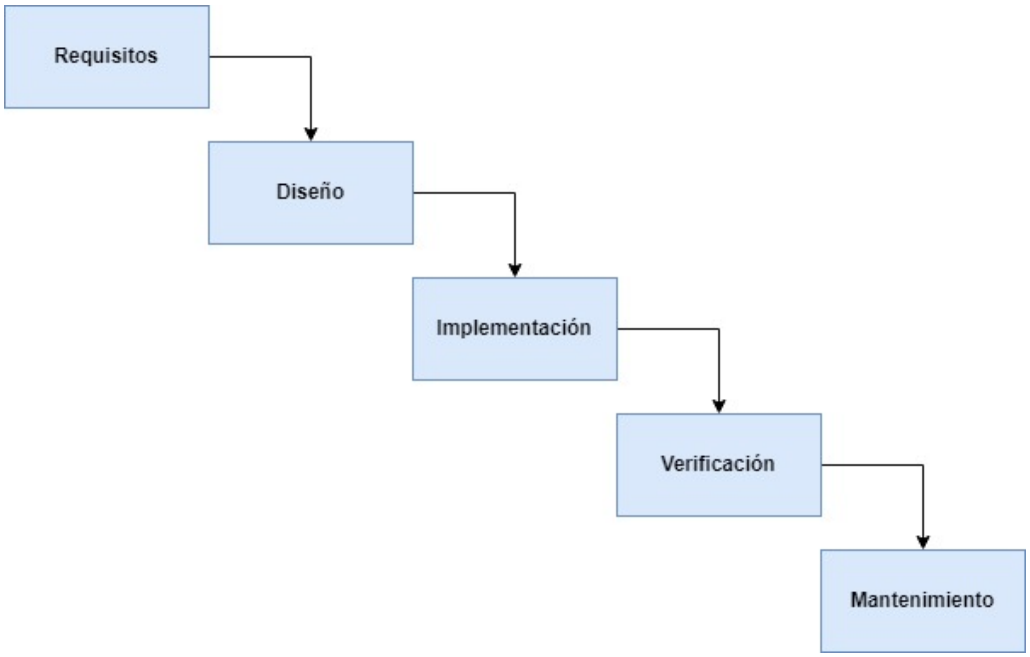

Figura 2.1: Modelo Cascada

2.3.1.2. Modelo V

El Modelo V es un modelo de desarrollo de software que sigue una estructura en forma de "V". Esta metodología está diseñada para poder garantizar el nivel de calidad del software a través de la validación y verificación exhaustiva en cada fase. A medida que se desciende por la "V", las fases de diseño y especificación se van refinando y se llevan a cabo pruebas correspondientes para validar cada una de ellas.

A continuación, se describen las fases clave del Modelo V y cómo se relacionan con las pruebas correspondientes [9]:

- **Definición de requerimientos**: En esta etapa inicial, se proponen y documentan los requisitos del modelo a desarrollar, estableciendo claramente las funcionalidades y características que debe tener el sistema. Los requerimientos pueden incluir aspectos funcionales, no funcionales y restricciones del sistema.
- **Diseño Funcional del Sistema**: En esta fase, se traducen los requerimientos definidos en la etapa anterior en un diseño funcional de alto nivel. Se establecen las principales funcionalidades, la estructura general del sistema y las interfaces con otros componentes o sistemas externos.
- **Diseño Técnico del Sistema**: A partir del diseño funcional, se realiza un desglose detallado del sistema en componentes más pequeños y se definen las especificaciones

técnicas. En esta fase, se establecen las interfaces entre los diferentes componentes y se planifica la estructura interna del sistema.

- **Especificación de Componentes**: En esta etapa, se detalla aún más el diseño técnico del sistema, definiendo las especificaciones de cada componente individual. Se establecen las interfaces, los algoritmos que se ejecutarán cada componente y las estructuras de datos que usarán dichos algoritmos.

- **Código**: Una vez que las especificaciones de los componentes están claras, se procede a la implementación del software. Los programadores escriben el código en el lenguaje de programación apropiado, siguiendo las especificaciones y estándares definidos en las etapas anteriores.

- **Pruebas Unitarias**: Después de completar la implementación del código, se llevan a cabo pruebas unitarias para verificar que cada componente realiza su función de manera esperada. Se prueban casos de prueba específicos para validar el comportamiento de cada unidad de código.

- **Pruebas de Componentes**: Durante esta etapa, se realiza la integración de los componentes individuales para formar el sistema completo. Se llevan a cabo pruebas de componentes para asegurar que los diferentes módulos se comuniquen correctamente y funcionen en conjunto como se espera.

- **Pruebas de Sistema**: Una vez que los componentes están integrados, se realizan pruebas de sistema para verificar que el sistema completo funcione de acuerdo con las especificaciones y los requerimientos establecidos. Se prueban los escenarios de uso, las funcionalidades y se identifican posibles problemas de interoperabilidad.

- **Pruebas de Aceptación**: En esta última fase, se llevan a cabo pruebas de aceptación en las cuales el sistema es evaluado por el cliente o los usuarios finales para determinar si cumple con los requisitos y expectativas establecidos. Estas pruebas validan el sistema completo y su capacidad de satisfacer las necesidades del modelo conceptualizado y los criterios de aceptación establecidos previamente.

En el Modelo V, cada fase de pruebas se alinea con una fase de diseño o especificación correspondiente. Esto significa que las pruebas realizadas en cada etapa tienen como objetivo validar y verificar el trabajo realizado en la fase anterior, asegurando así la calidad y la conformidad del software en cada paso del proceso.

Es importante destacar que el Modelo V resalta la importancia de las pruebas tempranas y frecuentes en el proceso de desarrollo de software. Esto ayuda a identificar problemas y corregirlos en etapas tempranas, lo que reduce el riesgo de fallos costosos y retrasos en etapas posteriores.

Aunque el Modelo V es una metodología clásica y secuencial, se puede adaptar y combinar con enfoques ágiles para permitir una mayor flexibilidad y colaboración durante el desarrollo. Esta combinación puede ayudar a abordar los cambios y los requisitos emergentes, manteniendo al mismo tiempo la estructura de validación y verificación rigurosa que ofrece el Modelo V. Este modelo se presenta en la Figura 2.2.

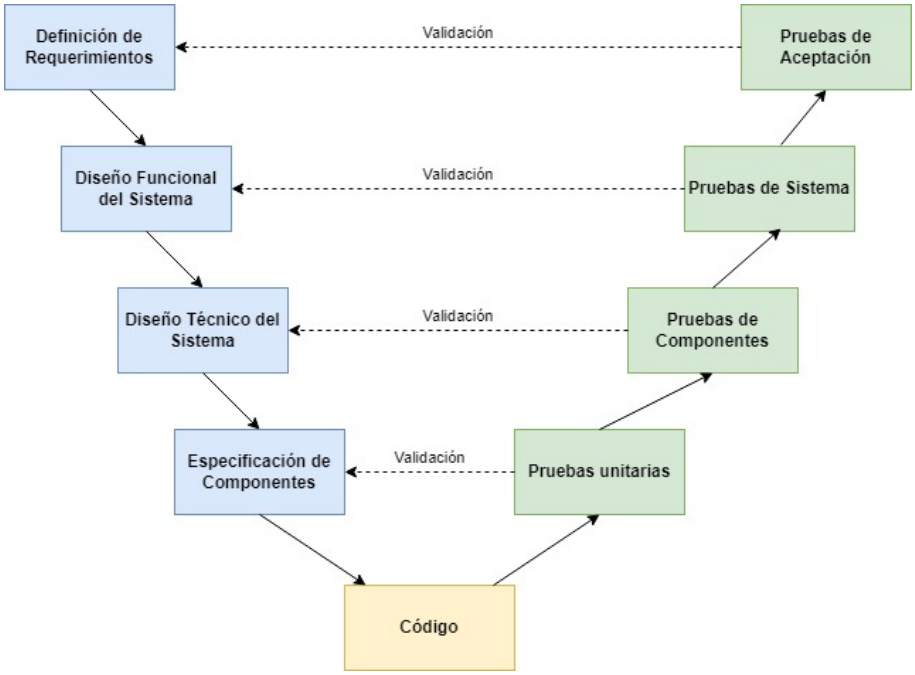

Figura 2.2: Modelo V

2.3.1.3. Modelo en Espiral

El Modelo en Espiral es una metodología clásica de desarrollo de software que combina elementos de desarrollo secuencial y en cascada con la gestión de riesgos y la retroalimentación del cliente. Fue propuesto por Barry Boehm en 1986 como una forma de abordar proyectos complejos y de alto riesgo, donde los requisitos pueden ser ambiguos o propensos a cambios.

El Modelo en Espiral sigue un enfoque iterativo en forma de espiral, donde cada iteración consta de cuatro fases principales [10]:

- **Planificación**: En esta fase inicial, se realiza la recolección de requisitos iniciales, donde se identifican y documentan los requisitos del software. Estos requisitos se utilizan para establecer una base inicial para el proyecto. En las iteraciones posteriores, la planificación se basa en los comentarios y la retroalimentación del cliente obtenidos en las iteraciones anteriores. Se definen los objetivos, los recursos y el cronograma del proyecto para la siguiente iteración.

- **Análisis de Riesgo**: En esta etapa, se realiza un análisis de riesgos basado en los requisitos recolectados en la primera iteración y en las iteraciones posteriores se actualiza el análisis basado en la retroalimentación del cliente. Se identifican los posibles riesgos y se evalúa su impacto en el proyecto. Esto permite tomar decisiones informadas sobre cómo mitigar o gestionar los riesgos identificados.

- **Ingeniería**: En esta fase, se lleva a cabo la ingeniería del software. En la primera iteración, se desarrolla un prototipo inicial del software basado en los requisitos recopilados. Este prototipo se utiliza para evaluar su viabilidad y obtener comentarios del cliente. En las iteraciones posteriores, se desarrollan prototipos iterativos basados en los comentarios del cliente para refinar y mejorar el software.

- **Evaluación del cliente**: En cada "vuelta" de la espiral, el cliente evalúa el software desarrollado hasta el momento. Se recopilan comentarios y retroalimentación del cliente para comprender sus necesidades y expectativas, y se utilizan para guiar las próximas iteraciones del proyecto. Esta evaluación del cliente es esencial para garantizar que el software cumpla con los requisitos y expectativas del cliente.

La característica clave del Modelo en Espiral es su enfoque en la gestión de riesgos. El análisis de riesgos se lleva a cabo de manera continua a lo largo del proyecto, y las decisiones tomadas en función de este análisis influyen en las etapas siguientes. Esto permite abordar y mitigar los riesgos de manera pro-activa, lo que aumenta las posibilidades de éxito del proyecto.

A medida que se avanza en las iteraciones de la espiral, el software se va refinando y mejorando en función de la retroalimentación del cliente. Esto asegura que el producto final sea una solución que se ajuste a las necesidades del cliente y que cumpla con los requisitos establecidos.

El Modelo en Espiral no es lineal ni rígido, ya que se trata de un enfoque iterativo y adaptativo. Esto permite que el desarrollo se ajuste a medida que se obtiene más información y se refina el producto final. Este modelo se presenta en la Figura 2.3.

