

Grupo 6

Plataforma Freelance para Jóvenes:
Conectando Talento con Oportunidades
Laborales

Juan Camilo Gómez Contento
Andrés Felipe Vélez Echeverry
Thomas Alejandro Orjuela Bello

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ, D.C.

2025

ISIST-2530-06

Plataforma Freelance para Jóvenes: Conectando Talento con Oportunidades Laborales

Autores:

Juan Camilo Gómez Contento
Andrés Felipe Vélez Echeverry
Thomas Alejandro Orjuela Bello

MEMORIA DE PROYECTO DE GRADO REALIZADO PARA CUMPLIR UNO
DE LOS REQUISITOS PARA EL TÍTULO EN INGENIERÍA DE SISTEMAS

Director

Ing. Pablo Andrés Márquez Fernández

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ, D.C.

Noviembre, 2025

**PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERÍA DE SISTEMAS**

Rector de la Pontificia Universidad Javeriana

P. Luis Fernando Múnera Congote, S.J.

Decano de la Facultad de Ingeniería

Ing. Diego Alejandro Patiño Guevara

Director de Carrera de Ingeniería de Sistemas

Ing. Andrea del Pilar Rueda Olarte

Director del Departamento de Ingeniería de Sistemas

Ing. Cesar Julio Bustacara Medina

Artículo 23 de la Resolución No. 1 de Junio de 1946

“La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia”

AGRADECIMIENTOS

Queremos expresar nuestro más sincero agradecimiento a nuestro director de tesis Pablo Márquez, quien nos estuvo orientando durante todo el proceso de desarrollo de nuestro trabajo de grado final; apoyándonos con su conocimiento y experiencia en ingeniería de sistemas, que fueron claves para aprender y alcanzar los objetivos propuestos.

Thomas Alejandro Orjuela Bello:

Quiero agradecer especialmente a Dios por haberme bendecido durante todo el proceso de mi carrera en ingeniería de sistemas y agradecerle de corazón a mi familia, mi madre, mi padre y mi hermano, quienes siempre me apoyaron de todas las maneras posibles, sin ellos, no hubiese sido posible llegar hasta este punto en mi carrera, ellos son mi más grande fuerza para seguir adelante, agradecer especialmente a mi abuelita, quien falleció durante este proceso, quiero agradecerle por el apoyo y por todo lo que hizo por mí y por nuestra familia.

Mi familia y Dios me dieron las fuerzas y motivación que necesitaba para seguir adelante durante mi proceso académico y les agradezco de todo corazón.

TABLA DE CONTENIDO

I. INTRODUCCIÓN	1
II. DESCRIPCIÓN GENERAL	2
1 Oportunidad, Problema	2
1.1 Contexto del Problema	2
1.2 Formulación del Problema	2
1.3 Propuesta de Solución	3
1.4 Justificación de la Solución	4
2 Descripción del Proyecto	5
2.1 Objetivo General	5
2.2 Objetivos Específicos	5
2.3 Entregables, Estándares y Justificación	5
III. CONTEXTO DEL PROYECTO	7
1 Transfondo	7
2 Análisis del Contexto	8
2.1 Soluciones Existentes a Nivel Global	8
2.2 Valor Agregado de la Solución Propuesta	8
IV. ANÁLISIS DEL PROBLEMA	10
1 Requerimientos	10
1.1 Requerimientos Funcionales	10
1.2 Requerimientos No Funcionales Críticos	10
2 Restricciones	11
2.1 Restricciones Técnicas	11
2.2 Restricciones Operacionales	11
2.3 Restricciones de Negocio	11
3 Especificación Funcional	12
V. DISEÑO DE LA SOLUCIÓN	15
1 Pruebas de Concepto	15
1.1 Implementaciones Evaluadas	15
1.2 Metodología Evaluación	17
1.3 Resultados Funcionales	17
1.4 Resultados de Rendimiento	20
2 Sobre el uso de DAMA	21
2.1 Fase 1: Gobierno de Datos (Data Governance)	21
2.2 Fase 2: Arquitectura de Datos (Data Architecture)	21
2.3 Fase 3: Desarrollo de Datos (Data Development)	21
2.4 Fase 4: Operaciones de Datos (Data Operations)	22
2.5 Fase 5: Calidad de Datos (Data Quality)	22
2.6 Fase 6: Decisión Arquitectónica	22
2.7 Lecciones Aprendidas DAMA	23
2.8 Conclusión	23
3 Diseño base de datos	25
3.1 Objetivos del Diseño	25
3.2 Descripción de Entidades	26

3.3	Justificación del Diseño	28
3.4	Conclusiones	28
4	Arquitectura basada en C4	28
4.1	Contexto	29
4.2	Capa de Contenedores	30
4.3	Capa de Componentes	32
5	Arquitectura de la solución	35
5.1	Microservicios	35
5.2	Microfrontends	38
6	Despliegue	41
6.1	Arquitectura General Basada en Kubernetes	41
6.2	Implementación de Microfrontends con Webpack Module Federation / Native Federation	42
6.3	Arquitectura de Microservicios Backend (Spring Boot + FastAPI)	43
6.4	Estrategia de Contenerización con Docker	44
6.5	Optimización de Imágenes y Seguridad	44
6.6	Kubernetes como Plataforma de Despliegue y Operación	44
6.7	Redes y Descubrimiento de Servicios	45
6.8	Pipeline de CI/CD con GitHub Actions	45
6.9	Flujo Completo de Despliegue End-to-End	45
6.10	Actualización en Clúster y Composición Dinámica	46
VI. DESARROLLO DE LA SOLUCIÓN	47
1	Planificación del cronograma	47
1.1	Metodología y dinámica de trabajo	49
1.2	Estructura de los Sprints en el Cronograma	50
2	Desarrollo y Pruebas	50
2.1	Flujo técnico	50
VII. RESULTADOS	52
1	Resultados del prototipo desarrollado	52
2	Resultados de la integración del motor de IA	52
3	Resultados del desarrollo y de las pruebas de calidad	52
4	Pruebas unitarias e integración	52
5	Pruebas funcionales	53
6	Pruebas de carga	53
7	Resultados frente a los objetivos del proyecto	53
8	Impacto y aporte del proyecto	53
9	Pruebas funcionales	53
9.1	Metodología aplicada	54
9.2	Prueba con Sujeto 1	55
9.3	Prueba con Sujeto 2	55
VIII. CONCLUSIONES	57
1	Análisis de Impacto del Proyecto	57
2	Trabajos Futuros	57
2.1	Implementación de un rol de administrador	57
2.2	Integración de colas de mensajería (Kafka)	57
2.3	Evolución del motor de IA	58

2.4	Extensión del modelo de reputación	58
2.5	Integración con herramientas externas	58
3	Conclusiones Generales	58
IX.	REFERENCIAS	59
X.	ANEXOS	60

LISTA DE FIGURAS

1	Tabla Prioridades Clasificación POCs	18
2	Esquema Tabla Evaluativa por Concepto	18
3	Resultados Implementación Embeddings	18
4	Resultados Implementación Filtros	19
5	Resultados Implementación RAG	19
6	Resultados Rendimiento POCs 200-10	20
7	Implementación Solución Híbrida	24
8	Rendimiento Implementación Híbrida	24
9	Modelo Entidad Relación	25
10	Diagrama Contexto - Arquitectura C4	29
11	Diagrama Contenedores - Arquitectura C4	31
12	Diagrama Componentes Proyectos - Arquitectura C4	33
13	Diagrama Componentes Componentes - Arquitectura C4	34
14	Diagrama de despliegue	41
15	Ciclo de vida proyecto	49
16	Flujo Funcional de la Aplicación	51

LISTA DE TABLAS

1	Entregables adicionales anexos al Trabajo de Grado	6
2	Especificación Funcional por Historias de Usuario	13
3	Criterios de Aceptación por Historia de Usuario	14
4	Microservicio 001 – Autenticación y Autorización	35
5	Microservicio 002 – Administración de Perfiles	36
6	Microservicio 003 – Ofertas y Proyectos	36
7	Microservicio 004 – Matchmaking (IA)	37
8	Microservicio 005 – Chat y Comunicación	37
9	Microservicio 006 – Evaluación y Feedback	38
10	Cronograma inicial del proyecto	48
11	Pruebas Funcionales - Sujeto 1	55
12	Pruebas Funcionales - Sujeto 2	56

ABSTRACT

This project implements a specialized web platform designed to match university students with short-term freelance opportunities using an AI-driven recommendation engine. A distributed architecture of seven microservices (Spring Boot, FastAPI) and Angular-based microfrontends supports identity management, CV processing, project lifecycle management, real-time communication, and bilateral evaluation. Student résumés are parsed with Natural Language Processing techniques, vectorized using PgVector, and matched to project requirements through semantic-similarity models. Continuous integration and deployment were achieved with Docker, Kubernetes, and GitHub Actions. Functional, integration, and load tests validated system reliability, achieving sub-5-second recommendation responses and stable operation under concurrent usage.

I INTRODUCCIÓN

El presente documento expone de manera detallada el diseño, desarrollo e implementación de una plataforma web orientada a conectar estudiantes universitarios con oportunidades laborales tempranas mediante un sistema de emparejamiento inteligente basado en inteligencia artificial. La iniciativa surge como respuesta a una problemática ampliamente reconocida en el ámbito académico y profesional: la brecha existente entre la formación impartida por las instituciones educativas y las competencias prácticas demandadas por el mercado laboral actual. Esta brecha se hace especialmente evidente en la dificultad que enfrentan los estudiantes y recién egresados para acceder a experiencias reales que fortalezcan su perfil profesional, desarrollos sus habilidades y faciliten su transición hacia el mundo laboral.

Con el fin de abordar esta necesidad, el proyecto propone una solución integral que centraliza perfiles estudiantiles, proyectos publicados por contratantes y un motor de recomendación capaz de identificar coincidencias relevantes entre ambas partes. Para ello, se realizó un análisis exhaustivo del contexto, que incluyó la identificación de los principales desafíos del sector, la evaluación de plataformas existentes y la definición de requerimientos funcionales y no funcionales que garantizaran la pertinencia, escalabilidad y eficacia del sistema.

A partir de este estudio, se diseñó una arquitectura distribuida moderna, compuesta por microservicios independientes, microfrontends altamente desacoplados y un módulo especializado de procesamiento del lenguaje natural (NLP), encargado de transformar hojas de vida en representaciones vectoriales y generar recomendaciones basadas en similitud semántica. Este enfoque permite optimizar la precisión de los emparejamientos, mejorar la experiencia del usuario y garantizar que el sistema pueda escalar de manera eficiente.

El documento se organiza de manera que el lector pueda comprender el proyecto de forma progresiva. Primero, se presenta el problema a resolver y la justificación de la solución propuesta; posteriormente, se describen los componentes arquitectónicos, las tecnologías utilizadas y las decisiones de diseño adoptadas. Finalmente, se exponen los resultados obtenidos durante las distintas fases de prueba —incluyendo pruebas funcionales, de integración y de rendimiento— así como las conclusiones derivadas del proceso de desarrollo.

II DESCRIPCIÓN GENERAL

1 Oportunidad, Problema

1.1 Contexto del Problema

El desempleo juvenil en América Latina representaba una problemática estructural significativa, con tasas que duplicaban el promedio regional general. En 2024, diferentes estudios reportaron que la tasa de desempleo juvenil en la región llegó a ser tres veces mayor que la de los adultos, lo cual evidencia una brecha persistente y amplia en la inclusión laboral de los jóvenes (Efe, 2025).

En el contexto colombiano, esta situación era particularmente crítica para los estudiantes universitarios y recién egresados, quienes enfrentaban una paradoja laboral: las empresas demandaban experiencia previa para contratar, pero los jóvenes carecían de oportunidades para adquirir dicha experiencia. Este fenómeno ha sido identificado como uno de los principales factores que dificultan la inserción laboral juvenil en el país (Infobae, 2025).

Investigaciones regionales sobre empleabilidad profesional señalan que un porcentaje importante de empleadores percibe que los recién graduados no poseen las competencias prácticas necesarias para integrarse inmediatamente al mercado laboral. Esta brecha entre formación académica y expectativas del mercado ha sido documentada en estudios sobre transiciones educativas y laborales en América Latina (CEPAL, 2006).

Simultáneamente, las pequeñas y medianas empresas —que constituyen el mayor generador de empleo en la región— enfrentaban limitaciones para acceder a talento joven capacitado que pudiera apoyar proyectos específicos con costos reducidos. Organismos internacionales han señalado que las PyMEs latinoamericanas suelen ser más sensibles a los costos asociados a la contratación, lo que incrementa la barrera de entrada para trabajadores sin experiencia (OIT, 2025).

La educación superior tradicional, aunque sólida en fundamentos teóricos, presentaba desconexiones con las demandas reales del sector productivo. Los estudiantes culminaban sus carreras con conocimientos académicos robustos, pero con escasas oportunidades de aplicarlos en contextos laborales reales antes de su graduación. Esta brecha entre formación y práctica ha sido ampliamente discutida en estudios sobre pertinencia educativa en la región (Banco Interamericano de Desarrollo, 2021).

Esta situación se exacerbaba por la falta de plataformas especializadas que facilitaran la interacción entre el talento universitario y las necesidades de las empresas, especialmente en modelos de trabajo por proyecto o en la creciente economía freelance.

1.2 Formulación del Problema

El problema específico identificado fue la ausencia de un mecanismo eficiente que conectara a estudiantes universitarios con oportunidades laborales tempranas y relevantes que les permitieran adquirir experiencia práctica mientras culminaban sus estudios. Esta carencia se manifestaba en tres dimen-

siones críticas:

Para los estudiantes, existía una dificultad para encontrar proyectos que se ajustaran a sus horarios académicos, nivel de experiencia y áreas de conocimiento específicas. Las plataformas freelance existentes estaban orientadas a profesionales con experiencia, creando una barrera de entrada casi infranqueable para quienes buscaban sus primeras oportunidades.

Para los contratistas, específicamente pymes y emprendedores, el proceso de encontrar talento joven calificado resultaba ineficiente y consumía recursos significativos. No existían herramientas especializadas que permitieran identificar estudiantes con las competencias específicas requeridas para proyectos puntuales.

Para las instituciones educativas, había una limitación en la capacidad de proporcionar a sus estudiantes experiencias prácticas diversificadas que complementaran su formación académica y mejoraran su empleabilidad al graduarse.

La relevancia de este problema radicaba en su impacto directo sobre la transición educación-empleo, un factor determinante en el desarrollo económico regional y en la realización profesional de las nuevas generaciones de profesionales.

1.3 Propuesta de Solución

Se desarrolló e implementó una plataforma web especializada en conectar estudiantes universitarios con oportunidades laborales tempranas mediante un sistema de emparejamiento inteligente. La solución se enmarca en el área de desarrollo de software e ingeniería de sistemas, específicamente en el diseño e implementación de aplicaciones web empresariales con arquitecturas distribuidas.

La plataforma incorporó funcionalidades clave que abordaban directamente las dimensiones del problema identificado: Para los estudiantes, se implementó un sistema de perfiles profesionales enriquecidos que permitía la carga y procesamiento automatizado de hojas de vida, facilitando la presentación de sus competencias de manera estandarizada. El sistema de búsqueda y recomendación de proyectos que consideraban factores relevantes de los perfiles de estudiantes como las habilidades técnicas.

Para los contratistas, se desarrolló un motor de recomendación basado en inteligencia artificial que analizaba los requerimientos de los proyectos y los comparaba con los perfiles estudiantiles disponibles, identificando automáticamente los candidatos más adecuados. Esto redujo significativamente el tiempo y esfuerzo requerido para encontrar talento especializado.

A nivel técnico, se adoptó una arquitectura de microservicios que garantizó escalabilidad, mantenibilidad y la capacidad de evolucionar funcionalidades de manera independiente. El frontend se desarrolló como una aplicación web responsive.

1.4 Justificación de la Solución

La solución planteada para abordar la falta de experiencia laboral entre estudiantes universitarios es completa y se adapta a las exigencias del mercado actual.

Uno de los mayores retos a los que se enfrentan los estudiantes al acercarse al final de su formación es la brecha entre su preparación académica y la experiencia práctica que exigen la mayoría de empleos. Esta plataforma freelance funciona como un puente que les permite acceder a proyectos reales mientras aún están estudiando, generando un entorno donde pueden aplicar lo aprendido en clases y fortalecer sus competencias técnicas y profesionales desde etapas tempranas.

La flexibilidad es uno de los pilares más importantes de esta solución. Debido a los horarios académicos, a muchos estudiantes les resulta difícil asumir empleos tradicionales. En cambio, el formato freelance les permite elegir proyectos compatibles con su disponibilidad, facilitándoles equilibrar sus estudios con trabajo práctico. Este modelo fomenta la autonomía, la responsabilidad y la capacidad de organización, habilidades esenciales en cualquier entorno laboral moderno.

La plataforma también abre la puerta a una variedad de proyectos y clientes, lo que permite a los estudiantes explorar diferentes áreas dentro de su disciplina. Por ejemplo, un estudiante de ingeniería informática podría participar en proyectos de desarrollo web, automatización, soporte técnico o análisis de datos, lo que le ayuda a descubrir sus áreas de interés y a construir un portafolio diverso. Esta exposición temprana a distintos tipos de tareas favorece la toma de decisiones sobre futuras especializaciones y fortalece su perfil profesional.

Otro aspecto relevante es la posibilidad de construir una red de contactos reales. A través de las colaboraciones con clientes y equipos, los estudiantes pueden establecer relaciones laborales que a futuro pueden convertirse en oportunidades de empleo o en conexiones estratégicas en su industria. En un mercado laboral cada vez más competitivo, contar con contactos directos es un factor decisivo que complementa la formación académica.

Aunque la plataforma no ofrece cursos ni sistemas formales de mentoría, la experiencia obtenida al enfrentarse a proyectos reales actúa como una forma de aprendizaje práctico. Los estudiantes desarrollan habilidades técnicas, comunicativas y de trabajo colaborativo al interactuar directamente con clientes y al adaptarse a las necesidades específicas de cada proyecto. Este aprendizaje basado en la práctica es especialmente valioso, ya que simula condiciones reales del entorno laboral y acelera el desarrollo de competencias profesionales.

El impacto directo en la empleabilidad es claro: al acumular experiencia comprobable, construir un portafolio sólido y generar conexiones profesionales, los estudiantes aumentan significativamente sus posibilidades de conseguir empleo una vez se gradúan. Esta solución no solo atiende las necesidades individuales de los estudiantes, sino que responde también a las expectativas de empresas que requieren talento joven con experiencia práctica y capacidad de adaptación.

En conclusión, esta plataforma freelance constituye una respuesta eficaz al problema de la falta de experiencia laboral entre estudiantes. Su enfoque flexible, orientado a proyectos reales y a la construcción de trayectorias profesionales tempranas, permite que los estudiantes se desarrolle de manera integral y que mejoren su competitividad en el mercado laboral. Esta iniciativa beneficia tanto al

talento en formación como a los sectores productivos que buscan profesionales mejor preparados para los desafíos actuales.

