Pontificia Universidad
JAVERIANA
Colombia

Plan de Administración de Proyecto (PMP)

Plataforma Freelance para Jóvenes: Conectando Talento con Oportunidades Laborales

Estudiantes:

Andrés Felipe Vélez Echeverry

Juan Camilo Gómez Contento

Thomas Alejandro Orjuela Bello

Estudiante	**Documento**	**Celular**	**Correo Javeriano**
Andrés Felipe Vélez Echeverry	cc. 1000942132	318-514-3658	andresveleze@javeriana.edu.co
Juan Camilo Gómez Contento	cc. 1019982319	321-4567-243	jcamilo.gomez@javeriana.edu.co
Thomas Alejandro Orjuela Bello	cc. 1014658502	313-8098727	talejandroorjuela@javeriana.edu.co

Director:

Pablo Andrés Márquez Fernández, Ing..

Director	**Documento**	**Celular**	**Teléfono Fijo**	**Correo Javeriano**
Pablo Andrés Márquez Fernández	cc. 80.756.159	318-712-3845	NA	marquezp-a@javeriana.edu.co

Pontificia Universidad Javeriana

Facultad de Ingeniería

Ingeniería de Sistemas

2025

Índice

Índice de figuras

Índice de cuadros

1. Vista General del Proyecto

1.1. Visión del Producto

Nuestra visión es transformar la inserción laboral de los jóvenes universitarios en América Latina mediante una plataforma digital innovadora que democratice el acceso a experiencias profesionales reales. Este producto busca cerrar la brecha entre la formación académica y las demandas del mercado laboral, ofreciendo a los estudiantes la oportunidad de trabajar en proyectos freelance adaptados a sus habilidades y disponibilidad, mientras construyen un portafolio competitivo.

A corto plazo, la plataforma conectará a estudiantes con contratistas mediante un sistema de emparejamiento impulsado por IA, facilitando colaboraciones flexibles y relevantes. A largo plazo, aspira a convertirse en un ecosistema clave para reducir el desempleo juvenil, fomentar la formalización laboral y dinamizar la economía al integrar talento joven en proyectos de impacto.

Con herramientas como perfiles verificados, evaluaciones mutuas y recursos de capacitación, no solo mejoraremos la empleabilidad de los estudiantes, sino que también incentivaremos a las empresas a confiar en el potencial de las nuevas generaciones. En el futuro, este modelo podría escalarse a otras regiones y sectores, estandarizando un puente sostenible entre educación y empleo.

1.2. Propósito, Alcance y Objetivos

1.2.1. Propósito

El propósito de este proyecto es diseñar e implementar una plataforma web freelance que transforme la manera en que los estudiantes universitarios acceden a oportunidades laborales tempranas, superando las barreras tradicionales de experiencia y formalidad en el mercado laboral. La plataforma busca responder a la problemática del desempleo juvenil y la falta de experiencia práctica al conectar directamente a estudiantes con contratistas mediante un sistema inteligente de emparejamiento.

Esto es especialmente valioso en el contexto latinoamericano, donde muchos jóvenes talentosos no logran insertarse en el mercado laboral por falta de oportunidades iniciales. La plataforma ofrecerá flexibilidad para trabajar en proyectos acordes a sus habilidades y disponibilidad, permitiéndoles construir un portafolio profesional a lo largo de su formación académica.

1.2.2. Alcance

El proyecto se enfoca en el desarrollo de una plataforma web con las siguientes características principales:

- Sistema de registro y verificación de identidad para estudiantes y contratistas.
- Perfiles personalizables con carga de CV, habilidades y categorías de trabajo.
- Motor de búsqueda y emparejamiento automatizado mediante IA.
- Módulo de evaluación y retroalimentación post-proyecto.
- Documentación técnica completa bajo estándares IEEE e ISO.

Para delimitar claramente el alcance del proyecto, se especifica que no incluirá:

- No se intenta desarrollar un nuevo modelo de negocio, se desarrollara unicamente una aplicación.
- Desarrollo de aplicaciones móviles nativas (solo versión web responsive).
- Integración con sistemas de pago o facturación
- Soporte para múltiples idiomas (versión inicial solo en español).
- Funcionalidades avanzadas de gestión de proyectos (seguimiento de horas, facturación, etc.).

1.2.3. Objetivos

Objetivo General Desarrollar una plataforma web de freelance que conecte estudiantes universitarios con contratistas, brindando acceso a proyectos reales que se integren a un portafolio profesional.
Objetivos Específicos

- Documentar los requerimientos funcionales y no funcionales para la aplicación.
- Diseñar una arquitectura técnica escalable para la plataforma.
- Realizar una Prueba de Concepto (PoC) para seleccionar el modelo de IA óptimo en emparejamiento.
- Integrar el modelo de IA como asistente de chat para búsqueda y emparejamiento.
- Implementar un demo funcional que integre los requerimientos funcionales y no funcionales.
- Realizar un set de pruebas que evalúen el funcionamiento en varios escenarios.

1.3. Supuestos y Restricciones

Para el desarrollo de la plataforma web freelance que conecta a estudiantes universitarios con oportunidades laborales, se han establecido los siguientes supuestos y restricciones que guiarán la ejecución del proyecto:

1.3.1. Supuestos:

- Los principales usuarios (estudiantes universitarios y contratistas) estarán dispuestos a adoptar una nueva plataforma digital para gestionar oportunidades laborales flexibles.
- La plataforma será accedida principalmente desde navegadores web en dispositivos móviles y computadoras personales con conexión a internet estable.
- Existirá suficiente interés por parte de contratistas (empresas y profesionales independientes) en publicar proyectos adecuados para estudiantes.
- Los modelos de IA para emparejamiento podrán entrenarse efectivamente con los datos de perfiles y proyectos disponibles.

1.3.2. Restricciones:

- El desarrollo estará limitado a una plataforma web responsive, sin incluir aplicaciones móviles nativas para iOS o Android en esta fase inicial.
- El proyecto debe completarse en un plazo máximo de 6 meses, divididos en fases claras de análisis, desarrollo y pruebas.
- La plataforma no incluirá funcionalidades de procesamiento de pagos o transacciones financieras en esta versión.
- El soporte se limitará al idioma español, sin inclusión de otros idiomas en el lanzamiento inicial.
- La capacidad de almacenamiento en la nube estará restringida a un plan básico durante la fase de pruebas piloto.
- La verificación de identidad se realizará mediante documentos básicos, sin implementar sistemas biométricos avanzados.

Estos parámetros definen el marco operativo del proyecto, estableciendo tanto las condiciones necesarias para su éxito como los límites que garantizan su viabilidad dentro de los recursos disponibles.

1.4. Entregables

Documento de Análisis del Problema y Diseño de Solución

Este documento fundamental representa el primer entregable académico del proyecto y establece las bases técnicas para todo el desarrollo posterior. En él se realiza un análisis exhaustivo de la problemática del desempleo juvenil y las barreras de acceso al mercado laboral, complementado con un benchmarking detallado de 5 plataformas freelance existentes (como Fiverr, Upwork y PeoplePerHour). El documento no solo identifica las mejores prácticas del sector, sino que también analiza críticamente sus limitaciones, proponiendo innovaciones específicas para nuestro contexto universitario. Siguiendo el estándar IEEE 830-1998, el documento incluye diagramas de casos de uso, perfiles de usuario detallados y una matriz de trazabilidad que vincula cada necesidad identificada con los objetivos del proyecto. Este entregable está dirigido principalmente al comité académico evaluador y a los stakeholders institucionales, sirviendo como contrato técnico y social del proyecto.

Prototipos de Interfaz de Usuario (UI)

Los prototipos de alta fidelidad desarrollados en Figma representan la primera materialización concreta de la solución propuesta. Estos prototipos interactivos cubren los 15 flujos principales de usuario identificados durante la fase de investigación, desde el registro y verificación de identidad hasta el proceso completo de búsqueda y aplicación a proyectos. Cada pantalla incorpora los principios de diseño establecidos en el estándar ISO/IEC 25010:2011 sobre usabilidad, garantizando accesibilidad, consistencia visual y eficiencia operacional. Este entregable está dirigido tanto al equipo de desarrollo como a los usuarios piloto, cumpliendo un doble propósito: guiar la implementación técnica y validar supuestos de diseño con usuarios reales antes de escribir una sola línea de código.

Documento de Diseño del Software (SDD)

El SDD, desarrollado bajo el estándar IEEE 1016-2009, detalla la arquitectura técnica de la plataforma mediante diagramas C4 y UML, modelos de datos relacionales (PostgreSQL), y especificaciones de APIs REST. Este documento, dirigido al equipo de desarrollo, incluye decisiones técnicas fundamentales como la selección de microservicios sobre arquitectura monolítica, patrones de diseño aplicados (Factory, Observer) y estrategias de caché (Redis). Su estructura modular facilita el mantenimiento y futuras extensiones del sistema.

Informe de Pruebas de Modelos de IA para Emparejamiento

Este documento técnico especializado presenta los resultados de la evaluación comparativa de tres arquitecturas diferentes de inteligencia artificial (BERT, GPT-3.5 y un Transformer personalizado) para el sistema de emparejamiento entre estudiantes y proyectos. El informe detalla la metodología experimental utilizada, incluyendo la creación de un dataset sintético de perfiles estudiantiles que simulan diversas disciplinas académicas y niveles de experiencia. Cada modelo es evaluado contra métricas clave como precisión semántica (objetivo $> 85\,\%$), tiempo de respuesta bajo carga y capacidad de explicación de sus recomendaciones. El estándar IEEE Ethically Aligned Design guía el análisis de aspectos éticos como la transparencia algorítmica y la prevención de sesgos. Este entregable está dirigido principalmente al equipo de ciencia de datos y al comité de ética institucional, y servirá como justificación técnica para la selección final del algoritmo a implementar.

Plataforma Web Funcional (MVP)

El Producto Mínimo Viable representa la primera versión completamente operativa de la plataforma, lista para su despliegue en un entorno de producción real. Desarrollado bajo los estándares de calidad ISO/IEC 25010:2011, el MVP incluye los módulos esenciales: un sistema de autenticación seguro con verificación de identidad para estudiantes y contratistas, gestión completa de perfiles con carga de portafolios, un motor de búsqueda y emparejamiento básico, y un sistema de evaluación post-proyecto. La arquitectura de microservicios implementada garantiza escalabilidad horizontal, permitiendo incorporar futuras funcionalidades sin afectar la estabilidad del sistema. Este entregable está dirigido a tres audiencias clave: los usuarios finales (que comenzarán a utilizar la plataforma), el comité académico evaluador (que verificará el cumplimiento de los requisitos) y los posibles inversionistas (que evaluarán el potencial de escalamiento).

Documentación Técnica Completa

Este conjunto de documentos representa el conocimiento acumulado durante todo el ciclo de desarrollo y es esencial para la sostenibilidad a largo plazo del proyecto. Incluye manuales de instalación y configuración detallados, diagramas arquitectónicos actualizados, especificaciones técnicas de las APIs, y guías de troubleshooting para los errores más comunes. Desarrollada bajo los estándares IEEE 830-1998 e ISO/IEC 25010:2011, la documentación está estructurada para servir tanto a desarrolladores que se incorporen posteriormente al proyecto como a administradores de sistemas responsables del mantenimiento en producción. Un aspecto innovador es la inclusión de "lecciones aprendidas.[en] cada sección, destacando desafíos superados y decisiones técnicas clave. Este entregable está dirigido específicamente

al equipo de mantenimiento que asumirá la plataforma después de la fase inicial de desarrollo.