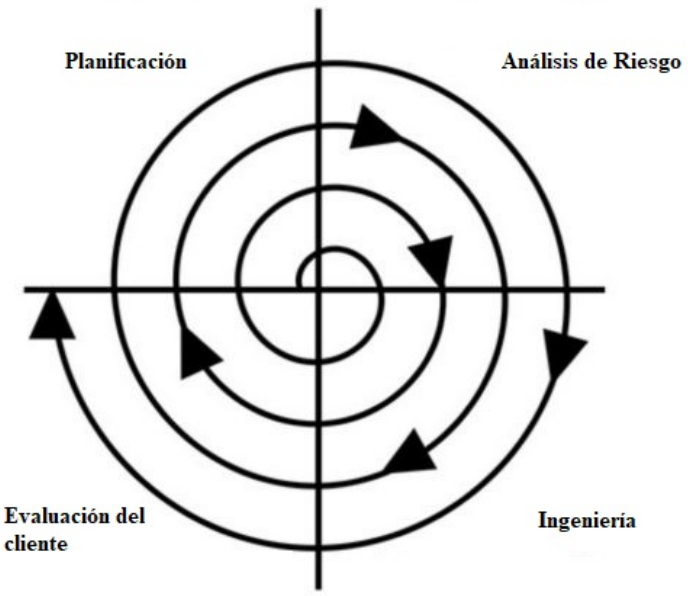

Figura 2.3: Modelo Espiral

2.3.1.4. Método de la Ruta Crítica

El Método de la Ruta Crítica (*Critical Path Method* (CPM)) es una metodología clásica de gestión de proyectos que se utiliza para planificar, programar y controlar las actividades involucradas en un proyecto. Se basa en una representación gráfica de las actividades, sus dependencias y los tiempos estimados de duración, lo que permite identificar la ruta crítica del proyecto, es decir, la secuencia de actividades que determina la duración mínima del proyecto [11]. Los pasos principales en la aplicación de este método son:

- **Identificación de actividades**: En esta etapa, se identifican todas las actividades necesarias para completar el proyecto. Cada actividad debe ser claramente definida y representar una tarea o un conjunto de tareas que requieren tiempo y recursos para su ejecución.
- **Secuenciación de actividades**: Una vez identificadas las actividades, se establecen las dependencias entre ellas. Esto implica determinar qué actividades deben completarse antes de que otras puedan comenzar. Se crean relaciones de precedencia que indican la secuencia lógica en la que deben llevarse a cabo las actividades.
- **Estimación de tiempos**: En esta fase, se estima la duración de cada actividad. Se pueden utilizar diferentes técnicas, como la experiencia previa, la consulta a expertos o el análisis histórico, para determinar el tiempo necesario para completar cada actividad.
- **Creación del diagrama de red**: Utilizando la información recopilada en los pasos anteriores, se crea un diagrama de red que representa las actividades, sus dependencias y los tiempos estimados de duración. El diagrama de red puede ser representado mediante la técnica de flechas (Diagrama de Flechas) o mediante el método de los nodos (Diagrama de Precedencia).
- **Determinación de la ruta crítica**: Una vez que se ha construido el diagrama de red, se determina la ruta crítica del proyecto. La ruta crítica es la secuencia de actividades que tiene el tiempo total más largo y, por lo tanto, determina la duración mínima del proyecto. Las actividades en la ruta crítica no tienen holgura, lo que significa que cualquier retraso en estas actividades retrasará todo el proyecto.
- **Programación y control del proyecto**: Con la ruta crítica identificada, se establece un cronograma de ejecución del proyecto. Se asignan los recursos necesarios para cada actividad y se establecen las fechas de inicio y finalización esperadas. Durante la ejecución del proyecto, se realiza un seguimiento del avance de las actividades y se comparan los resultados reales con el cronograma planificado para controlar el progreso y tomar acciones correctivas si es necesario.

Un ejemplo de ruta diagrama de red podría ser el que se presenta en la Figura 2.4.

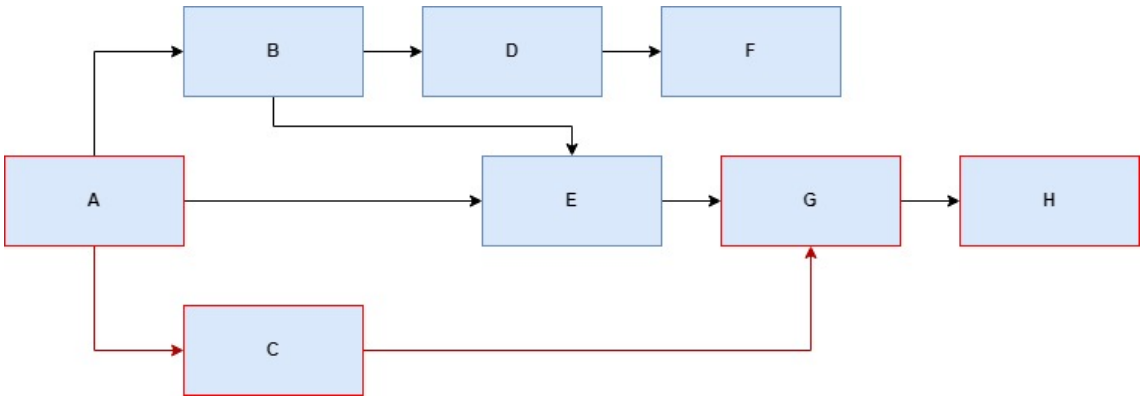

Figura 2.4: Ejemplo de ruta crítica (A-C-G-H). Se han omitido los tiempo de cada tarea.

El Método de la Ruta Crítica es especialmente útil para proyectos que tienen restricciones de tiempo y donde se busca optimizar los recursos disponibles. Al identificar

la ruta crítica y las actividades más críticas, se pueden asignar los recursos de manera más eficiente y anticiparse a posibles retrasos o problemas.

El CPM ofrece varias ventajas significativas. En primer lugar, proporciona una visión clara y visual del proyecto, lo que facilita la comprensión de las relaciones entre las actividades y la determinación de la ruta crítica. Esto permite a los equipos de proyecto identificar las actividades más críticas y priorizar sus esfuerzos y recursos en función de ellas.

Además, el CPM permite una programación precisa y realista del proyecto. Al estimar los tiempos de duración de las actividades y determinar la ruta crítica, se obtiene una estimación de la duración total del proyecto. Esto es especialmente valioso cuando se trata de proyectos con plazos ajustados o proyectos en los que se requiere una entrega oportuna.

El seguimiento y control del proyecto también se simplifican con el CPM. Al comparar el avance real con el cronograma planificado, se pueden identificar desviaciones y tomar acciones correctivas de manera oportuna. Esto ayuda a mantener el proyecto en el camino correcto y evitar retrasos costosos.

Sin embargo, es importante tener en cuenta que el CPM tiene algunas limitaciones. En primer lugar, se basa en estimaciones de tiempo y depende de la precisión de estas estimaciones. Si las estimaciones no son precisas, esto puede afectar la planificación y el control del proyecto. Además, el CPM no tiene en cuenta las limitaciones de recursos, lo que significa que es posible que se requiera una optimización adicional para asignar eficientemente los recursos disponibles.

En resumen, el Método de la Ruta Crítica es una metodología clásica de gestión de proyectos que se basa en la representación gráfica de actividades, dependencias y tiempos estimados. A través de la identificación de actividades, la secuenciación, la estimación de tiempos, la creación del diagrama de red, la determinación de la ruta crítica y la programación y control del proyecto, el CPM permite planificar y controlar de manera efectiva la ejecución de un proyecto.

2.3.2. Metodologías Ágiles

2.3.2.1. Kanban

La metodología ágil Kanban es un enfoque visual basado en el flujo de trabajo para la gestión de proyectos y tareas. Se originó en el ámbito de la fabricación y se ha adaptado exitosamente al desarrollo de software y a otros campos. Kanban se centra en la mejora continua, la eficiencia y la entrega de valor de manera constante.

A continuación, se detallan los elementos clave y los principios de la metodología ágil Kanban [12]:

- **Tablero Kanban**: El tablero Kanban es la representación visual del flujo de trabajo del proyecto o del equipo. Se divide en columnas que representan las etapas del proceso, como “Por hacer”, “En progreso” y “Finalizado”. Cada tarea o elemento de trabajo se representa mediante tarjetas, y se mueven de una columna a otra a medida que avanzan en el flujo de trabajo.
- **Límites de trabajo en proceso (*Work in progress* (WIP))**: Kanban establece límites claros para la cantidad máxima de elementos de trabajo que se pueden

tener en cada columna del tablero Kanban. Esto evita la sobrecarga de trabajo y ayuda a mantener un flujo constante y equilibrado. Los límites de WIP fomentan la finalización de las tareas existentes antes de comenzar nuevas, lo que ayuda a evitar la acumulación de trabajo en progreso.

- **Flujo continuo**: Kanban promueve un flujo de trabajo continuo y equilibrado. Las tareas se mueven de una columna a otra de acuerdo con el progreso real, sin saltos ni interrupciones innecesarias. El objetivo es identificar y eliminar los cuellos de botella y los obstáculos para mantener un flujo constante y entregar valor de manera más rápida y eficiente.

- **Mejora continua**: Kanban se basa en el principio de mejora continua. Se alienta a los equipos a realizar mejoras incrementales y evolutivas en su flujo de trabajo y procesos. A través de la retroalimentación constante y el análisis de métricas, se identifican oportunidades de mejora y se implementan cambios para optimizar el rendimiento y la eficiencia.

- **Transparencia y colaboración**: Kanban promueve la transparencia y la colaboración en todo el equipo. Al utilizar un tablero Kanban visible para todos los miembros del equipo, se fomenta la comunicación y la visibilidad de las tareas en curso. Esto facilita la colaboración, la asignación de recursos y la resolución conjunta de problemas.

- **Enfoque en el valor**: Kanban se centra en la entrega de valor de manera constante. A través de la priorización adecuada de las tareas y la entrega continua de elementos terminados, se busca maximizar el valor entregado al cliente o usuario final. Esto permite obtener retroalimentación temprana y ajustar las prioridades y el alcance del proyecto según sea necesario.

Existen numerosas herramientas web que proporcionan una implementación gratuita del tablero Kanban, por ejemplo, *Canva*. En la Figura 2.5.

Kanban es especialmente adecuado para proyectos y equipos que requieren una mayor flexibilidad y adaptabilidad. Su enfoque visual, basado en el flujo de trabajo y en la entrega continua de valor, permite una mayor visibilidad, control y capacidad de respuesta a medida que evolucionn los requisitos y las necesidades cambiantes del proyecto.

Algunos beneficios de utilizar Kanban en la gestión de proyectos son:

1. **Mayor visibilidad y transparencia**: El tablero Kanban hace posible tener una visión del estado de las tareas y el flujo de trabajo. Todos los integrantes del equipo tienen acceso a esta información en tiempo real, lo que facilita la coordinación, la toma de decisiones y la identificación de posibles obstáculos.

2. **Flexibilidad y adaptabilidad**: Kanban se adapta bien a entornos donde los requisitos y las prioridades pueden cambiar con frecuencia. Al no establecer plazos rígidos y permitir que el equipo se enfoque en un número manejable de tareas, se puede responder rápidamente a los cambios y ajustar el flujo de trabajo según sea necesario.

3. **Optimización del flujo de trabajo**: Al limitar la cantidad de trabajo en proceso en cada etapa del flujo, Kanban evita la sobrecarga y los cuellos de botella. Esto

conduce a un flujo de trabajo más equilibrado y eficiente, lo que mejora los tiempos de entrega y reduce el tiempo de respuesta.

4. **Enfoque en la calidad**: Kanban promueve la entrega de tareas completas y de alta calidad en lugar de simplemente avanzar en el proceso. Al establecer límites de trabajo en proceso y enfocarse en la finalización antes de comenzar nuevas tareas, se fomenta la calidad del trabajo realizado.

5. **Mejora continua y aprendizaje**: La metodología Kanban fomenta la mejora continua a través del análisis de métricas y la retroalimentación del equipo y los stakeholders. Al medir el rendimiento, identificar áreas de mejora y realizar ajustes graduales, el equipo puede aprender de sus experiencias y optimizar su flujo de trabajo con el tiempo.

La implementación exitosa de Kanban requiere un compromiso y una colaboración activa de todo el equipo. La comunicación abierta, el seguimiento constante del flujo de trabajo y la disposición para adaptarse a medida que surjan nuevos desafíos son fundamentales para aprovechar al máximo los beneficios de esta metodología ágil.

En resumen, Kanban es una metodología ágil que se basa en la visualización del flujo de trabajo y la entrega continua de valor. Al promover la transparencia, la adaptabilidad y la mejora continua, Kanban ayuda a los equipos a gestionar proyectos de manera eficiente, optimizando el flujo de trabajo y entregando valor de manera constante.

