Grupo 8

Observatorio de demanda laboral en América Latina

Nicolas Francisco Camacho Alarcón

Alejandro Pinzón Fajardo

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2025

CIS2025CP08

Observatorio de demanda laboral en América Latina

Autor(es):

Nicolas Francisco Camacho Alarcón
Alejandro Pinzón Fajardo

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

Director

Ing. Luis Gabriel Moreno Sandoval

Jurados del Proyecto de Grado

Ing. ¡¡Nombre Jurado 1¿¿
Ing. ¡¡Nombre Jurado 2¿¿

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
Noviembre, 2025

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

Rector de la Pontificia Universidad Javeriana

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. Carlos Andrés Parra Acevedo

Director del Departamento de Ingeniería de Sistemas

Ing. César 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

Nicolás Francisco Camacho Alarcón

Llegar hasta aquí ha sido un camino largo y sinuoso, lleno de dudas, decisiones difíciles y momentos en los que pensé que no podría seguir adelante. Tras un camino de búsqueda que me llevó por tres carreras diferentes y siete años de aprendizaje constante, hoy puedo decir que cada paso, cada tropiezo y cada nueva dirección me trajeron exactamente donde debía estar.

En primer lugar, quiero agradecer a mis padres, quienes desde pequeño me impulsaron a crecer académicamente, a alcanzar nuevas metas y a nunca conformarme. Gracias por apoyarme en cada cambio de rumbo, por entender mis búsquedas y por nunca dejar de creer en mí, incluso cuando yo mismo dudaba. Su amor y esfuerzo constante han sido el pilar que me sostuvo en los momentos más difíciles. A mi hermana, mi apoyo incondicional y prácticamente mi mejor amiga, gracias por estar siempre ahí, por escucharme, por animarme y por ser esa compañía invaluable en cada etapa de este proceso. A mi abuela, por su cariño y por ser parte fundamental de esta familia que me ha dado todo. A toda mi familia, gracias por comprenderme, por acompañarme en cada gran decisión, por ayudarme a levantarme cuando sentí que no podía más, y por estar presentes en cada momento importante de mi vida.

Un agradecimiento muy especial a Lucy, mi perrita, quien con su compañía silenciosa y su alegría incondicional trajo luz a cada día, sin importar lo difícil que fuera. Su presencia fue un refugio de paz y felicidad en medio de las noches largas y los momentos de mayor estrés.

A todos los amigos que he ido conociendo semestre a semestre desde que entré a primer semestre de música, pasando por electrónica y llegando a sistemas: gracias por hacer de estos años una experiencia llena de aprendizajes, risas y compañía. Cada uno de ustedes aportó algo valioso a mi camino y me ayudó a convertirme en quien soy hoy.

A todos mis profesores, gracias por inspirarme semestre tras semestre a ser mejor, a alcanzar nuevos niveles y a recordarme constantemente que amo aprender y que me apasiona sentirme retado. Su dedicación y enseñanzas fueron fundamentales para mantener viva esa chispa de curiosidad que me impulsa.

A Alejandro, mi compañero de tesis, gracias por haberme acompañado este año y por haber sido el apoyo que necesité en el tramo final para lograrlo. Tu paciencia y tu apoyo emocional fueron invaluables, y estoy profundamente agradecido por haber compartido este proceso contigo.

A Gabriel, nuestro director de tesis, por inspirarnos y ayudarnos a encontrar algo realmente retador e innovador que nos permitiera hacer algo nuevo en la industria. Tu guía fue clave para que este proyecto tomara forma y alcanzara su verdadero potencial.

A Sezzle, mi empresa, gracias por permitirme terminar mi carrera al tiempo que trabajo, y por apoyarme para salir adelante. Su comprensión y flexibilidad fueron fundamentales para lograr este

objetivo.

Finalmente, quiero agradecer a ese Nicolás que decidió levantarse una vez más, por haber tenido la valentía de buscar su verdadero camino y por haber confiado en que encontraría su lugar, incluso cuando el camino no estaba claro. Este logro también es tuyo.

A todos los que han sido parte de este camino: gracias. Este trabajo es el resultado no solo de mi esfuerzo, sino del amor, la paciencia y el acompañamiento de cada uno de ustedes.

Alejandro Pinzón Fajardo

La culminación de este trabajo representa mucho más que un logro académico; es el resultado de años de esfuerzo, constancia y del apoyo invaluable de las personas que me han acompañado a lo largo de este camino.

Mi familia ha sido el pilar fundamental de este proceso. Su amor, paciencia y fe inquebrantable me brindaron la fortaleza necesaria para avanzar incluso en los momentos más difíciles. Gracias por enseñarme el valor del trabajo honesto, la disciplina y la perseverancia que hoy hacen posible este resultado.

Durante mi formación en la Pontificia Universidad Javeriana, tuve el privilegio de aprender de profesores que, más allá de impartir conocimiento, despertaron en mí la curiosidad, el pensamiento crítico y la pasión por la ingeniería. Su dedicación y compromiso fueron esenciales para mi crecimiento profesional y personal.

El desarrollo de esta tesis no habría sido posible sin la guía del profesor Gabriel, cuya orientación y criterio fueron determinantes para dar forma y sentido al proyecto. Su apoyo y su exigencia académica nos permitieron alcanzar un resultado del que me siento profundamente orgulloso.

Finalmente, compartir este proceso con Nicolás fue una experiencia gratificante. Su compromiso, su disposición al trabajo en equipo y su acompañamiento hicieron de esta etapa un recorrido más llevadero y enriquecedor, en el que aprendimos tanto del proyecto como de nosotros mismos.

A todos ellos, mi gratitud más sincera.

CONTENIDO

RESUMEN

El mercado laboral tecnológico latinoamericano carece de sistemas automatizados para caracterizar la demanda de habilidades técnicas IT de manera sistemática y actualizada, enfrentando el desafío de capturar tecnologías emergentes que evolucionan más rápido que taxonomías oficiales. Este proyecto evaluó la viabilidad de construir un observatorio automatizado que recolectó 30,660 ofertas laborales de Colombia, México y Argentina mediante web scraping de seis portales, focalizándose en extracción de hard skills (lenguajes de programación, frameworks, herramientas cloud/DevOps, metodologías ágiles). Se implementó Pipeline A (NER + Regex) procesando el corpus completo con latencia de 0.97s por oferta, y se desarrolló Pipeline B experimental (LLM Gemma 3 4B) evaluado sobre gold standard de 300 ofertas anotadas manualmente con 6,174 hard skills. Los resultados demostraron superioridad de modelos de lenguaje para aproximar mapeo humano de competencias técnicas (F1=84.26 % vs 72.53 %), capturando habilidades implícitas inferibles del contexto y tecnologías emergentes ausentes en vocabularios controlados que métodos basados en patrones omiten. El sistema normalizó extracciones contra taxonomía ESCO v1.1.0 extendida con 152 habilidades técnicas de O*NET mediante matcher conservador de dos capas (exacto + difuso threshold 0.85), alcanzando 12.6 % de cobertura. Experimentos con matcher enhanced aumentaron cobertura a 25 %, pero ambos evidencian que 70-87 % de habilidades detectadas son emergentes no presentes en taxonomías estándar (Next.js, Tailwind CSS, Terraform, Bun), validando necesidad de captura automática de vocabulario IT actual. El clustering no supervisado (UMAP reducción 768D$\rightarrow$2D + HDBSCAN density-based) identificó 34-53 familias tecnológicas dependiendo de configuración pipeline, sin categorías predefinidas. Los hallazgos validan tres conclusiones: (1) viabilidad técnica de observatorios basados en scraping multi-portal; (2) superioridad de LLMs versus métodos deterministas para extracción semántica y detección de emergentes, con trade-off de costo computacional 43$\times$ mayor; (3) obsolescencia crítica de taxonomías oficiales para vocabulario IT actual, evidenciando necesidad de actualización continua.

INTRODUCCIÓN

El mercado laboral tecnológico en América Latina atraviesa una transformación profunda impulsada por la digitalización de la economía. La pandemia de COVID-19 aceleró este proceso, intensificando la demanda de habilidades técnicas IT especializadas (lenguajes de programación, frameworks, herramientas cloud/DevOps, metodologías ágiles) y exponiendo las brechas de capital humano en la región [1]. En este escenario, identificar con precisión qué competencias técnicas están siendo requeridas por el mercado se ha vuelto estratégico para gobiernos que diseñan políticas de formación, instituciones educativas que ajustan sus programas, y profesionales que planifican su desarrollo de carrera en el sector tecnológico.

Sin embargo, medir esta demanda de manera sistemática presenta desafíos importantes. Los portales de empleo en la región publican vacantes en formatos heterogéneos, sin vocabularios estandarizados, y con alta volatilidad [2]. Las encuestas tradicionales, aunque valiosas, suelen ser retrospectivas y de baja periodicidad, limitando su utilidad para capturar tecnologías emergentes que evolucionan más rápido que los ciclos de actualización de instrumentos de medición [3]. Los estudios previos en países como Colombia, México y Argentina han aportado evidencia empírica importante, pero se han basado principalmente en análisis de frecuencia de términos y clasificaciones manuales, enfoques que no logran capturar habilidades implícitas inferibles del contexto ni identificar tecnologías emergentes ausentes en taxonomías oficiales [4, 5].

Este proyecto evaluó la viabilidad técnica de construir un observatorio automatizado de demanda laboral tecnológica que recolectó 30,660 ofertas de empleo de Colombia, México y Argentina mediante web scraping de seis portales. Se implementó Pipeline A basado en NER y expresiones regulares para procesamiento escalable del corpus completo (latencia 0.97s por oferta), y se desarrolló Pipeline B experimental con LLM Gemma 3 4B evaluado sobre un gold standard de 300 ofertas anotadas manualmente con 6,174 hard skills técnicas. La comparación rigurosa mediante evaluación dual Pre-ESCO y Post-ESCO demostró que modelos de lenguaje aproximan mejor el mapeo humano de competencias (F1=84.26 % vs 72.53 % de métodos tradicionales), capturando habilidades implícitas inferibles del contexto y tecnologías emergentes que métodos basados en patrones omiten. El sistema normalizó extracciones contra taxonomía ESCO v1.1.0 extendida con 152 habilidades de O*NET mediante matching de dos capas (exacto + difuso), y aplicó clustering no supervisado con UMAP y HDBSCAN descubriendo 156 familias tecnológicas sin categorías predefinidas.

Las contribuciones principales de este proyecto son cuatro. Primero, la validación empírica de viabilidad técnica de observatorios automatizados basados en web scraping multi-portal y multi-país para caracterización de demanda laboral tecnológica a escala regional. Segundo, evidencia cuantitativa

de superioridad de modelos de lenguaje sobre métodos deterministas para aproximar juicio humano en extracción de habilidades técnicas (mejora de +11.73pp en F1-Score), incluyendo capacidad de inferir competencias implícitas y capturar tecnologías emergentes, con caracterización explícita de trade-off en costo computacional (43$\times$ mayor latencia). Tercero, identificación de limitaciones críticas de taxonomías internacionales para vocabulario IT moderno, evidenciando mediante experimentación con matcher evolutivo (sin bias $\rightarrow$ con mapeos curados) que proporción significativa de extracciones corresponden a tecnologías ausentes en estándares oficiales como ESCO v1.1.0. Cuarto, metodología de evaluación dual Pre-ESCO y Post-ESCO sobre gold standard anotado manualmente, permitiendo comparación sistemática de pipelines distinguiendo capacidad de extracción pura versus alineación con taxonomías controladas.

DESCRIPCIÓN GENERAL

2.1 Oportunidad y problema

2.1.1 Contexto del problema

El mercado laboral en América Latina se encontró, durante la última década, en una compleja encrucijada definida por la confluencia de dos fuerzas a menudo contrapuestas: una acelerada transformación digital y la persistencia de desafíos estructurales, como una elevada informalidad laboral y brechas de capital humano [2]. La pandemia de COVID-19 actuó como un catalizador sin precedentes, intensificando la adopción de tecnologías y, con ello, la demanda de competencias digitales, al tiempo que exponía la vulnerabilidad de los mercados de trabajo de la región [1]. Este dinamismo generó el riesgo de que la automatización y la digitalización, de no ser gestionadas estratégicamente, pudiesen exacerbar las desigualdades existentes, conduciendo a una mayor polarización y segmentación social [2].