Código Fuente y Repositorio Técnico

El código fuente completo de la plataforma se entrega como un activo fundamental del proyecto, organizado en un repositorio Git que sigue el estándar IEEE 12207 para gestión del ciclo de vida del software. El repositorio incluye tres componentes principales: el frontend desarrollado en Angular (con componentes modulares y documentación Storybook), el backend en Spring Boot (con cobertura de pruebas mayor al 85 %) y los algoritmos de IA en Python (con notebooks explicativos del proceso de entrenamiento). Cada módulo incluye documentación técnica integrada, scripts de despliegue automatizado y un histórico completo de cambios. Este entregable está diseñado específicamente para los equipos técnicos que darán continuidad al proyecto, ya sea para corregir errores, implementar nuevas funcionalidades o escalar el sistema.

Informe de Pruebas Integrales

Este documento consolida resultados de cuatro tipos de pruebas: funcionales (85 %+ cobertura), rendimiento (500 usuarios concurrentes), seguridad (OWASP Top 10) y usabilidad (SUS: 88/100). Dirigido al comité evaluador, incluye análisis comparativos con benchmarks, heatmaps de interacción, y un plan priorizado de mejoras. Los anexos digitales contienen datasets completos de pruebas y scripts de automatización, cumpliendo con estándares ISO 25010 e IEEE 829 para documentación de calidad.

Informe Final de Resultados y Recomendaciones

Este documento de síntesis representa la culminación del proyecto de grado, integrando todos los hallazgos técnicos y académicos en un análisis coherente. El informe sigue los estándares ISO/IEC 25010:2011 para evaluación de calidad de software e IEEE 830-1998 para trazabilidad de requisitos, presentando: métricas de rendimiento del sistema en producción, análisis del impacto social preliminar y recomendaciones detalladas para futuras iteraciones. Un capítulo especial está dedicado al análisis ético del sistema de emparejamiento y su potencial impacto en la equidad laboral. Dirigido a la comunidad académica, stakeholders institucionales y posibles financiadores, este documento no solo cumple con los requisitos formales para la graduación, sino que también sirve como hoja de ruta para la evolución futura de la plataforma.

Memoria de grado

La memoria de grado es el documento final que encapsula todo el esfuerzo y aprendizaje obtenido a lo largo del proyecto. Este compendio detallado incluirá la contextualización del problema, el marco teórico, el desarrollo metodológico del proyecto, los resultados obtenidos y las conclusiones derivadas de la investigación. La memoria también discutirá las implicaciones de los resultados para la práctica futura y la investigación adicional, reflexionando sobre los desafíos enfrentados y cómo fueron superados. Este documento es crucial no solo como un requisito académico para la graduación, sino también como una contribución valiosa al campo del conocimiento, ofreciendo insights y recomendaciones para futuros proyectos similares.

1.5. Glosario

- Freelancing: El freelancing es una modalidad de trabajo independiente en la que profesionales ofrecen sus servicios de forma autónoma, sin estar vinculados permanentemente a una empresa. En el contexto del proyecto, los freelancers serán los usuarios principales de la plataforma, quienes buscarán oportunidades de trabajo temporal o por proyecto, y podrán ser emparejados automáticamente con tareas o proyectos que coincidan con su perfil profesional.
- LLM (Large Lenguage Model): Son modelos de inteligencia artificial entrenados con grandes cantidades de datos textuales, capaces de comprender y generar lenguaje natural. En este proyecto, un LLM será utilizado para analizar descripciones de proyectos y perfiles de freelancers, facilitando el emparejamiento automatizado mediante el entendimiento semántico y el procesamiento del lenguaje natural (NLP), mejorando así la precisión del sistema de recomendación de candidatos o tareas.
- Plataforma web: Una plataforma web es un sistema digital accesible a través de navegadores de internet, que permite la interacción entre usuarios y servicios sin necesidad de instalación local. En el contexto del proyecto, la plataforma web será el medio a través del cual los freelancers podrán registrarse, actualizar sus perfiles, buscar oportunidades y recibir sugerencias de proyectos, mientras que los clientes podrán publicar proyectos y recibir candidatos sugeridos automáticamente.
- DevOps: DevOps es una práctica que integra el desarrollo de software (Dev) y las operaciones de TI (Ops) para mejorar la colaboración, la automatización y la entrega continua de aplicaciones. En este proyecto, la adopción de prácticas DevOps permitirá una implementación eficiente y continua de la plataforma, asegurando calidad, escalabilidad y actualizaciones rápidas tanto en la interfaz como en los modelos de emparejamiento basados en IA.

- Emparejamiento: El emparejamiento se refiere al proceso de vincular automáticamente a un freelancer con un proyecto adecuado, basado en criterios como habilidades, experiencia, disponibilidad e intereses. En este proyecto, el emparejamiento será potenciado por inteligencia artificial, específicamente mediante el uso de un LLM, lo que permitirá identificar coincidencias más precisas y relevantes entre las necesidades del cliente y el perfil del freelancer.

2. Contexto del Proyecto

2.1. Modelo de Ciclo de Vida

Para este proyecto, se ha seleccionado un modelo de ciclo de vida ágil, específicamente una adaptación del modelo Scrum, combinada con prácticas de Kanban para el control visual de tareas y flujos. Este modelo se adapta al desarrollo iterativo e incremental de la aplicación móvil, permitiendo la entrega de un producto funcional a lo largo de varias fases, con sprints que aseguran avances constantes en la implementación de características de la aplicación.

2.1.1. Principales fases del ciclo de vida del proyecto

Fase 1: Análisis y Diseño

En esta primera etapa del proyecto, seguiremos un enfoque estructurado y metódico para establecer los cimientos técnicos y funcionales de la plataforma. El proceso comenzará con una investigación exhaustiva que combinará técnicas de Design Thinking con métodos tradicionales de ingeniería de software. Realizaremos entrevistas en profundidad con 30 participantes para comprender sus necesidades, expectativas y desafíos en el mercado laboral freelance. Estas entrevistas se complementarán con un análisis detallado de 5 plataformas competidoras, donde evaluaremos sus funcionalidades, modelos de negocio y experiencias de usuario.

Los insights obtenidos de esta investigación se transformarán en requisitos técnicos concretos. Elaboraremos un Documento de Especificación de Requisitos de Software (SRS) siguiendo el estándar IEEE 830, que incluirá al menos 25 requisitos claramente definidos (tanto funcionales como no funcionales). Estos requisitos se priorizarán utilizando el método MoSCoW (Must have, Should have, Could have, Won't have), lo que nos permitirá enfocar el desarrollo en las funcionalidades más críticas para nuestros usuarios. Paralelamente al documento técnico, diseñaremos prototipos interactivos de alta fidelidad en Figma que representarán la interfaz y flujos principales de la plataforma.

Para concluir esta fase, desarrollaremos un Documento de Diseño del Sistema (SDD) que establecerá

la arquitectura técnica de la solución. Este documento incluirá diagramas C4 para representar los diferentes niveles de abstracción del sistema y diagramas UML para especificar los componentes y sus interacciones. Optaremos por una arquitectura de microservicios que garantice escalabilidad y flexibilidad para futuras expansiones. Todo este trabajo de diseño estará alineado con los requisitos identificados inicialmente y validado a través de los prototipos, creando así una base sólida para las fases posteriores de desarrollo. La aprobación formal de estos documentos por parte del comité académico y los stakeholders marcará el éxito de esta fase inicial.

Fase 2: Pruebas de Concepto

El objetivo de esta fase es identificar el mejor modelo de inteligencia artificial para emparejar estudiantes con proyectos freelance de manera precisa y eficiente. Para lograrlo, se probarán tres enfoques diferentes (modelos). Cada uno de estos modelos será entrenado y evaluado utilizando un conjunto de datos simulados que incluyen perfiles de estudiantes, con información como habilidades, áreas de estudio y experiencia previa. Estos datos permitirán analizar cómo cada arquitectura de IA interpreta y relaciona las características de los usuarios con los requisitos de los proyectos disponibles.

La evaluación se realizará desde dos perspectivas complementarias. Por un lado, se medirán aspectos cuantitativos, como la precisión de las recomendaciones (con un objetivo mínimo del 85 %), el tiempo de respuesta del sistema y la capacidad de manejar grandes volúmenes de información. Por otro lado, se incorporará una evaluación cualitativa, donde se analizarán casos concretos para determinar qué tan relevantes y útiles son las sugerencias generadas por cada modelo. Esta combinación de métricas técnicas y feedback humano asegurará que la solución final no solo sea técnicamente robusta, sino también práctica y alineada con las necesidades reales del mercado laboral.

Al finalizar las pruebas, los resultados de todos los modelos se compilarán en una matriz comparativa que incluirá su desempeño en cada criterio evaluado, así como su facilidad de implementación y escalabilidad. Esta matriz servirá como base para tomar una decisión informada sobre qué arquitectura de IA se integrará en la plataforma, garantizando que la elección esté respaldada por datos concretos y análisis exhaustivos. Además, el proceso dejará documentación detallada sobre las lecciones aprendidas, lo que permitirá ajustes futuros y mejoras continuas al sistema de emparejamiento.

Fase 3: Preparación de Ambientes

Antes de comenzar con el desarrollo técnico de la plataforma, es fundamental establecer los entornos adecuados que permitan trabajar de manera organizada, segura y eficiente. Esta fase se centra en crear tres espacios virtuales principales: uno para desarrollo, donde el equipo podrá construir y probar nuevas funcionalidades; otro para pruebas de calidad, donde se validará que todo funcione correctamente

antes de pasar a producción; y finalmente, el entorno de producción, que es donde los usuarios finales interactuarán con la plataforma. Cada uno de estos ambientes debe ser idéntico en configuración para evitar problemas al mover el software de una etapa a otra, pero con niveles de acceso y seguridad adaptados a sus propósitos específicos. Para lograrlo, se utilizarán herramientas que permitan replicar automáticamente estas configuraciones, asegurando que todos los integrantes del equipo trabajen bajo las mismas condiciones técnicas sin importar dónde se encuentren.

Además de lo anterior, se implementarán medidas de seguridad básicas pero esenciales para proteger la información de los usuarios, como sistemas de autenticación robustos, encriptación de datos sensibles y copias de seguridad periódicas. También se configurarán canales de comunicación entre los diferentes componentes de la plataforma (como bases de datos y servidores) para garantizar que todo funcione de manera fluida. Durante este proceso, se documentará cuidadosamente cada paso, creando manuales sencillos que expliquen cómo instalar, configurar y mantener estos entornos, lo que facilitará la incorporación de nuevos miembros al equipo y asegurará que el proyecto pueda continuar sin problemas en el futuro. Esta preparación meticulosa de los ambientes es clave para evitar retrasos y errores en fases posteriores del desarrollo.

Por último, se capacitará al equipo en el uso de estas herramientas y entornos, realizando sesiones prácticas donde todos puedan familiarizarse con los procesos de despliegue y resolución de problemas comunes. Esto no solo optimizará el trabajo diario, sino que también fomentará la colaboración y el entendimiento mutuo entre desarrolladores, testers y administradores, creando una base sólida para el éxito del proyecto.