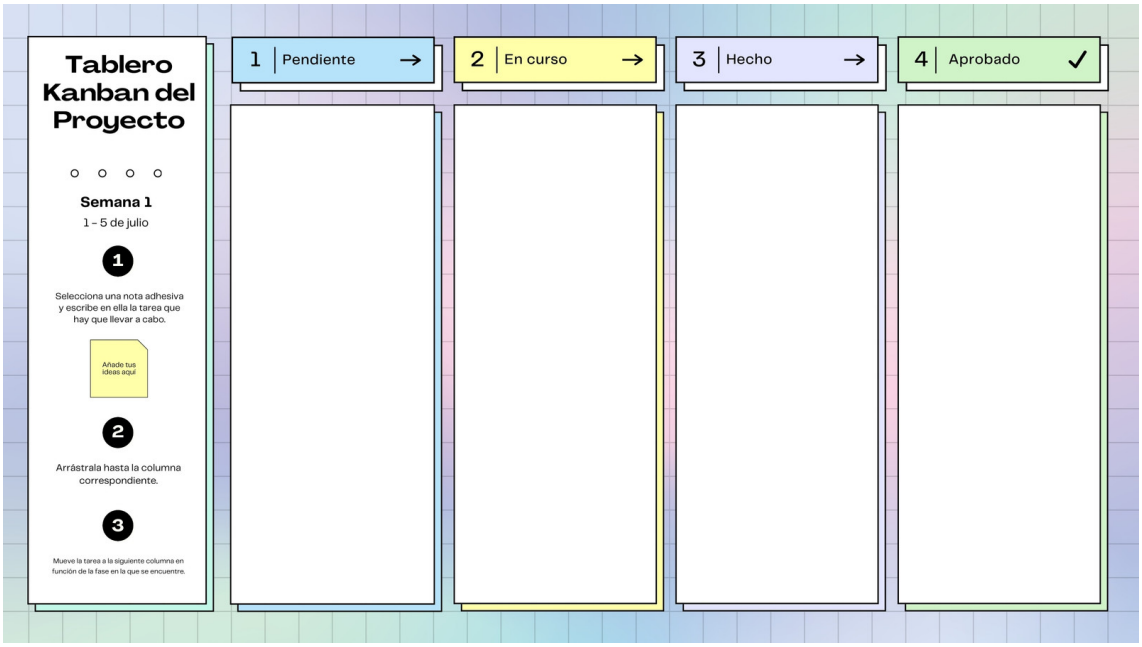

Figura 2.5: Ejemplo de tablero Kanban [1]

2.3.2.2. Programación extrema

Extreme Programming (XP) es una metodología ágil de desarrollo de software que se centra en la calidad del producto, la adaptabilidad a los cambios y la colaboración entre los miembros del equipo. XP se basa en una serie de prácticas y valores fundamentales que promueven un enfoque iterativo e incremental para la entrega de software [13].

A continuación, se detallan los elementos clave y los principios de la metodología ágil XP:

- Comunicación constante: se enfatiza la comunicación abierta y frecuente entre todos los miembros del equipo, incluyendo a los desarrolladores, los *stakeholders* y los clientes. La comunicación constante ayuda a comprender los requisitos y las expectativas, aclarar dudas y tomar decisiones rápidas y efectivas.
- Retroalimentación continua: XP se basa en la retroalimentación constante para mejorar el proceso de desarrollo. Los desarrolladores trabajan estrechamente con los clientes y los stakeholders para recibir comentarios sobre el producto en etapas tempranas y frecuentes. Esta retroalimentación se utiliza para ajustar y mejorar el software en cada iteración.
- Desarrollo incremental: se sigue un enfoque iterativo e incremental para el desarrollo de software. En lugar de esperar a tener todos los requisitos y funcionalidades definidos desde el inicio, se desarrolla y entrega software funcional en pequeñas iteraciones. Esto permite una entrega temprana de valor y la posibilidad de adaptarse a los cambios y requisitos emergentes.
- Pruebas unitarias: las pruebas unitarias son una parte integral de este método ágil. Los desarrolladores escriben pruebas automatizadas para cada pieza de código que producen. Estas pruebas ayudan a garantizar que el software funcione correctamente y que los cambios no introduzcan errores en el código existente. Además, las pruebas unitarias actúan como una documentación viva del sistema.
- Integración continua: lo que implica que los cambios realizados por los desarrolladores se integran y se prueban de manera frecuente. Esto permite detectar y corregir problemas de forma temprana, facilitando la entrega de software funcional y estable en todo momento.
- Diseño simple: se evita la sobreingeniería y se busca la solución más sencilla para cumplir con los requisitos. Esto facilita la comprensión y el mantenimiento del código, y permite adaptarse de manera ágil a los cambios en los requisitos.
- Juego de pruebas: El juego de pruebas, o "programación en parejas", es una práctica común en XP. Dos desarrolladores trabajan juntos en una misma estación de trabajo, compartiendo conocimientos y revisando el código del otro. Esta práctica fomenta la colaboración, el aprendizaje y la calidad del código.
- Propiedad compartida: En XP, se fomenta la propiedad compartida del código y la responsabilidad de todo el equipo. Todos los miembros tienen voz y voto en la toma de decisiones y se fomenta la colaboración y la contribución activa de cada persona.

En la figura 2.6 se representa el flujo de trabajo en XP.

XP se centra en la entrega temprana y frecuente de software funcional, la adaptabilidad a los cambios y la mejora continua. Al utilizar prácticas como la comunicación constante, las pruebas unitarias y la integración continua. Ayuda a los equipos a entregar productos de alta calidad de manera ágil y efectiva.

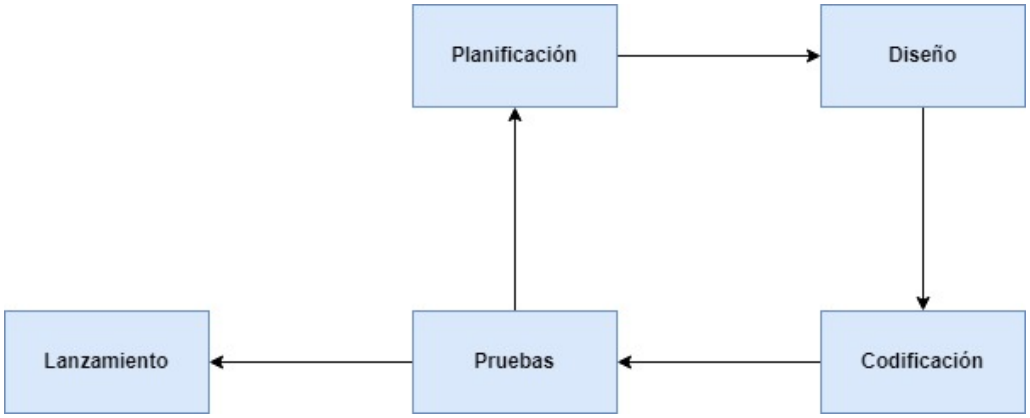

Figura 2.6: Flujo de trabajo en XP

2.3.2.3. Desarrollo basado en funcionalidades

Desarrollo basado en funcionalidades (*Feature Driven Development* (FDD)) es una metodología ágil de desarrollo de software que se centra en la entrega incremental y en la construcción de funcionalidades (*features*) clave en el desarrollo de un proyecto. FDD se basa en una serie de principios y prácticas orientadas a la colaboración efectiva, la comunicación clara y la entrega continua de software de calidad [14].

A diferencia de otras metodologías ágiles, se enfatiza la gestión por funcionalidades. Estas funcionalidades se dividen en características más pequeñas y manejables que se pueden desarrollar en un período de tiempo específico, generalmente en semanas. Cada funcionalidad se asigna a un equipo o a un desarrollador específico responsable de su implementación.

El proceso de desarrollo de FDD se divide en cinco etapas clave:

- Desarrollo del modelo global: En esta etapa, se identifican y definen las principales características del sistema. El equipo trabaja en conjunto para comprender los requisitos del sistema y crear un modelo global que describa la estructura y las interacciones de las funcionalidades.
- Lista de funcionalidades: En esta etapa, se crea una lista exhaustiva de todas las funcionalidades del sistema. Cada funcionalidad se describe en términos claros y concisos, y se prioriza según su importancia y valor para el cliente.
- Planificación por funcionalidades: En esta etapa, el equipo de desarrollo selecciona un conjunto de funcionalidades para ser implementadas en un período de tiempo determinado, generalmente en una iteración corta. Se asignan recursos y se define un plan detallado para el desarrollo de cada funcionalidad.
- Diseño por funcionalidades: En esta etapa, cada funcionalidad seleccionada se diseña de manera individual. Se establecen las interfaces, se definen los componentes necesarios y se realiza una estimación precisa del tiempo y los recursos requeridos para su implementación.
- Construcción por funcionalidades: En esta etapa, el equipo de desarrollo implementa cada funcionalidad de manera incremental. Se lleva a cabo la codificación, las pruebas

y la integración continua de las funcionalidades desarrolladas. Al finalizar cada iteración, se entrega una funcionalidad completa y probada.

FDD también hace hincapié en la colaboración activa y la comunicación efectiva dentro del equipo de desarrollo. Se fomenta la transparencia y se establecen mecanismos de seguimiento para garantizar el progreso adecuado del proyecto.

En resumen, FDD se enfoca en la entrega incremental de funcionalidades clave. Se basa en una serie de principios y prácticas que promueven la colaboración, la comunicación y la entrega continua de software de calidad.

2.3.2.4. Scrum

Scrum es una de las metodologías ágiles más reconocidas y ampliamente utilizadas, se destaca por su enfoque iterativo e incremental para la gestión de proyectos. En este apartado, exploraremos Scrum en detalle, analizando sus principios, roles, artefactos y eventos clave [15].

Scrum se basa en una serie de principios que orientan su enfoque ágil. Estos principios incluyen:

- Transparencia: Todos los aspectos del proyecto deben ser visibles y comprensibles para todos los involucrados.
- Inspección: Se deben realizar inspecciones periódicas de los artefactos y el progreso del proyecto para identificar posibles mejoras.
- Adaptación: Se fomenta la adaptación continua para abordar los cambios y desafíos que puedan surgir durante el proyecto.

Scrum define tres roles principales que desempeñan funciones clave en el proyecto:

- *Product Owner*: Es responsable de definir y priorizar los elementos del backlog del producto, representando los intereses del cliente o *stakeholders*.
- *Scrum Master*: Actúa como facilitador y defensor de Scrum, asegurando que se sigan las prácticas y ayudando al equipo a alcanzar su máximo potencial.
- Equipo de Desarrollo: Está compuesto por profesionales multifuncionales que trabajan juntos para desarrollar y entregar los incrementos del producto.

Por otro lado, Scrum utiliza tres artefactos principales para gestionar el flujo de trabajo y la entrega del producto:

- *Product Backlog*: Es una lista priorizada de todas las funcionalidades, requisitos y mejoras deseadas para el producto.
- *Sprint Backlog*: Es una lista de elementos seleccionados del *Product Backlog* para el *Sprint* actual, junto con un plan para su implementación.
- Incremento: Es el resultado tangible y potencialmente entregable de cada *Sprint*, que agrega valor al producto en desarrollo.

Por último, Scrum establece eventos específicos que ayudan a estructurar el flujo de trabajo y fomentar la colaboración:

- *Sprint*: Es un periodo de tiempo fijo y regular durante el cual se lleva a cabo el trabajo. Generalmente tiene una duración de dos a cuatro semanas.
- Reunión de Planificación del Sprint: En esta reunión, el equipo selecciona los elementos del *Product Backlog* para el próximo *Sprint* y crea el *Sprint Backlog*.
- Reunión Diaria (*Daily Scrum*): Es una breve reunión diaria en la que el equipo de desarrollo sincroniza su trabajo y actualiza el progreso.
- Revisión del *Sprint*: Al final de cada *Sprint*, el equipo muestra el incremento completado a los *stakeholders* y recibe su retroalimentación.
- Retrospectiva del *Sprint*: Es una reunión que se lleva a cabo al final de cada *Sprint*, donde el equipo reflexiona sobre el *Sprint* anterior y busca mejoras en su proceso de trabajo.

En definitiva, Scrum es una metodología ágil ampliamente adoptada que ofrece un enfoque iterativo e incremental para la gestión de proyectos. Sus principios, roles, artefactos y eventos clave trabajan en conjunto para fomentar la transparencia, inspección y adaptación continua. Al enfocarse en la entrega de incrementos de valor en intervalos regulares, Scrum permite poder dar respuesta de manera rápida y efectiva a cambios en los requisitos. A través de la colaboración estrecha y la retroalimentación constante, Scrum ayuda a los equipos a optimizar su rendimiento y a entregar productos de alta calidad. Al comprender y aplicar Scrum de manera efectiva, las organizaciones pueden mejorar su capacidad para gestionar proyectos de manera ágil y exitosa.

2.4. Metodología aplicada a este proyecto

La metodología de trabajo empleada en este proyecto fue Scrum, un marco ágil que se centra en la colaboración, la entrega de valor y la adaptabilidad a los cambios. Aunque Scrum generalmente se implementa con un equipo multidisciplinario, en este caso particular, asumí el desafío de llevar a cabo el proyecto de manera individual, siendo el único miembro del equipo de trabajo.

En este rol multifacético, asumí varias responsabilidades. Como *Product Owner*, fui responsable de definir los objetivos del proyecto y establecer las prioridades de las funcionalidades a desarrollar. Me aseguré de entender las necesidades y requerimientos de la Policía Nacional, para así poder representar sus intereses de la mejor manera posible.

Además, asumí el papel de Equipo de Desarrollo. Esto implicó diseñar, implementar y probar la aplicación por mi cuenta. Desde la concepción inicial hasta la fase de implementación final, trabajé en cada etapa del proceso, asegurándome de aplicar las mejores prácticas de desarrollo de software.