Para analizar este fenómeno regional de manera tangible y robusta, este proyecto seleccionó como casos de estudio a tres de las economías más grandes y digitalmente activas de habla hispana: Colombia, México y Argentina. La elección de estos países respondió a tres criterios estratégicos. Primero, su alto volumen de publicaciones de ofertas laborales en portales digitales aseguró la viabilidad de una recolección masiva de datos (web scraping), fundamental para el entrenamiento de modelos de lenguaje robustos [3-5]. Segundo, la existencia de estudios previos en cada país, aunque metodológicamente limitados, confirmó la pertinencia del problema y proporcionó una línea de base para la comparación [6, 7]. Y tercero, su diversidad en términos de realidades económicas, territoriales y de madurez digital permitió validar que la solución desarrollada fuese portable y adaptable a los distintos contextos que caracterizan a América Latina.

El caso de Colombia sirvió como una ilustración profunda de esta dinámica. El diagnóstico nacional previo al proyecto ya indicaba que el principal cuello de botella para la inclusión digital no era la falta de infraestructura, sino la brecha de capital humano. Específicamente, el "Índice de Brecha Digital" (IBD) del Ministerio de Tecnologías de la Información y las Comunicaciones reveló que la dimensión de "Habilidades Digitales" constituía el mayor componente individual de la brecha en el país. Esta evidencia fue posteriormente corroborada y cuantificada por el análisis empírico de la demanda laboral, el cual demostró que la pandemia generó un cambio estructural y persistente en el mercado. Se encontró que, en los 18 meses posteriores al inicio de la crisis sanitaria, las vacantes tecnológicas aumentaron en un 50 % en comparación con las no tecnológicas [3]. Este cambio no fue

solo cuantitativo, sino también cualitativo: se observó una marcada caída en la demanda de herramientas ofimáticas tradicionales como Excel (cuya mención en ofertas cayó del 35.8 % en 2018 al 17.4 % en 2023) y un surgimiento exponencial de tecnologías especializadas asociadas al desarrollo web y la gestión de datos, como bases de datos NoSQL (12.3 %), el framework Django (5.5 %) y la librería React (5.3 %) para el año 2023 [3].

2.1.2 Formulación del problema

A pesar de que el contexto del problema —la creciente e insatisfecha demanda de habilidades tecnológicas— estaba claramente identificado, los métodos existentes en la región para analizarlo presentaban limitaciones metodológicas significativas que impedían una comprensión profunda y ágil del fenómeno. Los estudios de referencia en los países seleccionados, si bien valiosos para establecer tendencias macro, se basaron en enfoques de análisis léxico y reglas manuales. En Colombia, el análisis se centró en un sistema de clasificación basado en la Clasificación Internacional Uniforme de Ocupaciones (CIUO), utilizando algoritmos de emparejamiento de texto con tokenización y métricas de similitud basadas en n-gramas [3]. De forma análoga, en Argentina, los estudios se concentraron en técnicas de minería de texto con análisis de frecuencias y bigramas para identificar patrones en las ofertas del sector TI [4]. En México, el enfoque combinó datos de encuestas con scraping de portales, apoyándose en el análisis de frecuencia de términos y la creación de tipologías manuales para segmentar las habilidades [5].

La limitación fundamental compartida por estos enfoques es su dependencia de la correspondencia léxica explícita, lo que los hace incapaces de capturar la riqueza semántica del lenguaje. Estos métodos no podían detectar habilidades implícitas (aquellas que se infieren del contexto de un cargo pero no se mencionan directamente), gestionar la ambigüedad del lenguaje informal o el uso de anglicismos técnicos (“Spanglish”), ni identificar clústeres de competencias emergentes que aún no forman parte de taxonomías estandarizadas. La alta variabilidad en la redacción de las ofertas laborales, la falta de estructuras normalizadas y la rápida aparición de nuevas tecnologías hacían que estos sistemas fueran metodológicamente frágiles y requirieran un constante mantenimiento manual [2, 8].

En consecuencia, el problema específico que este proyecto abordó fue la ausencia de una herramienta automatizada y de extremo a extremo que, adaptada a las particularidades lingüísticas y estructurales del español latinoamericano, permitiera superar las limitaciones de los análisis léxicos tradicionales. Se identificó la necesidad de un sistema capaz de extraer, estructurar y analizar la evolución de las habilidades tecnológicas de manera semántica, escalable y con un mayor grado de autonomía, integrando para ello técnicas avanzadas de Procesamiento de Lenguaje Natural (NLP), enriquecimiento contextual con Large Language Models (LLMs) y algoritmos de agrupamiento no supervisado.

2.1.3 Propuesta de solución

Para dar respuesta al problema formulado, se diseñó e implementó un observatorio de demanda laboral tecnológica basado en un pipeline modular y automatizado, un proyecto enmarcado en las áreas de Ingeniería de Sistemas y Ciencia de Datos. El sistema fue concebido como una solución de extremo a extremo que integró las etapas de recolección, procesamiento, análisis semántico y segmentación de ofertas de empleo publicadas en Colombia, México y Argentina. El objetivo fue crear una arquitectura robusta, replicable y adaptada a las complejidades del contexto latinoamericano, superando las limitaciones de los enfoques puramente léxicos o manuales.

La solución se materializó a través de un sistema compuesto por módulos secuenciales y cohesivos. El primer módulo consistió en un motor de adquisición de datos que, mediante técnicas de web scraping, extrajo de forma sistemática y ética decenas de miles de ofertas laborales de portales de empleo clave en la región. El núcleo del sistema fue su arquitectura de extracción dual, compuesta por dos pipelines paralelos.

El primero, denominado Pipeline A (tradicional), implementó un método de extracción basado en Reconocimiento de Entidades Nombradas (NER) utilizando un EntityRuler de spaCy, poblado con la taxonomía completa de ESCO, combinado con expresiones regulares para capturar un baseline de habilidades explícitas de alta precisión.

El segundo, Pipeline B (basado en LLMs), empleó Large Language Models (LLMs) como Llama 3 para realizar una extracción semántica, capaz de identificar no solo habilidades explícitas sino también de inferir competencias implícitas a partir del contexto de la vacante, siguiendo enfoques de vanguardia [9, 10].

Posteriormente, un módulo de mapeo de dos capas normalizó las habilidades extraídas por ambos pipelines contra la taxonomía ESCO. La primera capa realizó una coincidencia léxica (exacta y difusa), mientras que la segunda ejecutó una búsqueda de similitud semántica de alto rendimiento, utilizando embeddings multilingües (E5) y un índice FAISS pre-calculado, inspirado en las arquitecturas de herramientas como ESCOX [11]. Finalmente, un módulo de análisis no supervisado aplicó una secuencia metodológica de embeddings, reducción de dimensionalidad con UMAP y agrupamiento con HDBSCAN para identificar clústeres de habilidades y perfiles emergentes, un enfoque validado por la literatura para el descubrimiento de estructuras en el mercado laboral [8].

2.1.4 Justificación de la solución

La solución implementada se justificó como una alternativa superior y mejor adaptada para el análisis de la demanda de habilidades en América Latina, ya que abordó directamente las debilidades metodológicas identificadas en los estudios previos. A diferencia de los enfoques basados exclusivamente en reglas léxicas [3, 4] o en el uso aislado de LLMs [10], la arquitectura de dos pipelines paralelos permitió una validación empírica cruzada: combinó la auditabilidad y alta precisión para habilidades conocidas del Pipeline A con la potencia inferencial y la capacidad de descubrir habilidades

implícitas del Pipeline B. Este diseño comparativo proveyó un marco para evaluar objetivamente el rendimiento de los LLMs, en lugar de depender únicamente de su capacidad "black-box".

Técnicamente, el sistema representó un avance significativo en escalabilidad y eficiencia. La implementación de un índice FAISS para la búsqueda semántica de similitud (una mejora sobre la propuesta original de ESCOX) permitió procesar grandes volúmenes de datos a una velocidad órdenes de magnitud superior a las búsquedas en bases de datos vectoriales convencionales, haciendo factible el análisis de todo el corpus recolectado [8, 11]. Adicionalmente, el sistema fue diseñado explícitamente para la realidad del español latinoamericano. Este enfoque abordó directamente una limitación crítica de trabajos de vanguardia en LLMs, los cuales se han desarrollado y validado casi exclusivamente sobre datasets en inglés [9], ignorando las particularidades lingüísticas (como el "Spanglish") del dominio tecnológico en la región.

Finalmente, el valor agregado del proyecto residió en su síntesis estratégica de metodologías de vanguardia. El sistema no se limitó a una sola técnica, sino que articuló la cobertura del scraping regional, la potencia de los LLMs ajustados para generar salidas estructuradas [9], y la capacidad estructuradora del clustering semántico [8]. Al hacerlo, se desarrolló un observatorio más completo, robusto y metodológicamente transparente que las alternativas existentes, estableciendo una base sólida y replicable para el monitoreo dinámico de la demanda laboral en la región.

2.2 Descripción del proyecto

El proyecto se concibió como un observatorio automatizado para capturar, normalizar y analizar avisos de empleo en Latinoamérica. Se integraron múltiples portales (CO, MX y AR), se diseñó una base de datos relacional con soporte vectorial, y se implementó un pipeline de extracción de habilidades (NER/regex/LLM) alineadas a ESCO, con generación de indicadores, visualizaciones y reportes. Operativamente, se planificó escalar hasta 600.000 avisos para la defensa, garantizando calidad, trazabilidad y reproducibilidad.

2.2.1 Objetivo general

Desarrollar un sistema que permita procesar y segmentar la demanda de habilidades tecnológicas en Colombia, México y Argentina, mediante técnicas de procesamiento de lenguaje natural.

2.2.2 Objetivos específicos

- Construir un estado del arte exhaustivo para comparar trabajos existentes en el ámbito de observatorios laborales automatizados y técnicas de procesamiento de lenguaje natural en español.
- Diseñar una arquitectura modular, escalable y reutilizable para el observatorio laboral automatizado, fundamentada en las mejores prácticas identificadas en el estado del arte.

- Implementar e integrar técnicas de inteligencia artificial para la identificación, normalización y agrupación semántica de habilidades tecnológicas en ofertas laborales en español.
- Validar el desempeño y la robustez de la arquitectura y los modelos propuestos mediante métricas cuantitativas y estudios empíricos.

2.2.3 Entregables, estándares y justificación

El desarrollo del observatorio se materializó en un conjunto estructurado de entregables alineados con estándares de ingeniería de software y buenas prácticas de la industria. La Tabla 2.1 presenta cada componente desarrollado, los estándares técnicos que guiaron su implementación, y la justificación que fundamenta su adhesión a dichos estándares.

Tabla 2.1: Entregables, Estándares y Justificación Técnica

Entregable	**Estándares asociados**	**Justificación**
Repositorio de código (spiders, orquestador, pipelines)	PEP 8/257/484; Conv. Commits; SemVer	Mantenibilidad, legibilidad y control de versiones.
Esquema BD y migraciones (PostgreSQL + pgvector)	Normalización (3NF); SQL best practices	Integridad, trazabilidad y soporte a consultas vectoriales.
Spiders y configuración de scraping	Polite crawling (delays/retries); manejo anti-bots	Captura estable a escala y resiliencia ante cambios UI.
Orquestador CLI + scheduler	CLI UX (Typer); jobs idempotentes	Operación reproducible, programable y auditable.
Módulo de extracción/normalización de habilidades	ISO/IEC/IEEE 29148 (requisitos); ESCO	Consistencia semántica y comparabilidad entre países.
Embeddings y análisis (E5, UMAP, HDBSCAN)	Procedimientos reproducibles; semillas fijas	Descubrimiento de patrones y replicabilidad experimental.
Datasets consolidados (CSV/JSON) + diccionario de datos	Esquemas declarativos; control de versiones	Consumo externo y verificación de calidad.
Documentación técnica y de proyecto (SRS, SPMP, VFP, manuales)	IEEE 1058 (plan de proyecto); 29148 (requisitos)	Alineación con buenas prácticas y transferencia de conocimiento.
Reportes y visualizaciones (PDF/PNG/CSV)	Principios de visualización; metadatos	Comunicación clara de hallazgos a públicos no técnicos.
Plan de operación y mantenimiento (Docker/monitoring)	Buenas prácticas Docker/Logging	Despliegue consistente y observabilidad del sistema.

CONTEXTO DEL PROYECTO

3.1 Conceptos Fundamentales del Dominio

Antes de abordar los aspectos técnicos de la solución, es fundamental establecer el vocabulario específico del dominio del mercado laboral. Esta sección define los conceptos clave que estructuran el problema: las ofertas laborales como fuente primaria de datos, las habilidades como unidades de análisis, su tipología (hard vs. soft skills), y el rol de las taxonomías en la estandarización del conocimiento ocupacional.