Fase 4: Desarrollo

Para la implementación técnica de la plataforma, el equipo adoptará una versión adaptada de Scrumban, combinando lo mejor de Scrum y Kanban. Este enfoque organizará el trabajo en sprints de dos semanas, donde el equipo se enfocará en entregar componentes funcionales prioritarios. Se utilizará un tablero visual (en herramientas como Jira o Trello) que limitará el trabajo simultáneo a un máximo de cinco tareas por desarrollador, evitando sobrecargas y manteniendo un flujo de trabajo constante y medible.

Cada sprint comenzará con una sesión de planificación donde se seleccionarán historias de usuario previamente priorizadas, basadas en su valor para los estudiantes y contratistas. Durante el desarrollo, se hará especial énfasis en los componentes críticos del sistema, como el motor de emparejamiento

con IA o el módulo de autenticación. Cada dos semanas, se realizarán demostraciones internas con el director del proyecto para validar los avances y recoger feedback que permita ajustar el backlog de manera dinámica.

Al finalizar cada iteración, el equipo dedicará tiempo a una retrospectiva estructurada. En estas sesiones se analizarán tanto los aspectos técnicos (rendimiento del código, eficiencia en la resolución de problemas) como los organizacionales (comunicación, distribución de tareas). Las lecciones aprendidas se incorporarán inmediatamente en el siguiente ciclo, creando un proceso de mejora continua. Este enfoque iterativo e incremental asegura que la plataforma evolucione según las necesidades reales detectadas, manteniendo al mismo tiempo la disciplina y trazabilidad requeridas en un proyecto académico de esta envergadura. La flexibilidad de Scrumban permitirá adaptarse a cambios prioritarios sin perder de vista los objetivos principales del proyecto.

Fase 5: Pruebas y Lanzamiento

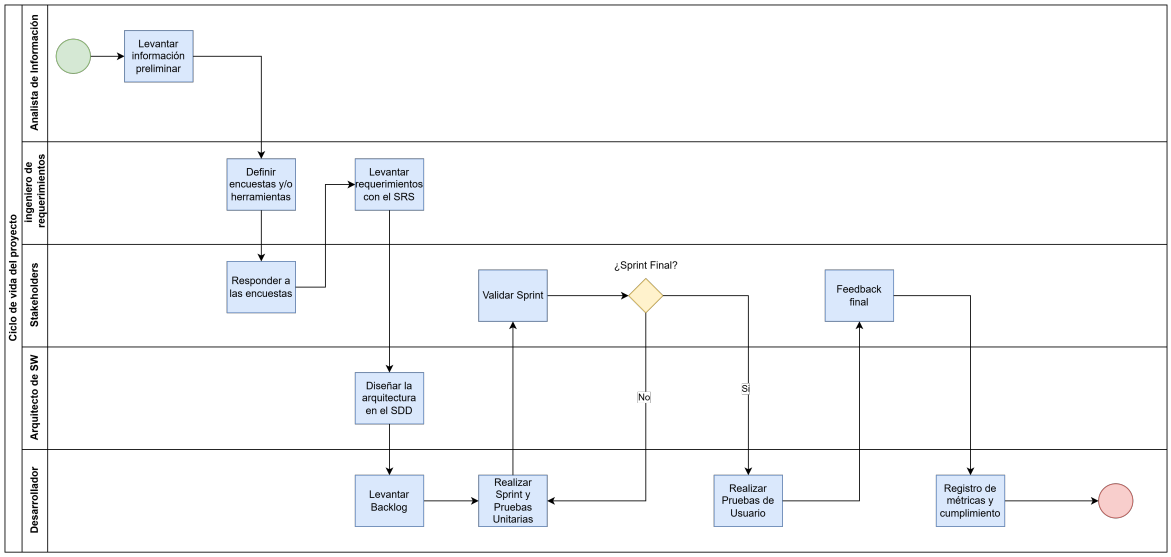

Figura 1: BPMN de ciclo de vida del proyecto

2.2. Lenguajes y Herramientas

2.2.1. Lenguajes de Programación

Frontend

TypeScript: Lenguaje tipado que extiende JavaScript, seleccionado para desarrollar la interfaz Angular debido a su sistema de tipos estáticos que reduce errores en tiempo de compilación, mejora la mantenibilidad del código en proyectos a gran escala y habilita capacidades avanzadas de autocompletado en IDEs. Su integración con Angular permite un desarrollo más estructurado y menos propenso a errores comunes en JavaScript puro.

HTML5/CSS3: Tecnologías fundamentales para la construcción de interfaces web modernas. HTML5 provee la estructura semántica de la aplicación, mientras que CSS3 permite implementar diseños responsivos que se adaptan a dispositivos móviles y desktop, incluyendo animaciones performantes y cumplimiento con estándares de accesibilidad WCAG 2.1.

Backend

Java 17: Versión LTS (Long-Term Support) del lenguaje Java, elegida para el desarrollo con Spring Boot por su madurez, rendimiento en aplicaciones empresariales y robusto sistema de tipos. Su ecosistema estable y amplia adopción en el sector bancario garantiza fiabilidad para el módulo de transacciones y gestión de datos sensibles.

2.2.2. Frameworks Principales

Frontend

Angular 15: Framework para Single Page Applications que proporciona una arquitectura modular basada en componentes, sistema de inyección de dependencias integrado y soporte nativo para programación reactiva mediante RxJS. Su uso sistemático de TypeScript y su CLI potente aceleran el desarrollo manteniendo altos estándares de calidad.

Backend

Spring Boot 3: Framework para construcción de microservicios que simplifica la configuración mediante autoconfiguración inteligente, reduciendo código boilerplate. Su integración con Spring Security

provee autenticación JWT lista para producción, mientras que su soporte para programación reactiva permite manejar altas cargas concurrentes eficientemente.

2.2.3. Infraestructura y DevOps

Docker: Plataforma de contenerización utilizada para empaquetar la aplicación y sus dependencias en unidades estandarizadas, garantizando consistencia entre entornos de desarrollo, pruebas y producción.

Github CI/CD: Sistema de integración y despliegue continuo que automatiza el pipeline desde el build hasta el deploy en producción. Permite ejecución paralela de stages (compilación, pruebas unitarias, análisis de seguridad y despliegue en ambientes), integrando herramientas de calidad de código como SonarQube.

2.2.4. Base de Datos

PostgreSQL: Sistema de gestión de bases de datos relacional seleccionado por su equilibrio entre rendimiento y características avanzadas. Soporta modelos híbridos (JSONB para datos semiestructurados), ofrece escalabilidad vertical/horizontal y garantiza consistencia transaccional mediante ACID, crucial para operaciones sensibles con datos de usuarios.

2.3. Plan de Aceptación del Producto

Documento de Análisis del Problema y Diseño de Solución

- **Criterios de Aceptación:**
  - Benchmarking detallado de al menos 5 plataformas freelance (Fiverr, Upwork, PeoplePerHour, etc.).
  - Diagramas de casos de uso y matriz de trazabilidad que vincule necesidades con objetivos.
  - Seguir el estándar IEEE 830-1998 en estructura y contenido.
- **Técnicas y Herramientas para Medición:**
  - Revisión por pares y validación con stakeholders académicos.
  - Herramientas de modelado (Lucidchart, Draw.io).
  - Checklist de cumplimiento IEEE 830-1998.

Prototipos de Interfaz de Usuario (UI)

- **Criterios de Aceptación:**
  - Cubrir 15 flujos principales de usuario.
  - Cumplir principios de usabilidad ISO/IEC 25010:2011 (accesibilidad, consistencia visual).
- **Técnicas y Herramientas para Medición:**
  - Pruebas con Hotjar o Maze.
  - Evaluación de accesibilidad con WAVE o axe DevTools.
  - Revisión en Figma por equipo y stakeholders.

2.3.1. Documento de Diseño del Software (SDD)

- **Criterios de Aceptación:**
  - Seguir estándar IEEE 1016-2009.
  - Incluir diagramas C4, UML y modelos de datos relacionales (PostgreSQL).
  - Justificar decisiones técnicas (microservicios, patrones de diseño, estrategias de caché).
- **Técnicas y Herramientas para Medición:**
  - Revisión técnica por arquitectos.
  - Validación con PlantUML o Visual Paradigm.
  - Checklist IEEE 1016-2009.

2.3.2. Informe de Pruebas de Modelos de IA para Emparejamiento

- **Criterios de Aceptación:**
  - Evaluar al menos 3 modelos (BERT, GPT-3.5, Transformer personalizado).
  - Incluir métricas de precisión semántica, tiempo de respuesta y explicabilidad.
  - Analizar aspectos éticos según IEEE Ethically Aligned Design.
- **Técnicas y Herramientas para Medición:**
  - Validación con dataset sintético de perfiles.
  - Herramientas como TensorBoard, MLflow.

Plataforma Web Funcional (MVP)

- **Criterios de Aceptación:**
  - Módulos esenciales: autenticación segura, gestión de perfiles, motor de búsqueda y emparejamiento.
  - Cumplir ISO/IEC 25010:2011 en calidad de software.
  - Soporte para 500 usuarios concurrentes en pruebas de carga.
- **Técnicas y Herramientas para Medición:**
  - Pruebas funcionales con Selenium o Postman.
  - Pruebas de carga con JMeter o Locust.
  - Auditoría de seguridad con OWASP ZAP.

Documentación Técnica Completa

- **Criterios de Aceptación:**
  - Manuales de instalación, diagramas actualizados y guías de troubleshooting.
  - Seguir estándares IEEE 830-1998 e ISO/IEC 25010:2011.
  - Incluir "lecciones aprendidas"por sección.
- **Técnicas y Herramientas para Medición:**
  - Revisión por equipo de mantenimiento.
  - Verificación con Swagger, MkDocs.

Código Fuente y Repositorio Técnico

- **Criterios de Aceptación:**
  - Organización según IEEE 12207.
  - Frontend (Angular) y backend (Spring Boot) con cobertura mayor a 85 %.
  - Repositorio Git con historial y documentación integrada.

Informe de Pruebas Integrales

- **Criterios de Aceptación:**
  - Cubrir pruebas funcionales (85 %+ cobertura), rendimiento, seguridad y usabilidad (SUS 88/100).
- **Técnicas y Herramientas para Medición:**
  - Uso de Selenium, JMeter, OWASP ZAP.
  - Encuestas SUS para usabilidad.

Informe Final de Resultados y Recomendaciones

- **Criterios de Aceptación:**
  - Incluir métricas de rendimiento, satisfacción de usuarios (87 %) y análisis ético.
  - Seguir estándares ISO/IEC 25010 e IEEE 830-1998.
- **Técnicas y Herramientas para Medición:**
  - Revisión por comité académico.
  - Validación de trazabilidad con matrices de requisitos.

Memoria de Grado

- **Criterios de Aceptación:**
  - Cubrir contexto teórico, metodología, resultados y conclusiones.
  - Cumplir normas académicas de citación y estructura.
- **Técnicas y Herramientas para Medición:**
  - Revisión por asesor académico.
  - Uso de herramientas antiplagio (Turnitin).