Aunque no hubo un *Scrum Master* específico en el proyecto, tomé la responsabilidad de guiar el proceso de acuerdo con los principios de Scrum. Me aseguré de seguir los rituales establecidos, como las reuniones de planificación, las revisiones de *Sprint* y las retrospectivas, adaptándolos a las necesidades del proyecto y manteniendo un enfoque ágil y flexible.

A pesar de que este proyecto se llevó a cabo de manera individual, reconocí la importancia de obtener feedback de los clientes. Por lo tanto, organicé reuniones periódicas con ellos para mostrarles el estado actual de la aplicación y recibir sus comentarios y sugerencias. Esta retroalimentación fue valiosa para ajustar las prioridades y realizar mejoras continuas en el producto.

Capítulo 3

Estado del Arte

3.1. Herramientas de trabajo utilizadas actualmente

En el actual método de trabajo que se quiere mencionar se destaca el uso extensivo de documentos Excel para gestionar una gran cantidad de información. Estos archivos son gestionados por cada investigador que participa en los proyectos, registrando las horas realizadas diariamente en cada proyecto en el que se trabaja. El gestor del archivo Excel se encarga de recopilar manualmente la información de todos estos archivos dispersos.

La gestión de estas hojas de cálculo en Excel tienen el propósito de satisfacer las labrores burocráticas presentadas por la necesidad de gestionar el presupuesto dado a cada proyecto. Sin embargo, este proceso de gestión de hojas de cálculo presenta diversas limitaciones y desafíos que dificultan la eficiencia y eficacia. El enfoque manual de recopilar los datos de cada hoja de cálculo es tedioso y consume un tiempo considerable, lo cual puede resultar en demoras, errores o pérdida de productividad.

Para superar estas limitaciones, se plantea realizar este este proceso mediante el uso de herramientas tecnológicas, en concreto, gestores de proyectos.

El proceso de trabajo de la Policía Nacional revela la necesidad de superar el enfoque manual y tedioso de gestión de documentos Excel. En consecuencia, se realiza una investigación sobre las herramientas que podrían solventar esta necesidad.

3.2. Gestores de proyectos existentes

En esta sección se llevó a cabo una búsqueda de diversos gestores de proyectos con el objetivo de evaluar cómo se alineaban con los requisitos identificados durante la investigación. Se exploraron diferentes alternativas disponibles en el mercado, analizando sus características, funcionalidades y capacidades para satisfacer las necesidades específicas del proyecto. El análisis comparativo de estas soluciones permitió obtener una visión del estado actual de las herramientas gestionadoras de proyectos y establecer una base sólida para la selección de la solución más adecuada. A continuación, se presentan los gestores estudiados:

3.2.1. Jira

Jira es un completo y versátil gestor de proyectos y seguimiento de problemas desarrollado por Atlassian. Es ampliamente utilizado en diversas industrias y organizaciones, desde pequeñas empresas hasta grandes corporaciones, así como en equipos de trabajo enfocados al desarrollo de código y departamentos de TI. Jira se destaca por su capacidad para facilitar la planificación, seguimiento y colaboración en proyectos complejos [16].

Una de las características principales de Jira es su sistema de seguimiento de problemas y gestión de tareas. Permite a los equipos crear, asignar y realizar un seguimiento de problemas, tareas y proyectos de manera eficiente. Los usuarios pueden crear tickets o problemas, agregar descripciones, establecer prioridades, asignar responsables y establecer fechas límite. Además, Jira proporciona herramientas para la gestión de flujos de trabajo personalizados, lo que permite adaptar el proceso de seguimiento de problemas a las necesidades específicas de cada equipo.

La capacidad de planificación ágil es otro aspecto destacado de Jira. Permite a los equipos implementar metodologías ágiles como *Scrum* o *Kanban*, lo que facilita la colaboración, el seguimiento del progreso y la gestión del backlog. Los tableros visuales de Jira proporcionan una visión clara del estado de las tareas, permitiendo a los miembros del equipo actualizar el estado de las tareas y realizar un seguimiento del progreso en tiempo real.

Jira también ofrece una amplia gama de funciones adicionales y complementos que permiten personalizar y ampliar su capacidad. Estos complementos brindan características adicionales, como paneles personalizables, informes horarios detallados, integración con otras herramientas y flujos de trabajo automatizados.

Además, Jira es altamente escalable y se integra fácilmente con otras herramientas de desarrollo y gestión de proyectos, lo que lo convierte en una opción viable para equipos que buscan una solución completa y flexible. Su interfaz intuitiva y su amplia documentación y comunidad de usuarios también hacen de Jira una herramienta accesible y fácil de adoptar.

Por último, Jira no es una plataforma de código abierto y su precio es comercial. Atlassian ofrece diferentes planes de precios y opciones de licencia para adaptarse a las necesidades y el tamaño de cada organización.

3.2.2. Asana

Asana es un destacado gestor de proyectos y herramienta de colaboración diseñada para impulsar la productividad y la eficiencia en los equipos. Con su enfoque en la organización y la gestión de tareas, Asana permite a los usuarios visualizar, asignar y dar seguimiento a las tareas y proyectos de manera efectiva.

Una de las características sobresalientes de Asana es su enfoque en la colaboración y la comunicación en equipo. Los usuarios pueden crear proyectos, asignar tareas a los miembros del equipo y establecer plazos claros. Además, pueden mantener conversaciones dentro de cada tarea, compartir archivos y realizar comentarios, lo que facilita la colaboración y el intercambio de ideas en tiempo real [17].

La interfaz *Kanban* intuitiva de Asana permite a los usuarios gestionar sus proyectos de manera clara y estructurada. Ofrece una variedad de vistas, como la vista de lista, la vista de tablero y la vista de calendario, para poder adaptarse a las preferencias de cada

usuario. Estas vistas flexibles permiten una gestión visual y personalizada de las tareas y proyectos.

Asana destaca por su capacidad de personalización. Los usuarios pueden crear plantillas y flujos de trabajo personalizados, lo que les permite adaptar la herramienta a sus propios procesos y necesidades específicas. Asimismo, ofrece una amplia gama de integraciones con otras herramientas populares, lo que permite a los usuarios conectar Asana con otras aplicaciones y maximizar su flujo de trabajo.

La gestión eficiente de tareas es otro aspecto clave de Asana. Los usuarios pueden asignar tareas a miembros específicos del equipo, establecer prioridades, agregar descripciones y adjuntar archivos relevantes. También pueden realizar un seguimiento del progreso de las tareas, establecer recordatorios y recibir notificaciones para mantenerse al día con los plazos y las actualizaciones importantes.

Asana va más allá de la gestión de tareas básica al ofrecer funciones avanzadas como automatizaciones y reglas. Estas características permiten automatizar tareas repetitivas, establecer flujos de trabajo automatizados y garantizar que se cumplan los plazos y las dependencias de las tareas.

El modelo de negocio que presenta Asana es el *Freemium*, dicho término consiste en la mezcla de las palabras *Free* y *Premium*, que significan gratis y de pago respectivamente. El modelo se basa en dar una serie de caracterísiticas gratuitas y otras especiales de pago.

En conclusión, Asana es un poderoso gestor de proyectos y colaboración en equipo que se destaca por su enfoque en la automatización. Con su interfaz intuitiva, capacidad de personalización y funciones avanzadas, Asana se posiciona como una solución confiable para equipos que buscan optimizar su productividad y lograr resultados exitosos en sus proyectos. Sin embargo, al igual que Jira, Asana no es una plataforma de código abierto.

3.2.3. Basecamp

Basecamp es una completa plataforma de gestión de proyectos y colaboración que se ha ganado una sólida reputación en el mundo empresarial. Con su enfoque simplificado y centrado en la comunicación efectiva, Basecamp ofrece una amplia gama de herramientas y funcionalidades diseñadas para ayudar a los equipos a organizar, colaborar y completar sus proyectos de manera eficiente [18].

La estructura principal de Basecamp se basa en proyectos, que funcionan como espacios de trabajo virtuales para cada iniciativa o área de trabajo. Dentro de cada proyecto, los usuarios pueden crear y organizar listas de tareas, establecer fechas de vencimiento, asignar responsabilidades y mantener un seguimiento detallado del progreso. Esta estructura jerárquica proporciona una visión clara de las tareas pendientes, en progreso y completadas, lo que facilita la gestión y la colaboración en equipo.

Una de las características distintivas de Basecamp es su enfoque en la comunicación transparente y centralizada. La herramienta ofrece una amplia gama de funcionalidades para fomentar la comunicación efectiva entre los miembros del equipo, incluyendo chats en tiempo real, comentarios en las tareas, foros de discusión y mensajes directos. Esta comunicación integrada elimina la necesidad de depender de correos electrónicos dispersos y facilita la colaboración en tiempo real.

Basecamp también ofrece un conjunto de herramientas adicionales que complementan

la gestión de proyectos, como la organización de documentos, el seguimiento de tiempo y el calendario de eventos. Estas funcionalidades adicionales permiten a los equipos mantener todos los aspectos de su proyecto en un solo lugar, evitando la necesidad de utilizar múltiples herramientas y plataformas.

La simplicidad y la facilidad de uso son elementos clave en Basecamp. La interfaz *Kanban* intuitiva y amigable permite a los usuarios comenzar rápidamente a utilizar la plataforma sin necesidad de un entrenamiento extenso. Los comandos y las funcionalidades están diseñados de manera clara y accesible, lo que facilita la adopción y el uso continuo por parte de los miembros del equipo.

Basecamp también se destaca por su enfoque en la privacidad y la seguridad de los datos. La plataforma implementa medidas rigurosas para proteger la información confidencial y ofrece opciones de control de acceso para garantizar que solo los miembros autorizados tengan acceso a determinados proyectos y recursos.

El modelo de negocio que presenta Basecamp consiste en pagar por los proyectos que se gestionan con la herramienta.

A modo de resumen, Basecamp es una plataforma de gestión de proyectos y colaboración que se centra en la comunicación efectiva y la simplicidad. Con sus herramientas y funcionalidades intuitivas, Basecamp ayuda a los equipos a organizar sus proyectos, mantener una comunicación transparente y completar sus tareas de manera eficiente. Ya sea para equipos pequeños o grandes, Basecamp proporciona una solución integral para la gestión de proyectos y la colaboración efectiva en el entorno empresarial. Al igual que sus predecesores explicandos anteriormente, se trata de una plataforma de código cerrado.

3.2.4. Redmine

Redmine es un poderoso gestor de proyectos de código abierto que ha ganado popularidad debido a su flexibilidad, funcionalidades completas y su naturaleza completamente gratuita. Esta plataforma basada en web brinda a los equipos una amplia gama de herramientas para planificar, coordinar y controlar sus proyectos de manera eficiente.

Una de las principales ventajas de Redmine es su naturaleza de código abierto, lo que significa que su código fuente está disponible para que los usuarios lo revisen, modifiquen y personalicen según sus necesidades específicas. Esto permite a los equipos adaptar Redmine para satisfacer sus requisitos de gestión de proyectos únicos y añadir funcionalidades adicionales según sea necesario [18].

La estructura de Redmine se basa en la organización de proyectos, donde los usuarios pueden crear y gestionar múltiples proyectos en un solo lugar. Dentro de cada proyecto, se pueden definir y asignar tareas, establecer fechas límite, hacer un seguimiento del progreso y asignar prioridades. Además, los usuarios pueden colaborar en tiempo real, compartir documentos y archivos, y mantener una comunicación efectiva utilizando foros de discusión y mensajes.

Una de las características destacadas de Redmine es su capacidad de personalización. Los usuarios pueden adaptar la interfaz de usuario, añadir campos personalizados, crear flujos de trabajo personalizados y utilizar complementos y extensiones disponibles en la comunidad de usuarios de Redmine. Esto brinda una gran flexibilidad para adaptar la plataforma a los requisitos específicos de cada equipo y proyecto.

Además, Redmine cuenta con una sólida comunidad de usuarios y desarrolladores que contribuyen activamente al desarrollo y mejora continua de la plataforma. Esta comunidad brinda soporte, comparte conocimientos y ofrece una amplia variedad de plugins y temas para ampliar las funcionalidades y personalización de Redmine.

Por último, la característica que hace resaltar a Redmine frente a los demás gestores de proyectos es que es completamente gratuito.