3.1.1 Ofertas Laborales y Avisos de Empleo

Una oferta laboral (job posting) es un anuncio público de una vacante que especifica título del cargo, descripción de funciones, requisitos de formación, habilidades técnicas requeridas y condiciones del puesto [3]. Las ofertas publicadas en portales de empleo constituyen una fuente de datos de alta frecuencia sobre demanda laboral, permitiendo capturar tendencias emergentes con granularidad temporal superior a encuestas tradicionales [3, 6]. Sin embargo, representan únicamente "demanda revelada", excluyendo vacantes cubiertas por redes internas [4]. Los portales digitales (elempleo.com, computrabajo.com, LinkedIn) operan como observatorios del mercado laboral LATAM, proveyendo texto no estructurado procesable mediante NLP.

3.1.2 Habilidades, Competencias y Skills

El término skill (habilidad) se define como la unidad básica de conocimiento, capacidad técnica o destreza requerida para un rol laboral [11]. Para este trabajo, se adopta la definición operacional de skill como cualquier mención textual en ofertas que describa capacidades requeridas: lenguajes ("Python"), metodologías ("Scrum"), herramientas ("Docker"), conceptos técnicos ("microservicios") o competencias transversales ("trabajo en equipo") [10]. Las skills operan como proxy de demanda laboral: frecuencia refleja intensidad de demanda, co-ocurrencia revela ecosistemas tecnológicos [8].

3.1.3 Tipología de Habilidades: Hard Skills vs. Soft Skills

Hard Skills (habilidades técnicas) son capacidades específicas, medibles y enseñables mediante educación formal, caracterizadas por ser específicas del rol, tener evaluación objetiva y experimentar evolución rápida [3]. Ejemplos IT: lenguajes (Python, Java), frameworks (React, Django), cloud

(AWS, Azure), bases de datos (PostgreSQL, MongoDB), metodologías (Agile, DevOps), herramientas (Docker, Kubernetes).

Soft Skills (habilidades transversales) son capacidades relacionales e interpersonales transversales a dominios con evaluación subjetiva y relevancia estable [4]: liderazgo, comunicación, trabajo en equipo, resolución de problemas, adaptabilidad.

Para NLP automatizado, las hard skills presentan ventaja por expresión léxica consistente ("Docker" con variantes limitadas), mientras soft skills exhiben alta variabilidad lingüística ("trabajo en equipo" = "colaboración", "espíritu colaborativo", "teamwork") [5].

3.1.4 Hard Skills IT en el Análisis Automatizado

El análisis automatizado de ofertas laborales típicamente se enfoca en hard skills IT por tres razones. Primero, las nomenclaturas estandarizadas globalmente facilitan extracción automatizada con alta precisión versus la ambigüedad semántica de soft skills [10]. Segundo, la trazabilidad a taxonomías ESCO (13k+ skills) y O*NET (233 skills subset) provee vocabularios controlados exhaustivos [3, 11]. Tercero, la identificación de tecnologías emergentes provee inteligencia accionable para actualización curricular [2].

Las hard skills IT abarcan categorías como: lenguajes de programación (Python, Java, JavaScript), frameworks web (Django, React, Spring Boot), bases de datos relacionales y NoSQL (PostgreSQL, MongoDB), plataformas cloud (AWS, Azure, GCP), herramientas DevOps (Docker, Kubernetes, Terraform), metodologías ágiles (Scrum, Kanban, CI/CD) y dominios especializados (Machine Learning, Data Science, Blockchain, Ciberseguridad).

3.1.5 Taxonomías de Habilidades y Ocupaciones

Una taxonomía de habilidades es un sistema jerárquico que organiza skills en categorías, establece relaciones semánticas y provee identificadores únicos. Cumplen tres funciones principales. Primero, la estandarización mediante la unificación de variantes léxicas ("JS" $\rightarrow$ "JavaScript" en ESCO) [11]. Segundo, la comparabilidad internacional ya que ESCO es multilingüe (27 idiomas UE), facilitando análisis transnacionales [12]. Tercero, el mapeo de relaciones al conectar skills con ocupaciones y sectores económicos.

Limitación fundamental de las taxonomías: actualización lenta versus cambio tecnológico acelerado. ESCO v1.1.0 (2021) excluye tecnologías post-2022: "ChatGPT", "LangChain", "Tailwind CSS", "dbt", "Terraform" [3], generando "skills emergentes" sin representación en vocabularios controlados. Esta brecha temporal entre ciclos de actualización taxonómica (cada 2-3 años) y aparición de nuevas tecnologías (semanas/meses) plantea un desafío metodológico para los sistemas de análisis automatizado.

3.2 Antecedentes Conceptuales

Para comprender el diseño y la justificación de la solución desarrollada, es necesario fundamentar el proyecto en una serie de conceptos clave provenientes de la ingeniería de sistemas, la ciencia de datos y, fundamentalmente, del Procesamiento de Lenguaje Natural (NLP). Estos conceptos no actúan de forma aislada, sino que se articulan en un flujo metodológico que va desde la adquisición de datos brutos hasta la generación de conocimiento estructurado sobre el mercado laboral.

3.2.1 Web Scraping y Adquisición de Datos

El punto de partida del observatorio es la recolección de datos a gran escala desde fuentes web públicas. Esta tarea se realiza mediante Web Scraping, una técnica de extracción automatizada de información desde el código HTML de las páginas web [13]. En el contexto del mercado laboral, esta técnica ha demostrado ser fundamental para obtener datos de alta frecuencia y granularidad directamente de los portales de empleo, superando las limitaciones de las encuestas y los reportes institucionales, que suelen ser retrospectivos y de baja periodicidad [3, 6].

El web scraping se distingue del simple *crawling* en que no solo navega páginas web, sino que extrae y estructura información específica. Las técnicas modernas incluyen parsers HTML (BeautifulSoup, lxml), headless browsers (Playwright, Puppeteer) para contenido dinámico, control de rate limiting con throttling y backoff exponencial, y rotación de user-agents para evitar bloqueos.

La implementación debe seguir principios éticos y legales: respeto del archivo `robots.txt`, delays entre peticiones, registro de fuentes con sellos de tiempo, validación de datos extraídos y monitoreo de cambios en la estructura del DOM.

3.2.2 Procesamiento de Lenguaje Natural (NLP)

Una vez extraído el contenido textual de las ofertas laborales, el siguiente paso es prepararlo para el análisis computacional mediante técnicas de Procesamiento de Lenguaje Natural.

El preprocesamiento es fundamental para estandarizar los datos textuales. La Tokenización consiste en segmentar el texto en unidades mínimas o "tokens" (generalmente palabras o signos de puntuación) [10], transformando cadenas continuas en secuencias discretas procesables. La Lematización reduce las palabras a su forma base o raíz gramatical, permitiendo agrupar variaciones morfológicas (por ejemplo, "programar", "programando" y "programado" se unifican bajo el lema "programar") [2].

Con el texto limpio y normalizado, el núcleo del desafío consiste en la extracción de habilidades mediante un enfoque híbrido. Las Expresiones Regulares (Regex) permiten identificar secuencias de texto específicas con formatos predecibles [8], siendo efectivas para capturar tecnologías con nomenclaturas estandarizadas. El Reconocimiento de Entidades Nombradas (NER) es una técnica de NLP

diseñada para identificar y clasificar entidades como habilidades y competencias [9], permitiendo reconocer habilidades en contextos gramaticales complejos.

3.2.3 Large Language Models (LLMs)

Para superar las limitaciones de la extracción de menciones explícitas, el proyecto incorpora Large Language Models (LLMs). Estos modelos de lenguaje a gran escala, como GPT o Llama 3, poseen capacidades de razonamiento contextual que permiten abordar desafíos más complejos [9].

A través del Prompt Engineering, es posible guiar a los LLMs para realizar tareas de enriquecimiento semántico: distinción entre habilidades explícitas e implícitas [10], normalización de variantes terminológicas, clasificación según taxonomías predefinidas y generación de salidas estructuradas en formatos como JSON.

Existen diferentes modalidades de aplicación: zero-shot learning (sin ejemplos previos), few-shot learning (con algunos ejemplos en el prompt) [10] y fine-tuning (re-entrenamiento sobre datasets específicos) [9, 14].

3.2.4 Embeddings Semánticos y Representación Vectorial

Las habilidades extraídas deben representarse de forma que permita su análisis cuantitativo. Los Embeddings Semánticos son representaciones vectoriales en un espacio de alta dimensionalidad donde la distancia entre vectores refleja la similitud semántica entre textos [12]. Esto permite capturar relaciones semánticas complejas, realizar búsquedas por similitud eficientemente y agrupar habilidades relacionadas.

Dado que las ofertas laborales en América Latina contienen términos técnicos en inglés (“Spanglish”), es crucial el uso de Embeddings Multilingües, modelos entrenados para que textos con el mismo significado en diferentes idiomas tengan representaciones vectoriales cercanas en el mismo espacio semántico [2, 12]. Modelos populares incluyen E5-large, Sentence Transformers basados en BERT y multilingual-e5-base.

3.2.5 Análisis No Supervisado: UMAP y HDBSCAN

Para descubrir patrones y estructuras latentes, se aplica un pipeline de análisis no supervisado. Debido a que los embeddings son vectores de muy alta dimensionalidad (768 dimensiones), lo que genera la “maldición de la dimensionalidad”, se aplica UMAP (Uniform Manifold Approximation and Projection), un algoritmo no lineal que reduce dimensiones preservando la estructura local y global, superior a métodos lineales como PCA [8].

Sobre los datos reducidos se aplica HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), un algoritmo de clustering basado en densidad. A diferencia de K-Means, HDBSCAN no requiere especificar el número de clústeres, identifica grupos de formas arbitrarias, separa puntos que no pertenecen a ningún grupo como “ruido” y funciona con clústeres de densidades

variables [8]. Esta secuencia metodológica permite la identificación automática de "ecosistemas de habilidades" y perfiles laborales emergentes.

3.2.6 Taxonomías Estandarizadas: ESCO e ISCO

Para asegurar la comparabilidad y estandarización de resultados, el sistema integra taxonomías internacionales. ESCO (European Skills, Competences, Qualifications and Occupations) es una taxonomía multilingüe desarrollada por la Comisión Europea que clasifica más de 13,000 habilidades y conocimientos de manera jerárquica [11]. ISCO-08 (International Standard Classification of Occupations) es un estándar internacional de la OIT para clasificar ocupaciones, proporcionando un marco común para comparar datos ocupacionales entre países. Adicionalmente, se toma como referencia O*NET, un portal del Departamento de Trabajo de EE. UU. que reporta habilidades de alta demanda [3].

3.3 Análisis del Contexto

El desafío de extraer, analizar y comprender la demanda de habilidades a partir de ofertas de empleo en línea ha sido abordado desde múltiples frentes en la literatura académica y aplicada. Si bien el objetivo es común traducir texto en conocimiento accionable sobre el mercado laboral, las aproximaciones metodológicas varían significativamente en complejidad, escalabilidad y profundidad semántica.

Para posicionar adecuadamente la contribución de este proyecto, fue necesario realizar un análisis crítico de las soluciones existentes a nivel global, agrupadas en tres grandes líneas de trabajo: primero, enfoques regionales en América Latina basados en análisis léxico y reglas manuales; segundo, la frontera de la extracción con Large Language Models (LLMs) mediante prompting y fine-tuning; y tercero, pipelines semánticos con descubrimiento no supervisado mediante embeddings y clustering.

El siguiente análisis demostrará que ninguna de estas líneas, de forma aislada, resolvía los desafíos metodológicos, geográficos y lingüísticos del mercado laboral tecnológico en América Latina. Esta fragmentación justificó la necesidad de una solución sintética y adaptada.

3.3.1 Enfoques Regionales: Caracterización del Mercado con Métodos Léxicos

La primera línea de trabajo comprende estudios pioneros en América Latina que validaron el uso de portales de empleo en línea como fuente de datos, pero emplearon metodologías de procesamiento de texto basadas en análisis léxico, frecuencias de términos y reglas manuales.

El estudio más completo fue el de Rubio Arrubla (2024) para el mercado colombiano [3]. Este trabajo construyó una base de datos masiva mediante web scraping del portal elempleo.com para el periodo 2018-2023, abarcando más de 500,000 ofertas laborales. Su principal aporte fue la caracterización cuantitativa del impacto de la pandemia, demostrando un cambio estructural: las vacantes

tecnológicas aumentaron 50 % en 18 meses post-pandemia. Se observó una caída en la demanda de herramientas ofimáticas tradicionales como Excel (35.8 % en 2018 a 17.4 % en 2023) y un surgimiento exponencial de tecnologías especializadas como NoSQL (12.3 %), Django (5.5 %) y React (5.3 %) para 2023 [3].

Metodológicamente, implementó una tipología propia de habilidades y clasificó vacantes mediante emparejamiento de texto basado en n-gramas y similitud contra la Clasificación Internacional Uniforme de Ocupaciones (CIUO). Sin embargo, su dependencia de la coincidencia léxica fue una limitación: el método perdía eficiencia a medida que aumentaban las palabras en los títulos, al no capturar el contexto general [3].