2 Descripción del Proyecto

2.1 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.

2.2 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.
- Diseñar y ejecutar pruebas funcionales y de rendimiento para validar la calidad, estabilidad y fiabilidad de la plataforma.

2.3 Entregables, Estándares y Justificación

Cuadro 1: Entregables adicionales anexos al Trabajo de Grado

Entregable (Anexo)	Estándares asociados	Justificación
Anexo 1 — Especificación de Requisitos (SRS)	IEEE 830; ISO/IEC/IEEE 29148	Establece los requisitos funcionales y no funcionales del sistema, garantizando trazabilidad y alineación con la solución desarrollada.
Anexo 2 — Documento de Diseño	IEEE 1016; C4 Model; ISO/IEC/IEEE 42010	Describe la arquitectura del sistema, el diseño de microservicios y microfrontends, los modelos de datos y las decisiones técnicas adoptadas. Facilita la comprensión estructural del prototipo.
Anexo 3 — Documento de Control de Calidad (Pruebas)	IEEE 29119	Incluye el plan de pruebas, casos ejecutados y resultados de pruebas unitarias, de integración, funcionales, de carga y usabilidad. Garantiza la calidad técnica del prototipo desarrollado.
Anexo 4 — Código fuente (Repositorio GitHub)	Buenas prácticas de ingeniería; GitFlow; Estándares REST	Contiene toda la implementación del sistema. Permite validar el desarrollo real del prototipo, asegurar reproducibilidad y habilitar futuras extensiones técnicas.

III CONTEXTO DEL PROYECTO

1 Transfondo

El desarrollo de este proyecto se fundamentó en la convergencia de diversas tendencias tecnológicas y socioeconómicas que han marcado profundamente la evolución del trabajo, la educación y los modelos de innovación digital en los últimos años. Estas tendencias permiten contextualizar la pertinencia y el valor de la solución implementada.

La economía gig ha transformado radicalmente los modelos tradicionales de empleo, caracterizándose por trabajos temporales, flexibles y basados en proyectos. Según estimaciones internacionales, una proporción significativa de la fuerza laboral mundial participa actualmente en modalidades freelance o de trabajo por encargo, lo cual refleja un cambio estructural hacia esquemas laborales más dinámicos y descentralizados (OIT, 2025). Sin embargo, esta modalidad presenta barreras específicas para estudiantes universitarios y profesionales junior, quienes suelen carecer de la experiencia, reputación y portafolio necesarios para competir en plataformas generalistas, dificultando su entrada al mercado de trabajo digital.

En paralelo, el procesamiento de lenguaje natural (NLP) y los modelos de lenguaje grandes (LLMs) experimentaron avances acelerados durante el periodo de desarrollo del proyecto. Tecnologías como BERT y GPT permitieron un análisis semántico más profundo de documentos no estructurados, como hojas de vida y descripciones de proyectos, superando las limitaciones de los sistemas tradicionales basados exclusivamente en coincidencias de palabras clave (Devlin et al., 2018; OpenAI, 2023). Este salto tecnológico abrió la puerta a soluciones de recomendación más precisas y centradas en el contenido.

Las arquitecturas de microservicios y microfrontends emergieron como el paradigma dominante para el desarrollo de aplicaciones empresariales escalables. Este enfoque permitió descomponer funcionalidades complejas en servicios independientes capaces de evolucionar, escalar y fallar de manera aislada, contrastando con la rigidez inherente de las arquitecturas monolíticas tradicionales en términos de mantenibilidad y velocidad de desarrollo. Diversos análisis técnicos han señalado que este modelo favorece la agilidad organizacional y la rápida iteración en proyectos de software distribuido (Arias Ortiz et al., 2021).

Finalmente, el concepto de matching inteligente evolucionó desde algoritmos basados en reglas simples hacia sistemas de recomendación sofisticados que incorporan análisis de similitud semántica y técnicas de aprendizaje automático. Esta evolución ha sido fundamental para abordar la complejidad del emparejamiento entre las competencias reales de los estudiantes y los requerimientos específicos de proyectos freelance, permitiendo mayor pertinencia en las asignaciones y mayor visibilidad del talento emergente en ecosistemas digitales.

2 Análisis del Contexto

2.1 Soluciones Existentes a Nivel Global

- **Plataformas Freelance Generalistas:** Upwork y Freelancer.com dominan el mercado global de trabajo freelance y concentran una parte significativa del tráfico y transacciones dentro de la economía gig. Estudios recientes indican que este tipo de plataformas han sido un motor clave en la expansión del trabajo por proyecto, especialmente entre profesionales independientes. Sin embargo, continúan presentando limitaciones específicas para estudiantes y recién graduados: exigen experiencia comprobable, funcionan bajo un esquema altamente competitivo basado en precios y carecen de mecanismos dedicados para validar habilidades formativas o académicas (OIT, 2025; CEPAL, 2015).
- **Plataformas de Empleo Tradicionales:** LinkedIn e Indeed incorporan funcionalidades de búsqueda y emparejamiento a partir de perfiles profesionales, algoritmos que consideran historial laboral, conexiones y credenciales verificables. No obstante, estas plataformas están diseñadas principalmente para perfiles con experiencia previa, lo que deja en desventaja a estudiantes o profesionales junior que aún no cuentan con trayectoria laboral sólida. Diversos análisis sobre inserción laboral juvenil en la región muestran que los mecanismos tradicionales de reclutamiento tienden a priorizar candidatos con experiencia formal, reforzando las brechas de acceso (CEPAL, 2006; Banco Interamericano de Desarrollo, 2021).
- **Plataformas Universidades-Industria:** Handshake en Estados Unidos y Graduateland en Europa representan alternativas que conectan estudiantes con empleadores en contextos académicos. Estas plataformas han alcanzado una alta adopción institucional, especialmente en universidades norteamericanas, donde su enfoque principal se dirige a prácticas, pasantías y primeros empleos de tiempo completo. Sin embargo, estudios recientes han señalado que este tipo de plataformas no cubren adecuadamente modalidades de proyectos freelance o trabajos cortos basados en entregables, lo que limita su alcance para economías emergentes donde el trabajo por proyecto es más dinámico (CEPAL, 2015; Banco Interamericano de Desarrollo, 2021).
- **Soluciones Regionales Latinoamericanas:** Workana, Bumeran y otras plataformas regionales han logrado una presencia significativa en América Latina, particularmente en mercados freelance generalistas. Workana, por ejemplo, ha reportado millones de usuarios activos en la última década, lo cual refleja un crecimiento sostenido del trabajo por proyecto en la región. A pesar de ello, análisis de organismos regionales han identificado que su modelo continúa siendo genérico, sin adaptaciones específicas para el entorno académico latinoamericano ni mecanismos para integrar competencias universitarias, créditos académicos o validación institucional, lo cual limita su utilidad para estudiantes y recién egresados (CEPAL, 2015; OIT, 2025).

2.2 Valor Agregado de la Solución Propuesta

- **Algoritmos de Matching Centrados en Potencial vs. Experiencia:** A diferencia de Upwork y LinkedIn, que priorizan historial laboral, nuestro motor de recomendación implementó análisis de competencias adquiridas, proyectos académicos y potencial de aprendizaje. Esto permitió identificar talento prometedor que sería invisible en plataformas tradicionales.
- **Arquitectura Modular vs. Plataformas Monolíticas:** La arquitectura planteada permitió evolucionar funcionalidades críticas, como el motor de IA, de manera independiente, en contraste

con la rigidez de plataformas establecidas donde cambios disruptivos requieren modificaciones estructurales complejas.

- **Modelo de Reputación Académico–Profesional Híbrido:** Se desarrolló un sistema de evaluación que combina resultados tradicionales de proyectos con métricas de desarrollo de competencias, brindando a los estudiantes retroalimentación valiosa para su crecimiento profesional.
- **Estrategias de Despliegue y Contenerización:** GitHub Actions actúa como la herramienta principal de automatización. Cada cambio en el repositorio activa un pipeline que construye la imagen Docker del servicio y, si todo es exitoso, publica la imagen en un registry. Luego, GitHub Actions actualiza los archivos de despliegue de Kubernetes para apuntar a la nueva versión y aplica estos cambios en el clúster del servidor. Kubernetes crea o actualiza el pod correspondiente utilizando la imagen recién generada y reemplaza los pods antiguos mediante *rolling updates*. Este proceso garantiza que cada cambio llegue al entorno de ejecución de forma consistente, automatizada y confiable.

IV ANÁLISIS DEL PROBLEMA

1 Requerimientos

1.1 Requerimientos Funcionales

Gestión de Identidad y Acceso El sistema implementó un módulo de autenticación robusto que permitió el registro diferenciado para estudiantes y contratistas. Los estudiantes pudieron registrarse verificando su condición académica, mientras que los contratistas proporcionaron información empresarial validada. Se incorporó autenticación mediante JWT con refresh tokens y un control de acceso basado en roles.

Perfiles Inteligentes con Procesamiento de CV Se desarrolló un sistema de gestión de perfiles que permitió a los estudiantes cargar sus hojas de vida en formato PDF. Estos documentos fueron procesados mediante técnicas de NLP, extrayendo competencias técnicas, experiencias relevantes y logros académicos. La información extraída fue vectorizada y almacenada para optimizar búsquedas semánticas, generando perfiles enriquecidos que superaron las limitaciones de los formularios tradicionales.

Motor de Matchmaking con IA El núcleo de inteligencia artificial implementó algoritmos de recomendación que analizaron diversas dimensiones de compatibilidad. La compatibilidad técnica se definió a partir del emparejamiento entre habilidades requeridas y competencias demostradas. La compatibilidad temporal se evaluó con base en la alineación entre la disponibilidad del estudiante y el cronograma del proyecto. La compatibilidad de experiencia consideró la adecuación entre la complejidad del proyecto y el nivel del estudiante. Finalmente, la compatibilidad semántica se estableció mediante análisis de similitud contextual entre descripciones de proyectos y perfiles.

Gestión Integral de Proyectos El sistema soportó el ciclo completo de vida de los proyectos, desde su publicación hasta la finalización y evaluación. Los contratistas pudieron crear proyectos con descripciones detalladas, presupuestos, líneas de tiempo y habilidades requeridas. Los estudiantes, por su parte, pudieron postularse mediante mensajes personalizados y realizar seguimiento al estado de sus aplicaciones.

Sistema de Comunicación en Tiempo Real Se implementó un módulo de mensajería asíncrona acompañado de notificaciones push que facilitaron la comunicación en todas las fases del proyecto. El sistema mantuvo un historial completo de conversaciones, contextualizado según cada proyecto.

1.2 Requerimientos No Funcionales Críticos

Rendimiento y Escalabilidad El sistema fue diseñado para soportar hasta 5,000 usuarios concurrentes, con capacidad de escalar a 80,000 sin degradación perceptible del rendimiento. El 95 % de las páginas cargaron en menos de tres segundos y el motor de recomendación mantuvo tiempos de respuesta inferiores a cinco segundos incluso bajo carga máxima.

Disponibilidad y Confiabilidad Se garantizó una disponibilidad del 99.9 % mediante ventanas de mantenimiento predictivas. El sistema fue capaz de procesar hasta 1,000 transacciones por minuto y las operaciones críticas se ejecutaron con garantías de integridad mediante transacciones ACID.

Seguridad y Privacidad Toda la información sensible fue cifrada tanto en tránsito (TLS 1.3) como en reposo (AES-256). Las contraseñas se almacenaron aplicando hashing con bcrypt y se implementaron controles de acceso granulares. El diseño cumplió con los principios de privacidad por diseño definidos por el GDPR.

2 Restricciones

2.1 Restricciones Técnicas

Stack Tecnológico Definido El proyecto estableció un stack tecnológico compuesto por Angular 15 con TypeScript 4.8 y Material Design en el frontend; Java 17 con Spring Boot 3 y Spring Security en el backend; PostgreSQL 15 con la extensión PgVector como base de datos; un microservicio de IA construido en Python 3.11 con FastAPI; y un módulo de mensajería basado en Node.js con NestJS y WebSockets. El despliegue se realizó mediante GitHub Actions y Kubernetes.

Arquitectura de Microservicios Cada funcionalidad principal se implementó como un microservicio independiente, con su propio ciclo de vida, base de datos y mecanismo de despliegue. Esta arquitectura impuso patrones de comunicación asíncrona y un modelo de consistencia eventual.

Limitaciones de Integración con IA El proyecto hizo uso de modelos preentrenados y APIs existentes, en lugar de desarrollar modelos desde cero, con el fin de equilibrar capacidades avanzadas con los plazos de desarrollo disponibles.

2.2 Restricciones Operacionales

Plazo de Desarrollo Todas las fases del proyecto se completaron en un plazo máximo de seis meses, divididos en iteraciones de dos semanas con entregables incrementales.

Presupuesto y Recursos El desarrollo se apoyó exclusivamente en herramientas de código abierto y en servicios cloud bajo planes gratuitos, lo que limitó parcialmente las capacidades de infraestructura.

Cobertura Geográfica Inicial La primera versión del sistema se limitó al mercado colombiano, con soporte exclusivo en español y uso de moneda local (COP).

2.3 Restricciones de Negocio

La plataforma no incluyó funcionalidades de procesamiento de pagos, delegando las transacciones a sistemas externos. Se adoptó un modelo freemium en el cual las funcionalidades básicas fueron gratuitas y las avanzadas se ofrecieron mediante planes premium. Finalmente, la plataforma se orientó exclusivamente a estudiantes universitarios y egresados recientes, excluyendo funcionalidades dirigidas a profesionales con experiencia.

3 Especificación Funcional

Para la especificación de las funcionalidades de la aplicación se usa el modelo de historias de usuario. Cada historia describe una funcionalidad clave desde la perspectiva de los distintos tipos de usuario, definiendo su objetivo y propósito dentro del sistema para garantizar una experiencia eficiente y coherente en la plataforma.

Cuadro 2: Especificación Funcional por Historias de Usuario

ID	Nombre	Usuario	Quiero	Para
HU01	Inicio sesión	Estudiante freelance o contratista	Poder iniciar sesión utilizando mis credenciales	Acceder a mi cuenta y funcionalidades.
HU02	Registro estudiante	Estudiante freelance	Registrarme ingresando mis datos personales, académicos y de contacto	Crear mi perfil y comenzar a postularme.
HU03	Registro contratista	Unidad productiva contratante	Registrarme proporcionando información básica y validación	Crear mi cuenta como contratista.
HU04	Emparejamiento con IA	Unidad productiva contratante	Recibir sugerencias automáticas de perfiles adecuados	Acelerar la selección de candidatos.
HU05	Perfil de estudiante	Estudiante freelance	Crear y editar un perfil con mis habilidades y experiencia	Presentarme profesionalmente.
HU06	Búsqueda de oferta	Estudiante freelance	Buscar ofertas filtrando por varios criterios	Encontrar oportunidades relevantes.
HU07	Búsqueda de perfiles	Unidad productiva contratante	Buscar estudiantes filtrando por habilidades o estudios	Identificar candidatos relevantes.
HU08	Gestión de oferta	Unidad productiva contratante	Crear, editar y eliminar ofertas	Mantener actualizada mi oferta.
HU09	Comunicación con el contratista	Estudiante freelance	Enviar mensajes privados al contratista	Coordinar tareas y resolver dudas.
HU10	Comunicación con el estudiante	Unidad productiva contratante	Enviar mensajes directos a estudiantes	Coordinar avances y dar retroalimentación.
HU11	Gestión de postulantes	Unidad productiva contratante	Ver, comparar, aceptar o rechazar postulaciones	Seleccionar al mejor candidato.
HU12	Postulación	Estudiante freelance	Postularme con mi perfil y un mensaje	Aumentar mis posibilidades de contratación.
HU13	Calificación de estudiante	Unidad productiva contratante	Calificar al estudiante al finalizar el trabajo	Reconocer su desempeño.
HU14	Calificación de contratista	Estudiante freelance	Calificar al contratista	Compartir mi experiencia con otros estudiantes.

Cuadro 3: Criterios de Aceptación por Historia de Usuario

ID	Criterios de Aceptación	Prioridad
HU01	El formulario valida campos requeridos. Verifica que el correo y contraseña correspondan a un usuario registrado. Redirige al panel de usuario.	Media
HU02	Todos los campos obligatorios son validados. El correo no debe estar registrado previamente. El usuario queda registrado y es redirigido a su perfil.	Media
HU03	Se validan los campos obligatorios. El correo no debe estar registrado. El contratista puede acceder a su panel para publicar ofertas.	Media
HU04	La API de emparejamiento se invoca correctamente. Se recibe y muestra una lista de estudiantes sugeridos según criterios del proyecto.	Alta
HU05	El perfil puede visualizarse y editarse. La información se guarda correctamente. Se valida que los campos sean coherentes y estén completos.	Alta
HU06	Se pueden aplicar filtros y la lista de resultados se actualiza correctamente según los criterios seleccionados.	Media
HU07	Se aplican filtros y la lista de resultados muestra perfiles que cumplen los criterios seleccionados.	Media
HU08	Las ofertas pueden crearse, editarse y eliminarse. Los cambios se reflejan en tiempo real y se propagan a funcionalidades relacionadas.	Media
HU09	Se puede iniciar un chat desde la oferta o perfil. El mensaje llega al contratista y se mantiene el historial.	Baja
HU10	El contratista puede enviar mensajes. El estudiante recibe notificaciones y puede responder.	Baja
HU11	El contratista puede ver postulaciones, acceder a perfiles y marcar decisiones (aceptar/rechazar).	Media
HU12	El sistema permite enviar postulaciones válidas. Se notifica al contratista y se refleja en la sección de postulantes.	Media
HU13	El contratista puede dejar calificación numérica y comentario. Estos se asocian al perfil del estudiante y son visibles.	Baja
HU14	El estudiante puede dejar calificación numérica y comentario. Estos se muestran en el perfil del contratista.	Baja

V DISEÑO DE LA SOLUCIÓN

1 Pruebas de Concepto

La fase de Pruebas de Concepto tuvo como objetivo determinar cuál enfoque de inteligencia artificial ofrecía el mejor desempeño para el módulo de matchmaking de la plataforma. Se evaluaron tres implementaciones independientes: Embeddings puros, Clasificación con Filtros y RAG bajo un protocolo controlado que midió precisión funcional, consistencia de ranking y rendimiento bajo carga.

1.1 Implementaciones Evaluadas

Embeddings El enfoque de embeddings se basa en representar contenido textual, como hojas de vida, perfiles profesionales, descripciones de proyectos y requerimientos, mediante vectores numéricos de alta dimensionalidad generados por modelos de lenguaje. Estos vectores codifican relaciones semánticas entre palabras, frases y documentos completos, permitiendo capturar significados, similitudes conceptuales y relaciones implícitas. En un sistema de recomendación, esto permite comparar perfiles y ofertas por la cercanía de sus representaciones vectoriales, reflejando cuán apropiado es un candidato para un proyecto particular. El modelo interpreta conceptos equivalentes, sinónimos y habilidades relacionadas, convirtiéndose en una forma de medir afinidad de manera más humana y contextual que los enfoques tradicionales basados en palabras clave.