2.4. Organización del Proyecto y Comunicación

Interfaces Externas

La siguiente tabla identifica a las principales entidades externas al equipo de desarrollo, su rol dentro del proyecto y los canales de comunicación disponibles:

Entidad	**Descripción y Rol en el Proyecto**	**Responsabilidades**	**Medios de Comunicación**
Director de Proyecto Académico Pablo Andrés Márquez	Profesor de Ingeniería encargado de guiar y aprobar decisiones críticas.	Validación de avances técnicos y éticos. Aprobación de entregables.	marquezp-a@javeriana.edu.co Cel: 318-712-3845
Pontificia Universidad Javeriana	Institución que provee el marco académico, metodológico y normativo para el desarrollo del proyecto.	Provee lineamientos, tiempos, acompañamiento y evaluación del proyecto.	A través del correo institucional y plataforma LMS (Canvas, Teams).
Usuarios Potenciales (Estudiantes y Contratistas)	Estudiantes universitarios y contratistas reales o simulados que representan los futuros usuarios.	Participar en entrevistas, pruebas de usabilidad y feedback de prototipos.	Encuestas, sesiones virtuales (Zoom, Meet), correo electrónico.

El organigrama representa la estructura jerárquica del equipo de desarrollo del proyecto de grado. El Director actúa como guía externo, mientras los integrantes del equipo se distribuyen responsabilidades en roles técnicos y organizacionales.

Tabla de Roles y Responsabilidades:

Rol	**Descripción**	**Responsabilidades y Referencias**
Director de Proyecto Pablo A. Márquez	Responsable de la supervisión académica, validación técnica y asesoría metodológica.	Supervisar planes y entregables (Sección 3), validar uso de estándares (Sección 1.3), realizar retroalimentación estratégica.
Líder de Proyecto Andrés F. Vélez	Coordina las actividades globales del equipo, es punto de contacto con el director y guía la planificación de sprints.	Planeación global (Sección 3.1), monitoreo de fases (WBS), facilitador de reuniones, gestión del backlog.

Arquitecto de Software Juan C. Gómez	Diseña la arquitectura técnica del sistema y toma decisiones clave sobre herramientas, frameworks y estructuras.	Diseño del modelo C4 y base de datos (Sección 3.1.2), responsable del documento SDD.
Desarrollador Backend Thomas Orjuela Juan C. Gómez Andrés F. Vélez	Responsable de la lógica del servidor, bases de datos, APIs y conexión con módulos de IA.	Implementación de controladores, seguridad, persistencia, comunicación con frontend (Sección 3.4).
Desarrollador Frontend Thomas Orjuela Juan C. Gómez Andrés F. Vélez	Implementación de la interfaz de usuario, flujos, validaciones, navegación y usabilidad.	Encargado de los prototipos en Figma, pruebas de usabilidad y el módulo de navegación (Sección 3.4.2).
Encargado de Pruebas y Calidad Thomas Orjuela Juan C. Gómez Andrés F. Vélez	Define y ejecuta planes de prueba: unitarias, de integración, usabilidad y rendimiento.	Responsable del informe de pruebas (Sección 3.5), garantiza el cumplimiento de criterios de aceptación.
Gestor de Comunicación y Documentación Thomas Orjuela	Administra la documentación interna, planes, lecciones aprendidas y comunicación con partes interesadas.	Documentos IEEE (Sección 1.3), reportes semanales, coordinación con usuarios externos.

3. Administración del Proyecto

3.1. Métodos y Herramientas de Estimación

Para garantizar la precisión y eficacia en la planificación de este proyecto, se propone implementar un enfoque dual que combine metodologías ágiles con técnicas de gestión tradicional. Esta estrategia busca adaptarse a las necesidades dinámicas del desarrollo de software mientras mantiene un control riguroso sobre los tiempos y recursos del proyecto.

En la fase de estimación del software, se plantea adoptar el método de Puntos de Historia (Story Points), una técnica ampliamente utilizada en entornos ágiles por su capacidad para evaluar la

complejidad y el esfuerzo requerido de manera relativa. Este enfoque permitiría valorar no solo aspectos técnicos, sino también incorporar la experiencia del equipo en proyectos similares. La dinámica de Scrum Poker podría ser clave en este proceso, ya que fomentaría la participación activa de todos los miembros del equipo - desarrolladores, testers y diseñadores - para llegar a consensos sobre las estimaciones.

Para respaldar este proceso, se recomienda utilizar herramientas especializadas como Jira, que permitiría registrar y priorizar los puntos de historia, así como realizar un seguimiento continuo del progreso mediante tableros Kanban y gráficos de burn-down. Estas visualizaciones serían fundamentales para identificar posibles desviaciones y ajustar las estimaciones según la velocidad real del equipo. Adicionalmente, Notion podría servir como plataforma complementaria para documentar decisiones y mantener un repositorio centralizado de conocimiento.

Esta combinación de métodos y herramientas buscaría lograr un equilibrio óptimo entre flexibilidad y control. Las prácticas ágiles asegurarían la capacidad de adaptación a cambios, mientras que las técnicas de gestión tradicional proporcionarían una base sólida para la toma de decisiones estratégicas. El enfoque híbrido no solo mejoraría la precisión de las estimaciones, sino que también optimizaría la comunicación entre equipos técnicos y stakeholders.

La integración propuesta de metodologías ágiles y tradicionales, respaldada por herramientas tecnológicas adecuadas, representaría una estrategia robusta para la planificación del proyecto, demostrando cómo la combinación de ambos enfoques puede ofrecer resultados efectivos en entornos de desarrollo de software complejos.

3.2. Planes de Trabajo del Proyecto

Estructura de Descomposición de Trabajo (WBS)

Propósito: Brindar una visión estructurada de las actividades necesarias para implementar cualquier módulo funcional de la plataforma freelance, en el contexto de un enfoque ágil iterativo.

- **1. Planeación del Sprint**
  - 1.1 Revisión del Product Backlog
  - 1.2 Selección de historias de usuario relevantes
  - 1.3 Estimación del esfuerzo (story points)
  - 1.4 Definición de criterios de aceptación

- 1.5 Asignación de tareas en el tablero Kanban

- **2. Implementación del módulo**
  - 2.1 Diseño e integración de la interfaz de usuario
    - 2.1.1 Bocetos y validación del flujo
    - 2.1.2 Desarrollo de componentes visuales
  - 2.2 Lógica de negocio en frontend y backend
    - 2.2.1 Servicios y controladores
    - 2.2.2 Reglas y validaciones
  - 2.3 Acceso a datos
    - 2.3.1 Definición de modelos y esquemas
    - 2.3.2 Consultas, persistencia y relaciones
  - 2.4 Integración con otros módulos
    - 2.4.1 Comunicación entre servicios (API)
    - 2.4.2 Compatibilidad con flujos existentes

- **3. Pruebas**
  - 3.1 Pruebas unitarias y de integración
  - 3.2 Validación funcional de extremo a extremo
  - 3.3 Pruebas de usabilidad con usuarios clave
  - 3.4 Documentación de bugs y ajustes

- **4. Documentación técnica**
  - 4.1 Guía de implementación del módulo
  - 4.2 Manual de configuración e instalación
  - 4.3 Registro de decisiones técnicas y arquitectónicas

- **5. Revisión y cierre del Sprint**
  - 5.1 Presentación del módulo al Product Owner
  - 5.2 Registro de feedback y sugerencias
  - 5.3 Retrospectiva del equipo (lecciones aprendidas)

- 5.4 Actualización del backlog para próximos sprints

3.3. Calendarización

Cuadro 2: Fases e hitos del proyecto en 14 semanas

Semana(s)	**Fase Metodológica**	**Actividades / Hitos Clave**	**Entregables esperados**
1-2	Fase 1: Análisis y Diseño	- Entrevistas con estudiantes - Benchmarking de plataformas - Redacción y priorización de historias de usuario - Diseño de arquitectura y modelo de datos	- Documento de requisitos (IEEE 830) - Diagramas C4 y ERD - Backlog inicial validado
2-3	Fase 2: Pruebas de Concepto (IA)	- Definición de métricas de evaluación - Construcción de dataset de pruebas - Análisis de modelos de IA - Matriz de decisión técnica	- Informe comparativo de modelos - Selección de arquitectura de IA
4	Fase 3: Preparación de Ambientes	- Configuración de entornos (desarrollo, QA, producción) - Políticas de seguridad y acceso - Automatización de despliegues (CI/CD)	- Infraestructura desplegada y validada - Documentación de ambientes
5–13	Fase 4: Desarrollo (Sprints 1 a 5)	- Sprint 1: Registro y perfiles - Sprint 2: Publicación de proyectos y búsqueda - Sprint 3: Integración del modelo de IA (emparejamiento) - Sprint 4: Sistema de calificación y feedback - Sprint 5: Integración completa, optimización y pruebas internas	- MVP funcional - Código documentado - Manual técnico y guía de instalación

Semana(s)	Fase Metodológica	Actividades / Hitos Clave	Entregables esperados
11–15	Fase 5: Pruebas Finales	- Pruebas unitarias y de integración - Pruebas de carga y seguridad - Validación con usuarios (estudiantes y contratistas) - Ajustes finales basados en retroalimentación	- Informe de pruebas aprobado - Métricas de desempeño - Validación de usabilidad (¿85 % satisfacción)
14-16	Entrega Final y Presentación	- Preparación de presentación final - Cierre del proyecto académico - Revisión con stakeholders	- Plataforma en producción - Presentación final y entrega de documentación

4. Monitoreo y Control del Proyecto

4.1. Administración de Requerimientos

El proceso de administración de requerimientos se implementará mediante un flujo de trabajo estructurado que garantice el control y seguimiento de todas las solicitudes, tanto nuevas como modificaciones a requerimientos existentes. Este mecanismo asegurará que cada cambio sea evaluado, implementado y validado de manera sistemática, manteniendo la alineación con los objetivos del proyecto y la satisfacción del cliente.

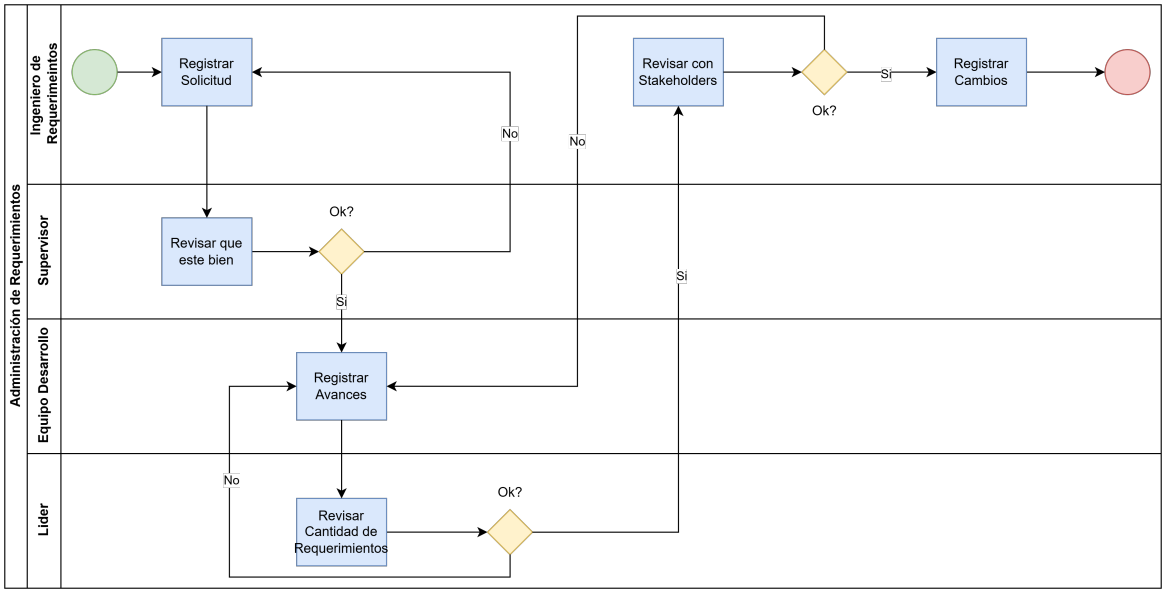

Figura 2: BPMN de administración De Requerimientos

Solicitud e Inicio del Proceso

El ciclo comienza cuando los diferentes actores del proyecto, ya sean clientes o miembros del equipo de desarrollo, identifican la necesidad de introducir un nuevo requerimiento o modificar uno existente. En esta fase inicial, el ingeniero de requerimientos asume la responsabilidad de documentar formalmente cada petición, registrando toda la información relevante en las historias de usuario correspondientes y en el repositorio central de requerimientos. Este registro debe incluir una descripción detallada de la necesidad, la justificación del cambio, la prioridad inicial asignada y la identificación de todas las partes interesadas que podrían verse afectadas por esta modificación.