De forma análoga, Aguilera y Méndez (2018) para Argentina extrajeron datos de ZonaJobs y Bumeran mediante análisis de frecuencias y bigramas [4]. Para estandarizar el vocabulario informal, construyeron una lista de palabras clave semi-manual, limitando la escalabilidad y adaptabilidad a nuevas tecnologías.

Para México, Martínez Sánchez (2024) combinó datos de encuestas oficiales con scraping, basándose en frecuencia de términos y tipología manual para segmentar habilidades [5].

Estos estudios regionales fueron cruciales para establecer la viabilidad de la recolección de datos, pero expusieron una brecha fundamental compartida: su dependencia de la correspondencia léxica explícita. Al basarse en frecuencias, n-gramas o listas predefinidas, estos sistemas eran metodológicamente frágiles ante la ambigüedad y variabilidad del lenguaje natural.

El Banco Interamericano de Desarrollo (BID) señaló la falta de pipelines modernos y automatizados en la región, destacando que la mayoría todavía se basa en reglas fijas o mapeos manuales sin incorporar embeddings ni NLP avanzado [2]. Esta constatación institucional refuerza la conclusión de un vacío sistémico: la ausencia de una solución que superara los enfoques léxicos para proporcionar análisis semántico, dinámico y escalable.

3.3.2 La Frontera de la Extracción: El Uso de Large Language Models

Paralelamente, una segunda línea de investigación a nivel internacional ha explorado el uso de LLMs para superar las limitaciones de los métodos léxicos, representando la frontera del estado del arte en extracción semántica.

Nguyen et al. (2024) investigaron el uso de LLMs de propósito general (GPT-3.5, GPT-4) en modalidad prompting sin re-entrenamiento (few-shot learning) [10]. Experimentaron con dos formatos de salida: extracción directa (“EXTRACTION-STYLE”) y etiquetado (“NER-STYLE”). Aunque los LLMs no igualaron la precisión de modelos supervisados tradicionales, demostraron capacidad superior para interpretar frases sintácticamente complejas. Sin embargo, el estudio advirtió sobre limitaciones: inconsistencia en formatos de salida, riesgo de “alucinaciones” (entidades no reales) y rendimiento cuantitativo inferior (F1-score entre 17.8 % y 27.8 %) [10].

Tomando estas limitaciones como punto de partida, Herandi et al. (2024) representaron la siguien-

te evolución: el fine-tuning específico de un LLM [9]. Ajustaron el modelo LLaMA 3 8B utilizando el dataset SkillSpan [14], diseñando un formato de salida estructurado en JSON que extraía la habilidad y su contexto textual. Este enfoque alcanzó el estado del arte (SOTA) con F1-score total de 64.8 % (skills: 54.3 %, knowledge: 74.2 %), superior a modelos supervisados previos y LLMs mediante prompting [9]. El método garantizó consistencia y auditabilidad, resolviendo problemas prácticos de los LLMs.

A pesar de su sofisticación técnica, estos estudios comparten una limitación crucial: fueron desarrollados y validados casi exclusivamente sobre datasets en idioma inglés. El trabajo de Herandi et al. (2024) se fundamentó en SkillSpan, que contiene únicamente ofertas en inglés [9]. Esta dependencia evidenció un vacío geográfico y lingüístico en la aplicación de técnicas de NLP avanzadas para el análisis del mercado laboral.

Si bien los LLMs representan la tecnología de punta, su aplicación efectiva no es trivial. El prompting simple resulta insuficiente en precisión y consistencia [10], y las metodologías de fine-tuning, aunque superiores, estaban limitadas por la barrera del idioma de los datos de entrenamiento [9].

3.3.3 Pipelines Semánticos y Descubrimiento No Supervisado

La tercera línea se centra en arquitecturas de análisis completas que van más allá de la extracción para estructurar datos y descubrir patrones latentes de manera no supervisada, respondiendo cómo se agrupan las habilidades y evolucionan los perfiles laborales.

Lukauskas et al. (2023) es el pilar fundamental de esta aproximación [8]. Su investigación en el mercado laboral de Lituania propuso y validó un pipeline de extremo a extremo que se ha convertido en referencia metodológica. El flujo comenzaba con extracción mediante Regex, seguido de vectorización con modelos basados en BERT (Sentence Transformers) para generar embeddings de 384 dimensiones. Conscientes de la "maldición de la dimensionalidad", compararon cinco métodos de reducción (PCA, t-SNE, UMAP, Trimap, Isomap), concluyendo que UMAP ofrecía los mejores resultados al preservar estructura local y global según la métrica de trustworthiness [8].

Finalmente, aplicaron y compararon algoritmos de clustering (K-means, DBSCAN, HDBSCAN, BIRCH, Affinity Propagation, Spectral), demostrando que HDBSCAN fue el más eficaz por su capacidad para identificar clústeres de formas y densidades variables y manejar ruido robustamente [8]. El gran aporte fue proporcionar validación empírica para la secuencia completa Regex $\rightarrow$ Embeddings BERT $\rightarrow$ UMAP $\rightarrow$ HDBSCAN como metodología de vanguardia para descubrimiento automático de perfiles laborales coherentes a partir de más de 500,000 ofertas.

En una línea complementaria enfocada en estandarización, se encuentra la herramienta open-source ESCOX, presentada por Kavargyris et al. (2025) [11]. ESCOX operacionaliza el mapeo semántico de texto no estructurado contra las taxonomías ESCO e ISCO-08. Su arquitectura usa un modelo Sentence Transformer pre-entrenado (all-MiniLM-L6-v2) para generar embeddings y calcula similitud del coseno contra entidades de ESCO, devolviendo aquellas que superan un umbral predefinido

(0.6 para skills, 0.55 para occupations). Ofrece backend Flask API, matching por cosine similarity, umbrales ajustables, deployment con Docker Compose y GUI no-code [11].

En un caso de estudio con 6,500 ofertas de EURES en software engineering, ESCOX extrajo aproximadamente 7,400 habilidades y 6,100 ocupaciones. Las skills más frecuentes fueron Java (27.7 %), SQL (19.2 %), DevOps (12.8 %), Work independently (10.1 %) y Python (5.9 %) [11]. El valor de ESCOX reside en su practicidad y naturaleza open-source. Sin embargo, sus autores reconocen que al ser un método basado en embeddings pre-entrenados sin fine-tuning, su precisión es inherentemente menor que modelos más especializados [11].

El trabajo de Kavas et al. (2024) abordó el desafío de clasificar ofertas laborales multilingües (en español e italiano) contra la taxonomía ESCO que está definida en inglés [12]. Los autores propusieron un modelo híbrido de tres etapas: primero, utilizan embeddings multilingües (E5-large) para recuperar las 30 ocupaciones ESCO más similares mediante similitud coseno; segundo, enriquecen el contexto del LLM mediante Retrieval-Augmented Generation (RAG) para reducir alucinaciones; y tercero, emplean LLM Llama-3 8B optimizado con Chain-of-Thought y DSPy para seleccionar el título ocupacional final.

El sistema fue evaluado sobre 200 ofertas reales de InfoJobs (100 de Italia y 100 de España). La Tabla 3.1 resume los resultados obtenidos, comparando el desempeño del modelo para ambos idiomas.

Tabla 3.1: Resultados del modelo híbrido de Kavas et al. (2024)

Componente	**Métrica**	**Italia**	**España**
LLM Llama-3 8B (CoT)	Precisión@5	0.32	0.28
	Recall@5	0.76	0.72
Embeddings E5-large	Recall@10	0.88	0.92

Los resultados superaron los baselines previos (SkillGPT, MNLI) y validaron que el enfoque híbrido de tres etapas (embeddings, RAG, LLM) es efectivo para contextos multilingües [12]. Este estudio demostró que los embeddings multilingües son fundamentales para lograr alta cobertura (recall), mientras que los LLMs permiten refinar la precisión mediante razonamiento contextual.

El estado del arte al inicio de este proyecto mostraba que ya existían pipelines robustos para análisis no supervisado y descubrimiento de perfiles [8], así como herramientas prácticas para estandarización semántica [11]. No obstante, estas capacidades no se habían integrado en una solución única que también incorporara la potencia inferencial de los LLMs de última generación [9]. Más importante aún, ninguna de estas arquitecturas avanzadas había sido desarrollada, adaptada o validada para el contexto específico del mercado laboral en América Latina y las particularidades lingüísticas del español en la región.

3.3.4 Análisis Comparativo del Estado del Arte

El análisis del contexto revela un panorama de investigación rico pero fragmentado, donde ninguna solución existente abordaba de manera integral los desafíos del mercado laboral tecnológico en América Latina. La Tabla 3.2 resume las características principales de las líneas de trabajo analizadas.

Tabla 3.2: Comparación de enfoques en el estado del arte

Enfoque	**Ventajas**	**Limitaciones**	**Refs.**
Enfoques Regionales (Léxicos)	Validación de web scraping; Datos de alta frecuencia; Contexto local	Dependencia léxica; Escalabilidad limitada; No captura semántica	[3-5]
LLMs Prompting	Flexibilidad; Sin entrenamiento; Captura contexto complejo	Inconsistencia de salida; Alucinaciones; F1 bajo (17-27 %)	[10]
LLMs Fine-tuned	SOTA en F1 (64.8 %); Salidas estructuradas; Auditabilidad	Requiere datasets anotados; Solo en inglés; Costoso computacionalmente	[9, 14]
Pipelines Semánticos	Descubrimiento no supervisado; Identificación de perfiles; Metodología validada	No incluye LLMs; Limitado a extracción regex inicial	[8]
Herramientas de Estandarización	Open-source; Integración ESCO/ISCO; Fácil de usar	Precisión limitada; No captura skills emergentes	[11]

3.3.5 Identificación de Brechas en la Literatura

El análisis comparativo evidencia cinco brechas fundamentales que ninguna solución existente resuelve de manera integral:

Brecha geográfica y lingüística: Los estudios de vanguardia con LLMs se desarrollaron exclusivamente sobre datasets en inglés [9, 14], dejando un vacío metodológico para el análisis del mercado laboral en español latinoamericano. Las particularidades lingüísticas del "Spanglish" técnico característico de la región no han sido abordadas sistemáticamente.

Brecha metodológica en la extracción: Los enfoques regionales validaron la recolección masiva de datos mediante web scraping [3-5], pero se limitaron a análisis léxico sin capacidad semántica. Inversamente, los estudios internacionales con LLMs demostraron potencia inferencial [9, 10] pero no fueron aplicados al contexto latinoamericano.

Fragmentación arquitectónica: Las tres líneas de investigación operan de forma aislada. Los pipelines semánticos no integran LLMs [8], los estudios de LLMs no implementan análisis no supervisado completo [9], y los enfoques regionales no incorporan embeddings ni clustering [3].

Ausencia de validación comparativa: Ningún estudio contrasta sistemáticamente métodos tradicionales (NER + Regex) contra LLMs en el mismo corpus, impidiendo cuantificar el valor agregado de cada enfoque. Los estudios evalúan técnicas de forma aislada sin proveer marco de validación empírica cruzada.

Barrera de actualización taxonómica: Las taxonomías estandarizadas como ESCO presentan actualización lenta frente al cambio tecnológico [3]. Las tecnologías emergentes post-2022 (frameworks, herramientas cloud, metodologías) no están representadas en vocabularios controlados, limitando la capacidad de los sistemas basados únicamente en matching taxonómico.

Estas brechas fundamentan la necesidad de una solución que sintetice las fortalezas de las distintas líneas metodológicas, adapte las técnicas de vanguardia al contexto lingüístico latinoamericano, e integre capacidades de extracción semántica con análisis no supervisado en una arquitectura de extremo a extremo validada empíricamente.

METODOLOGÍA

El proyecto adoptó un enfoque metodológico basado en la integración entre la metodología CRISP-DM y un esquema de desarrollo modular iterativo. Esta combinación permitió gestionar simultáneamente la complejidad analítica del proceso de minería de datos y las necesidades prácticas de construir un sistema compuesto por módulos independientes. CRISP-DM proporcionó la estructura conceptual que dio orden al ciclo de vida del proyecto, mientras que la modularidad garantizó que cada componente pudiera desarrollarse, validarse y refinarse de manera aislada antes de incorporarse al pipeline completo. El resultado fue un proceso sistemático capaz de equilibrar diseño metodológico y evolución técnica.

La aplicación del enfoque comenzó con una etapa de análisis en la que se clarificó el problema que motiva el observatorio: la falta de herramientas automatizadas que permitan capturar habilidades técnicas emergentes en el mercado laboral latinoamericano. A través de revisión documental, análisis de vacantes y evaluación del estado del arte, se estableció un conjunto de objetivos medibles y criterios de éxito que guiaron el desarrollo. Este entendimiento inicial fue determinante para definir el alcance, las métricas y las decisiones metodológicas posteriores.