Ventajas

- Captura relaciones semánticas profundas entre habilidades, experiencias y requisitos.
- Permite reconocer equivalencias y sinónimos sin requerir reglas explícitas.
- Produce rankings de candidatos altamente coherentes con la pertinencia real.
- Es un enfoque robusto ante variaciones en el vocabulario o estilos de redacción.

Limitaciones

- El cálculo y comparación de vectores puede ser computacionalmente costoso.
- La latencia aumenta a medida que crece el número de perfiles y consultas.
- Requiere mecanismos especializados (indexación vectorial, caching) para escalar.
- No incorpora razonamiento explícito ni explicaciones generadas.

Clasificación Previa + Filtros El enfoque de clasificación previa con filtros estructura la información mediante categorías explícitas, asignando cada perfil a un rol, nivel de experiencia o área profesional, y posteriormente aplicando filtros rígidos basados en condiciones determinísticas. Este tipo de enfoque funciona mediante reglas, etiquetas y umbrales definidos por criterios específicos (como tecnologías, años de experiencia, ubicación o disponibilidad), y determina si un perfil cumple o no con cada condición. Es una aproximación que organiza la información antes de cualquier análisis semántico.

Ventajas

- Se comporta de manera predecible y transparente, lo que facilita auditorías y explicaciones del proceso.
- Requiere pocos recursos computacionales, siendo muy adecuado para escenarios de baja capacidad.
- Permite controlar explícitamente qué candidatos pasan o no pasan ciertos criterios.
- Reduce el espacio de búsqueda, pudiendo servir como etapa preliminar para otros métodos más complejos.

Limitaciones

- Depende estrictamente de la precisión de las reglas y etiquetas, que pueden ser incompletas o demasiado rígidas.
- No reconoce equivalencias conceptuales ni similitudes implícitas entre habilidades.
- Es poco flexible frente a perfiles híbridos o nuevos roles que no se ajustan a los filtros definidos.
- Tiende a producir rankings poco representativos de la pertinencia real cuando la información está desestructurada o ambigua.

RAG — Retrieval-Augmented Generation El enfoque RAG combina técnicas de recuperación semántica con modelos generativos de lenguaje. Su funcionamiento parte de identificar, mediante embeddings u otros mecanismos de búsqueda contextual, los documentos más relevantes asociados a una consulta. Posteriormente, un modelo generativo procesa tanto la consulta como los documentos recuperados para producir una respuesta enriquecida que integra información contextual, justificaciones y razonamientos. En un sistema de recomendación, RAG no solo determina qué perfiles son los más adecuados, sino que también explica por qué, sintetizando evidencia textual y contextualizando la recomendación. Se trata de un enfoque híbrido que combina la precisión de la recuperación con la expresividad del razonamiento generado.

Ventajas

- Produce recomendaciones explicadas, mostrando cómo se relacionan los perfiles con los requerimientos.
- Permite actualizar la base de conocimiento sin reentrenar el modelo generativo.
- Maneja bien información no estructurada, adaptándose a distintos tipos de perfiles.
- Escala mejor que los embeddings puros, gracias al uso de almacenamiento optimizado para búsquedas.

Limitaciones

- La precisión del razonamiento depende de la calidad de la información recuperada.
- Puede generar respuestas correctas en forma pero menos rigurosas en términos de ordenamiento.
- Introduce costos y complejidad asociados al uso de modelos generativos y manejo de contexto.
- Puede verse limitado por restricciones de tokens, afectando la amplitud del análisis cuando existe mucha información relevante.

1.2 Metodología Evaluación

Para comparar de manera objetiva los tres enfoques considerados se definió una metodología en cuatro etapas, documentada en el diseño de la solución, cuyo propósito fue asegurar que todas las implementaciones fueran evaluadas bajo condiciones idénticas y que los resultados fueran comparables entre sí.

La primera etapa consistió en la definición de escenarios de búsqueda que representaran requerimientos reales, variados y con diferentes grados de ambigüedad. Se establecieron cinco escenarios, cada uno centrado en un tipo distinto de necesidad profesional —por ejemplo, desarrollo backend, diseño UI/UX, análisis de datos, QA y perfiles híbridos— con el fin de evaluar la capacidad de cada enfoque de adaptarse a diferentes dominios y niveles de complejidad.

Posteriormente se construyó un conjunto de diez perfiles profesionales, cada uno redactado con un estilo distinto y con combinaciones específicas de habilidades, experiencia técnica, proyectos y formación. Los perfiles fueron deliberadamente heterogéneos, incluyendo candidatos altamente especializados, perfiles híbridos, perfiles generalistas y perfiles con términos poco estándar. Esto permitió evaluar la sensibilidad de cada enfoque ante ambigüedad, variabilidad lingüística y señales implícitas.

Se estableció para cada escenario el ranking ideal de los cinco candidatos más adecuados. Esta tabla funcionó como punto de referencia, permitiendo comparar objetivamente los rankings generados por cada implementación. La tabla no solo indicaba qué candidatos eran los más apropiados, sino también el orden correcto de relevancia, lo que permitió evaluar el nivel de precisión y fidelidad de cada enfoque a la expectativa humana.

Finalmente, cada enfoque fue probado tres veces en cada escenario, para un total de 45 ejecuciones. Los resultados se registraron incluyendo el ranking producido, el nivel de coincidencia con la tabla de referencia, la dispersión del orden, los aciertos y desaciertos, así como métricas operativas de latencia por consulta y consistencia entre repeticiones.

Además de esta evaluación, se realizaron pruebas de carga independientes con 10, 50 y 200 usuarios concurrentes para medir estabilidad y degradación del rendimiento.

1.3 Resultados Funcionales

Para la evaluación se usó una matriz de referencia que definía el orden ideal de candidatos para cada escenario de búsqueda. Esta tabla, construida mediante consenso de los miembros del equipo,

establecía la clasificación correcta esperada para cada consulta, asignando prioridades de mejor a peor candidato según su adecuación al requerimiento específico.

Pregunta/Perfil	Front	Back	Full Stack	UX/UI	DBA	Electrónico	Data scientist	administrador de empresas	Contador	Artista
1	10	8	9	7	7	5	5	3	2	1
2	6	10	7	6	10	3	6		2	1
3	9	7	8	10	5	2	5		2	1
4	8	7	8	6	6	10	6		2	1
5	3	3	3	3	6	3	3	8	10	1

Figura 1: Tabla Prioridades Clasificación POCs

Pregunta	Intento	Front	Back	Full Stack	UX/UI	DBA	Electrónico	Data scientist	administrador de empresas	Contador	Artista	Total	Promedio/pregunta	Promedio Total
1	1	10	9	7	8	6	3	5	4	1	2	389	389	337.4
	2	10	9	7	8	6	3	5	4	1	2	389		
	3	10	9	7	8	6	3	5	4	1	2	389		
2	1	6	8	1	5	10	2	9	4	7	3	338	338	338
	2	6	8	1	5	10	2	9	4	7	3	338		
	3	6	8	1	5	10	2	9	4	7	3	338		
3	1	7	6	3	8	9	2	10	5	4	1	327	327	327
	2	7	6	3	8	9	2	10	5	4	1	327		
	3	7	6	3	8	9	2	10	5	4	1	327		
4	1	7	9	2	6	4	10	5	3	8	1	348	348	348
	2	7	9	2	6	4	10	5	3	8	1	348		
	3	7	9	2	6	4	10	5	3	8	1	348		
5	1	3	4	1	9	8	5	7	6	10	2	285	285	285
	2	3	4	1	9	8	5	7	6	10	2	285		
	3	3	4	1	9	8	5	7	6	10	2	285		

Figura 2: Esquema Tabla Evaluativa por Concepto

El enfoque de embedding puros mostró un comportamiento superior al resto en términos de precisión semántica y calidad de ranking. En la mayoría de los escenarios, los embeddings reprodujeron el orden esperado o se aproximaron estrechamente a la tabla de referencia. El modelo capturó relaciones implícitas entre competencias, identificó habilidades equivalentes y reconoció pertinencia incluso cuando los candidatos no utilizaban exactamente la misma terminología que el escenario planteado. La consistencia entre ejecuciones también fue alta: los resultados variaron muy poco entre repeticiones, evidenciando estabilidad en el proceso de similitud semántica. Los escenarios con mayor ambigüedad, como perfiles híbridos, reforzaron esta ventaja, ya que embeddings logró priorizar correctamente candidatos con mezcla de roles.

Pregunta	Intento	Front	Back	Full Stack	UX/UI	DBA	Electrónico	Data scientist	administrador de empresas	Contador	Artista	Total	Promedio/pregunta	Promedio Total
1	1	10	9	7	8	6	3	5	4	1	2	389	389	337.4
	2	10	9	7	8	6	3	5	4	1	2	389		
	3	10	9	7	8	6	3	5	4	1	2	389		
2	1	6	8	1	5	10	2	9	4	7	3	338	338	338
	2	6	8	1	5	10	2	9	4	7	3	338		
	3	6	8	1	5	10	2	9	4	7	3	338		
3	1	7	6	3	8	9	2	10	5	4	1	327	327	327
	2	7	6	3	8	9	2	10	5	4	1	327		
	3	7	6	3	8	9	2	10	5	4	1	327		
4	1	7	9	2	6	4	10	5	3	8	1	348	348	348
	2	7	9	2	6	4	10	5	3	8	1	348		
	3	7	9	2	6	4	10	5	3	8	1	348		
5	1	3	4	1	9	8	5	7	6	10	2	285	285	285
	2	3	4	1	9	8	5	7	6	10	2	285		
	3	3	4	1	9	8	5	7	6	10	2	285		

Figura 3: Resultados Implementación Embeddings

La estrategia de Clasificación + Filtros presentó el rendimiento funcional más limitado. Aunque siempre devolvió resultados coherentes con las reglas establecidas, los rankings producidos carecían de sensibilidad contextual. Varios candidatos relevantes fueron descartados por filtros estrictos, por ejemplo, la ausencia literal de una tecnología que sí era sustituible por experiencia equivalente, mientras que otros candidatos menos apropiados fueron promovidos debido a coincidencias superficiales. La comparación con la referencia manual mostró desviaciones amplias en el orden, especialmente en escenarios donde los perfiles eran complejos o multidimensionales. Además, este enfoque fue el que mostró menor adaptabilidad a perfiles redactados de manera no estándar.

Pregunta	intento	Front	Back	Full Stack	UX/UI	DBA	Electrónico	Data scientist	administrador de empresas	Contador	Artista	Total	Promedio/pregunta	Promedio Total
1	1	10	7	2	6	5	8	9	3	4	1	354	309.66666667	272.2
	2	10	7	9	6	5	3	2	8	4	1	255		
	3	9	2	7	6	1	4	8	3	5	10	320		
2	1	2	5	6	3	9	1	4	10	8	7	301	277	283
	2	2	8	5	3	9	1	4	10	7	6	281		
	3	3	7	4	5	9	6	10	8	1	2	249		
3	1	6	2	8	5	3	1	4	10	9	7	297	294.66666667	196.66666667
	2	6	2	8	5	3	1	4	10	9	7	297		
	3	6	2	4	9	7	5	10	8	1	3	255		
4	1	6	7	2	5	4	10	8	9	1	3	278	294.66666667	196.66666667
	2	6	7	2	5	4	10	8	9	1	3	278		
	3	1	5	2	6	3	10	7	4	8	9	328		
5	1	3	1	4	5	7	2	8	9	10	6	184	294.66666667	196.66666667
	2	3	1	4	5	7	2	8	9	10	6	184		
	3	2	4	1	7	3	6	8	5	10	9	222		

Figura 4: Resultados Implementación Filtros

El enfoque RAG logró resultados intermedios. En la fase de recuperación, identificó adecuadamente candidatos relevantes en la mayoría de los escenarios; sin embargo, la fase generativa tendió a reorganizar los elementos en función de la narrativa de la respuesta más que en función de una jerarquización estricta por similitud. Como consecuencia, algunos rankings mostraron variaciones significativas respecto al orden ideal, pese a que los candidatos recuperados eran en general correctos. El modelo, no obstante, exhibió una ventaja cualitativa notable: generó justificaciones textuales claras y bien articuladas, lo que permitió comprender las decisiones tomadas. Este aspecto lo posicionó como una alternativa válida para casos en los que la justificación sea prioritaria, aunque no fuera el más fuerte para precisión bruta de ranking.

Pregunta	intento	Front	Back	Full Stack	UX/UI	DBA	Electrónico	Data scientist	administrador de empresas	Contador	Artista	Total	Promedio/pregunta	Promedio Total
1	1	10	5	9	4	7	2	3	8	6	1	360	360	319,8
	2	10	5	9	4	7	2	3	8	6	1	360		
	3	10	5	9	4	7	2	3	8	6	1	360		
2	1	4	10	8	3	6	1	2	9	5	7	308	308	299
	2	4	10	8	3	6	1	2	9	5	7	308		
	3	4	10	8	3	6	1	2	9	5	7	308		
3	1	9	4	8	6	3	1	2	10	7	5	299	299	342
	2	9	4	8	6	3	1	2	10	7	5	299		
	3	9	4	8	6	3	1	2	10	7	5	299		
4	1	7	9	5	4	3	10	2	8	6	1	342	342	290
	2	7	9	5	4	3	10	2	8	6	1	342		
	3	7	9	5	4	3	10	2	8	6	1	342		
5	1	4	1	6	5	8	2	3	9	10	7	290	290	290
	2	4	1	6	5	8	2	3	9	10	7	290		
	3	4	1	6	5	8	2	3	9	10	7	290		

Figura 5: Resultados Implementación RAG

1.4 Resultados de Rendimiento

Las pruebas de rendimiento permitieron evaluar la viabilidad operativa de cada enfoque bajo condiciones de carga, evidenciando diferencias importantes.

El enfoque de Embeddings puros mostró la latencia más alta y la mayor degradación a medida que aumentaba el número de usuarios concurrentes. Esto se debió al cálculo intensivo de similitud vectorial, que se ve afectado conforme crece el número de perfiles comparados. En el escenario de 200 usuarios concurrentes, las consultas presentaron tiempos de respuesta aproximadamente el doble de los registrados por RAG. Aunque el sistema mantuvo estabilidad y no perdió solicitudes, la latencia acumulada demostró que, sin optimizaciones adicionales, el enfoque no es sostenible a gran escala.

El enfoque de Clasificación + Filtros mostró el mejor rendimiento en términos de velocidad. Su latencia fue mínima en todos los escenarios debido a la simplicidad de las operaciones, que se limitan a verificaciones lógicas y búsquedas determinísticas. Sin embargo, su rendimiento excepcional no compensó las deficiencias funcionales, por lo que su rol quedó relegado a posibles etapas preliminares o complementarias, pero no como mecanismo de recomendación principal.

El enfoque RAG demostró un rendimiento notablemente más eficiente que los embeddings puros. Dado que parte del trabajo está delegado a la capa de recuperación y no requiere múltiples comparaciones vectoriales en memoria, su latencia se mantuvo estable incluso bajo cargas elevadas. En el escenario más exigente (200 usuarios), la implementación RAG procesó las solicitudes con un tiempo total aproximadamente dos veces menor que el obtenido por el enfoque de embeddings. Esta eficiencia operativa, sumada a su consistencia, lo posiciona como un enfoque escalable y apto para escenarios de uso intensivo.

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/embeddings	1466	0	78000	80000	81000	73286.59	6084	82755	316	1	0
	Aggregated	1466	0	78000	80000	81000	73286.59	6084	82755	316	1	0
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/preguntar	2549	0	39000	49000	53000	38668.16	5332	66905	8799	5	0
	Aggregated	2549	0	39000	49000	53000	38668.16	5332	66905	8799	5	0

STATISTICS												
Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/preguntar	13123	0	3700	4400	4800	3639.98	75	6272	8893	38.5	0
	Aggregated	13123	0	3700	4400	4800	3639.98	75	6272	8893	38.5	0

Figura 6: Resultados Rendimiento POCs 200-10

2 Sobre el uso de DAMA

Se realizaron pruebas de concepto bajo el framework DAMA (Data Management Association) para determinar el mejor enfoque de inteligencia artificial para el módulo de matchmaking entre candidatos y proyectos. El objetivo principal fue definir la arquitectura de datos óptima que cumpliera con los principios fundamentales de calidad, escalabilidad y gobierno de datos, evaluando tres implementaciones distintas mediante un protocolo controlado que midió precisión funcional, consistencia de ranking y rendimiento bajo carga.

2.1 Fase 1: Gobierno de Datos (Data Governance)

En la fase de gobierno de datos se establecieron los estándares de calidad y políticas que rigieron todas las pruebas de concepto. Se definió una metodología basada en matrices de referencia con rankings ideales por escenario, donde los criterios de evaluación incluyeron precisión funcional versus referencia humana, consistencia entre ejecuciones y rendimiento bajo carga con simulaciones de 10, 50 y 200 usuarios concurrentes. Los artefactos generados comprendieron tablas de prioridades de clasificación, esquemas evaluativos estandarizados y protocolos de medición de métricas consistentes. Para garantizar la comparabilidad entre enfoques, se diseñaron 10 perfiles profesionales heterogéneos y 5 escenarios de uso reales que cubrían dominios como desarrollo backend, diseño UI/UX, análisis de datos, QA y perfiles híbridos, asegurando que todos los enfoques fueran evaluados con exactamente el mismo conjunto de datos bajo condiciones idénticas de prueba.

2.2 Fase 2: Arquitectura de Datos (Data Architecture)

En la arquitectura de datos se evaluaron tres implementaciones distintas con características técnicas y enfoques fundamentalmente diferentes. El enfoque de embeddings puros se basó en vectores numéricos de alta dimensionalidad generados por modelos de lenguaje para representación semántica, ofreciendo ventajas significativas en calidad de datos semántica al capturar relaciones implícitas y reconocer equivalencias conceptuales, aunque presentando limitaciones en costo computacional y desafíos de escalabilidad. El enfoque de clasificación con filtros implementó una arquitectura basada en reglas determinísticas y categorización explícita, utilizando filtros rígidos basados en condiciones específicas que proporcionaban transparencia y auditabilidad completa junto con bajo costo computacional, pero demostrando rigidez en el tratamiento de datos y poca adaptabilidad a variaciones semánticas. El enfoque RAG combinó recuperación semántica con generación contextual, integrando embeddings con modelos generativos para ofrecer explicabilidad completa de las recomendaciones y capacidad de actualización sin reentrenamiento, aunque introduciendo complejidad operacional y dependencia crítica de la calidad de la fase de recuperación inicial.