3.3. Comparativa de gestores de proyectos

A continuación se presenta una tabla que presenta la comparación entre los diferentes gestores de proyectos estudiados. Como se puede apreciar en la tabla, todos ofrecen prácticamente las mismas características, sin embargo, la diferencia clave se encuentra en el Código abierto y el Precio. A cada gestor se le ha asociado una puntuación, dicha puntuación se calcula de la siguiente manera: si se cumple una característica se obtienen 10 puntos. La característica de Precio se busca dar más puntos a aquellos gestores que sean fáciles de obtener, por tanto, aquellos gestores que tengan un modelo de negocio en el que haya que pagar, no obtienen puntos. Por otro lado, los modelos híbridos (como el *Freemium*) obtienen 10 puntos. Por último, aquellos que sean gratis obtienen 50 puntos.

Característica	**Jira**	**Asana**	**Basecamp**	**Redmine**
Tracking de Horas	Sí	Sí	Sí	Sí
Tableros Kanban	Sí	Sí	Sí	Sí
Gestión de Documentos	Sí	Sí	Sí	Sí
Colaboración en Tiempo Real	Sí	Sí	Sí	Sí
Personalización	Sí	Sí	Sí	Sí
Código abierto	No	No	No	Sí
Precio	Comercial	Freemium	Pago por proyecto	Gratis
Puntaje	50	60	50	110

Tabla 3.1: Comparativa de Gestores de Proyectos

Tras el análisis de las características brindadas por los gestores de proyectos anteriormente mencionados, se ha decidido elegir Redmine como herramienta gestora, dada su naturaleza de código abierto y su plan gratuito.

Capítulo 4

Implementación de proyectos con Redmine

4.1. Investigación y Análisis

A lo largo del desarrollo de la investigación se realizaron una serie de entrevistas cuya finalidad era poder especificar las necesidades que debería cubrir el gestor de proyectos. Estas entrevistas fueron realizadas a los investigadores y la supervisión de los directores del trabajo.

4.1.1. Especificación de requisitos

Tras las entrevistas se pudo realizar una toma de requisitos específica y se pudo entender el flujo de trabajo deseado, dando como resultado los los requerimientos funcionales de la aplicación. Estos se encuentran descritos en el apéndice A.

4.1.2. Flujo de trabajo

Además de obtener los requisitios funcionales de la aplicación, también se obtuvo el flujo de trabajo que deberían seguir las propuestas de proyectos que se presenta en la Figura 4.1.

En primer lugar, se recibe una propuesta por parte del consorcio. Esta propuesta es debatida por la Policía Nacional. Tras debatir, la propuesta puede ser rechazada o aceptada. Si se rechaza, termina el proceso. Si se acepta, se deben generar y emitir una serie de documentos necesarios para la gestión del proyecto, terminando el proceso. Para explicar cómo se ha llegado a la implementación final realizada, primero se explican unos conceptos clave relacionados con la implementación de *Redmine* y de *Ruby on rails*.

4.2. Implementación de Ruby on Rails

Redmine fue desarrollado en el lenguaje de programación *Ruby on Rails* [19] y cuenta con una amplia comunidad que contribuye a su desarrollo y mejora continua. A continuación,

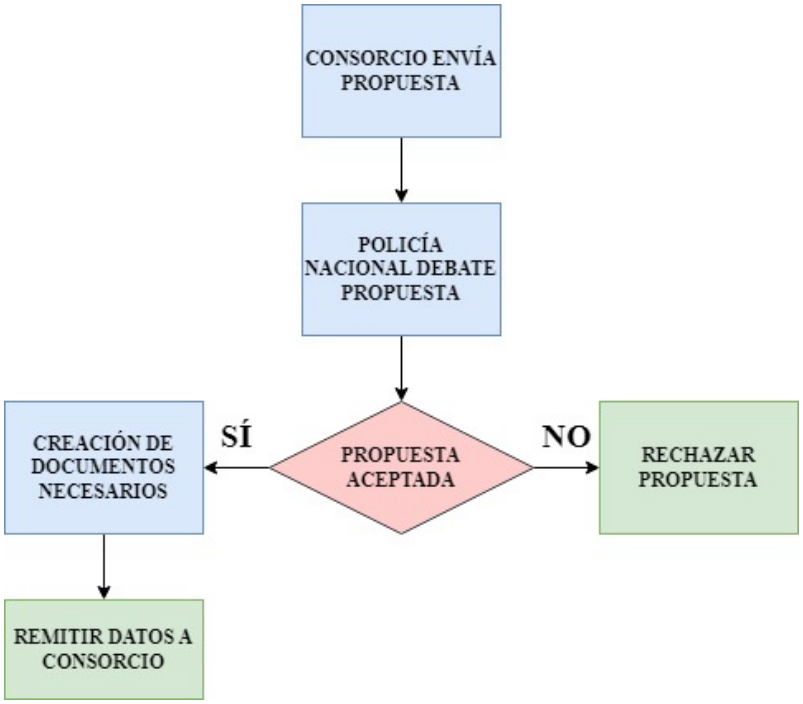

Figura 4.1: Flujo de la propuesta

se proporciona una explicación de los componentes más importantes de una aplicación desarrollada en *Ruby on Rails* [20].

4.2.1. Patrón Modelo-Vista-Controlador

El patrón *Modelo-Vista-Controlador* (MVC) [21] es un patrón de diseño arquitectónico ampliamente utilizado en el desarrollo de aplicaciones de software. Proporciona una estructura organizada y modularizada que separa la lógica de la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Cada uno de estos componentes tiene responsabilidades específicas y se comunica con los otros componentes de manera definida. A continuación, se explican en detalle cada uno de estos componentes y su funcionamiento:

- Modelo: El Modelo representa los datos y la lógica de negocio de la aplicación. Es responsable de gestionar y mantener el estado de los datos, así como de realizar las operaciones relacionadas con la persistencia y manipulación de los mismos. El Modelo encapsula la lógica de negocio y proporciona métodos para acceder, actualizar y manipular los datos subyacentes. Esto puede incluir operaciones como la lectura y escritura en la base de datos, cálculos y validaciones de datos, y la implementación de reglas de negocio. El Modelo no tiene conocimiento sobre la interfaz de usuario ni sobre cómo se presentan los datos al usuario.

- Vista: La Vista es responsable de la presentación de los datos al usuario y de la interacción con el mismo. Representa la interfaz de usuario y se encarga de mostrar los datos de manera visualmente atractiva y comprensible. La Vista recibe información del Modelo y la utiliza para generar la representación gráfica o textual que se muestra al usuario. Esto puede incluir la generación de páginas web, la creación

de interfaces de usuario interactivas, la visualización de informes y gráficos, entre otros. La Vista no tiene conocimiento directo sobre el Modelo ni sobre la lógica de negocio de la aplicación.

- Controlador: El Controlador actúa como intermediario entre el Modelo y la Vista. Es responsable de recibir las interacciones del usuario a través de la interfaz de usuario y de controlar el flujo de la aplicación. El Controlador interpreta las acciones del usuario y coordina las respuestas correspondientes del Modelo y la Vista. Cuando el usuario realiza una acción, como hacer clic en un botón o enviar un formulario, el Controlador recibe esta acción y toma las decisiones apropiadas. Puede invocar métodos del Modelo para realizar operaciones en los datos y actualizar su estado, y también puede actualizar la Vista para reflejar los cambios realizados en los datos. El Controlador no tiene conocimiento sobre cómo se almacenan los datos ni cómo se presentan visualmente.

El flujo de la aplicación en el patrón MVC se basa en la interacción entre estos tres componentes. Cuando el usuario realiza una acción en la interfaz de usuario, como enviar un formulario, el Controlador captura esta acción y decide cómo responder. Puede interactuar con el Modelo para obtener los datos necesarios, procesarlos y realizar operaciones, como validar la información ingresada por el usuario. Una vez que el Modelo ha realizado las operaciones requeridas, el Controlador actualiza la Vista correspondiente para reflejar los cambios realizados en los datos. Esto puede implicar la actualización de la interfaz de usuario, la generación de mensajes de confirmación, la visualización de errores, entre otros. La Vista, a su vez, muestra la información actualizada al usuario y permite nuevas interacciones. En la figura 4.2 se representa el flujo de trabajo en el patrón MVC.

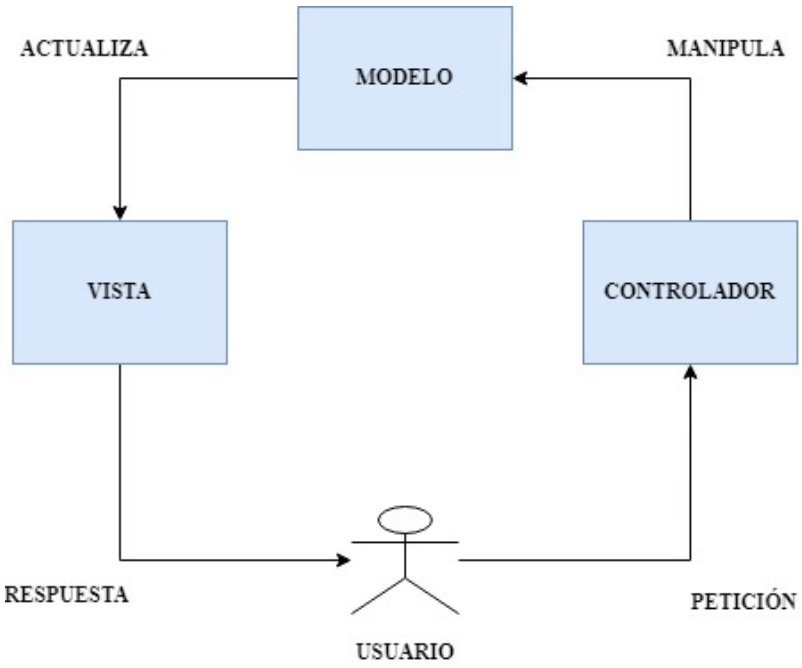

Figura 4.2: Modelo-Vista-Controlador

El patrón MVC proporciona varios beneficios y ventajas en el desarrollo de aplicaciones [21]:

- Separación de responsabilidades: Al dividir la lógica de la aplicación en tres componentes distintos, el patrón MVC permite una mejor organización y separación de las responsabilidades. Esto facilita el mantenimiento y la evolución de la aplicación, ya que cada componente tiene un propósito específico y puede modificarse de manera independiente.
- Reutilización de código: La separación de la lógica de negocio del Modelo y la presentación en la Vista permite reutilizar ambos componentes en diferentes contextos. Por ejemplo, se puede utilizar el mismo Modelo para diferentes interfaces de usuario o se puede utilizar una Vista para mostrar diferentes representaciones de los mismos datos.
- Facilidad de pruebas: El patrón MVC facilita las pruebas unitarias, ya que los componentes son independientes entre sí y se pueden probar por separado. Por ejemplo, se pueden realizar pruebas en el Modelo sin necesidad de la Vista o el Controlador, lo que permite una mayor cobertura de pruebas y una mejor detección de errores.
- Flexibilidad y escalabilidad: Al separar la lógica de la aplicación en componentes distintos, el patrón MVC permite una mayor flexibilidad y escalabilidad. Por ejemplo, se puede cambiar la Vista sin afectar el Modelo, o se puede agregar un nuevo Controlador para manejar nuevos flujos de la aplicación sin modificar los demás componentes.

En resumen, el patrón MVC proporciona una estructura clara y modularizada para el desarrollo de aplicaciones. Al separar la lógica de negocio, la presentación y el control de la aplicación, el patrón MVC mejora la organización, la reutilización de código, la facilidad de pruebas y la flexibilidad de la aplicación. Es ampliamente utilizado en el desarrollo de aplicaciones web y de software en general, y ha demostrado ser efectivo en la construcción de sistemas robustos y escalables.

4.2.2. Patrón Active Record

El patrón *Active Record* es un patrón de diseño de software que combina el manejo de datos y la lógica de negocio en una sola clase, lo que permite interactuar con la base de datos de forma sencilla y transparente. Este patrón se utiliza comúnmente en el desarrollo de aplicaciones que requieren un acceso persistente a datos, como aplicaciones web y sistemas de gestión de bases de datos.

En el patrón Active Record, cada clase del modelo representa una tabla en la base de datos y cada instancia de esa clase representa una fila en esa tabla. Cada atributo de la clase se mapea a una columna en la tabla, lo que permite acceder y manipular los datos de manera intuitiva.

El patrón Active Record se basa en varios principios clave [22]:

- Conexión a la base de datos: La clase Active Record se encarga de establecer la conexión con la base de datos y proporciona métodos para ejecutar consultas y transacciones.
- Mapeo objeto-relacional: El patrón Active Record utiliza técnicas de mapeo objeto-relacional para establecer la correspondencia entre los objetos de la clase

y las filas de la tabla en la base de datos. Esto implica mapear los atributos de la clase a las columnas de la tabla y proporcionar métodos para realizar operaciones *Create, Read, Update, Delete* (CRUD) en la base de datos.