Con los objetivos concretados, se avanzó hacia la caracterización de las fuentes de datos disponibles. Este proceso permitió identificar la estructura y variabilidad de los portales de empleo utilizados, así como la presencia de retos recurrentes relacionados con la heterogeneidad del formato, la mezcla de español e inglés técnico y la ausencia de campos estandarizados. A partir de este análisis se definió la necesidad de construir un conjunto de referencia manualmente anotado, así como la pertinencia de emplear ESCO como taxonomía base para la normalización de habilidades. Estas decisiones metodológicas sentaron las bases para garantizar evaluaciones objetivas en las etapas posteriores.

La preparación de datos se abordó mediante el desarrollo iterativo de los módulos de scraping, limpieza, normalización y deduplicación. Cada módulo atravesó ciclos de ajustes basados en inspección manual, pruebas controladas y verificación de coherencia, lo que permitió corregir errores, fortalecer la robustez del procesamiento y asegurar que los datos resultantes cumplieran criterios de calidad adecuados para el modelado. Paralelamente, se construyó el conjunto de 300 vacantes anotadas que funcionó como gold standard para la evaluación formal de los modelos.

El modelado se desarrolló siguiendo una lógica igualmente iterativa. Se diseñaron dos pipelines complementarios: uno basado en técnicas tradicionales —expresiones regulares, reglas lingüísticas y reconocimiento de entidades— y otro sustentado en modelos de lenguaje y mapeo semántico hacia ESCO. Ambos pipelines evolucionaron mediante ciclos constantes de experimentación y análisis de errores, donde cada ajuste se validó respecto al gold standard. Este proceso permitió mejorar de manera

progresiva la precisión, cobertura y coherencia en la extracción de habilidades.

La evaluación del sistema integró métricas cuantitativas, verificación cualitativa y análisis de robustez operativa. Los resultados obtenidos revelaron fortalezas y debilidades específicas, lo que habilitó nuevos ciclos de refinamiento en los modelos y los componentes del pipeline. Esta interacción continua entre modelado y evaluación consolidó un proceso de mejora incremental que fortaleció la calidad final del sistema.

Finalmente, el proyecto alcanzó su fase operativa mediante la integración de todos los módulos en una herramienta ejecutable con programación automática, interfaz de línea de comandos y documentación técnica completa. Esta etapa permitió validar el funcionamiento del observatorio en un entorno real, asegurando estabilidad, trazabilidad y capacidad de ejecución recurrente sin supervisión.

4.1 Diagrama de Flujo Metodológico

El flujo metodológico completo del proyecto integró las seis fases de CRISP-DM con las siete etapas del pipeline de software mediante ciclos iterativos de refinamiento. Cada fase de CRISP-DM se tradujo en actividades concretas de desarrollo de software, estableciendo un proceso sistemático que equilibró diseño metodológico y evolución técnica. Los resultados de la fase de Evaluación retroalimentaron continuamente la fase de Modelado, generando versiones mejoradas de los pipelines de extracción mediante ajuste de parámetros, refinamiento de reglas lingüísticas y optimización de prompts para LLMs. La representación visual completa del flujo metodológico, incluyendo las interconexiones entre fases, las iteraciones de mejora continua y los puntos de validación experimental, se presenta detalladamente en el Apéndice D (Figura 9.1).

ANÁLISIS DEL PROBLEMA

Este capítulo expone un análisis detallado del problema que el proyecto pretende solucionar, estableciendo el marco conceptual que guía el diseño de la solución técnica. Se presentan los principales requerimientos funcionales y no funcionales del sistema, las restricciones técnicas y metodológicas que delimitaron el alcance, y la especificación funcional de alto nivel de la arquitectura de pipeline. Este capítulo se enfoca en los aspectos fundamentales que determinaron las decisiones arquitectónicas posteriores.

5.1 Requerimientos del sistema

El observatorio debe satisfacer requerimientos funcionales, no funcionales y de datos que garanticen tanto las capacidades operativas como las propiedades de calidad del sistema.

5.1.1 Requerimientos funcionales

Los requerimientos funcionales definen las capacidades operativas que el observatorio debe implementar, organizadas según las siete etapas del pipeline de procesamiento. Los requerimientos clave incluyen:

- **Adquisición automatizada de datos**: Recolección de ofertas laborales desde portales de empleo en Colombia, México y Argentina mediante web scraping que soporte tanto contenido estático (HTML directo) como dinámico (JavaScript rendering).
- **Procesamiento de lenguaje natural**: Normalización y limpieza de texto adaptada al español latinoamericano técnico, incluyendo tokenización, lematización y manejo de "Spanglish".
- **Extracción dual de habilidades**: Pipeline A con NER y regex para habilidades explícitas; Pipeline B con LLMs para inferencia de habilidades implícitas.
- **Normalización taxonómica**: Mapeo de habilidades extraídas contra taxonomía ESCO mediante matching exacto y difuso (fuzzywuzzy $\geq$0.85).
- **Análisis no supervisado**: Generación de embeddings semánticos (768D), reducción dimensional (UMAP), y clustering jerárquico (HDBSCAN) para descubrir perfiles emergentes.
- **Trazabilidad completa**: Registro de metadatos de procesamiento (timestamps, versiones de modelos, parámetros) para reproducibilidad científica.

5.1.2 Requerimientos no funcionales

Los requerimientos no funcionales establecen las propiedades de calidad que garantizan la viabilidad técnica y operativa del sistema:

- **Escalabilidad**: Capacidad de procesar 600,000 ofertas en 6 meses con latencias aceptables (extracción $<$30 seg/oferta, clustering $<$4 horas).
- **Multilingüismo**: Soporte nativo para español, inglés y mezcla técnica ("Spanglish") en todos los modelos de NLP.
- **Reproducibilidad científica**: Semillas aleatorias fijas, versionado de modelos, y persistencia de datasets intermedios.
- **Eficiencia computacional**: Índices FAISS para búsqueda semántica en tiempo sub-lineal ($O(\log n)$).
- **Portabilidad y mantenibilidad**: Contenedorización Docker, adherencia a PEP 8, documentación técnica completa, y arquitectura modular extensible.

5.1.3 Requerimientos de datos

Los requerimientos de datos especifican las características del corpus de ofertas laborales:

- **Cobertura geográfica**: Ofertas de Colombia, México y Argentina con al menos 2 portales por país.
- **Volumen y calidad**: Mínimo 100,000 ofertas con descripción $>$100 caracteres y al menos 2 habilidades identificables, tras deduplicación y filtrado de calidad.
- **Metadatos temporales**: Fecha de publicación, recolección y expiración para análisis de evolución temporal.
- **Representatividad sectorial**: Aunque enfocado en tecnología, captura de múltiples sectores (TI, finanzas, manufactura, salud) para identificar demanda transversal de competencias digitales.

5.2 Restricciones

Las restricciones representan limitaciones inherentes al problema, al contexto de operación o a las decisiones de alcance del proyecto. Se organizan en tres categorías: técnicas, de datos y metodológicas.

5.2.1 Restricciones técnicas

Las principales restricciones técnicas que afectaron el diseño del sistema incluyen:

- **Medidas anti-bot de portales**: CAPTCHAs, rate limiting y bloqueos por IP que requieren delays adaptativos, rotación de user-agents y backoff exponencial.
- **Dinamismo del DOM**: Cambios frecuentes en la estructura HTML de portales que generan fragilidad en selectores CSS/XPath, requiriendo monitoreo de fallos y mecanismos de alerta.
- **Recursos computacionales limitados**: LLMs requieren GPU con 8GB+ VRAM, limitando el proyecto a modelos open-source de 3-4B parámetros ejecutables localmente.
- **Latencia de procesamiento LLM**: 40-45 segundos/oferta para procesamiento local versus métodos tradicionales, implicando trade-off entre calidad semántica y tiempos de procesamiento.

5.2.2 Restricciones de datos

Las características del dominio de ofertas laborales introducen restricciones significativas:

- **Heterogeneidad de formatos**: Ausencia de estándar para publicación de ofertas, con campos, nomenclaturas y niveles de detalle diferentes entre portales.
- **Incompletitud de información**: Muchas ofertas omiten salario, requisitos detallados o tecnologías específicas, limitando la profundidad del análisis.
- **Ruido lingüístico**: Errores ortográficos, abreviaciones no estándar, mezcla de idiomas y uso informal que reduce efectividad de técnicas NLP entrenadas en corpus formales.
- **Volatilidad temporal**: Vigencia típica de 30-60 días requiere estrategias de recolección periódica y versionado de datos.

5.2.3 Restricciones metodológicas

El contexto de español latinoamericano técnico presenta desafíos metodológicos únicos:

- **Ausencia de ground truth**: No existe dataset etiquetado de referencia para habilidades en ofertas laborales en español latinoamericano, dificultando evaluación cuantitativa rigurosa.
- **Subjetividad de "habilidad":** Definición inherentemente contextual (ej: ¿"trabajo en equipo" es técnica o blanda? ¿"Microsoft Office" debe segmentarse?).
- **Sesgo de fuente**: Portales de empleo excluyen ofertas en sitios corporativos directos, redes sociales o canales informales, introduciendo sesgo de formalidad y tamaño de empresa.

5.3 Especificación funcional

La especificación funcional describe el comportamiento de alto nivel del sistema mediante la definición de su arquitectura de pipeline de 7 etapas, las interfaces críticas entre módulos, y los casos de uso principales que el observatorio debe soportar.

5.3.1 Arquitectura de pipeline de 7 etapas

El observatorio se estructura como un pipeline secuencial de transformación de datos, donde cada etapa consume la salida de la anterior desde PostgreSQL, ejecuta su transformación especializada, y persiste resultados para la siguiente etapa:

1. **Scraping**: Recolecta ofertas laborales de portales mediante Scrapy + Selenium → `raw_jobs`
2. **Normalización**: Limpia y estandariza texto UTF-8 → `description_clean`, `combined_text`
3. **Extracción Pipeline A**: Aplica NER + Regex → `extracted_skills`
4. **Extracción Pipeline B**: Aplica LLM con prompt engineering → `enhanced_skills`
5. **Mapeo ESCO**: Normaliza contra taxonomía (exact + fuzzy) → `esco_uri`, `esco_preferred_label`
6. **Embeddings**: Genera vectores 768D con E5 Multilingual → `skill_embeddings`
7. **Clustering**: UMAP (768D→2-3D) + HDBSCAN → `analysis_results`

La orquestación se gestiona mediante CLI único (Typer) que permite ejecución manual de etapas individuales o automatización completa mediante scheduler (APScheduler).

5.3.2 Interfaces críticas

El sistema define cuatro interfaces críticas que garantizan la comunicación entre módulos:

- **Scraper-Database**: Pipeline personalizado `PostgreSQLPipeline` serializa items a JSON e inserta en `raw_jobs` con deduplicación por hash SHA-256.
- **Extractor-ESCO**: Módulo `ESCOMatcher` con métodos `find_exact_match()` y `find_fuzzy_match()`, implementando memoización que reduce procesamiento batch de 6.5h a 6.5min (60× aceleración).
- **LLM-Processor**: Módulo `LLMHandler` abstrae llamadas a modelos locales (llama.cpp) o APIs remotas (OpenAI) con interfaz unificada que retorna JSON validado (Pydantic).
- **Orquestador-Pipeline**: `MasterController` expone comandos CLI con control de estado persistente en base de datos para reinicio tras fallos.

5.3.3 Casos de uso principales

El observatorio soporta cuatro casos de uso fundamentales:

- **Recolección programada**: Scheduler (APScheduler) ejecuta spiders periódicamente para mantener corpus actualizado, con registro de métricas y alertas por caída de volumen.
- **Análisis temporal de demanda**: Generación automática de visualizaciones (heatmaps, gráficos de evolución) y reportes JSON con métricas de clustering por trimestre.
- **Validación de pipeline**: Comparación experimental de Pipeline A vs. B mediante métricas de solapamiento (Jaccard) y análisis cualitativo de habilidades únicas.
- **Extensibilidad geográfica**: Arquitectura modular permite agregar nuevos países implementando spiders heredados de `BaseSpider` sin modificaciones al pipeline de procesamiento.

Nota: La especificación técnica detallada de los requerimientos, restricciones y casos de uso presentados en este capítulo se encuentra en el Apéndice C.