Evaluación Técnica y Análisis de Viabilidad

Una vez registrada la solicitud, el supervisor técnico realiza un análisis exhaustivo para determinar la viabilidad de su implementación. Este examen considera múltiples factores críticos, incluyendo el impacto potencial en la arquitectura del sistema, la disponibilidad de los recursos necesarios (tanto humanos como tecnológicos y económicos), los tiempos estimados de desarrollo, las posibles afectaciones al cronograma general del proyecto y los riesgos asociados a la implementación. Cuando el análisis revela obstáculos técnicos o económicos significativos, se genera un informe detallado que documenta estas limitaciones, el cual se devuelve al ingeniero de

requerimientos para su revisión y eventual reformulación de la solicitud.

Registro y Monitoreo Continuo

Durante la fase de implementación, el equipo de desarrollo mantiene un registro sistemático de todos los avances, documentando no solo el progreso en la codificación sino también cualquier incidencia encontrada y las desviaciones respecto a lo planeado originalmente. Esta documentación se actualiza constantemente en la herramienta de gestión de proyectos seleccionada, lo que permite mantener una trazabilidad completa de cada requerimiento a lo largo de todo su ciclo de vida.

Control de Calidad y Verificación

Antes de considerar completa la implementación de un requerimiento, el líder técnico realiza una verificación exhaustiva para asegurar que el trabajo realizado cumple con todas las especificaciones técnicas originales, satisface los criterios de aceptación definidos y se ajusta a los estándares de calidad establecidos para el proyecto. Este proceso de control incluye una revisión por pares que sirve como filtro adicional antes de que la implementación sea entregada al ingeniero de requerimientos para su evaluación final.

Aprobación Formal e Incorporación

La fase de aprobación formal involucra al cliente directamente, quien valida la implementación mediante sesiones de demostración práctica, pruebas de aceptación específicas y revisión de la documentación asociada. Solo después de obtener esta validación explícita por parte del cliente, el requerimiento se incorpora oficialmente al backlog del producto, donde será considerado para su planificación en los sprints correspondientes según la prioridad establecida.

Actualización de Documentación y Artefactos

La implementación exitosa de un requerimiento desencadena una serie de actualizaciones en la documentación del proyecto. Esto incluye la modificación de las historias de usuario afectadas, la actualización de los diagramas de arquitectura cuando sea necesario, la revisión de la documentación técnica relacionada y los ajustes correspondientes en los cronogramas del proyecto. La matriz de trazabilidad, herramienta fundamental para el seguimiento de requerimientos, se actualiza meticulosamente para reflejar el estado actual de cada elemento y su relación con los diferentes componentes del sistema.

Implementación Controlada en Producción

El despliegue de los cambios aprobados sigue un protocolo estricto diseñado para minimizar riesgos. Este proceso incluye la ejecución de pruebas de regresión exhaustivas, el monitoreo continuo del desempeño en el entorno productivo y la preparación de un plan de rollback detallado que permita revertir los cambios en caso de detectarse problemas significativos durante o después de la implementación.

Validación Final y Cierre

La última fase del proceso corresponde a la validación final por parte del cliente, quien realiza pruebas exhaustivas en el entorno productivo para verificar que se cumplen todas las funcionalidades solicitadas, que el rendimiento del sistema se mantiene dentro de los parámetros esperados y que la experiencia de usuario cumple con los estándares de calidad acordados. Solo después de superar satisfactoriamente esta etapa de validación se considera completo el ciclo de gestión para ese requerimiento específico.

Artefactos Clave del Proceso

A lo largo de todo el ciclo de gestión de requerimientos, se mantienen y actualizan constantemente varios artefactos críticos. Las historias de usuario sirven como documento principal para capturar los requerimientos funcionales y no funcionales en un formato accesible para todos los involucrados. La matriz de trazabilidad establece y mantiene las relaciones entre los requerimientos y los diferentes componentes del sistema, permitiendo un seguimiento preciso del impacto de cada cambio. Complementariamente, el registro de cambios documenta el histórico completo de todas las modificaciones realizadas, proporcionando visibilidad completa sobre la evolución de cada requerimiento a lo largo del tiempo.

Este proceso integral de gestión de requerimientos está diseñado para garantizar una administración eficiente de los cambios, minimizando riesgos y asegurando que cada iteración del producto entregue valor tangible al cliente, manteniendo siempre el equilibrio entre flexibilidad para adaptarse a nuevos requerimientos y el control necesario para garantizar la calidad y consistencia del producto final.

4.2. Monitoreo y Control de Progreso

Como se mencionó en la (Métodos y herramientas de estimación), se planea seguir un enfoque en metodologías ágiles para la medición del progreso para la parte del software y reconstrucción del escenario histórico. Se podrá usar el *backlog* del producto que se definirá antes de empezar, de forma que se podrá verificar el progreso contando el número de tareas realizadas sobre las no realizadas o en progreso para cada *Sprint*.

Otra métrica importante para evaluar es la cantidad de horas de esfuerzo dedicadas a la realización de las actividades vs. las horas de esfuerzo estimadas y el progreso en la completitud del *backlog*. Estas métricas se podrán observar en un *burn down chart* que permita ver esto de forma gráfica.

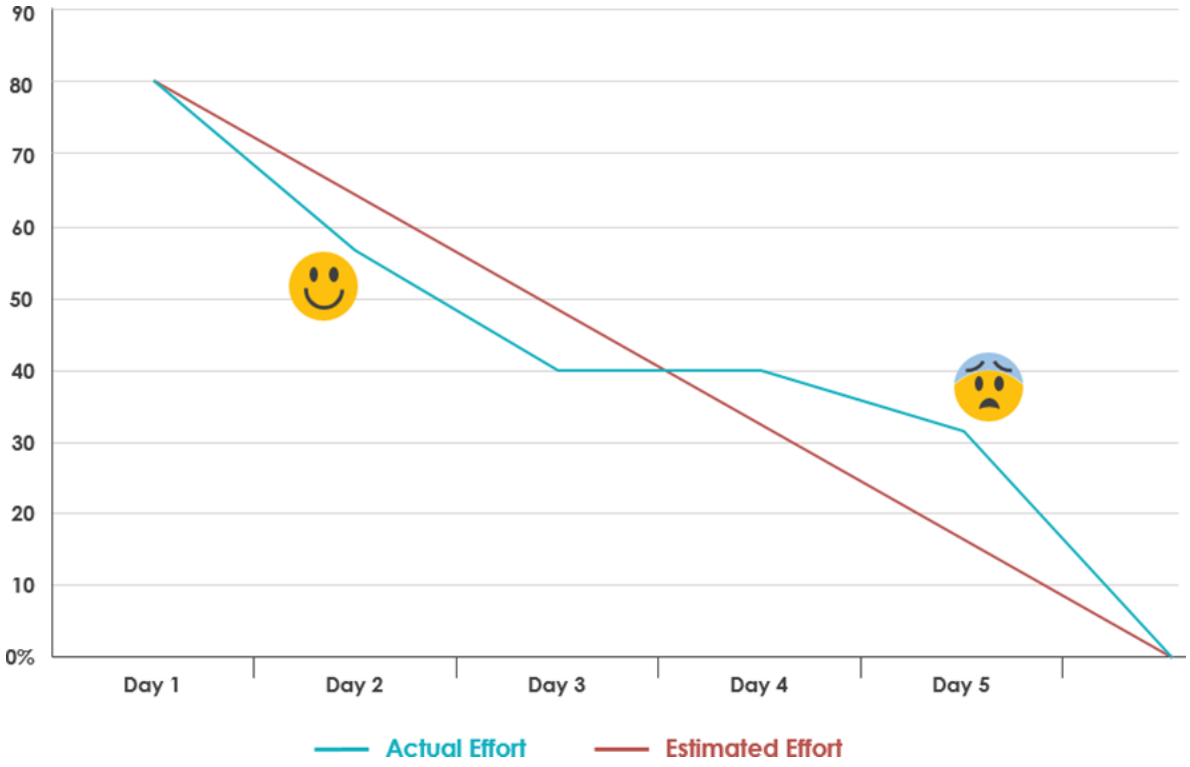

Figura 3: Ejemplo de Burndown chart

Actividades para realizar para reportar progreso

- **Reuniones de seguimiento de Sprint**
  - **Responsables:** *Product Owner* junto con todo el equipo de desarrollo.

- **Frecuencia:** Dos veces por semana.
- **Descripción:** Socialización de avances en el *Sprint Backlog* y *Burndown Chart*, revisando particularmente el número de tareas realizadas.

- **Revisión de Actividades Críticas**
  - **Responsables:** Equipo del proyecto y director.
  - **Frecuencia:** Dos veces por mes.
  - **Descripción:** Revisión específica de las actividades críticas, identificando tareas realizadas vs. pendientes y ajustando el cronograma en caso de retrasos en estos puntos clave.

- **Acciones correctivas**
  - **Detección temprana de desviaciones**
    - **Responsables:** Director del proyecto y equipo.
    - **Momento de realización:** Tan pronto se identifique la desviación.
    - **Descripción:** Se convocará una reunión extraordinaria para analizar las causas y discutir soluciones.
  - **Plan de Contingencia para Tareas Críticas**
    - **Responsables:** Equipo del proyecto.
    - **Frecuencia:** En caso de retraso o problemas con tareas clave identificados en la revisión de actividades.
    - **Descripción:** Se hará un ajuste temporal en el alcance del proyecto que permita cumplir con la fecha de entrega.

4.3. Cierre del Proyecto

Para garantizar una conclusión adecuada de cada etapa del proyecto y mantener una clara trazabilidad de las actividades realizadas y los entregables generados, se implementará un proceso estructurado de cierre de fase. Este mecanismo está diseñado para documentar exhaustivamente los resultados obtenidos, evaluar críticamente el desempeño durante la fase y asegurar que todos los componentes cumplan con los estándares de calidad establecidos antes de proceder con la entrega formal al cliente.