- Validación de datos: El patrón Active Record incluye la capacidad de validar los datos antes de almacenarlos en la base de datos. Esto se logra mediante la definición de reglas de validación en la clase Active Record, como la longitud mínima o máxima de un campo, el formato de un campo de fecha, etc. Estas reglas se aplican automáticamente antes de realizar cualquier operación de escritura en la base de datos, lo que ayuda a mantener la integridad de los datos.
- Relaciones entre objetos: El patrón Active Record permite establecer relaciones entre las clases del modelo, lo que refleja las relaciones entre las tablas en la base de datos. Por ejemplo, una clase puede tener una relación de uno a muchos con otra clase, lo que significa que una instancia de la clase puede tener varias instancias relacionadas en otra clase. Estas relaciones se definen mediante la asociación de claves primarias y externas en las tablas, y se proporcionan métodos en la clase Active Record para acceder y manipular estas relaciones.
- Persistencia de datos: El patrón Active Record se encarga de la persistencia de datos, es decir, de almacenar y recuperar objetos desde la base de datos. Proporciona métodos para guardar un objeto en la base de datos, actualizar sus atributos, eliminarlo de la base de datos y recuperar objetos de la base de datos basados en condiciones específicas.

Las ventajas que presenta el patrón son las siguientes:

- Simplicidad: Al combinar la lógica de negocio y el acceso a datos en una sola clase, el patrón Active Record simplifica el desarrollo y el mantenimiento del código.
- Transparencia: La interacción con la base de datos se realiza de forma transparente a través de los métodos de la clase Active Record, lo que facilita el manejo de datos persistentes.
- Rapidez de desarrollo: El patrón Active Record permite crear rápidamente modelos de datos y realizar operaciones CRUD sin necesidad de escribir consultas SQL personalizadas, lo que agiliza el desarrollo de la aplicación.
- Facilidad de uso: El patrón Active Record proporciona una interfaz sencilla y coherente para interactuar con los datos, lo que facilita su uso para desarrolladores que no están familiarizados con detalles específicos de la base de datos.
- Portabilidad: Al abstraer la capa de acceso a datos, el patrón Active Record permite que la aplicación sea más portátil, ya que los detalles específicos de la base de datos se encapsulan dentro de la implementación del Active Record.

4.3. Implementación de Redmine

La implementación de *Redmine* es bastante extensa y compleja, ya que se trata de un sistema de gestión de proyectos completo con múltiples funcionalidades. Comprendidos los

conceptos anteriormente explicados, se presentan algunos de los modelos más importantes en *Redmine* [23]:

- *Issue*: El modelo Issue representa una petición o tarea dentro de un proyecto en Redmine. Es uno de los modelos fundamentales y contiene información detallada sobre cada problema, como su título, descripción, estado, prioridad, asignado a, fecha de inicio, fecha de vencimiento, entre otros. Está estrechamente relacionado con muchos modelos ya que se trata de la herramienta de trabajo.
- *Tracker*: El modelo Tracker se utiliza para clasificar y organizar las peticiones o tareas (issues) dentro de un proyecto. El término "tracker"se puede traducir como rastreador.º "seguimiento.en español, y hace referencia a los tipos de peticiones que pueden existir en Redmine. Cada proyecto en Redmine puede tener diferentes Trackers definidos, lo que permite adaptar la herramienta a las necesidades específicas de cada proyecto. Por ejemplo, un proyecto de desarrollo de software tendrá Trackers diferente a un proyecto de gestión de ventas. El Modelo Tracker en Redmine almacena la información relacionada con cada tipo de problema, incluyendo su nombre, descripción y configuraciones adicionales. Algunas de las configuraciones comunes asociadas a los trackers son:
  - Flujo de trabajo: Cada Tracker puede tener su propio flujo de trabajo, que define los diferentes estados que un problema puede tener y las transiciones permitidas entre ellos.
  - Campos personalizados: Los Trackers permiten definir campos personalizados adicionales que pueden ser específicos de cada tipo de problema. Estos campos pueden ser de diferentes tipos, como texto, fecha, lista desplegable, entre otros, y se utilizan para capturar información adicional relevante para cada tipo de problema.
  - Configuración de visibilidad: Los Trackers pueden tener configuraciones de visibilidad que determinan quiénes pueden ver y acceder a los problemas asociados a ese Tracker. Esto permite restringir el acceso solo a usuarios específicos o grupos de usuarios.
- *Project*: El modelo Project representa un proyecto en Redmine. Contiene información general sobre el proyecto, como su nombre, descripción, identificador único, fecha de inicio, fecha de finalización, entre otros. Un proyecto es la unidad de organización de *Redmine*.
- *User*: El modelo User representa un usuario en Redmine. Almacena información sobre los usuarios registrados en el sistema, como su nombre, dirección de correo electrónico, contraseña, roles asignados, preferencias de visualización, entre otros.

4.3.1. Extensiones

Una extensión en Redmine tiene múltiples usos y aporta funcionalidades adicionales a la plataforma base. A continuación, se detallan algunas de las principales razones por las que las extensiones son valiosas en Redmine [24]:

- Extender la funcionalidad: Las extensiones permiten extender las capacidades de Redmine agregando nuevas características y herramientas. Pueden agregar módulos

específicos, como seguimiento de tiempo adicional, integración con sistemas externos, generación de informes personalizados, integración con control de versiones, entre otros. Esto permite adaptar Redmine a las necesidades específicas de un proyecto o una organización.

- Personalización y adaptabilidad: Las extensiones proporcionan la capacidad de personalizar y adaptar Redmine según los requisitos específicos de una organización. Esto puede incluir la personalización de la interfaz de usuario, la adición de campos personalizados, la creación de flujos de trabajo personalizados, la configuración de reglas de negocio específicas y la incorporación de procesos personalizados. Las extensiones permiten que Redmine se ajuste a los flujos de trabajo existentes y a las prácticas de una organización.

- Integración con otras herramientas y sistemas: Las extensiones pueden facilitar la integración de Redmine con otras herramientas y sistemas utilizados en una organización. Esto incluye integraciones con servicios de almacenamiento en la nube como *Dropbox* o *Google Drive* , sistemas de automatización como *Jenkins*, entre otros. Estas integraciones mejoran la productividad y la colaboración al permitir que Redmine se conecte y comparta información con otras herramientas.

- Mejora de la productividad y eficiencia: Las extensiones pueden automatizar tareas repetitivas, proporcionar acceso rápido a información relevante y optimizar procesos dentro de Redmine. Por ejemplo, una extensión podría agregar plantillas predefinidas para la creación rápida de nuevos elementos, proporcionar atajos de teclado personalizados, agregar recordatorios o notificaciones automatizadas, o generar informes y métricas para el seguimiento del progreso del proyecto. Estas mejoras ayudan a ahorrar tiempo y aumentan la eficiencia en el uso de Redmine.

- Complementar las capacidades existentes: Algunas veces, Redmine puede carecer de ciertas características o tener limitaciones en su funcionalidad base. Las extensiones pueden llenar esas brechas agregando las capacidades faltantes. Por ejemplo, una extensión podría agregar soporte para diagramas de Gantt, gráficos para seguimiento ágil, capacidades avanzadas de gestión de documentos, integración con herramientas de chat y colaboración en tiempo real, entre otros. Esto permite aprovechar al máximo Redmine y ampliar su alcance.

4.4. Implementación del flujo de trabajo

Para implementar el flujo de trabajo dentro de *Redmine*, en primer lugar el Consorcio debe enviar la propuesta, dicha propuesta se crea con un formulario en el que se introducen los siguientes datos, a petición de la Polcía Nacional, estos datos están nombrados en inglés:

- Proposal Acronym
- Call
- Type of Action
- Topic
- Budget

- Budget allocate to Spanish National Police
- Keywords
- Abstract
- Workload expeted from Spanish National Police
- Coordinating organization
- Main Contact Person
- Email address
- Department
- Telephone
- Deadline

El formulario resultante es el representado por la figura 4.3.

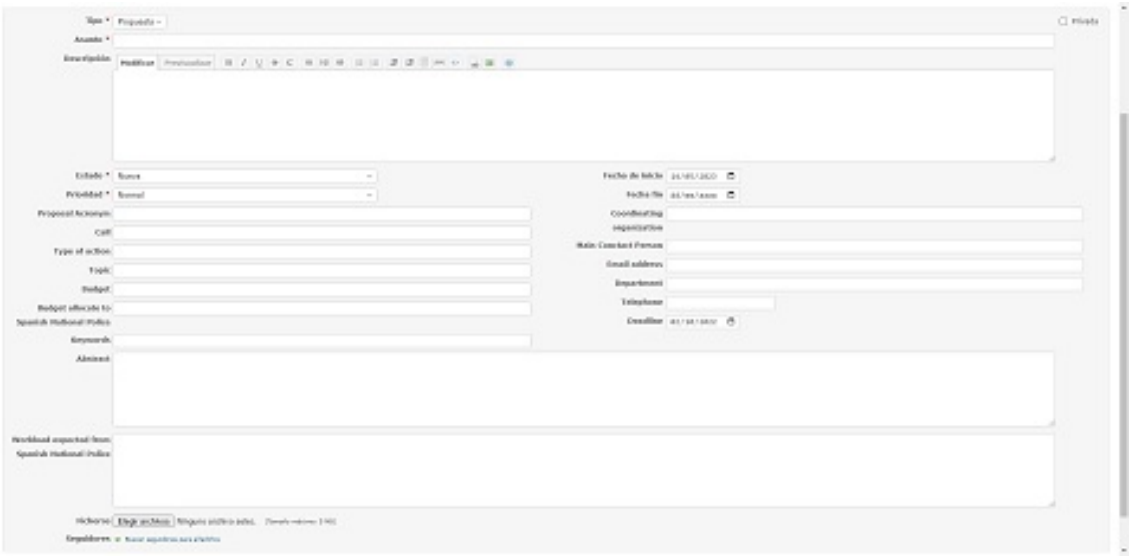

Figura 4.3: Formulario de propuesta

Una vez cumplimentado dicho formulario, la propuesta se crea, haciendo que el departamento que gestiona los proyectos del Consorcio sea avisado mediante correo electrónico. En este punto, la Policía Nacional lleva a cabo una serie de reuniones para aceptar o rechazar la propuesta, cambiando el estado de la propuesta a En negociaciones. En caso de que se rechace la propuesta, simplemente se cambia su estado a Rechazada y se guarda en la base de datos una entrada con la propuesta rechazada. En el caso de que la propuesta resulte aceptada, se crea un proyecto asociado a dicha propuesta para que se puedan generar los informes que pide el Consorcio.

4.5. Pruebas y Resultados

4.5.1. Introducción

En esta sección, se presentan los resultados obtenidos después de la implementación del gestor de proyectos en *Redmine*. Para garantizar la calidad y la adecuación a los requisitos, se utilizaron varias estrategias, incluyendo la realización de reuniones en las que se hacían pruebas de concepto y la aplicación de pruebas unitarias.

4.5.2. Pruebas de concepto

Para asegurar que la implementación del gestor de proyectos en *Redmine* cumplía con las necesidades específicas de la Policía Nacional, se llevaron a cabo reuniones con los representantes de la institución. Estas reuniones permitieron mostrar las características desarrolladas hasta el momento y obtener una retroalimentación de los usuarios finales del gestor.

Tal como se describe en la metodología Scrum, tras recibir la retroalimentación de la Policía Nacional, se realizaban los ajustes y modificaciones necesarias en la implementación del gestor de proyectos. Este proceso de coordinación y retroalimentación se repetía en cada iteración, permitiendo una colaboración estrecha y asegurando que el sistema se ajustara a las necesidades reales de los usuarios.

4.6. Pruebas de funcionamiento

Además de las reuniones con la Policía Nacional, se realizaron pruebas funcionales para garantizar la calidad y el correcto funcionamiento del gestor de proyectos en Redmine. Se utilizaron pruebas unitarias tanto de *Redmine* como de los plugins desarrollados.

Las pruebas unitarias consistieron en la ejecución de las pruebas automatizados proporcionados por Redmine y de las extensiones. Estas pruebas verificaban el funcionamiento de los componentes individuales del sistema, como los modelos, controladores y vistas. Se aseguraba que las funcionalidades desarrolladas cumplían con los criterios de aceptación definidos durante la etapa de análisis y diseño.

La ejecución de pruebas unitarias no solo permitió verificar el comportamiento esperado de la aplicación, sino que también facilitó la detección temprana de posibles errores y problemas de integración. Se realizaron pruebas exhaustivas para cubrir diferentes escenarios y se corrigieron los problemas identificados antes de proceder con la implementación final [24].