DISEÑO DE LA SOLUCIÓN

6.1 Antecedentes Teóricos

La construcción de un observatorio de demanda laboral automatizado requiere la integración de múltiples técnicas del estado del arte en procesamiento de lenguaje natural, aprendizaje automático y análisis de datos. Esta sección presenta los fundamentos teóricos que sustentan las decisiones de diseño del sistema, estableciendo el marco conceptual sobre el cual se construyó la solución propuesta. El análisis comparativo de estas técnicas, junto con su evaluación empírica en el contexto del español latinoamericano, proporciona la base para las decisiones arquitectónicas presentadas en secciones posteriores.

6.1.1 Reconocimiento de Entidades Nombradas (NER)

El Reconocimiento de Entidades Nombradas es una tarea fundamental de NLP que consiste en localizar y clasificar entidades en texto dentro de categorías predefinidas [15]. Los sistemas NER modernos se basan en modelos de aprendizaje supervisado, particularmente arquitecturas basadas en transformers [16]. Para el dominio de habilidades técnicas, NER presenta ventajas teóricas significativas: alta precisión para entidades conocidas en datasets balanceados ($>$90 %), contextualización bidireccional para desambiguación, y eficiencia computacional con latencias de milisegundos por documento [17]. Sin embargo, enfrenta limitaciones críticas: dependencia del vocabulario de entrenamiento, baja cobertura en dominios especializados, y la incapacidad de inferir habilidades implícitas no mencionadas explícitamente en el texto [18].

En este proyecto se adoptó un enfoque híbrido que combina el modelo base `es_core_news_lg` de spaCy con un EntityRuler personalizado poblado con 666 patrones de habilidades técnicas de la taxonomía ESCO, permitiendo reconocimiento directo de terminología técnica no presente en el modelo pre-entrenado [19]. Los resultados experimentales revelaron que NER en este contexto específico presentó desempeño mixto: si bien aporta cobertura adicional de skills contextuales (mejora de +6.91pp F1 Pre-ESCO), introduce ruido que degrada precisión Post-ESCO (-6.64pp versus configuración Regex-Only). A pesar de este trade-off, NER se mantuvo en Pipeline A dado que incrementa recall detectando menciones contextuales que patrones deterministas omiten, como se documenta en la sección de pruebas de modelos.

6.1.2 Extracción basada en Expresiones Regulares

Las expresiones regulares (regex) son patrones de búsqueda basados en lenguajes formales que permiten identificar secuencias de caracteres con estructuras específicas [20]. En el contexto de extracción de habilidades técnicas, regex fue particularmente efectiva para tecnologías con nomenclaturas estandarizadas: versiones numeradas ("Python 3.x", "Java 8+"), frameworks con sufijos convencionales (".js", "SQL"), y acrónimos técnicos ("REST API", "CI/CD"). La implementación final consistió en 548 patrones organizados en 18 categorías técnicas (lenguajes de programación, frameworks, bases de datos, plataformas cloud, DevOps, ciencia de datos, entre otras). Sus ventajas incluyeron precisión determinística del 100 % en patrones bien definidos, transparencia y auditabilidad, velocidad extrema (microsegundos por documento), y ausencia de requisitos de entrenamiento. Las limitaciones principales fueron fragilidad ante variaciones ortográficas, mantenimiento manual intensivo de patrones, y ausencia de comprensión contextual [21].

6.1.3 Extracción basada en Modelos de Lenguaje Grandes (LLMs)

Los Large Language Models representaron un cambio de paradigma en NLP, basándose en arquitecturas transformer pre-entrenadas sobre corpus masivos mediante objetivos de modelado de lenguaje [22, 23]. A diferencia de NER y regex, los LLMs poseen capacidades de razonamiento contextual que permiten: inferencia de habilidades implícitas (deducir que un "Científico de Datos" requiere estadística y Python aunque no se mencione), normalización semántica automática (identificar que "React", "React.js" y "ReactJS" son equivalentes), desambiguación contextual, adaptación al lenguaje informal y "Spanglish", y razonamiento explicable mediante prompt engineering [24-26].

Sin embargo, presentaron limitaciones significativas: latencia 15-25x mayor que Pipeline A (tiempo típico de procesamiento de 15-25s por documento vs. 1s), no-determinismo con temperatura > 0, alucinaciones que generaron habilidades incorrectas, alto costo computacional (GPUs 4-6GB o APIs de pago), y sesgo lingüístico hacia el inglés con rendimiento degradado en español técnico [27-29].

6.1.4 Justificación del Enfoque Dual

La Tabla 6.1 presenta una comparación sistemática de las tres técnicas según criterios relevantes para el observatorio.

Tabla 6.1: Comparación de Técnicas de Extracción de Habilidades

Criterio	**Pipeline A (NER+Regex)**	**Regex Solo**	**LLMs (Gemma 3 4B)**
Precision Post-ESCO	65.50 %	86.36 %	**89.25 %**
Recall Post-ESCO	**81.25 %**	73.08 %	79.81 %
F1-Score Pre-ESCO	24.98 %	18.07 %	**46.23 %**
F1-Score Post-ESCO	72.53 %	79.17 %	**84.26 %**
Inferencia implícita	No soportado	No soportado	Sí (capacidad arquitectónica)
Latencia	0.97 s/job	∼0.32 s/job	18s (P50), 15-25s (P25-P75)
Costo computacional	Bajo (CPU)	Muy bajo (CPU)	Alto (GPU 4-6 GB)
Mantenimiento	Alto (filtros)	Alto (patrones)	Bajo (prompt eng.)
Español LATAM	Medio	Alto	Alto

Nota: Los valores presentados en la Tabla 6.1 fueron obtenidos mediante evaluación experimental sobre el conjunto de datos gold standard de 300 ofertas laborales (100 por país: CO, MX, AR). El F1-Score Post-ESCO refleja el rendimiento después de normalización mediante el mismo código de mapeo ESCO para comparación justa. Ambos pipelines operan de forma independiente sobre el mismo texto de entrada.

Dado que ninguna técnica individual satisface todos los requisitos, se implementaron dos pipelines en paralelo para comparación empírica: Pipeline A (NER + Regex) diseñado para procesamiento escalable del corpus completo mediante reglas determinísticas; y Pipeline B (LLMs) implementado experimentalmente sobre gold standard para evaluar enriquecimiento semántico y detección de habilidades implícitas. Ambos pipelines procesan las mismas ofertas de forma independiente, permitiendo comparación directa de sus capacidades. La evaluación experimental determinó que Pipeline B supera cuantitativamente a Pipeline A pero con mayor costo computacional, validando la necesidad de arquitectura dual que permita seleccionar el pipeline óptimo según el caso de uso: Pipeline A para análisis masivo y Pipeline B para validación cualitativa o detección de skills emergentes (evaluación comparativa detallada en el capítulo de Resultados) [30].

6.1.5 Modelos de Lenguaje Grandes: Selección y Evaluación

Habiendo establecido que el Pipeline B requirió capacidades de LLMs para inferencia de habilidades implícitas y normalización semántica, el siguiente desafío consistió en seleccionar un modelo específico que balanceara rendimiento lingüístico con restricciones computacionales del proyecto.

Se identificaron cuatro modelos LLM de código abierto como candidatos principales, evaluados según criterios de costo computacional, rendimiento en español latinoamericano, soporte multilingüe y capacidad de despliegue local sin dependencias de APIs comerciales.

Candidatos Evaluados:

Gemma 3 4B es un modelo ligero desarrollado por Google basado en la arquitectura Gemini. Con 4 mil millones de parámetros, está optimizado para despliegue eficiente en hardware limitado. Sus ventajas incluyen: tamaño compacto que permite ejecución en CPU o GPUs de gama media (4-6 GB VRAM con cuantización Q4), licencia permisiva Gemma para uso académico y comercial, tokenización eficiente heredada de la familia Gemini, y soporte multilingüe con cobertura de español latinoamericano. Las limitaciones principales son: menor capacidad de razonamiento complejo comparado con modelos más grandes, vocabulario técnico potencialmente limitado debido al tamaño reducido, y documentación aún en desarrollo al ser un modelo relativamente nuevo.

Llama 3 3B (Meta AI) es la versión compacta de la familia LLaMA 3, entrenado sobre un corpus masivo multilingüe con enfoque en eficiencia. Con 3 mil millones de parámetros, representa el balance extremo entre rendimiento y recursos computacionales. Ventajas: requisitos mínimos de memoria (3-4 GB VRAM con cuantización Q4), arquitectura optimizada con Grouped-Query Attention para inferencia rápida, tokenización eficiente de la familia LLaMA 3, licencia LLaMA 3 Community License permisiva para proyectos académicos, y latencia reducida ideal para procesamiento batch. Limitaciones: capacidad de razonamiento más limitada que modelos grandes, posible degradación en tareas complejas de inferencia, y menor cobertura de vocabulario técnico especializado.

Qwen 2.5 3B (Alibaba Cloud) es parte de la familia Qwen (Qianwen) optimizada para multilingüismo con énfasis en idiomas asiáticos y europeos. Con 3 mil millones de parámetros, destaca por su arquitectura de atención eficiente. Ventajas: precisión superior en tareas de extracción estructurada, tokenización eficiente multilingüe, requisitos moderados de memoria (3-4 GB VRAM con Q4), y licencia Apache 2.0 permisiva. Limitaciones: conservadurismo excesivo con recall bajo en contextos ambiguos, menor cobertura de vocabulario técnico latinoamericano, y tendencia a omitir skills implícitas.

Phi-3.5 Mini (Microsoft Research) es un modelo ultra-compacto de 3.8 mil millones de parámetros entrenado con datos sintéticos de alta calidad. Ventajas: arquitectura extremadamente eficiente con baja latencia, capacidad de razonamiento avanzado para su tamaño, soporte multilingüe, y licencia MIT. Limitaciones: inconsistencias en generación de JSON estructurado causando pérdidas durante parsing, menor robustez en instrucciones complejas comparado con modelos más grandes, y vocabulario técnico limitado en español.

La Tabla 6.2 resume la comparación cuantitativa entre los cuatro modelos evaluados.

Tabla 6.2: Comparación de Large Language Models para Extracción de Habilidades

Criterio	**Gemma 3 4B**	**Llama 3.2 3B**	**Qwen 2.5 3B**	**Phi-3.5 Mini**
Parámetros	4B	3B	3B	3.8B
F1 Pre-ESCO	**46.23 %**	39.7 %	38.9 %	35.2 %
Jobs evaluados	299	10	10	10
Skills/job prom.	27.8	24.7	11.2	15.8
Alucinaciones	**0**	7 (DS)	0	0
JSON válido	99 %	90 %	95 %	60 %
Latencia (GPU)	18s (P50)	15.24s	N/D	N/D
VRAM (Q4)	4-6 GB	3-4 GB	3-4 GB	3-4 GB
Licencia	Gemma	LLaMA 3 CL	Apache 2.0	MIT

Nota: Los cuatro modelos fueron evaluados comparativamente sobre el gold standard de 300 ofertas laborales anotadas manualmente (100 por país: CO, MX, AR). Gemma 3 4B procesó 299/300 ofertas completas (99.3 % cobertura; 2 jobs con mode collapse); Llama, Qwen y Phi procesaron 10 ofertas cada uno para comparación preliminar, siendo descartados por menor rendimiento y problemas técnicos. F1 Pre-ESCO mide rendimiento antes de normalización taxonómica. Alucinaciones (DS) indica skills de Data Science extraídas erróneamente. JSON válido mide porcentaje de respuestas con formato estructurado correcto.

Resultados de la Evaluación Comparativa:

Se evaluaron los cuatro modelos candidatos (Gemma 3 4B, Llama 3.2 3B, Qwen 2.5 3B, Phi-3.5 Mini) mediante un experimento comparativo inicial sobre 10 ofertas laborales del gold standard. La evaluación midió Precision, Recall, F1-score para habilidades explícitas e implícitas, latencia promedio, throughput y presencia de alucinaciones. Basándose en estos resultados preliminares, Gemma 3 4B fue seleccionado como ganador y posteriormente procesó el conjunto completo de 300 ofertas laborales anotadas manualmente (100 por país: CO, MX, AR) para validación exhaustiva.

El gold standard se construyó con distribución balanceada por país y representación diversa de roles técnicos: Backend (33.33 %), QA (16.33 %), Frontend (14.00 %), DevOps (11.67 %), Data Science (9.00 %), y otros roles especializados (15.67 %). El protocolo de anotación contempló la revisión de cada oferta identificando tres aspectos principales: primero, habilidades técnicas explícitas (hard skills) mencionadas directamente en el texto, totalizando 6,174 anotaciones de 1,914 skills únicas; segundo, habilidades blandas (soft skills) inferibles del contexto del cargo, totalizando 1,674 anotaciones de 306 skills únicas; y tercero, mapeo manual a conceptos ESCO cuando aplicable. El gold standard resultante contiene 7,848 anotaciones totales con promedio de 26.16 skills por oferta (20.58 hard +

5.58 soft), distribución de seniority Senior (54 %), Mid (41 %), Junior (5 %), y cobertura idiomática de Español (83.33 %) e Inglés (16.67 %).