2.3 Fase 3: Desarrollo de Datos (Data Development)

El desarrollo de datos implementó una metodología de evaluación rigurosa que ejecutó 45 pruebas completas distribuidas en tres repeticiones por cada combinación de cinco escenarios y tres enfoques. Las variables se controlaron estrictamente mediante el uso de los mismos perfiles de entrada, idénticos escenarios de búsqueda y condiciones ambientales equivalentes para todas las ejecuciones. Las métricas de calidad incluyeron grado de coincidencia con el ranking de referencia manual, dispersión del orden entre ejecuciones, análisis detallado de aciertos y desaciertos, y medición precisa de latencia por consulta. Complementariamente, se realizaron pruebas de rendimiento escaladas en tres niveles de

carga con 10, 50 y 200 usuarios concurrentes, midiendo tiempos de respuesta, estabilidad del sistema, patrones de degradación del rendimiento y capacidades de escalado bajo condiciones operacionales realistas.

2.4 Fase 4: Operaciones de Datos (Data Operations)

En las operaciones de datos, los resultados funcionales revelaron patrones distintivos para cada enfoque evaluado. El enfoque de embeddings puros demostró calidad superior en precisión semántica, con alta consistencia entre ejecuciones y rankings que se aproximaban más estrechamente a la referencia humana, destacándose especialmente en la captura de relaciones implícitas y equivalencias conceptuales entre habilidades y experiencias. El enfoque de clasificación con filtros exhibió un rendimiento operacional excepcional en velocidad de procesamiento y transparencia completa del proceso decision-making, aunque su rigidez estructural resultó en baja adaptabilidad a variaciones semánticas y contextos complejos, limitando su aplicabilidad a etapas preliminares de filtrado más que a recomendaciones finales. El enfoque RAG logró un balance intermedio en precisión pero destacó en capacidad explicativa, generando justificaciones contextuales detalladas y demostrando mejor escalabilidad operacional que los embeddings puros, posicionándose como solución viable para casos que requieren transparencia y auditabilidad. Los resultados de rendimiento confirmaron que los embeddings presentaban la mayor latencia y degradación bajo carga, los filtros ofrecían el mejor rendimiento pero con limitaciones funcionales significativas, y RAG proporcionaba un balance eficiente entre capacidades funcionales y escalabilidad operacional, llevando a la conclusión fundamental de que ningún enfoque individual satisfacía completamente todos los requisitos del sistema.

2.5 Fase 5: Calidad de Datos (Data Quality)

La evaluación de calidad de datos implementó un conjunto comprehensivo de métricas que abarcaron múltiples dimensiones de performance. La precisión se midió mediante el grado de coincidencia con las referencias humanas establecidas, la consistencia mediante la variabilidad entre ejecuciones repetidas, el rendimiento mediante latencia y escalabilidad bajo carga, y la explicabilidad mediante la capacidad de justificar recomendaciones de manera comprensible. Los hallazgos cualitativos demostraron que los embeddings ofrecían la máxima calidad semántica pero con costo operacional elevado, los filtros proporcionaban calidad predictible pero fundamentalmente limitada, y RAG entregaba calidad explicativa superior con precisión intermedia, estableciendo claros trade-offs entre los diferentes atributos de calidad que informaron la decisión arquitectónica final.

2.6 Fase 6: Decisión Arquitectónica

La decisión arquitectónica final seleccionó una solución híbrida basada en embeddings con búsqueda vectorial optimizada mediante pgvector, justificada bajo el framework DAMA por su capacidad de combinar las fortalezas de múltiples enfoques mientras mitiga sus debilidades individuales. Desde la perspectiva de gobierno, esta arquitectura integra los mejores aspectos de cada enfoque; en términos arquitectónicos, optimiza la calidad semántica con eficiencia operacional; respecto a calidad, mantiene la precisión superior de los embeddings mientras mejora sustancialmente el rendimiento; y en operaciones, demuestra escalabilidad robusta mediante mecanismos de búsqueda vectorial especializados. Los componentes de la solución final integran embeddings para garantizar calidad semántica, pgvector para eficiencia en búsquedas masivas, arquitectura escalable para operaciones en producción, y

mecanismos de optimización de rendimiento que abordan las limitaciones identificadas durante las pruebas.

2.7 Lecciones Aprendidas DAMA

Las lecciones aprendidas abarcaron todas las dimensiones del framework DAMA, comenzando con el gobierno de datos donde se destacó la importancia crítica de matrices de referencia estandarizadas, la necesidad de múltiples métricas de evaluación complementarias y el valor fundamental de la consistencia en datos de prueba para garantizar comparabilidad. En arquitectura, se demostró que no existe solución única para todos los requisitos, que el balance entre calidad y rendimiento resulta crítico para la viabilidad operacional, y que la escalabilidad debe considerarse desde las etapas iniciales de diseño. Las operaciones revelaron que las pruebas de carga son esenciales para decisiones arquitectónicas informadas, que la latencia impacta directamente la viabilidad operacional, y que la transparencia conlleva costos computacionales significativos. Finalmente, en calidad se estableció que la precisión semántica requiere recursos computacionales sustanciales, que la explicabilidad agrega valor tangible pero introduce complejidad, y que el contexto específico de uso determina los requisitos de calidad prioritarios para cada implementación.

2.8 Conclusión

Las pruebas de concepto demostraron que ninguno de los enfoques evaluados satisface por sí solo todos los requisitos del sistema. Embeddings puros ofrecieron la mejor precisión y el ranking más cercano a la referencia humana, pero su latencia aumentó significativamente bajo carga. Clasificación + Filtros mostró un rendimiento excelente y gran estabilidad, aunque con una precisión insuficiente debido a su rigidez y baja sensibilidad semántica. RAG logró un equilibrio entre velocidad y capacidad explicativa, pero su ordenamiento no alcanzó la consistencia obtenida con embeddings.

A partir de estos resultados, se concluyó que la solución óptima debía combinar la capacidad semántica de los embeddings con la eficiencia operativa observada en las técnicas de recuperación. Por ello, se seleccionó una arquitectura híbrida basada en embeddings con búsqueda vectorial en pgvector, capaz de ofrecer recomendaciones precisas, rápidas y escalables.

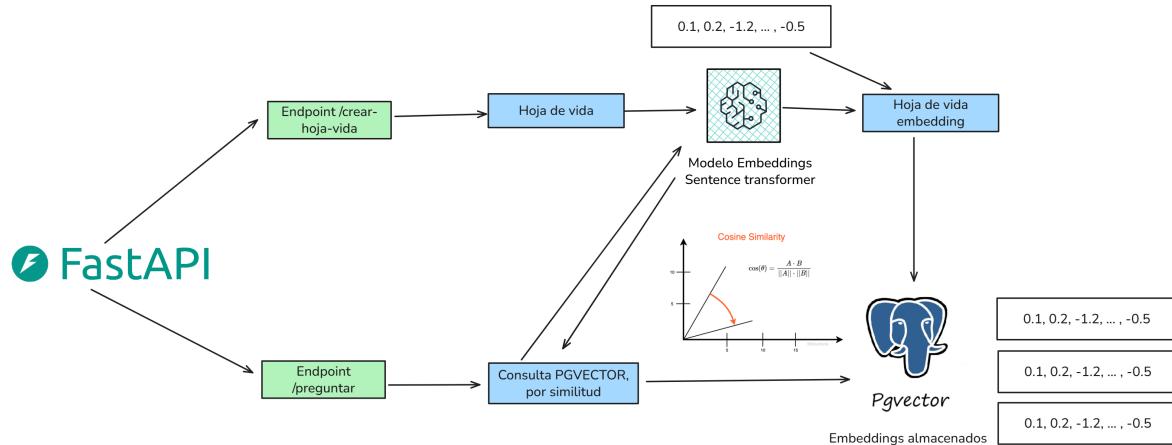
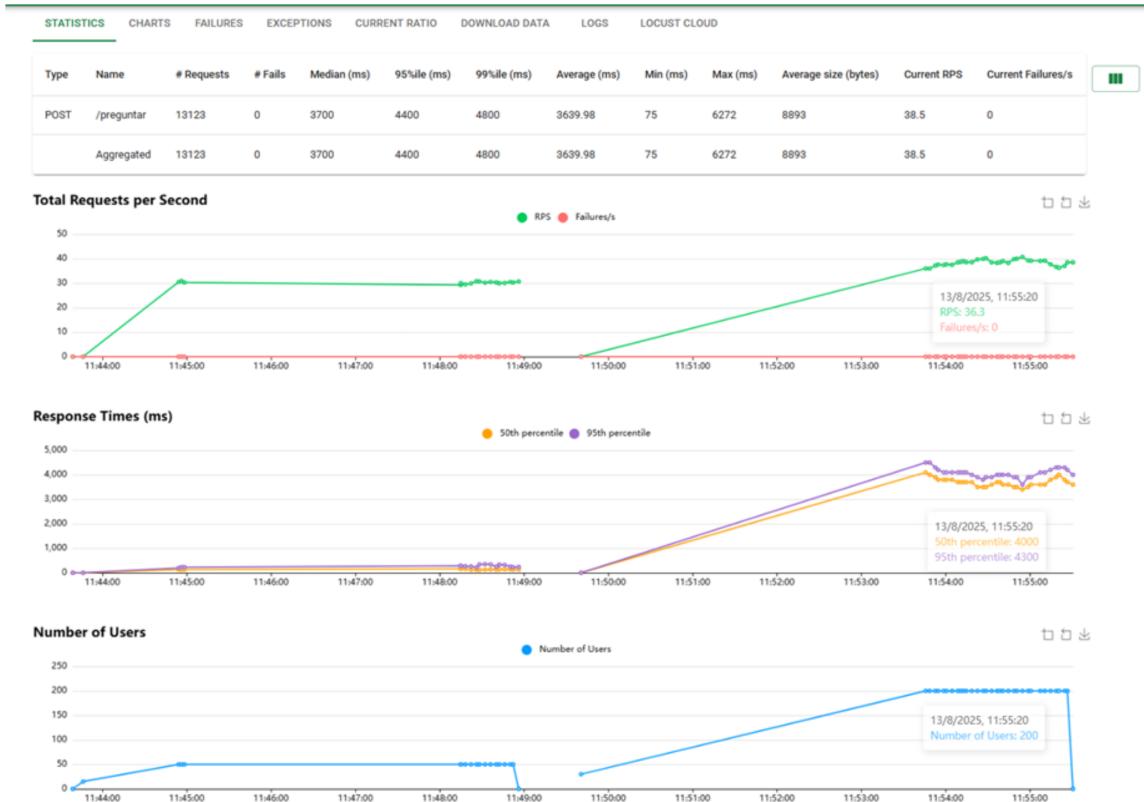


Figura 7: Implementación Solución Híbrida



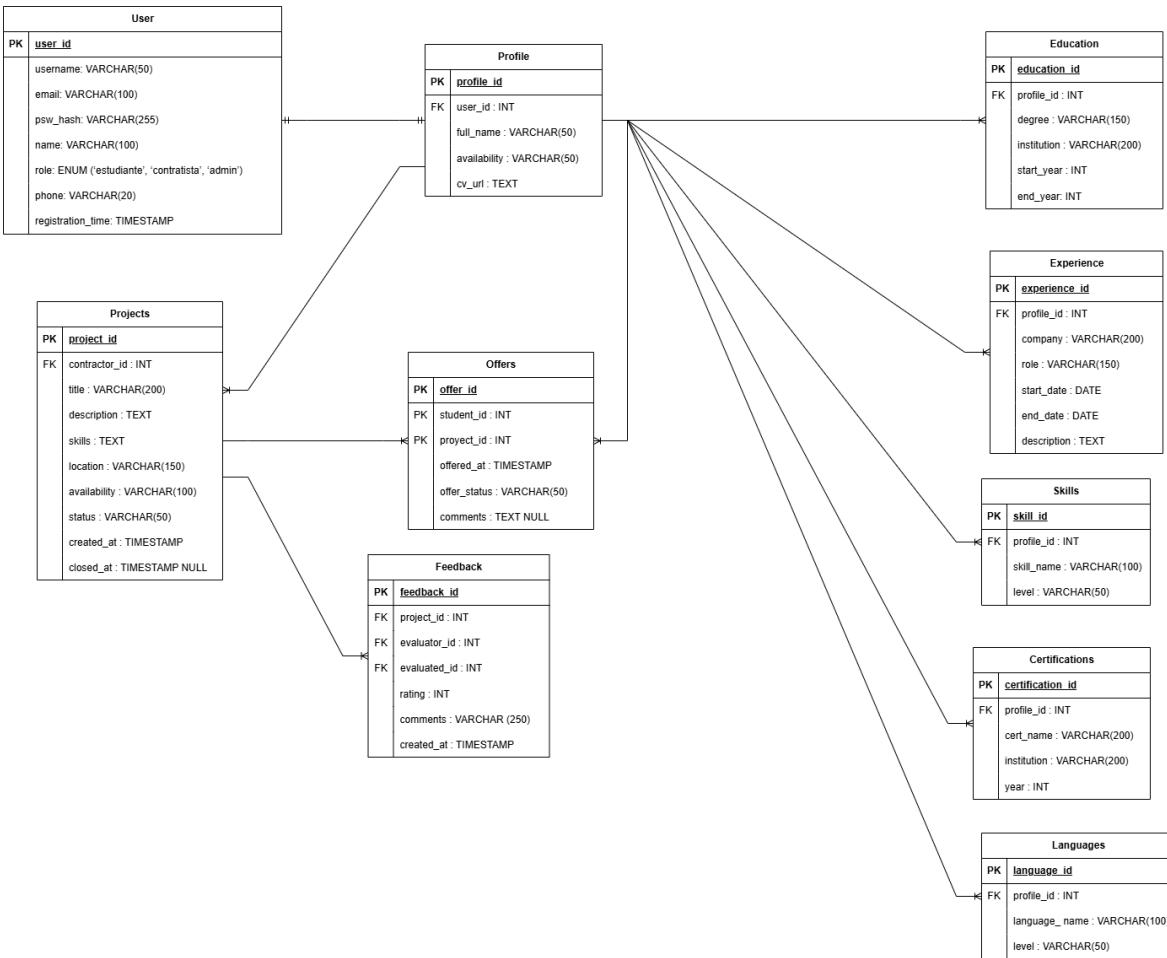


Figura 9: Modelo Entidad Relación

3 Diseño base de datos

El diseño sigue un modelo entidad–relación (MER) normalizado en tercera forma normal (3FN), garantizando consistencia, integridad, modularidad y facilidad de mantenimiento.

3.1 Objetivos del Diseño

- Organizar la información de usuarios y sus perfiles profesionales.
- Gestionar proyectos publicados por contratistas.
- Administrar ofertas o postulaciones de los estudiantes.
- Implementar un sistema de reputación basado en evaluaciones.

- Mantener integridad referencial entre todas las entidades.
- Asegurar escalabilidad mediante un modelo modular.

3.2 Descripción de Entidades

User La entidad User gestiona a los usuarios registrados. Incluye credenciales, información personal y el rol asignado. Contiene los atributos:

- **user_id** (PK)
- username, email, psw_hash, name, phone
- role (estudiante, contratista, admin)
- registration_time

Cada usuario posee un único perfil extendido, modelado mediante una relación uno a uno con **Profile**.

Profile La entidad Profile extiende la información del usuario con atributos profesionales:

- **profile_id** (PK)
- user_id (FK)
- full_name, availability, cv_url

Es la entidad central desde la cual se relacionan múltiples componentes académicos y profesionales.

Education Asocia información académica al perfil mediante una relación 1:N. Sus atributos son:

- **education_id** (PK)
- profile_id (FK)
- degree, institution, start_year, end_year

Experience Registra la trayectoria laboral del usuario:

- **experience_id** (PK)
- profile_id (FK)
- company, role, start_date, end_date, description

Skills La entidad Skills almacena habilidades específicas asociadas a un perfil:

- **skill_id** (PK)
- profile_id (FK)
- skill_name, level

Certifications Almacena certificaciones profesionales:

- **certification_id** (PK)
- **profile_id** (FK)
- cert_name, institution, year

Languages Registra los idiomas que domina el freelancer:

- **language_id** (PK)
- **profile_id** (FK)
- language_name, level

Projects Representa los proyectos creados por contratistas. Sus atributos son:

- **project_id** (PK)
- contractor_id (FK → User)
- title, description, skills, location
- availability, status
- created_at, closed_at

Un usuario contratista puede crear múltiples proyectos.

Offers Representa las postulaciones de los freelancers a proyectos:

- **offer_id** (PK)
- student_id (FK → User)
- project_id (FK)
- offered_at, offer_status, comments

Es la tabla puente que establece una relación muchos a muchos entre usuarios (freelancers) y proyectos.

Feedback La entidad Feedback almacena evaluaciones realizadas entre usuarios:

- **feedback_id** (PK)
- project_id (FK)
- evaluator_id (FK → User)
- evaluated_id (FK → User)
- rating, comments, created_at

Este diseño permite evaluaciones bidireccionales entre estudiante y contratista.

Relaciones del Sistema Las relaciones clave del sistema son:

- User 1 — 1 Profile
- Profile 1 — N Education
- Profile 1 — N Experience
- Profile 1 — N Skills
- Profile 1 — N Certifications
- Profile 1 — N Languages
- User 1 — N Projects
- User N — N Projects (mediante Offers)
- User N — N User (mediante Feedback)

Este diseño relacional está optimizado para normalización, rendimiento y modularidad.

3.3 Justificación del Diseño

- El modelo está normalizado en 3FN, evitando redundancia.
- La modularidad del **Profile** permite agregar nuevos componentes sin alterar el núcleo.
- La tabla **Offers** gestiona correctamente la relación muchos a muchos entre freelancers y proyectos.
- La entidad **Feedback** permite un sistema de reputación confiable.
- La estructura promueve escalabilidad y consistencia referencial.

3.4 Conclusiones

El modelo de datos propuesto ofrece una estructura robusta, flexible y escalable para una plataforma freelance moderna. El uso de entidades independientes pero conectadas mediante relaciones bien definidas facilita la implementación de funcionalidades claves del sistema: perfiles completos, postulaciones, gestión de proyectos y evaluaciones.

El diseño cumple con los principios de normalización, integridad y modularidad, constituyendo una base sólida para la implementación de la plataforma.

4 Arquitectura basada en C4

Para describir la solución en diferentes niveles de abstracción se escogió el modelo de arquitectura C4, facilitando la comunicación entre distintos perfiles interesados. Propone cuatro vistas jerárquicas: Conceptual (qué es el sistema y con quién interactúa), Contenedores (cómo se organiza en aplicaciones, bases de datos o servicios), Componentes (cómo se estructuran internamente esos contenedores).

4.1 Contexto

Al ser el nivel más alto es el que ofrece un mayor nivel de abstracción, aquí se presentan los actores principales en el ambiente del sistema, tanto personas como sistemas externos en comunicación con el sistema de software.