4.7. Resultados y conclusiones

Gracias a la aplicación de reuniones de retroalimentación con la Policía Nacional y la realización de pruebas unitarias, se lograron obtener resultados satisfactorios en la implementación del gestor de proyectos en Redmine.

Las reuniones con los usuarios finales del gestor permitieron recopilar valiosos comentarios y sugerencias que se tuvieron en cuenta para ajustar y mejorar la

implementación del sistema. La metodología de implementación iterativa y las reuniones periódicas de coordinación fueron fundamentales para asegurar la adaptación del gestor de proyectos a las necesidades específicas de la institución.

Por otro lado, las pruebas unitarias de Redmine y de las extensiones garantizaron el correcto funcionamiento de las funcionalidades implementadas. Se detectaron y solucionaron errores antes de la puesta en producción, lo que contribuyó a la estabilidad y confiabilidad del sistema. Las pruebas unitarias permitieron verificar el comportamiento esperado de cada componente, asegurando que cumpliera con los criterios de aceptación definidos previamente.

En conclusión, la combinación de entrevistas con los usuarios y la realización pruebas unitarias fue crucial para evaluar y validar la implementación del gestor. Estas estrategias garantizaron que el sistema cumpliera con los requisitos específicos de la Policía Nacional, al tiempo que se aseguraba la calidad y el correcto funcionamiento del software. La retroalimentación recibida y los ajustes realizados en cada etapa del desarrollo contribuyeron a la satisfacción de los usuarios finales y al éxito de la implementación del gestor de proyectos.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En conclusión, el trabajo realizado brindó una oportunidad invaluable para aplicar los conocimientos adquiridos durante el estudio de la Ingeniería.

Este proyecto permitió explorar en profundidad las distintas metodologías de trabajo utilizadas en la industria y comprender su importancia para la eficiencia y el éxito de los proyectos de desarrollo de software. Además, se analizaron en detalle las principales herramientas informáticas disponibles para la gestión de proyectos, brindando una visión integral de cómo utilizar eficazmente estas herramientas para planificar, coordinar y controlar el progreso del trabajo.

El trabajo de investigación y análisis llevado a cabo en este proyecto proporcionó una comprensión sólida de las diferentes metodologías de trabajo, incluyendo enfoques tradicionales y ágiles como Scrum, Kanban y Cascada. Se evaluaron sus ventajas, desafíos y áreas de aplicación más adecuadas, lo que permitió una visión crítica y fundamentada sobre cuál metodología es más adecuada para cada tipo de proyecto.

Además, se profundizó en el uso de herramientas informáticas específicas para la gestión de proyectos, como Jira o Redmine, examinando sus características, funcionalidades y mejores prácticas para su implementación exitosa. Se exploraron aspectos clave, como la gestión de tareas, el seguimiento del progreso, la asignación de recursos y la comunicación efectiva dentro de los equipos de trabajo.

Se empleó la experiencia en técnicas de desarrollo de software, comprensión de requisitos y diseño de sistemas para abordar las necesidades y desafíos presentes en la gestión de proyectos con herramientas informáticas.

Este proyecto no solo proporcionó una visión práctica y aplicada de las metodologías de trabajo y la gestión de proyectos, sino que también permitió desarrollar habilidades de investigación, análisis crítico y toma de decisiones. Estas habilidades son fundamentales para enfrentar los desafíos del mundo laboral y contribuir de manera efectiva en proyectos de desarrollo de software.

5.2. Trabajo Futuro

Como posibles trabajos futuros pueden señalarse los siguientes:

- Ampliar funcionalidad del gestor: actualmente el gestor implementado permite la gestión de propuestas de proyecto recibidos desde Horizonte Europa, entre otros. Dada la estrecha relación del grupo de investigación al que pertenencen los directores del trabajo con la Policía Nacional, se propone ampliar con la gestión de Trabajos de Fin de Grado que se llevan a cabo en esta facultad.
- Búsqueda de extensiones: se propone la investigación de extensiones nuevas para satisfacer otro tipo de necesidades que la Policía Nacional no posee, pero que podrían ser de su interés. Como por ejemplo: *Redmine Agile Plugin*, *Redmine Finance Plugin* o *Redmine Messenger Plugin*.
- Actualizar versión: se propone también la migración de *Redmine* a una versión más nueva, teniendo en cuenta que algunas extensiones usadas podrían no ser compatibles con esta nueva versión.

Capítulo 6

Introduction

6.1. Motivation

Project management tools have become indispensable assets for organizations in a variety of sectors. These streamline project workflows, improve collaboration and ensure project delivery success. The motivation behind the development and use of project management tools stems from several key factors that address the challenges faced by the address the challenges faced by project teams and organizations. This section explores the main motivations underlying the adoption of project management tools and their impact on project success.

One of the main motivations for using project management tools is to achieve efficient task management. These tools provide a centralized platform where project teams can create, assign and track tasks throughout the project lifecycle. With task management capabilities, team members can prioritize assignments, set deadlines and monitor progress. This motivation stems from the need to ensure optimal resource allocation, avoid delays and effectively meet project milestones.

Effective collaboration is vital to project success, especially when dealing with cross-functional teams or remote work environments. Project management tools offer collaboration features such as shared calendars, shared documents, discussion forums and real-time communication channels. By facilitating seamless collaboration, these tools promote knowledge sharing, transparency and effective decision making among team members. The motivation for improving collaboration is to improve communication, foster teamwork and ultimately drive project success.

Efficient resource management is crucial to project success. Project management tools offer features to effectively plan, allocate and track resources. With features such as resource calendars, workload tracking and visibility into resource availability, these tools help project managers optimize resource allocation, avoid over- or under-utilization and ensure that the right resources are assigned to the right tasks at the right time. The motivation behind resource planning and allocation is to maximize resource utilization, minimize bottlenecks and improve overall project efficiency.

Tracking project progress and generating enlightening reports are essential for stakeholders to stay informed and make informed decisions. Project management tools offer robust tracking and reporting capabilities, allowing project managers to monitor milestones, track tasks and generate customized reports on project status, resource

utilization, budget and other critical metrics. The motivation behind progress tracking and reporting is to provide stakeholders with accurate and up-to-date information, identify potential risks or delays, and enable timely interventions for project success.

Every project has inherent risks that can affect its results. Project management tools help identify, assess and mitigate risks. These tools enable project teams to document risks, assign owners, define mitigation strategies and track risk mitigation progress. The motivation for risk management is to proactively identify and address potential risks, minimize their impact on project objectives, and improve overall project resilience.

In other words, the motivation behind project management tools lies in addressing the challenges faced by project teams, improving collaboration, optimizing resource allocation, enabling progress tracking and managing project risks. By adopting these tools, organizations can streamline project workflows, improve communication and ensure successful project execution. The continuous development and refinement of project management tools demonstrates an ongoing commitment to meet the evolving needs of project teams and organizations in an increasingly complex and dynamic business landscape.

6.2. Context

The National Police faces the challenge of managing numerous complex projects that require efficient task management. Initially, the National Police used disparate systems, primarily Microsoft Excel for task tracking and Microsoft Access for database management. However, the fragmented nature of these tools made it difficult to manage projects effectively, resulting in tedious communication and reporting processes. To address these challenges, the National Police sought a comprehensive solution that could unify its task and database management needs. Redmine emerged as the ideal project management tool, offering a centralized platform that seamlessly integrated task tracking and database management functions. By adopting Redmine, the National Police aimed to consolidate its project management processes, streamline communication and collaboration, ensure accurate data management and generate detailed reports for informed decision making.

This Final Degree Project aims to satisfy the need of the National Police to manage all those projects proposed by the Horizon Europe Program, successor of Horizon 2020. For this purpose, the Department of Software Engineering and Artificial Intelligence of the Faculty of Computer Science of the Complutense University of Madrid has been asked for help. The work has been developed jointly with the aforementioned department and the Innovation and Development Department of the National Police.

6.3. Work plan

The development of this work was carried out in three stages:

1. **Investigation**: The Investigation phase consisted of answering the question why it is necessary to organize work methodologies when developing a software project. Likewise, a specific work methodology was chosen. The results of this phase laid the foundations for the two subsequent phases, by providing a guide for the Development and Experimentation process.

2. **Development**: The development phase was divided into two parts, the first one consisted in carrying out an exhaustive research and analysis in order to know in depth the objectives of the project, the requirements and the specific needs of the National Police in the management of projects within the framework of the Horizon Europe Program. This phase involved an in-depth review of the working methodology currently used by the National Police and interviews to gather the necessary information to meet the proposed requirements.

   Also, an investigation of the existing project management tools was carried out, comparing them with each other and selecting the one that best met the needs posed by the requirements specified in the previous part.

   The second part focuses on the creation and implementation of the project management solution for the National Police. Based on the findings of the Research phase, this phase involved the customization of the selected project management tool and the integration of the necessary functionalities to meet the specific requirements of the Horizon Europe Program. The Development phase included tasks such as database design, user interface development, workflow implementation and system testing to ensure the functionality, usability and reliability of the solution.

3. **Testing**: The testing phase aimed to validate the effectiveness and efficiency of the developed project management solution. During this phase, the solution was deployed in a controlled environment within the *fly.io* platform, and real Horizon Europe Program scenarios and projects were simulated. The objective was to evaluate the performance of the solution, its ability to manage tasks and project-related data, and its compatibility with existing National Police workflows and processes. The results of this phase provided valuable information to refine and optimize the solution before its final implementation.

Development and Testing were repeated throughout the work due to the nature of the work methodology chosen in the research.

Below is a Gantt chart showing the activities developed during the work.

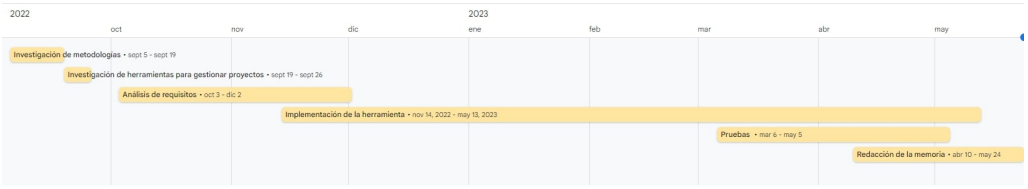

Figura 6.1: Diagrama de Gantt

6.4. Work structure

The rest of the work is organized in 7 chapters with the structure discussed below: Chapter 2 presents the theoretical framework of the work, in which it is investigated what work methodologies are and why they should be used. In addition, an investigation of the most currently used work methodologies was carried out.

Chapter 3 presents the State of the Art of the currently existing project managers, lists the most used project managers and describes the characteristics of each one. Finally,

a comparative table is added to gather the information of each manager, allowing to understand why *Redmine* is chosen as the final manager.

In Chapter 4, research is first carried out to obtain the specific requirements to be met by the project manager requested by the National Police. A formal proposal of requirements is made. Also, the workflow that the proposals received should follow is obtained. This is followed by a breakdown of the implementation of *Ruby on Rails* based applications. This is followed by an explanation of the implementation of *Redmine* and its possible extensions, along with the implementation of the workflow described. Finally, it describes what process has been carried out to ensure the effectiveness of the implementation performed.

Chapter 5 shows the main conclusions of this work and future lines of research.

Chapters 6 and 7 are English translations of the Introduction and Conclusions.

Capítulo 7

Conclusions and Future Work

7.1. Conclusions

In conclusion, the work performed provided an invaluable opportunity to apply the knowledge acquired during the study of engineering.

This project allowed to explore in depth the different work methodologies used in the industry and to understand their importance for the efficiency and success of software development projects. In addition, the main software tools available for project management were analyzed in detail, providing a comprehensive view of how to effectively use these tools to plan, coordinate and control the progress of the work.

The research and analysis work carried out in this project provided a solid understanding of the different working methodologies, including traditional and agile approaches such as Scrum, Kanban and Waterfall. Their advantages, challenges and most suitable areas of application were evaluated, allowing for a critical and informed view on which methodology is most suitable for which type of project.

In addition, the use of specific software tools for project management, such as Jira or Redmine, was discussed, examining their features, functionalities and best practices for their successful implementation. Key aspects such as task management, progress tracking, resource allocation and effective communication within work teams were explored.

Expertise in software development techniques, requirements understanding and system design was used to address the needs and challenges of managing projects with software tools.

This project not only provided a practical and applied vision of work methodologies and project management, but also allowed the development of research, critical analysis and decision making skills. These skills are fundamental to face the challenges of the working world and contribute effectively in software development projects.