Los modelos fueron evaluados con prompt engineering específico para español latinoamericano, incluyendo instrucciones para manejar "Spanglish", ejemplos contextuales con ofertas reales, normalización con taxonomía ESCO, y solicitud de formato JSON estructurado. Los parámetros de inferencia utilizados fueron: temperatura 0.3 (balance entre determinismo y creatividad), max_tokens 3072, y system prompt estandarizado.

Los resultados experimentales sobre las 300 ofertas determinaron la selección definitiva de Gemma 3 4B para el Pipeline B, fundamentado en cinco hallazgos principales: precisión superior, ausencia de alucinaciones sistemáticas detectadas en otros modelos, cobertura efectiva de habilidades implícitas, captura de tecnologías emergentes no presentes en ESCO, y robustez operacional con tasa de éxito superior al 99 %. La validación manual confirmó que Gemma genera outputs limpios sin fabricar skills, mientras que alternativas como Llama 3.2 presentaron alucinaciones sistemáticas en ofertas técnicas (métricas cuantitativas detalladas en el capítulo de Resultados).

El trade-off de latencia fue considerado aceptable para procesamiento batch, donde mayor tiempo de inferencia se compensa con eliminación de alucinaciones y mayor calidad semántica. La arquitectura final despliega Gemma 3 4B con cuantización Q4 (4-6 GB VRAM), temperatura 0.3, y max_tokens 3072, ejecutándose localmente sin dependencias de APIs comerciales y garantizando privacidad de datos laborales sensibles.

6.1.6 Embeddings Semánticos y Búsqueda de Similitud

Habiendo establecido las técnicas de extracción de habilidades (NER, Regex, LLMs) y los modelos para procesamiento lingüístico (Gemma/Llama), el siguiente desafío arquitectónico consistió en normalizar las habilidades extraídas contra una taxonomía de referencia y agruparlas para descubrir patrones emergentes. Esta normalización es crítica para eliminar variantes sintácticas ("React", "React.js", "ReactJS"), mapear términos coloquiales a conceptos ESCO formales, y habilitar análisis comparativo entre países y portales.

En el observatorio, los embeddings semánticos cumplieron dos roles principales: primero, la normalización de habilidades extraídas contra taxonomía ESCO mediante búsqueda de similitud coseno; y segundo, el agrupamiento de habilidades para descubrir perfiles emergentes. El modelo `intfloat/multilingual-` fue seleccionado por su soporte multilingüe nativo (768D, 100 idiomas, entrenado con contrastive learning) [31], normalización L2 integrada, tamaño compacto (278M parámetros ejecutables en CPU $<$100ms/batch), y licencia MIT.

Limitaciones Empíricas y Decisión Arquitectónica:

La evaluación experimental con 15 habilidades técnicas agregadas manualmente a ESCO reveló limitaciones críticas del modelo E5. Con threshold de similitud coseno = 0.75, se obtuvo tasa de matches correctos de 6.7 % (1/15) y falsos positivos de 93.3 % (14/15). Ejemplos: "React" $\rightarrow$ "neoplasia"

(0.828), "Docker" → "Facebook" (0.825), "Machine Learning" → "gas natural" (0.825). El análisis de causa raíz identificó tres factores: entrenamiento en lenguaje natural vs. vocabulario técnico, confusión entre brand names y palabras comunes, y contaminación del espacio vectorial por 13,939 habilidades ESCO de dominios no técnicos.

La evaluación de thresholds demostró que no existe un valor que balancee precision y recall: thresholds bajos ($<$0.80) generan falsos positivos críticos, mientras que thresholds altos ($\geq$0.85) excluyen matches exactos. Con base en esta evidencia, se tomó la decisión de deshabilitar la capa de búsqueda semántica (Layer 3), operando el sistema con estrategia de dos capas: Layer 1 (Exact Match) mediante búsqueda SQL con confidence = 1.00, y Layer 2 (Fuzzy Match) con threshold = 0.85 para capturar variantes ortográficas.

Para búsqueda de similitud en clustering, se adoptó FAISS (Facebook AI Similarity Search) con índice IndexFlatIP (exact search con inner product). FAISS demostró performance superior: 30,147 queries/segundo, latencia 0.033ms por query, speedup 25x sobre PostgreSQL pgvector, superando ampliamente el objetivo de diseño (100 q/s) por un margen de 301x [32].

6.1.7 Técnicas de Clustering No Supervisado

Una vez que las habilidades fueron extraídas, normalizadas y representadas como embeddings vectoriales de 768 dimensiones, el objetivo final del observatorio consistió en descubrir patrones y perfiles laborales emergentes sin categorías predefinidas. El sistema integró dos algoritmos complementarios para proyección dimensional y agrupamiento basado en densidad.

Configuración de UMAP:

Para el observatorio se utilizó UMAP con los siguientes parámetros: `n_neighbors=15` (balance entre estructura local y global), `min_dist=0.1` (compactación de puntos cercanos), y `metric='cosine'` (métrica apropiada para embeddings normalizados). UMAP redujo vectores de 768 dimensiones a 2-3 dimensiones visualizables manteniendo propiedades semánticas.

Configuración de HDBSCAN:

Se seleccionó HDBSCAN sobre K-Means dado que el número de perfiles emergentes era desconocido a priori. Los parámetros de producción fueron: `min_cluster_size=12` (tamaño mínimo de clúster válido para interpretabilidad, determinado tras 70+ experimentos), `min_samples=3` (puntos mínimos para núcleo de densidad), `cluster_selection_method='eom'` (Excess of Mass para clusters más estables), y `metric='euclidean'` (post-reducción dimensional con UMAP).

6.1.8 Taxonomías de Habilidades Laborales

Si bien las técnicas de extracción, embeddings y clustering constituyeron el núcleo analítico del observatorio, todas estas operaciones dependieron de un componente fundamental: una taxonomía de referencia que permitiera normalizar las habilidades extraídas del texto crudo y mapearlas a conceptos estandarizados. Sin esta normalización, habilidades equivalentes como "React", "React.js" y

"ReactJS" hubieran sido tratadas como entidades distintas, fragmentando el análisis y degradando la calidad del clustering. La selección de la taxonomía apropiada requirió balancear cobertura, actualización, soporte multilingüe y accesibilidad. Se evaluaron tres alternativas principales.

Alternativas Consideradas:

O*NET (Occupational Information Network) es la taxonomía del Departamento de Trabajo de EE.UU. con 1,000+ ocupaciones y 20,000+ habilidades. Ventajas: altamente estructurada con relaciones jerárquicas, actualización periódica, y datos salariales asociados. Limitaciones: enfoque en mercado estadounidense, escasa cobertura de tecnologías emergentes, y ausencia de soporte multilingüe nativo. A pesar de sus limitaciones, se extrajeron 152 habilidades técnicas modernas (Hot Technologies) de O*NET para complementar ESCO, especialmente en áreas de tecnologías emergentes donde ESCO v1.1.0 presenta gaps de cobertura.

ESCO (European Skills, Competences, Qualifications and Occupations) es la taxonomía oficial de la Unión Europea con 13,000+ habilidades, 3,000+ ocupaciones, y soporte para 27 idiomas [19]. Ventajas: etiquetas nativas en español e inglés, cobertura amplia de habilidades tecnológicas, estructura ontológica con URIs únicos, y respaldo institucional de la Comisión Europea garantizando mantenimiento. Limitaciones: actualización menos frecuente que el mercado tecnológico (v1.1.0 data de 2016-2017), y menor cobertura de frameworks JavaScript modernos.

Taxonomías propietarias (LinkedIn Skills, Burning Glass Technologies): Ventajas: actualizadas con mayor frecuencia, cobertura de tecnologías emergentes. Limitaciones críticas: acceso mediante API de pago, licencias restrictivas, y falta de transparencia metodológica.

Decisión Final:

Se seleccionó ESCO v1.1.0 como base taxonómica fundamentado en cuatro razones principales. En primer lugar, el soporte multilingüe nativo (español/inglés) eliminó la necesidad de traducción automática. En segundo lugar, la licencia Creative Commons BY 4.0 permitió uso académico y comercial sin restricciones. En tercer lugar, la estructura ontológica con URIs persistentes facilitó la integración con LLMs y sistemas de recomendación. En cuarto lugar, la cobertura de 13,939 habilidades resultó suficiente para establecer baseline. La taxonomía ESCO fue extendida con 152 habilidades técnicas de O*NET (Hot Technologies) y 124 habilidades agregadas manualmente identificadas en el análisis exploratorio, totalizando 14,215 skills. La taxonomía extendida se almacenó en PostgreSQL (tabla `esco_skills`) con índices en `preferred_label_es`, `preferred_label_en` y `alt_labels`, permitiendo búsquedas eficientes durante el matching.

Habiendo establecido las bases teóricas de las técnicas de extracción (NER, Regex, LLMs), modelos de lenguaje (Gemma, Llama), embeddings semánticos (E5 Multilingual), algoritmos de clustering (UMAP, HDBSCAN) y taxonomías de referencia (ESCO), la siguiente sección presentó la validación empírica de estas tecnologías sobre datos reales del mercado laboral latinoamericano. Las pruebas ejecutadas determinaron qué combinación de técnicas optimizó el balance entre precisión y cobertura para el contexto específico del español técnico, proporcionando evidencia cuantitativa que fundamentó las decisiones arquitectónicas del sistema.

6.2 Pruebas de Modelos

Los fundamentos teóricos presentados en la sección anterior establecieron las bases conceptuales para la selección de técnicas de extracción (NER, Regex, LLMs), tecnologías de embeddings (E5 Multilingual), y algoritmos de clustering (UMAP, HDBSCAN). Sin embargo, la viabilidad de estas técnicas en el contexto específico del español latinoamericano técnico requiere validación empírica con datos reales del dominio. Esta sección presenta los resultados de las validaciones experimentales ejecutadas sobre el sistema de extracción y matching de habilidades. A diferencia de benchmarks teóricos sobre datasets en inglés, estas pruebas se realizaron con ofertas laborales reales de portales latinoamericanos, proporcionando evidencia empírica específica del dominio que fundamentó decisiones arquitectónicas clave presentadas en la sección posterior.

6.2.1 Conjunto de Datos

El dataset del observatorio se compone de ofertas laborales tecnológicas recolectadas mediante web scraping de seis portales principales en tres países: Computrabajo, Bumeran y ElEmpleo (Colombia); Computrabajo, Bumeran, InfoJobs y OCC Mundial (México); y Computrabajo, Bumeran y ZonaJobs (Argentina). El corpus final contiene 30,660 ofertas laborales únicas distribuidas como: México 17,835 (58.16 %), Colombia 9,479 (30.91 %), y Argentina 3,346 (10.93 %). Las ofertas abarcan el período de octubre 2018 a octubre 2025 (7 años de datos históricos), con 71.23 % de cobertura temporal (21,839 jobs con posted_date válido). El trimestre más reciente (Q4 2025) representa 69 % del dataset total, reflejando la naturaleza dinámica del mercado laboral tecnológico.

Cada oferta contiene los siguientes campos estructurados: `job_id` (UUID único), `portal` (origen de la oferta), `country` (CO/MX/AR), `title` (título del cargo), `company` (empresa), `description` (descripción detallada del cargo), `requirements` (requisitos y habilidades), `salary_raw` (rango salarial cuando disponible), `contract_type` (tipo de contrato), `remote_type` (modalidad presencial/remota/híbrida), `posted_date` (fecha de publicación), y `content_hash` (hash SHA-256 para deduplicación).

6.2.2 Construcción del Conjunto de Datos

Proceso de Scraping:

La recolección de datos se ejecutó mediante Scrapy 2.11, un framework asíncrono de scraping en Python que permitió procesamiento concurrente de múltiples requests. La estrategia de extracción se adaptó al tipo de portal: para sitios con HTML estático o APIs accesibles, se utilizó `requests` directo obteniendo datos en formato JSON desde endpoints que alimentan las vistas frontend (más eficiente); para portales con contenido dinámico cargado mediante JavaScript (Bumeran, ZonaJobs), se integraron Selenium 4.15 y ChromeDriver para renderizado completo de páginas antes de la extracción. El proceso implementó técnicas de "polite crawling": delays adaptativos entre requests (2-5

segundos), rotación de user-agents, límites de concurrencia por dominio, y reintentos con backoff exponencial ante errores HTTP 429/503.

La deduplicación de ofertas se realizó mediante hashing SHA-256 del contenido normalizado (title + company + description limpiados), almacenando el hash en campo `content_hash` con restricción UNIQUE en PostgreSQL. Esta estrategia evitó re-procesar ofertas republicadas por portales múltiples o reposteadas por el mismo portal.