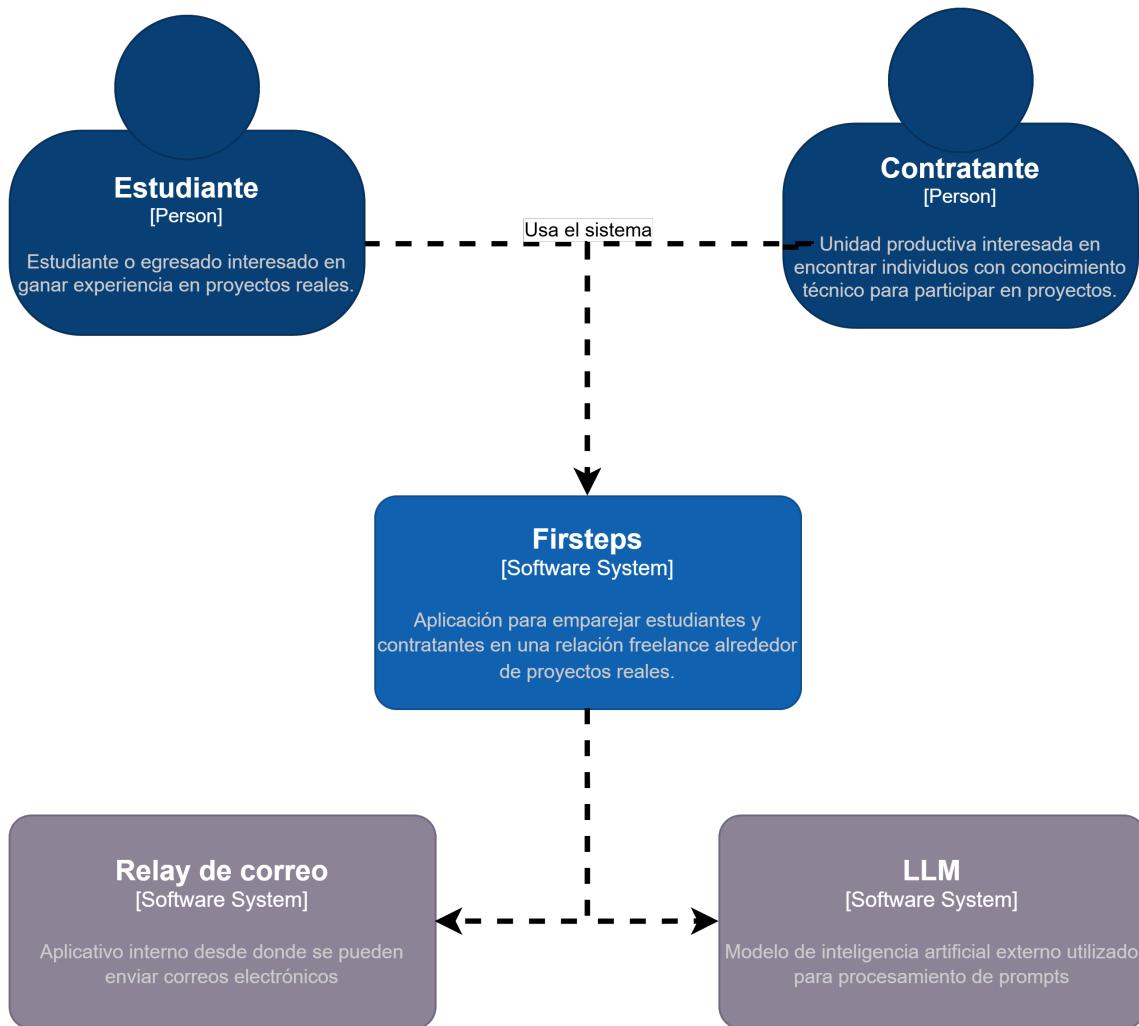


Figura 10: Diagrama Contexto - Arquitectura C4

Actores

Estudiante Es el usuario principal de la plataforma y corresponde a estudiantes o egresados interesados en adquirir experiencia práctica mediante la participación en proyectos reales. Su moti-

vación radica en fortalecer su perfil profesional, aplicar los conocimientos adquiridos en su formación académica y generar ingresos adicionales. El estudiante interactúa directamente con el sistema para postularse a iniciativas propuestas por los contratantes, recibir notificaciones y dar seguimiento al avance de sus proyectos.

Contratante Representa a una unidad productiva, empresa o profesional independiente que busca talento para el desarrollo de proyectos específicos. Su rol principal es proponer iniciativas, detallar los requisitos y seleccionar a los estudiantes que cumplan con las competencias necesarias para llevarlas a cabo. El contratante utiliza la plataforma como un medio confiable para acceder a perfiles calificados y dar seguimiento a la ejecución de las actividades, evaluando el desempeño de los participantes y generando un círculo de colaboración que beneficia a ambas partes.

Sistemas Externos

Relay de correo Corresponde al servidor de correo electrónico que funciona como intermediario entre la aplicación y los buzones de los destinatarios. Este servicio SMTP es fundamental para garantizar la entrega confiable de los mensajes, ya que maneja la autenticación, el enrutamiento y la seguridad en la transmisión. Dentro del sistema, el relay es utilizado para enviar notificaciones, confirmaciones de registro, actualizaciones sobre proyectos y cualquier otro tipo de comunicación automatizada que deba llegar a los usuarios finales.

LLM Se trata de un modelo de inteligencia artificial avanzado, diseñado para procesar lenguaje natural y generar respuestas contextuales a partir de prompts. Este componente externo se integra en la plataforma como una herramienta de apoyo para procesamiento de texto y categorización de información.

4.2 Capa de Contenedores

En este nivel se detallan los elementos y aplicaciones del sistema que realizan operaciones, procesan y almacenan información, en general los que cumplan funcionalidades importantes del sistema, separados por responsabilidades.

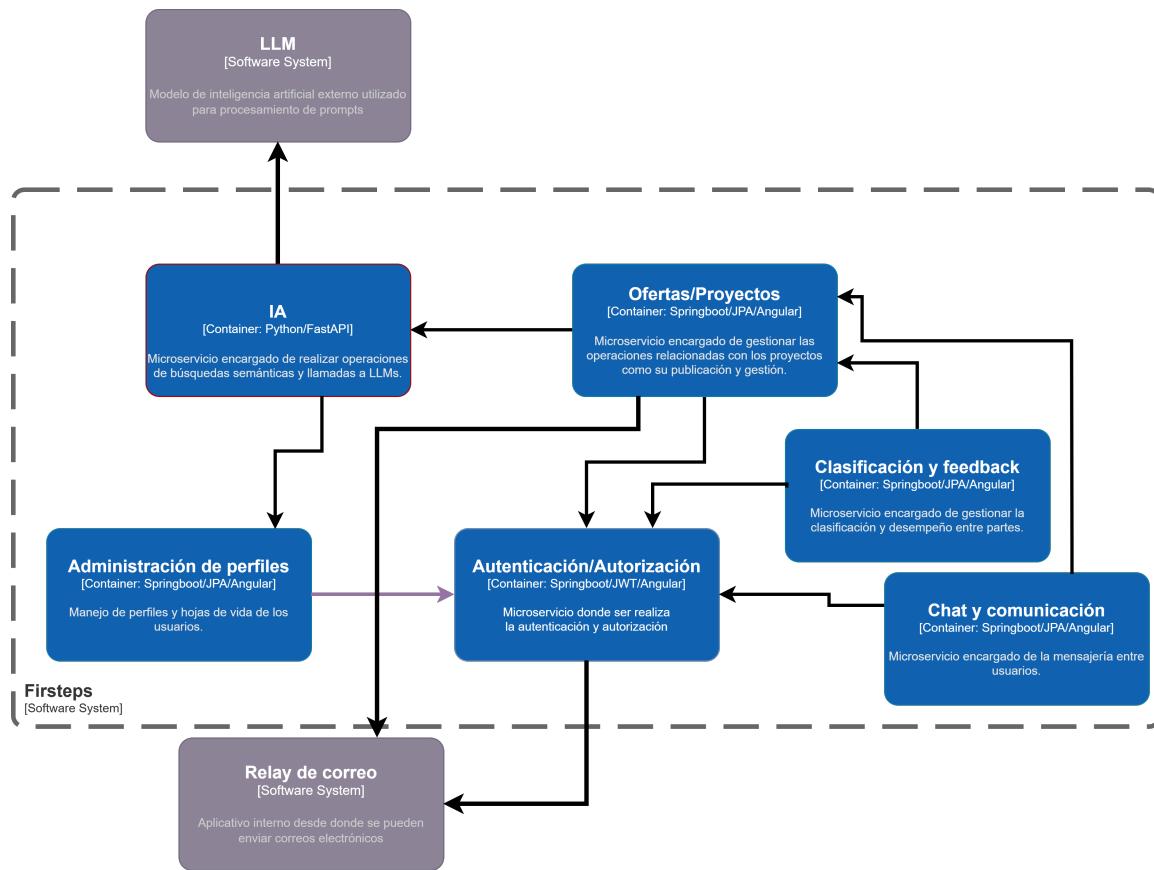


Figura 11: Diagrama Contenedores - Arquitectura C4

Autenticación y Autorización Este contenedor implementa los mecanismos de control de acceso a la plataforma, garantizando que solo usuarios autenticados puedan utilizar los servicios disponibles. Integra el uso de tokens JWT para manejar sesiones seguras y asegura que cada usuario tenga los permisos adecuados según su rol (estudiante, contratante o administrador). Su importancia radica en que constituye la primera capa de seguridad del sistema, evitando accesos no autorizados y protegiendo la información sensible que se almacena en la aplicación.

Administración de Perfiles Este contenedor se encarga de centralizar la gestión de perfiles de usuario y las hojas de vida que los estudiantes cargan en la plataforma. Su propósito es mantener organizada la información académica y profesional de cada participante, permitiendo al contratante acceder fácilmente a sus competencias y experiencias previas. A través de esta funcionalidad, el sistema facilita que los estudiantes actualicen sus datos de forma sencilla y que los contratantes cuenten con criterios claros al momento de seleccionar candidatos para sus proyectos.

Ofertas y Proyectos Este microservicio administra todo el ciclo de vida de los proyectos publicados por los contratantes. Incluye la creación, edición, búsqueda y gestión de ofertas, además de permitir que los estudiantes se postulen a las iniciativas de su interés. Su diseño está orientado a garantizar que los contratantes tengan un control adecuado sobre sus publicaciones y que los estudiantes encuentren oportunidades acordes a su perfil.

Matchmaking IA Este microservicio implementa la lógica de recomendación inteligente que conecta las peticiones de contratistas con los perfiles de los estudiantes. Su función principal es procesar solicitudes, analizar los datos de las hojas de vida y generar coincidencias que se ajusten a las competencias requeridas. Gracias a este motor de búsqueda semántica e inteligencia artificial, el sistema mejora la experiencia de los contratantes, reduciendo el tiempo de selección y aumentando la probabilidad de encontrar candidatos altamente compatibles con los requerimientos de cada iniciativa.

Del mismo modo, para los estudiantes este microservicio recibe las hojas de vida para extraer información relevante, categorizar mejor el perfil y almacenar los documentos en formato vectorial para futuras búsquedas y consultas.

Clasificación y Feedback Este microservicio está orientado a la gestión de la evaluación y retroalimentación entre estudiantes y contratantes. Permite que ambas partes asignen calificaciones y comentarios al finalizar un proyecto, fortaleciendo la confianza dentro de la plataforma y promoviendo la construcción de un historial de desempeño. A partir de esta funcionalidad, los contratantes pueden identificar a los estudiantes con mejor rendimiento y los estudiantes pueden mejorar su perfil con base en la retroalimentación recibida, generando un ciclo de mejora continua en la comunidad.

Chat y Comunicación Este contenedor ofrece un canal de mensajería interna para que los estudiantes y contratantes se comuniquen directamente dentro de la plataforma. Su objetivo es centralizar las conversaciones relacionadas con los proyectos, evitando depender de herramientas externas. A través de este medio, los usuarios pueden intercambiar mensajes, coordinar actividades y resolver dudas de forma eficiente.

4.3 Capa de Componentes

En este nivel se detallan contenedores principales describiendo los componentes que componen el flujo y funcionamiento de estos. Para este sistema se describen los componentes de Ofertas y Proyectos e Inteligencia Artificial. Para una mayor especificación referirse al documento, diseño de la solución.

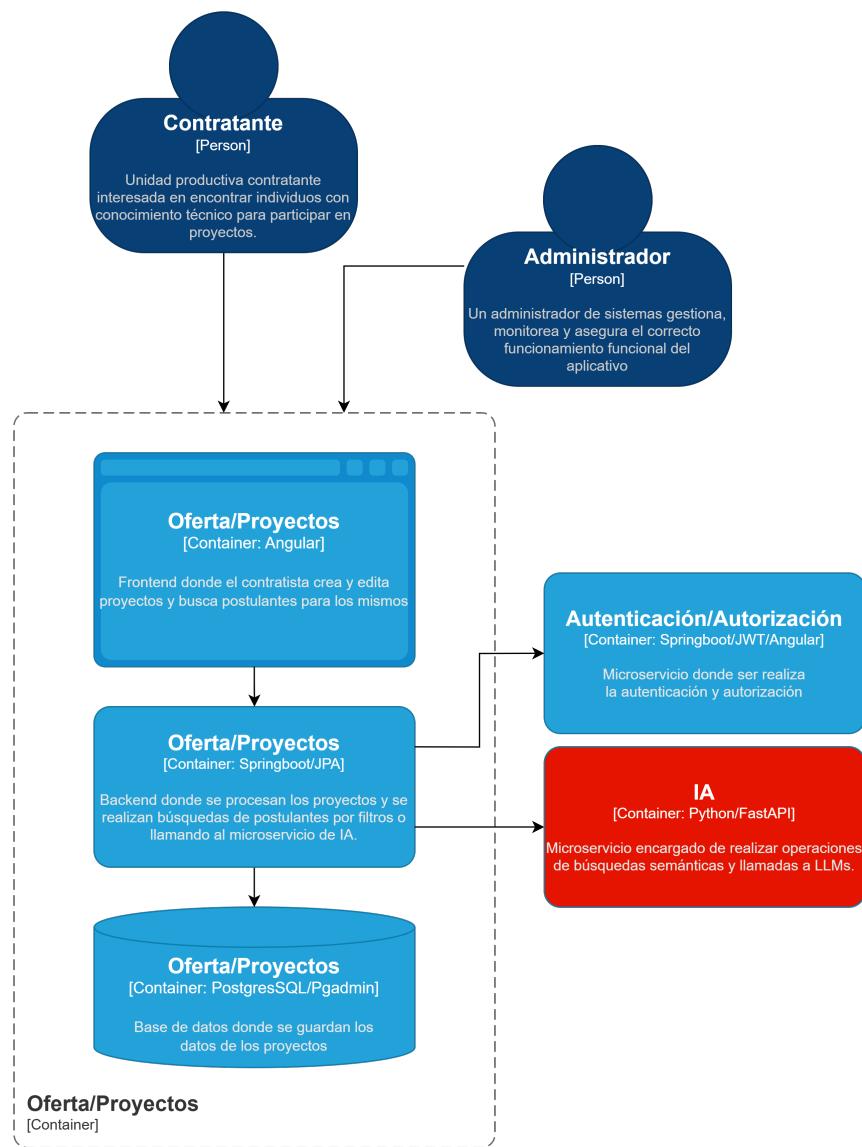


Figura 12: Diagrama Componentes Proyectos - Arquitectura C4

Oferta y proyectos El módulo Ofertas/Proyectos se divide en un frontend desarrollado con Angular y un backend implementado en Spring Boot con JPA. El frontend permite a los contratistas crear, editar y visualizar proyectos, mientras que el backend procesa la lógica de negocio, gestiona la información y realiza consultas sobre la base de datos PostgreSQL, donde se almacenan los datos de los proyectos.

Para el control de acceso y la seguridad del sistema, cada solicitud recibida genera una petición al

microservicio de Autenticación y Autorización el cual verifica que el token sea válido y esté vigente. De ser así retorna información asociada al usuario.

Para procesar solicitudes relacionadas a la búsqueda de candidatos se hacen llamadas al microservicio de Inteligencia Artificial que procesa las especificaciones que quiere el contratista y retorna los perfiles más cercanos.

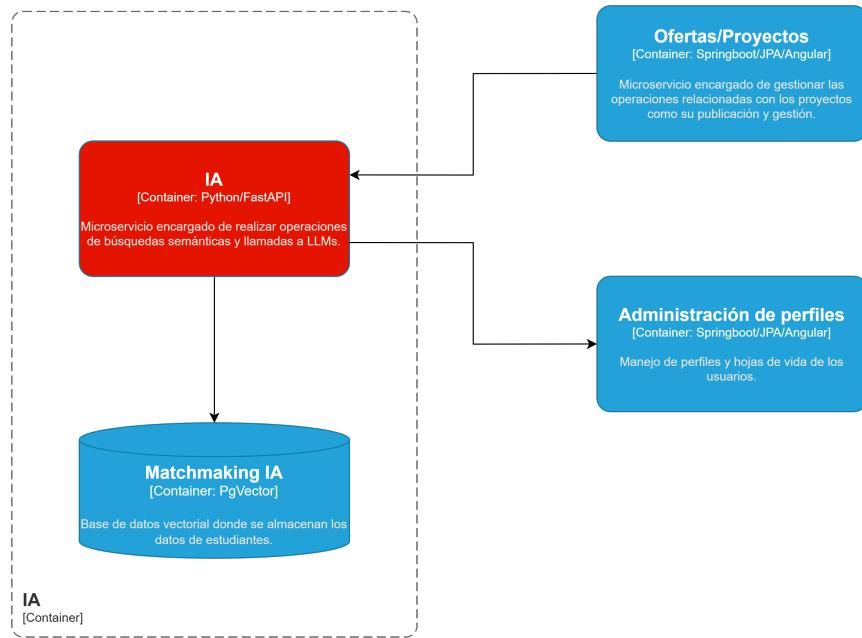


Figura 13: Diagrama Componentes Componentes - Arquitectura C4

IA El componente de Inteligencia Artificial (IA) desarrollado con Python y FastAPI, se integra con los módulos de Ofertas/Proyectos y Administración de Perfiles para ofrecer resultados inteligentes y recomendaciones basadas en similitud conceptual entre lo que el contratista busca y perfiles de los postulantes.

El flujo de interacción entre los componentes sigue una secuencia clara. En primer lugar, dentro del microservicio de Administración de Perfiles, los estudiantes crean su perfil y cargan su hoja de vida. En este punto, se realiza una llamada al componente de Inteligencia Artificial (IA), que procesa y clasifica la información relevante utilizando un modelo de lenguaje externo (LLM). Luego, los datos son vectorizados y almacenados en la base de datos vectorial para su posterior análisis.

Posteriormente, los contratistas, a través de un chatbot, solicitan un conjunto de perfiles que se ajusten a una descripción determinada. El microservicio de Ofertas/Proyectos envía esta solicitud al componente de IA, el cual vectoriza la descripción proporcionada por el contratante y la compara

con los vectores previamente almacenados de los perfiles. De esta manera, se identifican aquellos que presentan una mayor similitud semántica con la solicitud. Finalmente, los perfiles más relevantes son devueltos al microservicio de Ofertas/Proyectos, que se encarga de presentarlos al usuario.