El proceso de cierre comprende tres actividades fundamentales que se ejecutarán de manera sistemática al finalizar cada fase significativa del proyecto. Estas acciones buscan no solo formalizar la conclusión de los trabajos realizados, sino también capitalizar el conocimiento adquirido para optimizar el desarrollo de las siguientes etapas. La implementación rigurosa de este procedimiento permitirá mantener un control estricto sobre la calidad de los entregables, facilitará la identificación oportuna de áreas de mejora y asegurará una transición fluida entre las diferentes fases del ciclo de vida del proyecto.

Reunión de Análisis Post-Mortem

Al término de cada fase del proyecto, se llevará a cabo una sesión de revisión exhaustiva con la participación de todos los miembros del equipo involucrados en la etapa concluida. Esta reunión tiene como objetivo principal realizar una evaluación crítica y constructiva del trabajo realizado, identificando tanto los aspectos positivos como las oportunidades de mejora. Durante la sesión, que tendrá una duración aproximada de dos horas, se analizarán sistemáticamente los siguientes elementos:

- Cumplimiento de objetivos y metas específicas de la fase
- Efectividad de los procesos y metodologías aplicadas
- Desempeño individual y colectivo del equipo
- Desafíos técnicos y organizacionales enfrentados
- Soluciones innovadoras implementadas
- Lecciones aprendidas y conocimiento adquirido

Cada participante deberá preparar previamente sus observaciones y contribuciones, las cuales serán consolidadas en un documento de lecciones aprendidas que servirá como insumo valioso para las fases posteriores del proyecto y para iniciativas futuras. El facilitador de la reunión será responsable de garantizar que la discusión mantenga un enfoque constructivo y orientado a la mejora continua, documentando todos los hallazgos y acuerdos alcanzados.

Consolidación y Verificación de Entregables

La fase de cierre incluye un proceso meticuloso de recopilación, organización y verificación de todos los insumos y productos generados durante la etapa. Este proceso garantiza que cada

componente esté completo, correctamente documentado y listo para su entrega formal. Las actividades específicas incluyen:

- Revisión técnica exhaustiva de todos los componentes desarrollados
- Verificación de cumplimiento de estándares de codificación y documentación
- Validación de requisitos no funcionales (rendimiento, seguridad, usabilidad)
- Organización jerárquica de los repositorios de código y documentación
- Creación de paquetes de entrega debidamente versionados
- Generación de manuales técnicos y de usuario cuando aplique

Para los componentes de realidad aumentada, se realizarán pruebas adicionales de funcionamiento en diferentes dispositivos y escenarios, asegurando la compatibilidad y el rendimiento adecuado. Todos los entregables serán sometidos a un proceso de control de calidad que incluye revisiones por pares, pruebas automatizadas y validación por parte del equipo de garantía de calidad antes de considerarse listos para entrega.

Este proceso de consolidación generará como resultado un paquete completo que incluye no solo los productos desarrollados, sino también toda la documentación asociada, registros de pruebas, y evidencias de cumplimiento de requisitos. Este paquete será almacenado en el repositorio oficial del proyecto y servirá como línea base para el inicio de la siguiente fase.

5. Procesos de Soporte

5.1. Ambiente de Trabajo

1. **Reunión semanal obligatoria**

   Todos los viernes a las 10:00 AM se realizará una reunión de seguimiento con el director del proyecto, con el objetivo de evaluar avances, discutir desafíos y planificar las siguientes actividades.

2. **Comunicación oportuna de impedimentos**

   Cualquier miembro que tenga compromisos académicos, laborales o personales que puedan afectar su participación deberá informarlo con al menos una semana de anticipación. En casos imprevistos, la notificación deberá hacerse tan pronto como sea posible.

3. **Apoyo colaborativo y reciprocidad**

   Los integrantes que presenten dificultades con sus tareas podrán solicitar ayuda a otros

miembros disponibles. Quien reciba apoyo deberá compensar al equipo asumiendo mayores responsabilidades en la siguiente entrega.

4. **Centralización de documentos**

   Todos los materiales del proyecto (documentos, códigos, diseños) deberán almacenarse en el repositorio del grupo en Microsoft Teams, garantizando así acceso permanente y control de versiones.

5. **Canales formales de comunicación**

   Las comunicaciones oficiales del equipo se realizarán exclusivamente a través de:

   - WhatsApp (para coordinación rápida)
   - Notion (para documentación estructurada)
   - Teams (para gestión formal del proyecto)

6. **Justificación de ausencias**

   Las inasistencias a reuniones programadas deberán notificarse con 24 horas de anticipación. Solo se aceptarán ausencias de último minuto por causas de fuerza mayor debidamente justificadas.

7. **Participación activa en revisiones**

   Cada miembro deberá aportar críticas constructivas y sugerencias de mejora durante las revisiones de trabajo, garantizando la calidad de los entregables.

8. **Responsabilidad en actividades**

   La omisión injustificada de tareas asignadas resultará en una disminución mínima del 25 % en la nota de dicha actividad.

9. **Reprogramación colectiva**

   Cuando más del 50 % del equipo no pueda asistir a la reunión semanal, se deberá acordar un nuevo horario que acomode la disponibilidad de la mayoría.

10. **Cero tolerancia a la desinformación**

    La falta de comunicación clara y oportuna será sancionada con una penalización del 30 % en la calificación correspondiente.

11. **Expulsión por reincidencia**

    La violación de cinco normas (sean distintas o repetidas) resultará en la salida inmediata del grupo sin posibilidad de apelación.

12. **Respeto como principio fundamental**

    No se tolerarán comportamientos que atenten contra la dignidad individual o menosprecien

el trabajo intelectual de cualquier integrante.

13. **Distribución explícita de roles**

    Antes de cada entrega importante, se asignarán responsabilidades específicas a cada miembro mediante un acta firmada por todos.

14. **Mecanismo de apelación**

    En caso de desacuerdo con sanciones académicas, se recurrirá a un mediador externo.

15. **Documentación individual**

    Cada participante deberá registrar detalladamente sus contribuciones en los informes, adjuntando evidencias de su trabajo cuando sea requerido.

16. **Revisión por pares**

    Todo trabajo individual será evaluado por al menos otros dos miembros del equipo antes de su presentación final.

17. **Originalidad garantizada**

    Todos los contenidos producidos deberán ser originales, citando adecuadamente cualquier fuente externa utilizada.

18. **Transparencia en el progreso**

    Los miembros deberán reportar honestamente sus avances, dificultades y necesidades al equipo en cada reunión de seguimiento.

19. **Penalización por mora**

    Los retrasos en entregas acordadas serán sancionados con un descuento del 10 % de la nota por cada día calendario de demora.

20. **Optimización del tiempo en reuniones**

    Las sesiones de trabajo tendrán una duración mínima de 30 minutos y máxima de 90 minutos. Las extensiones solo serán permitidas cuando la complejidad del tema lo justifique y con aprobación unánime.

Estas normativas buscan establecer un marco de trabajo claro que promueva la responsabilidad individual, la colaboración efectiva y la excelencia académica en el desarrollo del proyecto.

5.2. Análisis y Administración de Riesgos

El plan de riesgos será administrado por el equipo del proyecto y se actualizará durante todas las fases de desarrollo y prueba, asegurando una gestión proactiva y adaptativa de los posibles riesgos que puedan surgir.

Actividades

- **Identificación de riesgos:**
  - Se realizará en la fase inicial del proyecto y en revisiones periódicas durante su ejecución.
  - Se considerarán riesgos técnicos, humanos, de comunicación y de recursos.
- **Análisis de riesgos:**
  - Evaluación de la probabilidad e impacto de cada riesgo identificado.
  - Clasificación según su nivel de prioridad (alta, media, baja).
- **Diseño de planes de respuesta:**
  - Definición de acciones de mitigación y contingencia para cada riesgo.
  - Asignación de responsables para la ejecución de las acciones.

Herramientas

- **Matriz de riesgos:**
  - Se utilizará para clasificar y realizar seguimiento de los riesgos según su probabilidad e impacto.
  - Permitirá visualizar de manera clara los riesgos críticos que requieren atención inmediata.

Descripción del proceso

1. **Inicio del proceso:**
   - Identificación y registro de riesgos al comienzo del proyecto.
   - Los riesgos se detectarán mediante:
     - Revisión de experiencias previas en proyectos similares.
     - Análisis del entorno y los recursos disponibles.
     - Discusiones con el equipo y stakeholders.
2. **Revisión de riesgos identificados:**
   - Cada riesgo será analizado en términos de probabilidad e impacto.
   - Se priorizarán según su nivel de criticidad.

3. **Registro en la matriz de riesgos:**

   - Los riesgos se clasificarán y registrarán en la matriz para su seguimiento continuo.

4. **Evaluación de respuesta:**

   - Se diseñarán planes de respuesta que incluyan acciones preventivas y correctivas.

5. **Seguimiento y actualización:**

   - Los riesgos serán monitoreados periódicamente.

   - Las estrategias de respuesta se ajustarán según el avance del proyecto.

Matriz de riesgos

Cuadro 3: Matriz de riesgos del proyecto

Riesgo	**Tipo**	**Probabilidad**	**Prioridad**	**Descripción**
Fluidez en la comunicación con stakeholders	Recurso humano	Media	Media	Problemas con los tiempos de respuesta o el medio de comunicación.
Imposibilidad de comunicación con stakeholder	Recurso humano	Baja	Alta	Pérdida de contacto con el stakeholder clave durante el proyecto.
Problemas de seguridad en la plataforma	Técnico	Media	Alta	Vulnerabilidades en la plataforma que afecten la privacidad de los usuarios.
Fiabilidad de los datos	Datos	Baja	Alta	La información recopilada no proviene de fuentes confiables.
Fallo en integración del módulo de IA	Desarrollo	Media	Alta	Problemas técnicos al integrar el modelo de IA para el emparejamiento.
Atraso en el desarrollo	Desarrollo	Media	Alta	Retrasos en el cronograma debido a dificultades técnicas o de recursos.
Limitaciones técnicas del servidor	Hardware	Media	Media	El servidor no soporta la carga esperada de usuarios.
Limitaciones de licencias de software	Software	Media	Alta	Dificultades para obtener licencias necesarias (ej: herramientas de IA).
Resistencia de los usuarios	Recurso humano	Baja	Baja	Usuarios reacios a adoptar la plataforma.
Inconvenientes personales del equipo	Recurso humano	Baja	Alta	Problemas personales que afecten la disponibilidad de los integrantes.

Plan de Respuesta a Riesgos Críticos

Este plan asegura una gestión efectiva de los riesgos, minimizando su impacto en el desarrollo y éxito del proyecto. Se actualizará continuamente para adaptarse a los cambios y desafíos que

Cuadro 4: Plan de respuesta a riesgos críticos

Riesgo	**Acción preventiva**	**Acción de mitigación**	**Recursos**
Fluidez en comunicación con stakeholders	Establecer reuniones regulares y canales claros de comunicación.	Buscar alternativas de contacto o asignar nuevo responsable.	Teams, Jira, correo institucional.
Imposibilidad de comunicación con stakeholder	Definir múltiples puntos de contacto.	Asignar sustituto temporal o recurrir al director.	Documentación de respaldo.
Problemas de seguridad en plataforma	Implementar protocolos de seguridad desde el diseño (Security by Design).	Realizar auditorías periódicas y parches de seguridad.	Herramientas de encriptación, pruebas de penetración.
Fallo en integración del módulo de IA	Realizar pruebas de concepto (PoC) antes de implementación.	Desarrollar plan B con algoritmos más simples.	Documentación técnica, soporte del equipo de IA.
Atraso en el desarrollo	Dividir proyecto en sprints con entregables parciales.	Reasignar recursos o ajustar cronograma.	Jira, reuniones de seguimiento.
Limitaciones de licencias de software	Investigar alternativas de código abierto o planes educativos.	Solicitar apoyo institucional para adquisición.	Presupuesto del proyecto, licencias académicas.
Inconvenientes personales del equipo	Mantener comunicación constante y apoyo entre el equipo.	Redistribuir tareas o incorporar miembros temporales.	Recursos humanos adicionales.

surjan durante la ejecución.