7.2. Future Work

Possible future work may include the following:

- Expand functionality of the manager: currently the implemented manager allows the management of project proposals received from Horizon Europe, among others. Given the close relationship of the research group to which the directors of the work belong with the National Police, it is proposed to expand with the management of Final Degree Projects carried out in this faculty.
- Search for extensions: it is proposed to research new extensions to satisfy other types of needs that the National Police does not have, but that could be of interest to them. For example: *Redmine Agile Plugin*, *Redmine Finance Plugin* or *Redmine Messenger Plugin*.
- Upgrade version: it is also proposed to migrate *Redmine* to a newer version, taking into account that some extensions used may not be compatible with this new version.

Apéndice A

Requisitos funcionales

Tabla A.1: Organización de usuarios en proyectos

ID	RF-001
Actores	Usuario gestor
Precondición	Se encuentran usuarios y proyectos creados
Descripción	La herramienta debe dar la posibilidad de asignar usuarios a proyectos Estos usuarios pueden tener diferentes roles, teniendo así diferente visibilidad en las tareas que se encuentra en el proyecto.
Postcondición	Se confirma la unión del usuario al proyecto

Tabla A.2: Uso de conteo de horas

ID	RF-002
Actores	Cualquier usuario
Precondición	Usuario se encuentra autentificado en el sistema y forma parte del proyecto al que se quiere añadir horas
Descripción	Los usuarios serán capaces de añadir sus propias a horas a determinados tipos de tareas de un proyecto dado
Postcondición	Tras el ingreso de horas, se confirmará con un mensaje

Tabla A.3: Acceso mediante navegador

ID	RF-003
Actores	Cualquier usuario
Precondición	Usuario tiene acceso a Internet e introduce dominio en su navegador web
Descripción	La herramienta será accesible vía protocolo TCP/HTTP a través de navegadores HTML Chrome, Firefox e Internet Explorer.
Postcondición	Será mostrada la herramienta en el navegador web

Tabla A.4: Autentificación de usuarios

ID	RF-004
Actores	Cualquier usuario
Precondición	Usuario introduce credenciales que son su identificador de usuario y contraseña
Descripción	Los usuarios deben ser capaces de autentificarse en el sistema mediante uso de usuario y contraseña
Postcondición	Usuario es identificado en el sistema, en caso de que las credenciales no sean correctas se vuelven a pedir los datos

Tabla A.5: Registro de usuarios

ID	RF-005
Actores	Cualquier usuario
Precondición	Usuario introduce los datos necesarios para la creación de su cuenta. Entre estos datos se encuentra un correo electrónico válido, este no puede estar siendo utilizado por otro usuario del sistema
Descripción	Los usuarios deben ser capaces de registrarse en el sistema, introduciendo los datos necesarios para ello. Una vez registrado un usuario, un administrador debe activar su cuenta, este proceso se debe poder automatizar
Postcondición	El usuario recibe confirmación de su creación de cuenta de usuario y debe esperar a que se le active la cuenta, puede recibir confirmación por su correo electrónico

Tabla A.6: Modificación de datos de usuario

ID	RF-006
Actores	Cualquier usuario
Precondición	Usuario está autentificado en la aplicación
Descripción	Los usuarios de la aplicación deben ser capaces de modificar su información, esto es, su nombre, apellido y correo electrónico
Postcondición	Usuario recibe confirmación del cambio de los datos

Tabla A.7: Borrado de usuarios

ID	RF-007
Actores	Gestor
Precondición	Se encuentran usuarios creados en la aplicación
Descripción	La aplicación debe dar la posibilidad de eliminar los datos de un usuario
Postcondición	Usuario recibe confirmación del borrado

Tabla A.8: Bloqueo de usuarios

ID	RF-008
Actores	Gestor
Precondición	Se encuentran usuarios creados en la aplicación
Descripción	La aplicación debe dar la posibilidad de bloquear usuarios. Los usuarios bloqueados no podrán acceder a la aplicación hasta que se desbloqueen. Tampoco se les podrá asignar tareas, asignarlos como observadores o recibir correos electrónicos
Postcondición	Usuario recibe confirmación del bloqueo

Tabla A.9: Generar contraseña automáticamente

ID	RF-009
Actores	Gestor
Precondición	Se encuentran usuarios creados en la aplicación
Descripción	La aplicación debe dar la posibilidad de cambiar y generar una contraseña segura para un usuario. Estas nuevas credenciales podrán ser enviadas por correo electrónico al usuario
Postcondición	Usuario recibe confirmación del cambio de contraseña

Tabla A.10: Creación de grupos de usuarios

ID	RF-010
Actores	Gestor
Precondición	N/A
Descripción	Los usuarios deben ser capaces de organizarse en grupos, hay dos grupos de usuarios por defecto: Usuarios no registrados y Usuarios anónimos (no autentificados)
Postcondición	Tras la introducción de datos necesarios para crear un grupo, se recibe su confirmación

Tabla A.11: Organización de usuarios en grupos

ID	RF-011
Actores	Gestor
Precondición	Se encuentran usuarios y grupos creados en la aplicación
Descripción	Los usuarios deben ser capaces de organizarse en grupos, los grupos son simplemente un conjunto de usuarios bajo un mismo nombre. Los grupos pueden ser añadidos a proyectos, de la misma manera que se añaden usuarios normales
Postcondición	Se recibe confirmación de que el usuario ha sido añadido al grupo

Tabla A.12: Creación de proyectos

ID	RF-012
Actores	Gestor
Precondición	N/A
Descripción	Los usuarios gestores deben ser capaces de crear proyectos. Estos proyectos podrán tener subproyectos dependientes
Postcondición	Tras la introducción de los datos necesarios para la creación de un proyecto, el usuario recibe confirmación de su creación

Tabla A.13: Editar información de proyectos

ID	RF-013
Actores	Gestor
Precondición	N/A
Descripción	La aplicación debe dar la posibilidad de cambiar la información de un proyecto. Por defecto la información de un proyecto es su nombre, su descripción. Se puede modificar también los módulos que utiliza un proyecto, los módulos son las funcionalidades como conteo del tiempo, creación de tareas, etc.
Postcondición	Usuario recibe confirmación del cambio de información

Tabla A.14: Creación de roles y permisos

ID	RF-014
Actores	Gestor
Precondición	N/A
Descripción	La aplicación debe dar la posibilidad de crear roles. Un rol es una colección de permisos que aplican a un proyecto, un usuario puede tener varios roles dentro de un mismo proyecto
Postcondición	Tras la introducción de los datos necesarios para crear un rol, se confirma su creación

Tabla A.15: Asignar roles a usuarios

ID	RF-015
Actores	Gestor
Precondición	Se encuentran usuarios creados en la aplicación. El gestor se encuentra autentificado en el sistema
Descripción	Los usuarios podrán ser asignados roles, dependiendo del rol que sea, se tendrán unos permisos u otros
Postcondición	Usuario recibe confirmación de la asignación

Tabla A.16: Creación de tipos de tareas

ID	RF-016
Actores	Gestor
Precondición	N/A
Descripción	El tipo de una tarea es una manera de organizar información. Para cada tipo de tarea, se puede definir su nombre, el estado por defecto de las tareas, su flujo de trabajo y campos especiales creados por el gestor
Postcondición	Usuario recibe confirmación de la asignación

Tabla A.17: Creación de campos personalizados

ID	RF-017
Actores	Gestor
Precondición	N/A
Descripción	La aplicación debe ser capaz de crear y asignar campos personalizados. Los campos personalizados es información arbitraria que se puede añadir a cualquier entidad del sistema, entre ellos: tipos de tareas, tareas, tiempo invertido en una tarea, proyectos, usuarios, grupos de usuarios, etc.
Postcondición	Usuario recibe confirmación de la creación del campo personalizado

Tabla A.18: Repositorio de datos

ID	RF-018
Actores	Cualquier usuario
Precondición	Usuario tiene permiso para hacer subida de datos
Descripción	Los usuarios deben ser capaces de subir archivos al sistema. Las extensiones de archivos aceptadas y el tamaño máximo de subida se debe poder configurar en la aplicación
Postcondición	Usuario recibe confirmación de la subida

Tabla A.19: Capacidad para configurar notificaciones por correo electrónico

ID	RF-019
Actores	Gestor
Precondición	N/A
Descripción	Los usuarios pueden ser notificados de ciertos cambios por correo electrónico, por ejemplo, se debería poder avisar cada vez que se crea una tarea o cuando un usuario es asignado a una tarea
Postcondición	Usuario recibe confirmación de la creación de notificaciones por correo electrónico

Bibliografía

[1] Canva. Tablero kanban. https://marketplace.canva.com/EAElcD3jpX0/1/0/1600w/canva-memphis-tablero-kanban-lluvia-de-ideas-8flm3lKElDQ.jpg, 2023 (accessed May 15, 2023).

[2] Julie Delisle. Working time in multi-project settings: How project workers manage work overload. *International Journal of Project Management*, 38(7):419–428, 2020. Actors, Practices and Strategy Connections in Multi-Project Management.

[3] Pablo Lledó and Gustavo Rivarola. *Gestión de proyectos*. Pearson Educación Buenos Aires, 2007.

[4] Esteban Gabriel Maida and Julián Pacienzia. Metodologías de desarrollo de software. *Tesis de Licenciatura en Sistemas y Computación*, pages 12–30, 2015.

[5] AK Munns and BF Bjeirmi. The role of project management in achieving project success. *International Journal of Project Management*, 14(2):81–87, 1996.

[6] Juan David Yepes González, César Jesús Pardo Calvache, and Omar Salvador Gómez Gómez. Revisión sistemática acerca de la implementación de metodologías ágiles y otros modelos en micro, pequeñas y medianas empresas de software. *Revista Tecnológica-ESPOL*, 28(5), 2015.

[7] Kumar Gaurav and Bhatia Pradeep. Comparative Analysis of Software Engineering Models from Traditional to Modern Methodologies. pages 189–191, 2014.

[8] Winston W Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*, pages 328–329, 1987.

[9] Sundramoorthy Balaji and M Sundararajan Murugaiyan. Waterfall vs. v-model vs. agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, 2(1):26–30, 2012.

[10] R Galo Fariño. Modelo espiral de un proyecto de desarrollo de software. *Obtenido de http://www.ojovisual.net/galofarino/modeloespiral.pdf*, 2011.

[11] Jesse Santiago and Desirae Magallon. Critical path method. *CEE320, Winter 2013*, 2009.

[12] Rob Cole and Edward Scotcher. *Brilliant Agile project management: a practical guide to using Agile, Scrum and Kanban*. Pearson UK, 2016.

[13] José Joskowicz. Reglas y prácticas en extreme programming. *Universidad de Vigo*, 22, 2008.

[14] Christian Misobi Budoya, Mussa M Kissaka, and Joel S Mtebe. Instructional design enabled agile method using addie model and feature driven development process. *International Journal of Education and Development Using Information and Communication Technology*, 15(1):n1, 2019.

[15] Ken Schwaber. Scrum development process. In *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*, pages 117–134. Springer, 1997.

[16] Aleksandar Arnautović. Managing project using jira software. *Serbian Journal of Engineering Management*, 7(2):40–46, 2022.

[17] Tânia Ferreira, Juncal Gutiérrez-Artacho, and Jorge Bernardino. Freemium project management tools: Asana, freedcamp and ace project. In *Trends and Advances in Information Systems and Technologies: Volume 1 6*, pages 1026–1037. Springer, 2018.

[18] Muhammad Sajad, Muhammad Sadiq, Khawar Naveed, and Muhammad Shahid Iqbal. Software project management: Tools assessment, comparison and suggestions for future development. *International Journal of Computer Science and Network Security (IJCSNS)*, 16(1):36–37, 2016.

[19] David Heinemeier Hansson. Ruby on rails guides. https://rubyonrails.org, 2023 (accessed April 15, 2023).

[20] Julia Plekhanova. Evaluating web development frameworks: Django, ruby on rails and cakephp. *Institute for Business and Information Technology*, 20:3–15, 2009.

[21] James Bucanek. Model-view-controller pattern. *Learn Objective-C for Java Developers*, pages 353–356, 2009.

[22] Paweł Łuczak, Aneta Poniszewska-Maranda, and Vincent Karovič. The process of creating web applications in ruby on rails. *Developments in Information & Knowledge Management for Business Applications: Volume 1*, pages 375,382–386, 2021.

[23] Denis O Zmeev, Oleg A Zmeev, and Daniil V Tamazlykar. Implementation of essence practice into project management system redmine. In *2019 Actual Problems of Systems and Software Engineering (APSSE)*, pages 119–123. IEEE, 2019.

[24] Alex Bevilacqua. *Redmine plugin extension and development*. Packt Publishing Ltd, 2014.