5 Arquitectura de la solución

Para la implementación de la aplicación se decide usar una arquitectura de Microservicios y Microfrontends para aumentar la modularidad de la aplicación facilitando su desarrollo y despliegue. En el backend de la aplicación el uso de microservicios permite distribuir las responsabilidades entre componentes y a su vez facilitar el trabajo independiente en el equipo sobre funcionalidades específicas. En el frontend, la implementación de Microfrontends reduce el acoplamiento entre módulos de la aplicación que, junto a la exposición y consumo de elementos gráficos, mejora el mantenimiento, actualización y tiempo de carga de las páginas.

5.1 Microservicios

Cuadro 4: Microservicio 001 – Autenticación y Autorización

Nombre	Autenticación, autorización de usuario
Descripción	Gestiona el registro, autenticación y control de acceso de usuarios.
Funcionalidad	<ul style="list-style-type: none"> • Registro de estudiantes y contratistas. • Verificación de identidad. • Generación de tokens JWT. • Gestión de roles y permisos.
Entradas	<ul style="list-style-type: none"> • Datos de registro (POST /register). • Credenciales de login (POST /login).
Procesamiento	<ul style="list-style-type: none"> • Validación de credenciales. • Consulta en la base de datos. • Generación de token JWT.
Salidas	<ul style="list-style-type: none"> • Token JWT al frontend. • Datos del usuario autenticado a otros microservicios.
Persistencia	PostgreSQL (users, roles).
Interacciones	<ul style="list-style-type: none"> • Perfiles. • Ofertas.
Especificación Técnica	<p>Java 17 + Spring Boot 3. Endpoints: POST /auth/register, POST /auth/login, GET /auth/validate. Autenticación: JWT.</p>

Cuadro 5: Microservicio 002 – Administración de Perfiles

Nombre	Administración de Perfiles
Descripción	Administra la información de perfiles, CVs y experiencia profesional.
Funcionalidad	<ul style="list-style-type: none"> • CRUD de perfiles. • Almacenamiento de CVs en bucket seguro. • Extracción de metadatos.
Entradas	<ul style="list-style-type: none"> • Solicitudes POST/PUT desde frontend. • Archivos PDF cargados.
Procesamiento	<ul style="list-style-type: none"> • Validación de datos. • Subida de CV a S3. • Extracción de datos clave.
Salidas	<ul style="list-style-type: none"> • Datos de perfil al frontend. • Información procesada al servicio de IA.
Persistencia	PostgreSQL (profiles). Bucket para CVs.
Interacciones	<ul style="list-style-type: none"> • Autenticación. • Matchmaking IA.
Especificación Técnica	Java 17 + Spring Boot 3. Endpoints: POST /profiles, GET /profiles/id, PUT /profiles/id. Autenticación: JWT.

Cuadro 6: Microservicio 003 – Ofertas y Proyectos

Nombre	Ofertas / Proyectos
Descripción	Permite publicar, gestionar y cerrar proyectos.
Funcionalidad	<ul style="list-style-type: none"> • CRUD de proyectos. • Filtros avanzados. • Estados del proyecto.
Entradas	<ul style="list-style-type: none"> • Datos POST /projects. • Filtros GET.
Procesamiento	<ul style="list-style-type: none"> • Validación. • Consulta y filtrado.
Salidas	<ul style="list-style-type: none"> • Lista de proyectos al frontend.
Persistencia	PostgreSQL (projects, offers).
Interacciones	<ul style="list-style-type: none"> • Autenticación. • Matchmaking IA.
Especificación Técnica	Java 17 + Spring Boot 3. Endpoints: POST/GET/PUT/DELETE /projects. Autenticación: JWT.

Cuadro 7: Microservicio 004 – Matchmaking (IA)

Nombre	Matchmaking (IA en Python)
Descripción	Genera recomendaciones automáticas basadas en perfiles y proyectos.
Funcionalidad	<ul style="list-style-type: none"> • Procesamiento de perfiles. • Ejecución modelo de IA. • Ranking de coincidencias.
Entradas	<ul style="list-style-type: none"> • Datos de perfiles. • Datos de proyectos.
Procesamiento	<ul style="list-style-type: none"> • Limpieza y normalización. • Ejecución del modelo IA. • Generación de ranking.
Salidas	<ul style="list-style-type: none"> • Recomendaciones al frontend. • Métricas de rendimiento.
Persistencia	Base de datos vectorial.
Interacciones	<ul style="list-style-type: none"> • Perfiles. • Ofertas. • Administración y Monitoreo.
Especificación Técnica	<p>Python 3.11 + FastAPI. Endpoints: POST /match, GET /metrics. Autenticación: JWT.</p>

Cuadro 8: Microservicio 005 – Chat y Comunicación

Nombre	Chat y Comunicación
Descripción	Mensajería en tiempo real y envío de documentos.
Funcionalidad	<ul style="list-style-type: none"> • Chat en tiempo real. • Envío de archivos. • Notificaciones.
Entradas	<ul style="list-style-type: none"> • Mensajes enviados por usuarios. • Archivos adjuntos.
Procesamiento	<ul style="list-style-type: none"> • Validación de remitente/destinatario. • Almacenamiento. • Notificación en tiempo real.
Salidas	<ul style="list-style-type: none"> • Mensajes en tiempo real. • Historial de chat.
Persistencia	MongoDB (chats, messages).
Interacciones	<ul style="list-style-type: none"> • Autenticación.
Especificación Técnica	<p>Node.js + NestJS. WebSocket + REST. Endpoints: GET /chats/id, POST /chats/id/messages. Autenticación: JWT.</p>

Cuadro 9: Microservicio 006 – Evaluación y Feedback

Nombre	Evaluación y Feedback
Descripción	Registro de calificaciones, comentarios y métricas.
Funcionalidad	<ul style="list-style-type: none">• Calificación numérica.• Comentarios.• Cálculo de métricas.
Entradas	<ul style="list-style-type: none">• Datos de evaluación.• Identificación del proyecto.
Procesamiento	<ul style="list-style-type: none">• Validación.• Cálculo de estadísticas.
Salidas	<ul style="list-style-type: none">• Métricas al frontend.• Estadísticas globales.
Persistencia	PostgreSQL (feedback, ratings).
Interacciones	<ul style="list-style-type: none">• Ofertas.• Administración y Monitoreo.
Especificación Técnica	Java 17 + Spring Boot 3. Endpoints: POST /feedback, GET /feedback/user/id. Autenticación: JWT.

5.2 Microfrontends

En el diseño de nuestra plataforma de emparejamiento entre estudiantes y contratistas, adoptamos una arquitectura moderna de microfrontends utilizando Module Federation de Angular. Esta se alinea estratégicamente con la arquitectura de microservicios en el backend, permitiendo una aplicación modular, escalable y mantenible. Cada microfrontend se especializa en un dominio particular y se comunica eficientemente con los microservicios correspondientes.

Nuestra plataforma de matching académico-profesional enfrenta desafíos complejos de escalabilidad, mantenibilidad y desarrollo paralelo. La arquitectura de microfrontends aborda estos desafíos mediante:

Cada módulo se especializa en un dominio específico, permitiendo que los equipos trabajen sobre funcionalidades claras y aisladas. A su vez, la autonomía técnica habilita la evolución independiente de componentes. La resiliencia operativa permite aislar fallos en módulos particulares sin afectar el sistema completo. Finalmente, se aumenta la velocidad de desarrollo gracias a despliegues independientes.

Aspecto	Detalle Técnico
Microservicio Principal	001 - Autenticación de Usuarios
Endpoints	POST /auth/register, POST /auth/login, GET /auth/validate
Funcionalidades Clave	Registro, validación de identidad, generación de JWT, control de roles, recuperación de credenciales
Comunicación Cross-MFE	Eventos RxJS, interceptores HTTP, estado compartido
Persistencia Local	Tokens en localStorage, sesión en sessionStorage, preferencias en IndexedDB
Arquitectura	Microfrontend especializado y aislado
Justificación	Separación del dominio de seguridad, cumplimiento normativo, actualizaciones independientes

La autenticación constituye un *bounded context* bien definido. Su separación permite ejecutar actualizaciones de seguridad sin afectar otros módulos, implementar múltiples estrategias de autenticación, realizar pruebas especializadas de flujos sensibles y cumplir regulaciones de protección de datos.

Aspecto	Detalle Técnico
Microservicios Asociados	002 - Perfiles, 004 - Matchmaking IA
Endpoints	POST/GET/PUT /profiles, POST /match, GET /metrics
Funcionalidades Clave	Gestión de CV, upload de documentos, análisis IA, dashboard de métricas
Integración IA	Recomendaciones, priorización y aprendizaje de preferencias
Comunicación Cross-MFE	Con autenticación y proyectos
Arquitectura	Microfrontend especializado
Justificación	Dominio complejo, flujos Enriquecidos, integración con IA, UX específica

Este módulo se separa debido a la complejidad del dominio: manipulación avanzada de documentos, integración con servicios de IA, flujos de enriquecimiento de datos profesionales y una UX especializada para estudiantes.

Aspecto	Detalle Técnico
Microservicio Principal	003 - Ofertas/Proyectos
Endpoints	POST/GET/PUT/DELETE /projects, GET /offers
Funcionalidades Clave	Publicación, filtros, estados, matching, métricas, notificaciones
Integraciones	Autenticación, IA, Chat
Arquitectura	Microfrontend transaccional
Justificación	Lógica de negocio compleja y picos de demanda

Este microfrontend encapsula transacciones de proyectos, reglas de negocio específicas, búsquedas de alto rendimiento y escalabilidad independiente en momentos de alta demanda.

Aspecto	Detalle Técnico
Microservicios Asociados	004 - Matchmaking IA
Componentes Clave	Chat IA, dashboards, screening de CV, comparador
Integración IA	NLP, ranking, scoring, predicción, recomendaciones
Arquitectura	Microfrontend de inteligencia de negocio
Justificación	UX especializada, procesamiento intensivo, colaboración

La separación responde a necesidades de UX especializadas para contratistas, cargas intensivas de procesamiento y múltiples integraciones backend.

Aspecto	Detalle Técnico
Microservicio Principal	005 - Chat y Comunicación
Tecnologías	WebSockets
Funcionalidades Clave	Mensajería en tiempo real
Arquitectura	Microfrontend en tiempo real
Justificación	Escalabilidad, conectividad persistente, tiempo real

Este módulo requiere separación debido a los altos requisitos técnicos de tiempo real, conectividad persistente y estrategias específicas de cache y sincronización.

Aspecto	Detalle Técnico
Rol Principal	Orquestador principal
Responsabilidades	Routing, orquestación de MFES, estado global, interceptores, configuración
Comunicación Interna	Custom Events, RxJS Subjects, localStorage, parámetros de ruta
Servicios Compartidos	Autenticación, notificaciones, configuración, analítica, manejo de errores

6 Despliegue

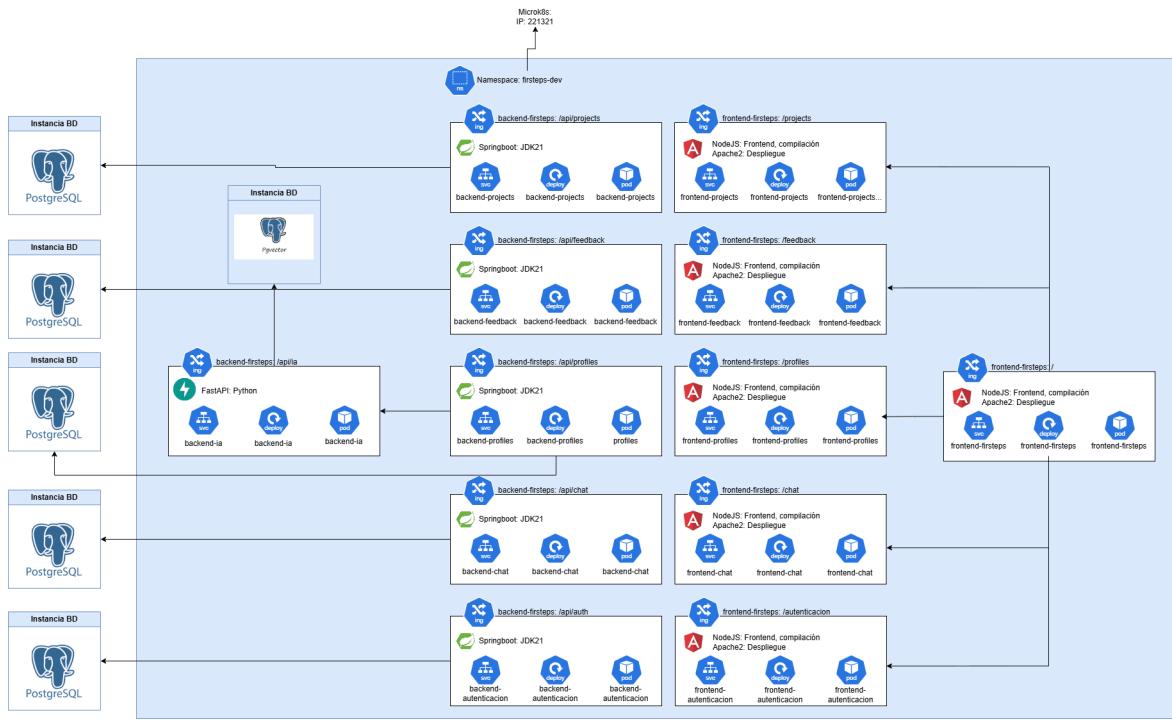


Figura 14: Diagrama de despliegue

6.1 Arquitectura General Basada en Kubernetes

Plataforma de Orquestación Core El despliegue de la plataforma se ejecuta íntegramente sobre un clúster Kubernetes, específicamente utilizando MicroK8s como solución de orquestación containerizada. La elección de Kubernetes no es meramente incidental sino estratégica, representando la columna vertebral de toda la infraestructura tecnológica. El clúster opera bajo el namespace firststeps-dev, proporcionando un aislamiento lógico que permite una gestión granular de recursos y políticas de seguridad específicas para el entorno de desarrollo.

Kubernetes funciona como el sistema nervioso central de la arquitectura, orquestando no solo la ejecución de contenedores sino también proporcionando mecanismos sofisticados de auto-recuperación, escalado automático basado en métricas personalizadas, y gestión del ciclo de vida de las aplicaciones. La plataforma garantiza alta disponibilidad mediante distribuciones inteligentes de pods across nodes, checks de salud continuos mediante liveness y readiness probes, y reinicios automáticos de contenedores fallidos.

Componentes Arquitectónicos Fundamentales La arquitectura se compone de múltiples capas tecnológicas que interactúan de manera sinérgica. En el núcleo encontramos microservicios backend diseñados bajo principios de dominio bounded context, cada uno con autonomía operativa completa. Complementando esta estructura, el frontend se implementa mediante microfrontends integrados a través de Native Module Federation, permitiendo una composición modular de interfaces de usuario.

La persistencia de datos se gestiona mediante instancias PostgreSQL aisladas por dominio de negocio, incluyendo una configuración especializada con la extensión pgvector para operaciones de inteligencia artificial sobre embeddings. La capa de red está gobernada por un Ingress Controller que centraliza y gestiona todo el tráfico entrante, mientras que los servicios internos se comunican mediante Service Discovery nativo de Kubernetes.

Cada unidad funcional despliega su propio conjunto de objetos Kubernetes: Services para descubrimiento interno, Deployments para la gestión del ciclo de vida, Pods como unidades de ejecución mínimas, y ConfigMaps/Secrets para la gestión de configuración sensible. El empaquetado universal se realiza mediante contenedores Docker, asegurando consistencia desde desarrollo hasta producción.

Automatización y Pipeline CI/CD El flujo de integración y despliegue continuo está implementado con GitHub Actions, constituyendo un pipeline altamente automatizado que abarca desde el commit inicial hasta el despliegue en producción. Este pipeline incorpora prácticas modernas de DevSecOps, incluyendo escaneo estático de código, análisis de vulnerabilidades en dependencias, pruebas de seguridad en contenedores, y validación de configuraciones Kubernetes.

6.2 Implementación de Microfrontends con Webpack Module Federation / Native Federation

Paradigma de Frontend Distribuido Una de las innovaciones más significativas en el despliegue actual es la adopción de una arquitectura de frontend distribuido. Cada sección funcional de la plataforma —gestión de proyectos, sistema de feedback, perfiles de usuario, chat en tiempo real, autenticación y interfaz principal— se construye y despliega como un microfrontend completamente independiente.

Esta aproximación representa un cambio paradigmático respecto a las aplicaciones monolíticas tradicionales. Cada funcionalidad cuenta con su propio pipeline de build basado en Angular/Node, su servidor web Apache2 dedicado, y su conjunto completo de recursos Kubernetes. La independencia se extiende también al versionado, donde cada microfrontend puede evolucionar siguiendo su propio roadmap y ciclo de release sin dependencias sincrónicas con otros equipos.

Tecnología Native Federation: Evolución y Ventajas La integración en runtime de estos microfrontends distribuidos se realiza mediante Native Federation, representando la evolución moderna del Module Federation clásico de Webpack. Esta tecnología permite una composición dinámica de aplicaciones frontend donde los módulos se cargan y ejecutan en contextos aislados pero interoperables.

Native Federation habilita capacidades avanzadas como el sharing de módulos entre aplicaciones en tiempo real, la carga bajo demanda de componentes remotos sin recarga completa de página, y la coexistencia de múltiples versiones de librerías cuando es necesario. Los microfrontends pueden exponer componentes específicos como entradas remotas y consumir componentes de otras aplicaciones como dependencias dinámicas.

Beneficios Operativos y Organizativos Los beneficios prácticos de esta arquitectura son sustanciales. Desde una perspectiva operativa, permite actualizaciones incrementales donde la modificación de un módulo específico no requiere el redeploy completo de la aplicación. Esto reduce ventanas de mantenimiento y mitiga riesgos de deployment. La escalabilidad se gestiona de forma granular, pudiendo asignar recursos específicos a microfrontends con mayor demanda computacional.

Organizacionalmente, el modelo permite que equipos distribuidos trabajen de forma autónoma, con independencia tecnológica y de release cycles. El aislamiento de fallos se ve significativamente mejorado, ya que un error en un microfrontend no propaga su efecto a toda la aplicación. La velocidad de entrega se acelera al permitir deployments frecuentes y focalizados.

6.3 Arquitectura de Microservicios Backend (Spring Boot + FastAPI)

Diseño Basado en Dominios de Negocio El backend está estructurado como un ecosistema de microservicios independientes donde cada dominio del negocio se encapsula en un servicio autónomo con boundaries bien definidos. Los servicios desarrollados en Spring Boot utilizando JDK21—backend-projects, backend-feedback, backend-profiles, backend-chat y backend-autenticacion— siguen principios de diseño domain-driven design (DDD).