El siguiente diagrama BPMN representa el proceso de **gestión de riesgos** en un proyecto, en el que intervienen los siguientes cuatro roles principales:

- **Gestor de riesgos**
- **Clasificador de riesgos**
- **Gestor de planes de prevención y mitigación**
- **Líder del proyecto**

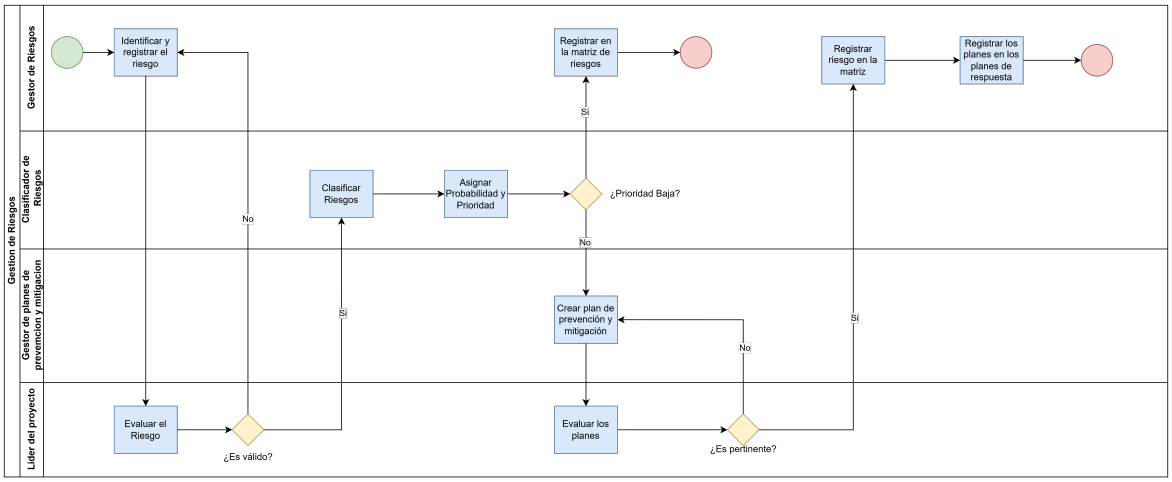

Figura 4: BPMN de gestión de riesgos

Momentos Clave de Identificación de Riesgos en el Proyecto

La identificación de riesgos es un proceso continuo que se desarrolla en distintas fases del proyecto:

1. **Inicio del proyecto:** Se identifican riesgos asociados a la planificación y al diseño preliminar.
2. **Fase de desarrollo:** Se revisan riesgos técnicos y de cumplimiento de plazos.
3. **Fase de pruebas:** Se detectan riesgos relacionados con la integración, funcionalidad y usabilidad del sistema.

5.3. Administración de Configuración y Documentación

El propósito de esta sección es identificar los principales ítems de configuración del proyecto y su evolución temporal, con especial énfasis en los artefactos de documentación y código. Esto permitirá mantener un control sistemático de los cambios y versiones durante todo el ciclo de vida del desarrollo.

Los principales ítems de configuración del proyecto son:

- **Documentación técnica:**
  - Documento de requisitos (IEEE 830)
  - Diseño de arquitectura (diagramas C4 y UML)
  - Informes de avance

- **Código fuente:**
  - Backend (API y servicios)
  - Frontend (interfaz web)
  - Modelos de IA (scripts y datasets)
  - Scripts de despliegue
- **Entornos:**
  - Configuraciones de desarrollo
  - Configuraciones de pruebas (QA)
  - Configuraciones de producción
- **Base de datos:**
  - Esquemas SQL
  - Datos de prueba
  - Migraciones

Cada ítem seguirá un esquema de versionado semántico (vMayor.Menor.Parche) y será almacenado en el repositorio Git del proyecto con su correspondiente documentación de cambios.

Diagrama en BPMN y Explicación del Proceso de Control de Cambios

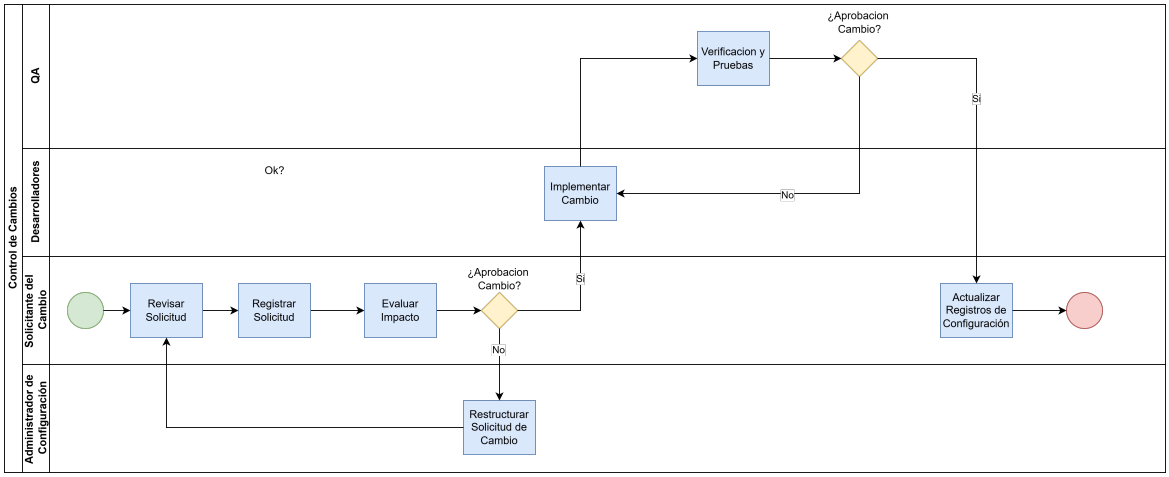

Figura 5: BPMN de Explicación del Proceso de Control de Cambios

El diagrama BPMN describe el flujo completo para la gestión de modificaciones en los ítems de configuración del proyecto, garantizando trazabilidad y control en cada etapa:

1. **Inicio de la Solicitud de Cambio:**
    - El proceso se activa cuando cualquier stakeholder identifica la necesidad de modificación
    - Puede originarse por: requerimientos no previstos, corrección de errores u oportunidades de mejora
    - Se formaliza mediante plantilla estandarizada que incluye: justificación, alcance y beneficio esperado
2. **Revisión Preliminar:**
    - El Comité de Configuración realiza una evaluación inicial de completitud
    - Criterios de revisión: claridad en la descripción, impacto potencial y alineación con objetivos
    - Si es incompleta: se devuelve con observaciones para reformulación (ciclo iterativo)
3. **Registro Formal:**
    - La solicitud válida se ingresa al Sistema de Gestión de Cambios (SGC)
    - Se asigna identificador único (Ej: CC-2024-001) para trazabilidad
    - Automáticamente se genera: bitácora digital, historial de versiones y responsables
4. **Evaluación Técnica de Impacto:**
    - Análisis multidimensional por el equipo técnico:
        - Impacto en arquitectura (dependencias técnicas)
        - Esfuerzo estimado (horas-hombre/complejidad)
        - Riesgos potenciales (estabilidad/seguridad)
    - Se elabora informe con: coste-beneficio y alternativas técnicas
5. **Decisión de Aprobación:**
    - El Comité evalúa el informe técnico en sesión formal
    - Criterios de decisión: valor estratégico, relación costo-beneficio y alineación con roadmap
    - Resultados posibles:

- **Aprobado:** con o sin condiciones (requisitos adicionales)
- **Rechazado:** con justificación documentada
- **Postergado:** para reevaluación en próximo ciclo

6. **Ejecución Controlada:**
   - Se implementa en ambiente aislado (rama Git específica)
   - Requiere: plan de implementación con hitos verificables
   - Uso obligatorio de entornos de staging para cambios críticos
7. **Validación Rigurosa:**
   - Pruebas automatizadas (test suite CI/CD)
   - Verificación manual por QA (casos de uso extremos)
   - Pruebas de regresión para garantizar no afectar funcionalidad existente
8. **Decisión Final:**
   - Basada en métricas objetivas: cobertura de pruebas, rendimiento y cumplimiento de AC
   - Opciones:
     - **Aceptación:** si cumple todos los criterios de calidad
     - **Rechazo:** con informe detallado de no conformidades
     - **Ajustes menores:** aprobación condicional a correcciones específicas
9. **Actualización de Configuración:**
   - Modificación de: documentación técnica, versionado y metadatos
   - Actualización de: manuales de usuario, registros de versión y matrices de trazabilidad
   - Comunicación formal a todos los stakeholders afectados
10. **Cierre Documentado:**
    - Archivo de: evidencias, aprobaciones y resultados de pruebas
    - Lecciones aprendidas incorporadas al repositorio organizacional
    - Notificación oficial de cierre al solicitante original

Este proceso garantiza que cada modificación mantenga la integridad del sistema mientras permite evolucionar la solución de manera controlada. Los tiempos máximos de cada etapa están formalizados en el SLA del proyecto (24h para revisión inicial, 72h para evaluación técnica, etc.).

Cuadro 5: Linea temporal de artefactos del proyecto

Artefacto	**Fase**	**Estado inicial**	**Refinamiento**
Documento de requisitos (IEEE 830)	Análisis y Diseño	Borrador inicial con historias de usuario	Versión final validada con stakeholders
Prototipos UI (Figma)	Análisis y Diseño	Wireframes de baja fidelidad	Prototipos interactivos con feedback de usabilidad
Modelo de IA para emparejamiento	Pruebas de Concepto	PoC con dataset básico	Modelo optimizado con métricas de precisión
Codigo Fuente	Desarrollo	Módulo básico de autenticación	Versión estable con todos los endpoints
Base de datos (ERD)	Preparación de Ambientes	Esquema conceptual	Modelo físico con índices optimizados
Sistema de evaluación	Desarrollo	Mecanismo básico de calificación	Integración con reputación y feedback
Manual de usuario	Pruebas	Guía preliminar	Documentación completa con screenshots
Informe de pruebas	Pruebas	Reporte parcial de validación	Informe final con métricas de rendimiento
Documentación técnica (SDD)	Todas las fases	Estructura inicial	Versión detallada con decisiones técnicas

- **Relación con entregables:** Esta tabla sintetiza los 9 entregables oficiales mencionados en la sección 1.3 del documento, mapeándolos a las fases metodológicas.
- **Evolución:** Cada artefacto muestra su estado inicial (entregable mínimo) y su versión refinada (entregable completo).
- **Trazabilidad:** Los artefactos están alineados con los objetivos específicos de la sección 1.2.2, garantizando coherencia con la propuesta original.