Cada microservicio Spring Boot constituye una unidad de despliegue independiente con su propia API REST, estrategia de persistencia dedicada, y capacidad de auto-scaling individual. La independencia se extiende a la gestión de datos, donde cada servicio mantiene su propia base de datos PostgreSQL, evitando acoplamientos a nivel de schema y permitiendo evoluciones independientes.

Servicio Especializado de IA con FastAPI El servicio backend-ia representa una especialización tecnológica dentro del ecosistema, diseñado específicamente para tareas de inteligencia artificial y procesamiento de embeddings. Desarrollado con Python y FastAPI, aprovecha las ventajas del ecosistema Python para machine learning junto con la performance y capacidades asincrónicas de FastAPI.

La arquitectura de este servicio incorpora PostgreSQL con la extensión pgvector para el almacenamiento y consulta eficiente de embeddings vectoriales. Esto permite operaciones de similarity search, clustering y clasificación directamente en la base de datos, reduciendo latencias y complejidad. El servicio está optimizado para inferencias ligeras y exposición de endpoints ML como servicios REST.

Principios Arquitectónicos Aplicados La arquitectura de microservicios aplica consistentemente el principio de single responsibility, donde cada servicio tiene una razón única para cambiar. Los servicios se comunican mediante APIs REST bien definidas, con contratos versionados y documentados.

La observabilidad se implementa mediante logging estructurado, métricas personalizadas y distributed tracing.

La resiliencia se construye mediante patrones como circuit breaker, retry con backoff exponencial y timeouts configurables. La seguridad se implementa en múltiples capas, incluyendo autenticación JWT, autorización basada en roles, encryptación de datos sensibles y validación exhaustiva de inputs.

6.4 Estrategia de Contenerización con Docker

Aislamiento y Consistencia de Entornos Cada componente del sistema —frontends, backends y servicios especializados— está empaquetado en contenedores Docker, proporcionando un aislamiento completo entre runtimes y dependencias. Esta contenerización permite que cada microservicio y microfrontend opere en su entorno optimizado: Java JDK21 para aplicaciones Spring Boot, Python 3.x con las librerías específicas para FastAPI y machine learning, y NodeJS con Apache2 para las aplicaciones Angular.

El uso de Docker garantiza una consistencia absoluta entre entornos, eliminando el clásico problema "funciona en mi máquina". El mismo contenedor que se prueba en el entorno de CI/CD es el que se despliega en producción, asegurando que las dependencias del sistema, variables de entorno y configuración sean idénticas en todas las etapas.

6.5 Optimización de Imágenes y Seguridad

Las imágenes Docker se construyen siguiendo mejores prácticas de optimización y seguridad. Se emplean construcciones multi-stage para separar el entorno de build del runtime final, resultando en imágenes mínimas que contienen solo el artefacto ejecutable y sus dependencias esenciales. Esto reduce la superficie de ataque, mejora los tiempos de descarga y despliegue.

Las imágenes se versionan con tags semánticos y se firman digitalmente para garantizar integridad. Se integran escáneres de vulnerabilidades en el pipeline que analizan las imágenes en busca de CVE conocidos, aplicando políticas de seguridad que pueden bloquear despliegues con vulnerabilidades críticas. Los registries se configuran con retención automática y políticas de garbage collection.

6.6 Kubernetes como Plataforma de Despliegue y Operación

Orquestación Avanzada de Recursos El clúster Kubernetes, implementado con MicroK8s, sirve como plataforma integral de orquestación, proporcionando capacidades avanzadas de gestión del ciclo de vida de aplicaciones. Cada módulo de la aplicación se despliega utilizando los objetos nativos de Kubernetes, configurados para optimizar disponibilidad, performance y resiliencia.

Los Deployments gestionan la estrategia de actualización mediante rolling updates que garantizan cero downtime durante despliegues. Se configuran readiness probes para determinar cuándo un contenedor está listo para recibir tráfico, y liveness probes para detectar y recuperar automáticamente contenedores en estado incorrecto. Los resource limits y requests aseguran una distribución equitativa

de recursos compute across pods.

6.7 Redes y Descubrimiento de Servicios

La capa de red está cuidadosamente diseñada para proporcionar conectividad segura y eficiente. Los Services exponen cada Deployment internamente mediante DNS bajo el dominio svc.cluster.local, permitiendo el descubrimiento dinámico de servicios. El Ingress actúa como punto de entrada único, enruteando el tráfico externo basado en reglas de path y host.

Las reglas de Ingress están configuradas para mapear rutas específicas a sus servicios correspondientes: /api/projects hacia backend-projects, /projects hacia frontend-projects, /api/chat hacia backend-chat, /chat hacia frontend-chat, y /api/ia hacia backend-ia. Se implementan políticas de network policies para restringir comunicación entre pods solo a los puertos y protocolos necesarios.

6.8 Pipeline de CI/CD con GitHub Actions

Integración Continua Avanzada El pipeline de CI/CD implementado con GitHub Actions automatiza exhaustivamente el proceso de entrega de software. En la fase de integración continua, se ejecutan workflows paralelos para cada microservicio y microfrontend, incluyendo la instalación de dependencias, análisis estático de código con linters especializados, ejecución de suites de pruebas unitarias e integrales, y generación de reports de cobertura.

Despliegue Continuo y Estrategias de Release En la fase de despliegue continuo, el pipeline gestiona el push de imágenes al registry container, la actualización de manifiestos Kubernetes, y la ejecución de estrategias de deployment avanzadas.

El pipeline soporta múltiples estrategias de release como blue-green deployments y canary releases, permitiendo una transición gradual de tráfico hacia nuevas versiones. La integración con sistemas de observabilidad permite rollbacks automáticos basados en métricas de performance o tasa de errores.

6.9 Flujo Completo de Despliegue End-to-End

Proceso Automatizado de Entrega El flujo completo de despliegue constituye un proceso altamente automatizado que se inicia con el commit de un desarrollador en GitHub. Este evento activa automáticamente el pipeline de GitHub Actions, que ejecuta en paralelo la construcción y testing de todos los microservicios y microfrontends afectados por los cambios.

Tras la fase de construcción y validación, se generan las imágenes Docker optimizadas y se taggean con versiones semánticas derivadas del conventional commit. Las imágenes se push al registry container previa verificación de seguridad y calidad. El pipeline actualiza entonces los manifiestos Kubernetes en el repositorio de GitOps, triggering la sincronización automática con el clúster.

6.10 Actualización en Clúster y Composición Dinámica

Kubernetes ejecuta rolling updates para cada microservicio actualizado, garantizando disponibilidad continua durante la transición. El Ingress Controller refleja inmediatamente los cambios de routing cuando nuevos pods pasan sus readiness checks. En el frontend, Native Federation permite la carga dinámica de nuevas versiones de microfrontends sin afectar los módulos no modificados.

El sistema incorpora mecanismos de verificación post-despliegue que monitorizan métricas clave de performance y negocio, ejecutando rollbacks automáticos si se detectan anomalías. Los logs de despliegue se centralizan y se correlacionan con métricas de observabilidad para proporcionar visibilidad completa del proceso.

VI DESARROLLO DE LA SOLUCIÓN

El desarrollo de la solución se realizó siguiendo un enfoque ágil, orientado a entregar incrementos funcionales cada semana, permitiendo así iteraciones rápidas para el progreso en los microservicios y la integración con el frontend. La planificación y ejecución estuvieron alineadas con el cronograma ajustado específicamente para este proyecto y con los entregables definidos (MVP, documentación, pruebas y revisión ética).

Las principales actividades implementadas en esta etapa fueron: planificación de sprints, implementación de microservicios, integración del motor IA (PoC y prototipo), pruebas unitarias de integración, validaciones de usabilidad y la revisión ética y el tratamiento de datos en la fase final.

1 Planificación del cronograma

La planificación de los sprints fue de gran utilidad, ya que permitió establecer un flujo de trabajo iterativo, donde cada semana se definieron metas claras, entregables concretos y responsables específicos. La estructura ágil facilitó priorizar tareas críticas, ajustar plazos y mantener un ritmo constante de desarrollo.

Durante la ejecución del proyecto fue necesario realizar dos ajustes formales al cronograma original. Ambos ajustes se concentraron en la fase de Prueba de Concepto (PoC) y en los primeros sprints de Desarrollo, debido a la complejidad técnica identificada durante la implementación inicial del motor de IA y la construcción de los microservicios base. La PoC requirió iteraciones adicionales para evaluar modelos de similitud y clasificación, lo cual extendió esta fase más allá del tiempo inicialmente previsto. De manera similar, los sprints iniciales de desarrollo tomaron más tiempo debido a la necesidad de refinar servicios, ajustar modelos de datos y estabilizar la comunicación entre microservicios. Aunque estos cambios no alteraron los entregables finales, permitieron garantizar una base técnica más sólida antes de avanzar a las etapas de validación, pruebas de rendimiento y usabilidad.

Cuadro 10: Cronograma inicial del proyecto

Semana	Fechas aprox.	Fase	Actividades principales	Entregables esperados
1	15–21 julio	Análisis PoC (1/2)	Entrevistas, benchmarking, análisis IA, exploración de enfoques	Documento de requisitos, inicio informe IA
2	22–28 julio	Análisis PoC (2/2)	Evaluación de modelos IA, definición de criterios de comparación, exploración de arquitectura	Informe comparativo de modelos, preselección de arquitectura IA
3	29 julio – 4 agosto	Diseño + Desarrollo PoC	Diseño de pruebas, desarrollo del prototipo PoC, ejecución de pruebas, comparación de resultados	Backlog inicial validado, selección de arquitectura IA, pruebas de PoC documentadas
4	5–11 agosto	Desarrollo (Sprint 1)	Módulo de registro y perfiles, conexión inicial frontend-backend	Módulo funcional inicial
5	12–18 agosto	Desarrollo (Sprint 2)	Módulo de publicación de proyectos y buscador	Módulo de proyectos
6	19–25 agosto	Desarrollo (Sprint 3)	Integración del modelo IA en el flujo de matchmaking	Matchmaking funcional
7	26 agosto – 1 septiembre	Desarrollo (Sprint 4)	Módulo de calificación y feedback	Sistema de retroalimentación
8	2–8 septiembre	Desarrollo (Sprint 5)	Integración completa y optimización	MVP funcional integrado
9	9–15 septiembre	Ajustes técnicos finales	Refactorización, documentación técnica, validación de ambientes	Código documentado, guía técnica, ambientes validados
10	16–22 septiembre	Validaciones internas	Revisión de backlog, manual técnico, pruebas preliminares	Manual técnico, guía de instalación
11	23–29 septiembre	Ejecución de Pruebas (1/3)	Pruebas funcionales (unitarias e integración)	Informe de pruebas funcionales aprobado
12	30 septiembre – 6 octubre	Ejecución de Pruebas (2/3)	Pruebas de carga y seguridad	Informe de rendimiento y seguridad
13	7–13 octubre	Ejecución de Pruebas (3/3)	Pruebas de usabilidad con usuarios reales	Validación ($\geq 85\%$ satisfacción)
14	14–20 octubre	Revisión final	Ajustes post-feedback, consolidación de documentación, revisión ética y datos de usuarios	Entregables ajustados, informe de revisión ética aprobado
15	21–27 octubre	Preparación Final	Diapositivas, pitch, ensayo con director	Presentación final lista
16	28 octubre – 3 noviembre	Entrega Final	Entrega de código, memoria e informe; despliegue en producción	Entrega oficial completa

1.1 Metodología y dinámica de trabajo

El proyecto se gestionó mediante una metodología ágil adaptada, basada en ciclos semanales de planeación y ejecución. Esta estructura permitió mantener un ritmo constante de trabajo, verificar avances de forma periódica y mejorar iterativamente los entregables.

Las reuniones con el director de tesis se realizaron todos los miércoles durante una hora. En estos encuentros se presentaron avances, se resolvieron problemas técnicos y se recibió retroalimentación académica esencial para el progreso del proyecto.

El equipo también llevó a cabo reuniones internas varias veces por semana con el fin de coordinar tareas, definir prioridades y revisar pendientes. Durante estas sesiones se actualizó el tablero Kanban semanal y se aseguraba la alineación del equipo respecto al ciclo de desarrollo.

En cada reunión con el director se elaboraron actas breves registrando acuerdos, tareas pendientes y compromisos para la siguiente semana. Esta documentación permitió mantener trazabilidad y claridad en la toma de decisiones.

La universidad proporcionó una plantilla de Kanban en Excel que se actualizaba cada lunes. Este tablero permitió registrar el avance de las tareas, identificar bloqueos y garantizar la trazabilidad de los entregables del proyecto.

La comunicación interna se realizó mediante canales de mensajería, utilizados para la coordinación diaria y seguimiento de subtareas, asegurando fluidez y eficiencia en la colaboración del equipo.

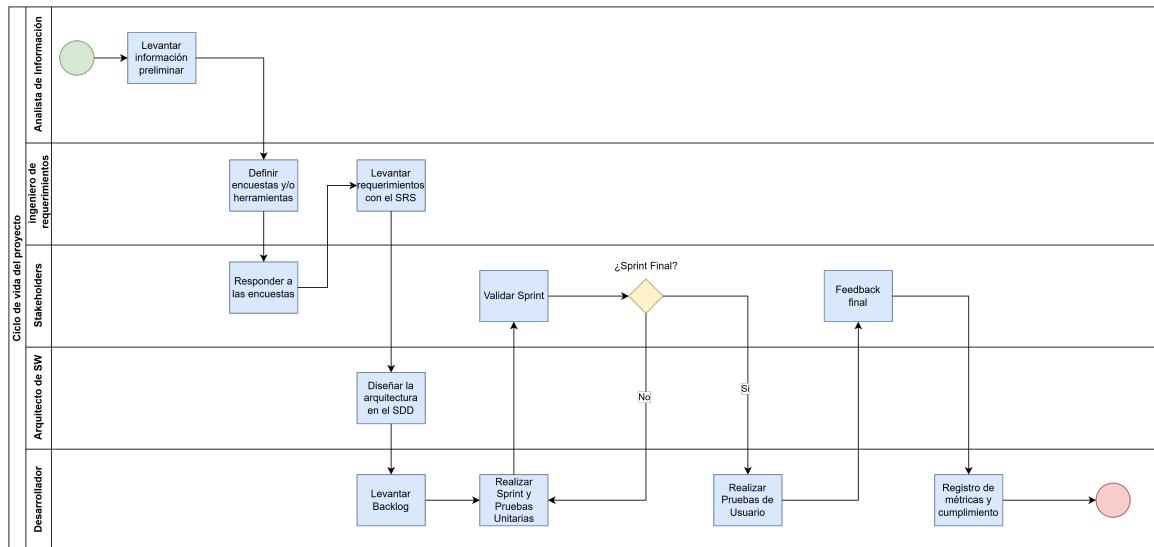


Figura 15: Ciclo de vida proyecto

1.2 Estructura de los Sprints en el Cronograma

Semana	Periodo	Fase/Sprint	Objetivo Principal	Entregables
1–4	15 jul – 11 ago	Análisis PoC + Diseño	Exploración de enfoques IA, definición de arquitectura y pruebas de concepto	Documento de requisitos, informe IA, backlog inicial
5–7	12 ago – 1 sep	Desarrollo (Sprints 1–2)	Implementación de microservicios iniciales, conexión básica backend	Microservicios iniciales funcionando
8–10	2 sep – 22 sep	Desarrollo extendido (Sprints 3–5)	Ampliación de módulos, integración parcial IA, optimización	MVP parcial
11	23 sep – 13 oct	Validaciones + Pruebas Funcionales	Pruebas preliminares unitarias e integración	Informe de validación preliminar
12	14 oct – 20 oct	Ejecución de Pruebas (1/2)	Pruebas de carga y seguridad	Informe de rendimiento
13	21 – 27 oct	Ejecución de Pruebas (2/2)	Usabilidad con usuarios reales	Validación $\geq 85\%$ satisfacción
14	28 oct – 3 nov	Revisión final + Ética	Ajustes finales y revisión ética sobre datos de usuarios	Informe de revisión ética aprobado
15	4 nov – 10 nov	Preparación Final	Ensayo, pitch y elaboración de diapositivas	Presentación final lista
16	11 nov – 17 nov	Entrega Final	Memoria final, código y despliegue en producción	Entrega oficial completa

2 Desarrollo y Pruebas

El desarrollo técnico se centró en construir un sistema basado en microservicios conectados a un modelo de inteligencia artificial, con una arquitectura escalable y modular. Paralelamente se aplicaron pruebas de integración y calidad, garantizando la estabilidad del sistema en cada fase.

2.1 Flujo técnico

- **Backend:** desarrollado en Spring Boot, implementando microservicios para usuarios, proyectos y matching.
- **Frontend:** construido con Angular, asegurando una interfaz adaptable e intuitiva.
- **Base de datos:** PostgreSQL, para gestión relacional de perfiles, métricas y resultados del modelo IA.
- **Integración IA:** se usó un enfoque de clasificación y similitud textual (TF-IDF, SVM, embeddings) para asociar perfiles con requerimientos.

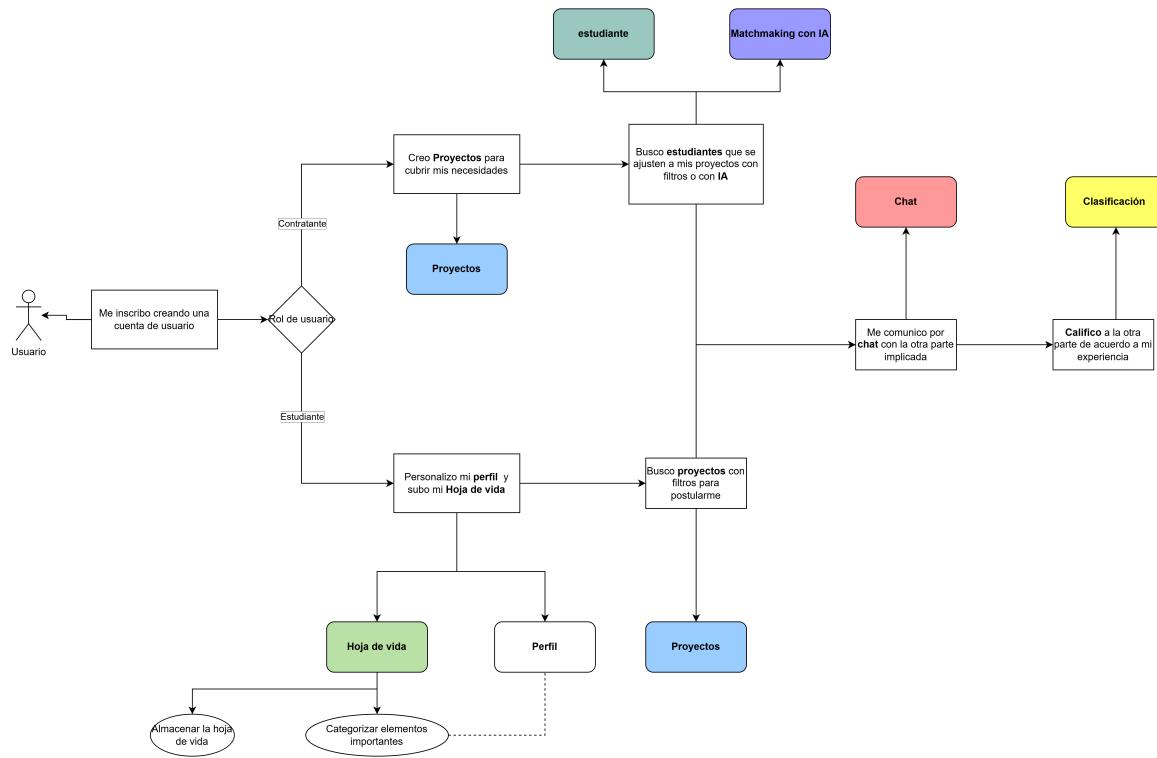


Figura 16: Flujo Funcional de la Aplicación

Durante el proceso de aseguramiento de calidad se ejecutaron tres tipos principales de pruebas:

- **Pruebas unitarias:** orientadas a verificar el correcto funcionamiento de módulos individuales en los microservicios desarrollados. Estas pruebas se enfocaron en la validación de controladores, servicios y componentes críticos del backend y del motor de IA.
- **Pruebas funcionales:** encargadas de verificar el comportamiento integral del sistema desde la perspectiva del usuario final. Incluyeron flujos completos entre frontend, backend y el microservicio de IA, además de validaciones de usabilidad con usuarios reales, quienes evaluaron la claridad, accesibilidad y coherencia de la plataforma durante la interacción.
- **Pruebas de carga:** ejecutadas para evaluar el rendimiento del sistema bajo condiciones de estrés. Se midió la capacidad de respuesta de los microservicios, la eficiencia del motor de recomendación y la estabilidad del frontend ante múltiples solicitudes concurrentes.

En conjunto, estas pruebas permitieron validar la estabilidad del sistema, la coherencia de los flujos funcionales y el desempeño de los componentes críticos bajo diferentes escenarios de uso.

VII RESULTADOS

La implementación del proyecto permitió obtener resultados significativos tanto en términos técnicos como en relación con los objetivos planteados inicialmente. El desarrollo del prototipo, la integración del motor de inteligencia artificial y la ejecución de las pruebas funcionales y de calidad evidenciaron la viabilidad de la solución, así como su capacidad para apoyar el proceso de emparejamiento entre estudiantes y unidades productivas contratantes.

1 Resultados del prototipo desarrollado

El sistema final correspondió a una plataforma funcional construida bajo una arquitectura de microservicios, integrando un frontend en Angular, un backend modular en Spring Boot y una base de datos en PostgreSQL. Este enfoque permitió desarrollar un producto escalable, desacoplado y fácil de mantener.

El prototipo incluyó todos los módulos esenciales establecidos en la planificación: gestión de usuarios (estudiantes y contratantes), gestión de ofertas, postulaciones, comunicación interna, calificaciones y un motor de recomendación basado en análisis de similitud textual. La integración del modelo de IA, inicialmente validada mediante la Prueba de Concepto (PoC), permitió automatizar la sugerencia de candidatos en función de habilidades, proyectos previos y criterios definidos por los contratantes. El sistema fue desplegado en un entorno funcional y acompañado de la documentación técnica correspondiente.

2 Resultados de la integración del motor de IA

La PoC del modelo IA permitió comparar distintos enfoques de similitud y clasificación, evaluando su aplicabilidad dentro del entorno del proyecto. A partir de esta experimentación se seleccionó un enfoque híbrido que combinó técnicas de clasificación y embeddings para mejorar la precisión en el emparejamiento.

Si bien la fase de PoC requirió más tiempo del estimado inicialmente, lo cual motivó el primer ajuste del cronograma, su resultado permitió definir una arquitectura de recomendación estable, reproducible y técnicamente compatible con los microservicios. La integración final demostró que el modelo era capaz de analizar perfiles y requerimientos y generar recomendaciones útiles y consistentes dentro de los flujos funcionales previstos.

3 Resultados del desarrollo y de las pruebas de calidad

El sistema fue sometido a distintos tipos de pruebas con el fin de garantizar estabilidad, coherencia funcional y una experiencia de usuario adecuada:

4 Pruebas unitarias e integración

Se validaron controladores, servicios y componentes críticos del backend. Estas pruebas permitieron identificar y corregir errores tempranos, mejorar la robustez de los microservicios y asegurar una correcta comunicación mediante API REST.

5 Pruebas funcionales

Se evaluaron los flujos completos entre frontend, backend y el motor de IA. Estas pruebas aseguraron que los usuarios pudieran registrarse, gestionar ofertas, postularse, intercambiar mensajes y recibir recomendaciones de forma consistente. La validación funcional permitió confirmar que el MVP cumplía con las expectativas de uso definidas inicialmente.

6 Pruebas de carga

El sistema fue evaluado bajo condiciones de mayor concurrencia para determinar su comportamiento frente a múltiples solicitudes simultáneas. Estas pruebas permitieron identificar oportunidades de optimización y confirmar la estabilidad general del sistema en escenarios demandantes.

7 Resultados frente a los objetivos del proyecto

La solución desarrollada permitió cumplir el objetivo general del trabajo de grado: diseñar e implementar un sistema prototípico que facilite el emparejamiento entre estudiantes y unidades productivas mediante técnicas de inteligencia artificial.

Los objetivos específicos también fueron alcanzados:

- **Diseño arquitectónico:** se construyó un sistema modular basado en microservicios.
- **Implementación del prototipo:** el MVP incluyó los módulos centrales previstos.
- **Integración del motor IA:** se realizó una PoC exitosa y su integración al flujo operacional.
- **Validación técnica:** las pruebas funcionales y de carga confirmaron el desempeño esperado.
- **Evaluación de usabilidad:** los usuarios finales validaron la utilidad del sistema.

En conjunto, los resultados evidencian que la solución es técnicamente viable, funcional y alineada con las necesidades de los actores involucrados.

8 Impacto y aporte del proyecto

El prototipo desarrollado constituye un avance significativo en la automatización del proceso de selección de talento en contextos educativos y productivos. La plataforma mejora la eficiencia operativa de las unidades productivas y facilita a los estudiantes acceder a oportunidades relevantes de acuerdo con su perfil.

Asimismo, la arquitectura diseñada y la documentación generada ofrecen una base sólida para futuras extensiones, tales como modelos más avanzados de recomendación, integración de analítica avanzada o la ampliación del sistema a otros programas académicos o instituciones.

9 Pruebas funcionales

Las pruebas funcionales se diseñaron con el objetivo de validar el comportamiento completo de la plataforma Firststeps desde la perspectiva del usuario final, evaluando los flujos más representativos del sistema para dos tipos de actores: estudiante freelance y unidad productiva contratante.

A diferencia de las pruebas unitarias, que evalúan en aislamiento la lógica interna de los microservicios, las pruebas funcionales verifican la interacción entre componentes: frontend, backend, microservicios de IA, bases de datos y mensajería. Su propósito es asegurar que el sistema responda correctamente ante escenarios reales de uso y que las funcionalidades principales operen de principio a fin sin errores.

Estas pruebas se elaborarán siguiendo flujos completos para ambos usuarios, simulando un caso real de interacción. El diseño de los casos se basa en las historias de usuario, pero su estructura NO corresponde a los criterios de aceptación, sino a un formato formal de prueba funcional, tal como lo requieren los estándares de aseguramiento de calidad.

9.1 Metodología aplicada

Para garantizar consistencia, claridad y facilidad de ejecución, las pruebas funcionales se diseñaron siguiendo los principios:

- **Validación end-to-end:** cada caso recorre todas las capas del sistema (UI → API → servicios internos → BD → respuestas).
- **Escenarios representativos:** se simularán acciones reales tanto del estudiante como del contratista.
- **Resultados observables:** cada prueba debe generar un resultado verificable por interfaz o por respuesta del sistema.
- **Repetibilidad:** cualquier miembro del equipo debe poder ejecutar la prueba siguiendo los pasos descritos.
- **Evidencia visual opcional:** se podrán anexar pantallazos del flujo para reforzar cada prueba.

Las pruebas no se limitan a un solo punto del sistema; cada una involucra varios microservicios actuando de manera conjunta, lo que permite evaluar la integridad del backend y la correcta comunicación entre módulos.

9.2 Prueba con Sujeto 1

Cuadro 11: Pruebas Funcionales - Sujeto 1

ID Prueba	Función Evaluada	Precondiciones	Observaciones	Estado
PF-S01-01	Registro de contratista	No existe cuenta previa en la plataforma con la información utilizada	No se puede visualizar la contraseña antes de crearla. Hay opción de recuperar contraseña antes de crear la cuenta	Aprobada
PF-S01-02	Publicación de oferta	Contratista autenticado	El nombre del proyecto indica que debe tener como mínimo 4 caracteres, en realidad necesita 8.	Aprobada
PF-S01-03	Emparejamiento IA	Oferta creada; algoritmo activo	No hay límite al número de meses que se puede solicitar un perfil. Error de ortografía	Aprobada
PF-S01-04	Gestión de postulaciones	Estudiantes postulados	NA	Aprobada
PF-S01-05	Registro de estudiante	No existe cuenta previa en la plataforma con la información utilizada	Igual que PF-S01-01	Aprobada
PF-S01-06	Edición de perfil	Inicio de sesión exitoso; perfil creado	Se debería redirigir a otra página después de subir la hoja de vida	Aprobada
PF-S01-07	Búsqueda de ofertas	Perfil funcional y datos de prueba cargados	NA	Aprobada

9.3 Prueba con Sujeto 2

Cuadro 12: Pruebas Funcionales - Sujeto 2

ID Prueba	Función Evaluada	Precondiciones	Observaciones	Estado
PF-S02-01	Registro de contratista	Credenciales nuevas sin uso previo en el sistema	Sin comentarios	Aprobada
PF-S02-02	Publicación de oferta	Usuario con sesión activa y rol de contratista	La validación de longitud del título es más estricta de lo indicado en la interfaz. Sugerencia: unificar criterios	Aprobada
PF-S02-03	Emparejamiento IA	Proyecto publicado y servicio de matching operativo	Pequeño bug a la hora de avanzar en la escogencia del candidato, hace falta un mejor loader	Aprobada
PF-S02-04	Gestión de postulaciones	Existencia de candidatos aplicados al proyecto	Flujo completo funcional sin inconvenientes	Aprobada
PF-S02-05	Registro de estudiante	Datos personales y académicos válidos	Redireccion del correo no valida	Aprobada
PF-S02-06	Edición de perfil	Perfil de estudiante creado previamente	No se sabe si ya quedo cargado el perfil, ni donde fue cargado	Aprobada
PF-S02-07	Búsqueda de ofertas	Información de perfil completa en el sistema	Funcionalidad de búsqueda y filtrado operando adecuadamente	Aprobada

VIII CONCLUSIONES

1 Análisis de Impacto del Proyecto

El proyecto desarrollado generó un impacto significativo tanto a nivel tecnológico como metodológico. En primer lugar, permitió demostrar la viabilidad de una plataforma inteligente de emparejamiento entre estudiantes y unidades productivas, integrando microservicios, un motor de recomendación y una arquitectura modular basada en principios actuales de ingeniería de software. El uso de modelos de similitud y análisis de texto permitió acelerar el proceso de selección, aportando criterios más objetivos y fomentando la identificación de talento no visible a través de métodos tradicionales.

Desde la perspectiva de arquitectura, el impacto se refleja en la adopción de un enfoque moderno, escalable y desacoplado. La implementación con Docker y Kubernetes facilita la orquestación, despliegue y actualización continua del sistema, permitiendo que los diferentes servicios del prototipo puedan ejecutarse en ambientes distribuidos de manera confiable. Esto hace que la solución sea altamente escalable en otros entornos, tanto académicos como empresariales, ya que puede desplegarse en nubes públicas, privadas o en infraestructura universitaria sin necesidad de cambios sustanciales.

Asimismo, el proyecto contribuye al proceso formativo institucional al ofrecer una base funcional que puede ser extendida en próximos semestres, permitiendo que otros estudiantes estudien, repliquen y amplíen la solución con nuevos modelos de IA, microservicios adicionales o metodologías de interacción más avanzadas. A nivel social, el sistema también genera impacto al facilitar oportunidades laborales y proyectos para estudiantes, incrementando sus posibilidades de inserción profesional.

En conjunto, el proyecto entrega una solución técnicamente robusta, metodológicamente sólida y con un potencial de expansión considerable hacia escenarios de mayor complejidad y adopción real.

2 Trabajos Futuros

Aunque el prototipo alcanzó los objetivos planteados, existen múltiples oportunidades de mejora y expansión que podrían abordarse en iteraciones posteriores o incluso en nuevos trabajos de grado. Entre las más relevantes se destacan:

2.1 Implementación de un rol de administrador

Agregar un rol administrativo permitiría supervisar toda la plataforma, gestionar usuarios y ofertas, monitorear métricas globales y acceder a dashboards de actividad. Esto aumentaría el control, la seguridad y la trazabilidad dentro del sistema.

2.2 Integración de colas de mensajería (Kafka)

La incorporación de Apache Kafka o tecnologías similares permitiría optimizar el intercambio de eventos entre microservicios, promoviendo un sistema más robusto, desacoplado y completamente asincrónico. Esto facilitaría:

- Procesamiento de grandes volúmenes de solicitudes
- Mayor resiliencia ante fallos

- Escalabilidad horizontal más eficiente

Especialmente útil para módulos de notificaciones, registro de actividad o el motor de recomendación.

2.3 Evolución del motor de IA

El modelo actual puede extenderse hacia enfoques más avanzados, como embeddings neuronales de última generación, modelos de lenguaje más robustos o técnicas híbridas de clasificación y ranking. Esto permitiría mejorar la precisión del emparejamiento.

2.4 Extensión del modelo de reputación

Ampliar el sistema actual con métricas adicionales, retroalimentación longitudinal y validación cruzada entre estudiantes y contratantes permitiría enriquecer los perfiles y fortalecer la confianza en la plataforma.

2.5 Integración con herramientas externas

La conexión con sistemas institucionales, proveedores de identidad, plataformas de portafolios académicos o servicios externos mediante APIs abriría nuevas posibilidades de automatización e interoperabilidad.

3 Conclusiones Generales

El desarrollo del proyecto permitió validar la factibilidad técnica de una plataforma inteligente de emparejamiento entre estudiantes y unidades productivas contratantes. La integración de microservicios, un motor de IA basado en similitud textual y una arquitectura escalable demostró que es posible automatizar procesos de selección manteniendo precisión, estabilidad y usabilidad. La Prueba de Concepto (PoC) y las fases posteriores de desarrollo evidenciaron que el enfoque adoptado es adecuado, reproducible y alineado con las necesidades identificadas.

La metodología aplicada —basada en iteraciones semanales y ajustes incrementales— permitió responder de manera flexible a los retos técnicos encontrados, especialmente en etapas de experimentación con IA y ajustes en la comunicación entre microservicios. El sistema resultante cumple con el objetivo general del trabajo de grado y deja sentadas las bases para su evolución futura, tanto a nivel técnico como funcional.

IX REFERENCIAS

Referencias

- [1] Efe, A. (2025, 26 de febrero). La tasa de desempleo de los jóvenes en Latinoamérica es tres veces mayor que de los adultos. www.revistaeyn.com.https://www.revistaeyn.com/centroamericaymundo/la-tasa-de-desempleo-de-los-jovenes-en-latinoamerica-es-tres-veces-mayor-que-de-los-adultos-LB24528762
- [2] González, D. M. (2025, 13 de febrero). Colombia redujo el número de 'ninus', pero el desempleo juvenil sigue sin solución, estas son las cifras. *Infobae*. [https://www.infobae.com/colombia/2025/02/13/colombia-redujo-el-numero-de-ninus-pero-el-desempleo-juvenil-sigue-sin-solucion-estas-son-las-cifras/](http://www.infobae.com/colombia/2025/02/13/colombia-redujo-el-numero-de-ninus-pero-el-desempleo-juvenil-sigue-sin-solucion-estas-son-las-cifras/)
- [3] Comisión Económica para América Latina y el Caribe (CEPAL). (2006). *Los jóvenes y el empleo en América Latina: desafíos y perspectivas ante el nuevo escenario laboral*. Santiago: CEPAL. <https://repositorio.cepal.org/handle/11362/1902>
- [4] Organización Internacional del Trabajo. (2025). *Juventud en cambio 2025: Informe sobre el trabajo juvenil y las transiciones laborales*. Ginebra: OIT. <https://www.ilo.org/sites/default/files/2025-02/Informe%20juventud%20en%20cambio%202025.pdf>
- [5] Arias Ortiz, E., Cruz-Aguayo, Y., & Prada, M. F. (2021). *El futuro del trabajo en América Latina y el Caribe: ¿Cuáles son las tendencias en educación postsecundaria?* Banco Interamericano de Desarrollo. <https://publications.iadb.org/es/publications/spanish/viewer/El-futuro-del-trabajo-en-America-Latina-y-el-Caribe-cuales-son-las-tendencias-en-educacion-postsecundaria.pdf>
- [6] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805*. <https://arxiv.org/abs/1810.04805>
- [7] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., ... & Zoph, B. (2023). GPT-4 Technical Report. *arXiv:2303.08774*. <https://arxiv.org/abs/2303.08774>
- [8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv:1706.03762*. <https://arxiv.org/abs/1706.03762>
- [9] Comisión Económica para América Latina y el Caribe (CEPAL). (2015). *Juventudes: realidades y retos para un desarrollo con igualdad*. Santiago: CEPAL. <https://repositorio.cepal.org/server/api/core/bitstreams/12c9d979-7f2c-4ad4-ba61-e4834289794b/content>
- [10] DAMA International. (2025, 8 de octubre). *DAMA® Data Management Body of Knowledge (DAMA-DMBOK®)*. <https://dama.org/learning-resources/dama-data-management-body-of-knowledge-dmbok>
- [11] Iurillo, M. (2025, 26 de junio). Todo sobre #DMBOK 3.0. *DAMA España*. <https://www.damaspain.org/articulos/todo-sobre-dmbok-3-0>

X ANEXOS

1. Diseño de la Solución

Archivo: Disenio_de_la_Solucion.pdf

2. Documentación de Pruebas – Control de Calidad

Archivo: Documentacion_de_Pruebas_Control_de_Calidad.pdf

3. Especificación de Requerimientos

Archivo: Especificacion_de_Requerimientos.pdf

4. Plan de Proyecto

Archivo: Plan_de_Proyecto.pdf

5. Propuesta de Trabajo de Grado

Archivo: Propuesta_de_Trabajo_de_Grado.pdf

6. Código Fuente

Archivo: <https://github.com/orgs/FIRSTEPSJAV/repositories>