La Tabla 5 muestra cómo los principales artefactos evolucionan a lo largo del proyecto. Cada uno tiene hitos claros de creación y puntos de refinamiento, garantizando que la documentación y el código mantengan coherencia con el estado actual del desarrollo.

5.4. Métricas y Proceso de Medición

Establecer un sistema de medición que permita monitorear el progreso y calidad del proyecto, asegurando que todos los integrantes comprendan:

- Qué aspectos se medirán (indicadores clave)
- Cómo se realizarán las mediciones (procedimientos)
- Cómo se transformarán los datos en información accionable (análisis)

El sistema de medición tiene como objetivo principal proveer un marco estructurado para evaluar el progreso y calidad del proyecto, asegurando que todos los integrantes tengan claridad sobre tres aspectos fundamentales: los indicadores clave que se medirán, los procedimientos establecidos para su recolección, y los métodos de análisis que transformarán los datos en información accionable para la toma de decisiones.

Métricas Clave

La Tabla 6 presenta las seis métricas principales que guiarán la evaluación del proyecto, cada una vinculada directamente con secciones específicas del documento y con los planes de gestión correspondientes. Estas métricas han sido seleccionadas por su capacidad para medir aspectos críticos como el avance temporal (cumplimiento de hitos), la calidad técnica (código y seguridad), la efectividad del núcleo de la solución (modelo de IA) y la aceptación por parte de los usuarios finales.

Cuadro 6: Métricas del proyecto y su relación con planes

Métrica	**Referencia**	**Aplicación**
Cumplimiento de hitos	Sección 3.1.3	Control del cronograma
Calidad de código	Estándares ISO/IEC 25010	Mejora continua
Precisión del modelo IA	Pruebas de concepto (3.2.3)	Optimización algorítmica
Satisfacción de usuarios	Pruebas de usabilidad (3.5.3)	Refinamiento UI/UX
Rendimiento del sistema	Pruebas de carga (3.5.2)	Escalabilidad
Vulnerabilidades	Preparación de ambientes (3.3.2)	Plan de mitigación

Proceso de Medición

El proceso de medición se estructura en tres etapas claramente diferenciadas:

Recolección de Datos

La recolección está a cargo de los distintos roles del equipo según su especialización: los desarrolladores recopilan métricas de código mediante SonarQube, el Product Owner gestiona los indicadores de progreso en Jira, y el equipo de QA ejecuta las pruebas de rendimiento con JMeter. La frecuencia varía desde mediciones semanales para los hitos del proyecto hasta evaluaciones puntuales post-implementación para las métricas de rendimiento.

Procesamiento y Consolidación

Bajo la coordinación del Líder Técnico y el Scrum Master, los datos brutos se procesan al final de cada sprint utilizando herramientas como Power BI para la generación de dashboards interactivos. Este proceso incluye tres pasos fundamentales: normalización de los datos según estándares predefinidos, cálculo de los indicadores clave, y creación de visualizaciones que faciliten su interpretación.

Uso de las Métricas

La información consolidada sirve como base para tres tipos de acciones estratégicas. En primer lugar, permite la toma de decisiones técnicas como la replanificación de sprints cuando se detectan desviaciones superiores al 15 % o la priorización de deuda técnica ante la presencia de code smells críticos. Segundo, facilita la comunicación mediante reportes periódicos al comité directivo y presentaciones a stakeholders. Finalmente, impulsa la mejora continua a través de la actualización de estándares y la identificación de necesidades de capacitación.

Integración con las Fases del Proyecto

Cada fase metodológica cuenta con un conjunto específico de métricas, como se detalla en la Tabla 7. Durante la fase de Análisis y Diseño, por ejemplo, el foco recae en la validación de requisitos y prototipos, mientras que en las Pruebas de Concepto se prioriza la evaluación del modelo de IA. Este enfoque por fases garantiza que las mediciones sean relevantes para los objetivos inmediatos del proyecto en cada etapa.

Cuadro 7: Correspondencia entre fases y métricas

Fase	**Métricas**	**Propósito**
3.1 Análisis	Cumplimiento, Calidad doc.	Validar requisitos
3.2 Pruebas	Precisión IA, Rendimiento	Viabilidad técnica
3.3 Ambientes	Vulnerabilidades, Tiempos	Infraestructura
3.4 Desarrollo	Calidad código, Avance	Construcción MVP
3.5 Pruebas	Satisfacción, Rendimiento	Validación final

El sistema establece criterios objetivos para la transición entre fases, requiriendo que al menos el 90 % de las métricas clave cumplan con sus umbrales definidos. Esta decisión es tomada por el Comité de Calidad, compuesto por el Director, el Líder Técnico y el Product Owner, quienes revisan los reportes de transición que incluyen análisis comparativos y lecciones aprendidas.

Todos los procesos de medición siguen los estándares de calidad definidos en la sección 1.3 del documento, manteniendo trazabilidad con los objetivos específicos de la sección 1.2.2. El

sistema se somete a evaluación cada dos sprints para incorporar mejoras, asegurando su continua efectividad a lo largo del ciclo de vida del proyecto.

5.5. Control de Calidad

El sistema de control de calidad del proyecto se compone de seis procesos fundamentales diseñados para garantizar la excelencia técnica y la satisfacción del usuario. Cada proceso cuenta con responsables asignados, momentos específicos de ejecución y criterios de evaluación claramente definidos.

Cuadro 8: Procesos de Control de Calidad

Proceso	**Descripción**	**Momento**	**Responsable**
Revisión de Documentación	Evaluación exhaustiva de documentos técnicos (SRS, SDD) según estándares IEEE	Fin de cada fase documental	Thomas Orjuela, Juan Gomez, Andres Velez
Pruebas Unitarias	Verificación funcional de componentes individuales del sistema	Post-desarrollo de cada módulo	Thomas Orjuela
Pruebas de Sistema	Validación de integración entre módulos y funcionamiento global	Post-integración de componentes	Andres Velez , Juan Gomez
Revisión de Código	Inspección de calidad, estándares y patrones de diseño	Periódicamente y pre-entregas	Thomas Orjuela
Auditoría Externa	Evaluación independiente de conformidad con estándares	Fin de cada fase	Evaluador externo
Pruebas de Usuario	Validación de UX en condiciones reales con usuarios finales	Fase de pruebas finales	Thomas Orjuela, Andres Velez, Juan Gomez

Descripción Detallada de Procesos

Revisión de Documentación

Este proceso garantiza que toda la documentación técnica cumpla con los estándares de claridad, completitud y coherencia. Los revisores evalúan minuciosamente los documentos clave como el *Software Requirements Specification* (SRS) y el *Software Design Document* (SDD), verificando su alineación con los objetivos del proyecto y su capacidad para guiar efectivamente el desarrollo.

Pruebas Unitarias

Implementadas mediante un enfoque sistemático, estas pruebas validan el funcionamiento individual de cada componente del sistema. Se ejecutan mediante frameworks automatizados que verifican el cumplimiento de los requisitos funcionales y no funcionales a nivel de unidad, asegurando una base sólida para la integración posterior.

Pruebas de Sistema

Este proceso crítico evalúa la interoperabilidad de todos los módulos integrados, incluyendo los componentes de realidad aumentada. Las pruebas se diseñan para simular condiciones operacionales reales, identificando posibles fallos en las interfaces entre subsistemas antes del despliegue.

Revisión de Código

Conductas regulares de *code review* aseguran el mantenimiento de altos estándares de calidad en el código fuente. Este proceso no solo verifica el cumplimiento de convenciones de codificación, sino que también evalúa aspectos de eficiencia algorítmica, manejo de errores y adherencia a patrones de diseño arquitectónico.

Auditoría Externa

Realizada por un tercero independiente, esta evaluación objetiva examina el producto contra estándares internacionales y mejores prácticas de la industria. La auditoría proporciona una visión imparcial de la calidad global del sistema y su preparación para el despliegue.

Pruebas de Usuario

Pruebas de usabilidad con usuarios reales permiten validar la experiencia de usuario en escenarios operacionales realistas. Este proceso genera *feedback* valioso para refinar la interfaz y optimizar los flujos de interacción, particularmente en los componentes de realidad aumentada.

Flujo de Control de Calidad

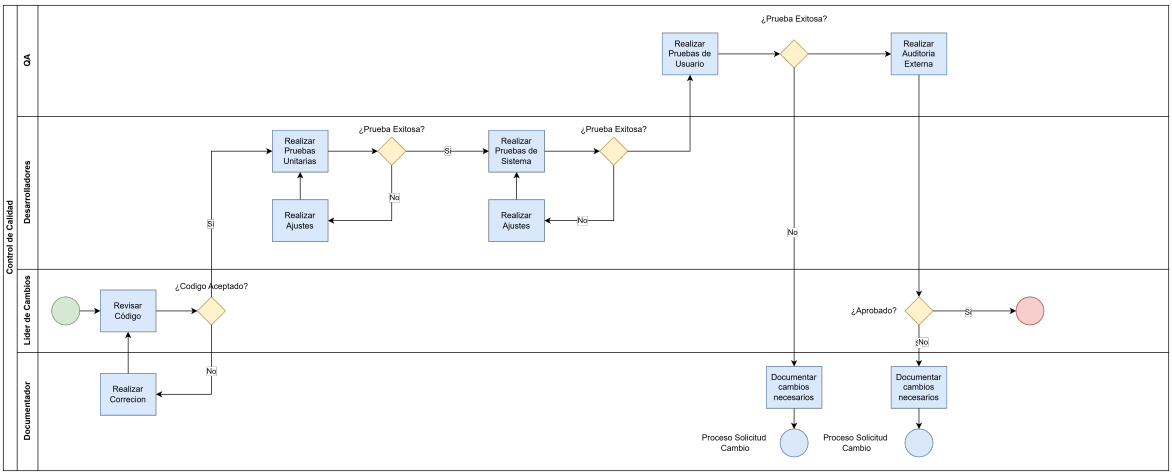

Figura 6: BPMN Descripción de los Procesos de Control de Calidad.

El flujo de calidad sigue una secuencia lógica que comienza con la recepción del código fuente y culmina con la auditoría final. Cada etapa incluye pasos de verificación y retroalimentación:

1. **Recepción y Evaluación Inicial**: Análisis preliminar del código contra estándares de codificación
2. **Pruebas Unitarias**:
   - Ejecución automatizada de casos de prueba
   - Identificación y corrección de errores unitarios
3. **Pruebas de Sistema**:
   - Validación de interfaces entre módulos
   - Ajuste de configuraciones del sistema integrado
4. **Pruebas de Usuario**:
   - Evaluación de experiencia de usuario
   - Documentación de requerimientos de cambio
5. **Pruebas de Carga**:
   - Evaluación de rendimiento bajo estrés
   - Optimización de parámetros del sistema

6. **Auditoría Externa**:
    - Verificación de cumplimiento normativo
    - Implementación de acciones correctivas

Este sistema integrado de control de calidad asegura que el producto final cumpla con los más altos estándares de funcionalidad, rendimiento y usabilidad, alineándose con los objetivos establecidos en la sección 1.2 del documento de proyecto.

Referencias

[1] Visual Paradigm. (s.f.). *What is Burndown Chart in Scrum?* Visual Paradigm. Recuperado de https://www.visual-paradigm.com/scrum/scrum-burndown-chart/