Limpieza y Normalización:

Las ofertas extraídas presentaron variabilidad significativa en formato y calidad. Se aplicó un pipeline de limpieza: eliminación de caracteres HTML residuales (`<br>`, ` `), normalización de espacios múltiples y saltos de línea, conversión de encoding a UTF-8, y extracción de texto plano de campos con formato rich text. Los disclaimers legales (equal opportunity statements, privacy policies) se identificaron mediante patrones regex y se separaron en metadatos sin afectar el análisis de habilidades.

6.2.3 Validación de Técnicas de Extracción (Pipeline A)

Se evaluó el rendimiento de los dos métodos del Pipeline A (Regex y NER) sobre el gold standard completo de 300 ofertas laborales manualmente anotadas (100 por país: Colombia, México, Argentina).

Resultados de Extracción con Expresiones Regulares:

El módulo regex implementó 548 patrones organizados en 18 categorías técnicas para tecnologías con nomenclatura estructurada. Resultados sobre el gold standard de 300 ofertas: F1-Score solo regex Pre-ESCO 18.07 %, F1-Score Post-ESCO 79.17 %, precision 86.36 % (post-ESCO), recall 73.08 %, latencia $\sim$0.32 s/job (320 ms). Las skills detectadas correctamente incluyeron Python, React, AWS, Docker, PostgreSQL, Kubernetes, Git, JavaScript, SQL, FastAPI. Los falsos positivos pre-ESCO correspondieron a acrónimos ambiguos (“ML” en contextos no técnicos) y nombres de empresas similares a tecnologías (“Oracle” empresa vs. “Oracle Database”), pero fueron efectivamente filtrados mediante matching con ESCO. Conclusión: regex demostró alta precision post-normalización (86.36 %) validando su uso como método base, con velocidad submilisegundos permitiendo procesamiento masivo.

Resultados de Extracción con NER y spaCy:

El módulo NER utilizó `es_core_news_lg` con EntityRuler poblado con 666 patrones ESCO y múltiples filtros post-extracción (200+ stopwords NER, 60+ technical generic stopwords). Resultados sobre gold standard: Pipeline A (NER+Regex) alcanzó F1-Score Pre-ESCO 24.98 % y F1-Score Post-ESCO 72.53 %, con recall de 64.43 % y precision post-filtrado de 9.3 % (pre-ESCO), latencia 50-80ms. El análisis reveló que NER aportó +6.91pp de mejora Pre-ESCO respecto a regex solo, pero generó -6.64pp de degradación Post-ESCO debido a extracción de variantes textuales que no mapearon a ESCO. Dado que ESCO no cubre exhaustivamente el vocabulario técnico latinoamericano emergen-

te, se priorizó optimizar el F1-Score Pre-ESCO (captura de habilidades reales mencionadas) sobre el Post-ESCO (normalización a taxonomía), por lo que NER se mantuvo activo en la arquitectura final para maximizar cobertura de detección.

Estrategia Combinada y Matching con ESCO:

La arquitectura final combinó ambos métodos con deduplicación posterior, logrando signal-to-noise ratio de 0.98 después de matching con ESCO. El matching contra taxonomía ESCO extendida (14,215 skills totales: 13,939 ESCO v1.1.0 + 152 O*NET + 124 agregadas manualmente) se ejecuta en dos capas: Layer 1 (exact match) mediante SQL `ILIKE` en `preferred_label_es/en` con confidence 1.00, y Layer 2 (fuzzy match) con `fuzzywuzzy` ratio $\geq$ 0.85 para variantes ortográficas con confidence = ratio/100.

6.2.4 Evaluación del Sistema Completo con Gold Standard

Se ejecutó un test end-to-end procesando 300 ofertas laborales del gold standard (100 por país: Colombia, México, Argentina) con el pipeline completo utilizando el matcher ESCO implementado de dos capas (exact + fuzzy). Resultados globales: jobs procesados exitosamente 300/300 (100 %), total skills extraídas 8,268 (27.6 skills/job promedio), skills matched con ESCO 1,038 (12.6 % match rate), emergent skills sin match 7,230 (87.4 %), latencia promedio 1.82 segundos/job. Posteriormente se desarrolló un matcher experimental mejorado con mapeos manuales curados que incrementó el match rate a aproximadamente 25 %, permitiendo identificar y cuantificar con mayor precisión las habilidades emergentes ausentes en ESCO v1.1.0. Sin embargo, esta versión experimental no fue integrada al pipeline productivo para evitar introducir sesgos en la comparación entre Pipeline A y Pipeline B, manteniendo condiciones de evaluación equitativas donde ambos pipelines utilizan el mismo matcher sin bias.

La distribución de matching por capa fue: Layer 1 (exact match) 149 skills (43.1 % del matched, confidence 1.00), Layer 2 (fuzzy match) 197 skills (56.9 % del matched, confidence 0.85-0.99). El análisis por país reveló variación: Colombia 15.3 % match rate (mayor proporción de stacks enterprise tradicionales: Java, .NET, Oracle, SAP), México 11.3 % (mayor adopción de frameworks modernos), Argentina 12.5 % (balance intermedio).

Las top 10 skills matched con ESCO fueron: Python (14 menciones), Agile (13), SQL (10), JavaScript (10), Git (8), FastAPI (8), AWS Lambda (8), Kubernetes (6), Go (6), GitLab CI/CD (6). Las top 10 emergent skills (sin match ESCO, verificadas manualmente) fueron: Data Science (67), HTML5 (35), CSS3 (21), Matplotlib (15), Dashboards (13), Data Modeling (9), Data Analysis (9), Flux (8), Relay (8), SOAP (8). Interpretación: las emergent skills confirman predominancia de conceptos amplios de análisis de datos (Data Science, Dashboards, Data Analysis) y versiones específicas de tecnologías (HTML5, CSS3) que ESCO v1.1.0 no distingue granularmente. Adicionalmente, se identificaron skills post-2022 de IA generativa (ChatGPT, LLM, Generative AI, Fine-tuning de LLMs) con frecuencias bajas pero alta relevancia estratégica.

El match rate de 12.6 % fue obtenido con el matcher baseline de dos capas (exact + fuzzy 0.92) operando sobre la taxonomía ESCO extendida de 14,215 skills (13,939 ESCO v1.1.0 + 152 O*NET + 124 agregadas manualmente). Este porcentaje es bajo pero esperado considerando dos factores principales. Primero, el corpus contiene alta frecuencia de conceptos amplios (Data Science, Data Analysis) y versiones específicas (HTML5, CSS3) que ESCO trata genéricamente sin distinguir granularidad. Segundo, el mercado laboral latinoamericano presenta alta demanda de tecnologías post-2022 (ChatGPT, Generative AI, LLM) ausentes en taxonomías europeas. Iteraciones posteriores con matcher enhanced de 4 capas incrementaron el coverage a $\sim$25 %, validando que las habilidades no matched (87.4 % baseline, 74.7 % enhanced) corresponden genuinamente a emergent skills y representan señal valiosa de innovación del mercado para análisis exploratorio posterior.

Estos resultados tienen implicaciones arquitectónicas importantes para el sistema. El alto porcentaje de emergent skills valida la decisión de implementar el Pipeline B con LLMs, ya que estas habilidades modernas probablemente corresponden a términos técnicos actuales que ESCO v1.1.0 no cubre pero que un LLM pre-entrenado puede reconocer contextualmente. Asimismo, la variación del match rate por país (CO 15.3 %, MX 11.3 %, AR 12.5 %) sugiere diferencias regionales en adopción tecnológica que deben considerarse en el análisis de clustering, potencialmente requiriendo ajuste de parámetros HDBSCAN por región. Finalmente, la alta frecuencia de conceptos amplios de datos (Data Science con 67 menciones) y skills de IA generativa post-2022 confirman la relevancia temporal del corpus y su alineación con tendencias del mercado tecnológico latinoamericano.

Para maximizar el valor analítico de las emergent skills, se propone un proceso de curación semiautomática que combinaría clustering semántico de las habilidades no matched mediante embeddings E5 y HDBSCAN, seguido de revisión manual de los clústeres más frecuentes. Los términos válidos y recurrentes (threshold de cinco o más menciones) se agregarían manualmente a la taxonomía ESCO local, mientras que los términos rechazados se documentarían con su justificación correspondiente. Este proceso iterativo permitiría que el match rate evolucione orgánicamente con el corpus, balanceando cobertura y control de calidad. En la implementación actual, las 124 habilidades agregadas manualmente fueron identificadas mediante análisis exploratorio inicial del corpus sin automatización del proceso de clustering.

6.3 Arquitectura

Las pruebas experimentales presentadas en la sección anterior proporcionaron evidencia empírica crítica que determinó las decisiones arquitectónicas del sistema. Los resultados demostraron que el enfoque dual NER+Regex alcanza precision de 78-95 % con latencias submilisegundos, que el matching con ESCO requiere expansión manual debido al bajo match rate inicial (12.6 %), y que los embeddings E5 presentan limitaciones para búsqueda semántica en vocabulario técnico especializado. Adicionalmente, las características del dominio – procesamiento batch de ofertas laborales sin requisitos de tiempo real, corpus objetivo de 600,000 ofertas, y recursos limitados propios de un proyecto

académico – restringieron las opciones arquitectónicas viables.

El sistema se diseñó como un observatorio automatizado end-to-end que integra siete etapas especializadas de procesamiento, desde la adquisición de ofertas laborales hasta la generación de visualizaciones analíticas. La arquitectura fue fundamentada en los principios de modularidad, escalabilidad y separación de responsabilidades, permitiendo desarrollo incremental, pruebas unitarias por componente, y evolución independiente de cada módulo. Esta sección presenta la selección del estilo arquitectónico, la especificación de los componentes principales del sistema, y el diseño de la base de datos como mecanismo de persistencia entre etapas.

6.3.1 Selección del Estilo Arquitectónico

Se evaluaron tres estilos arquitectónicos para el observatorio: arquitectura de microservicios, arquitectura orientada a eventos, y arquitectura de pipeline lineal. La Tabla 6.3 presenta la comparación según criterios relevantes para el contexto académico y operativo del proyecto.

Tabla 6.3: Comparación de Estilos Arquitectónicos

Criterio	**Microservicios**	**Event-Driven**	**Pipeline Lineal**
Complejidad	Alta	Media-alta	Baja
Escalabilidad horizontal	Excelente	Excelente	Limitada
Trazabilidad	Media	Media	Excelente
Debugging	Difícil	Medio	Fácil
Overhead operativo	Alto	Medio	Bajo
Time to market	Lento	Medio	Rápido

Se seleccionó arquitectura de pipeline lineal fundamentado en cuatro razones principales. En primer lugar, la simplicidad operativa dado que es un proyecto académico con equipo de 2 desarrolladores y recursos computacionales limitados (1 servidor, sin infraestructura Kubernetes/Docker Swarm). En segundo lugar, la trazabilidad completa ya que el flujo unidireccional de datos permite debugging determinístico y auditoría de transformaciones etapa por etapa. En tercer lugar, la velocidad de desarrollo considerando que la implementación de microservicios requiere 3-4x más tiempo en configuración de comunicación inter-servicios, service discovery, y manejo de fallos distribuidos. En cuarto lugar, la naturaleza batch del dominio debido a que el análisis de demanda laboral no requiere procesamiento en tiempo real (latencias de horas/días son aceptables), eliminando ventajas principales de arquitecturas asíncronas.

Si bien la arquitectura de pipeline lineal satisface los requisitos del proyecto, es importante reconocer sus limitaciones inherentes. El procesamiento secuencial sincrónico impide el aprovechamiento de paralelismo entre etapas, resultando en latencias acumulativas estimadas de 30 a 60 segundos por ofer-

ta para el pipeline completo cuando se incluye el procesamiento con LLM. Asimismo, la escalabilidad horizontal está limitada por la naturaleza monolítica del orquestador, lo cual requeriría migración a arquitectura de microservicios si el volumen superara las 100,000 ofertas mensuales. Adicionalmente, la ausencia de mecanismos de tolerancia a fallos distribuidos implica que el fallo de una etapa detiene el pipeline completo, aunque esta limitación se mitiga mediante persistencia intermedia en PostgreSQL y capacidad de reinicio desde checkpoints.

Estas limitaciones fueron consideradas aceptables dado que el análisis de demanda laboral no requiere procesamiento en tiempo real, mientras que el volumen objetivo de 600,000 ofertas es procesable en 5 a 10 horas con el hardware disponible. Además, la simplicidad operativa reduce significativamente el tiempo de desarrollo, estimado en 3 a 4 meses comparado con 9 a 12 meses que requeriría una arquitectura de microservicios. La arquitectura permite evolución futura mediante refactorización incremental de módulos críticos a servicios independientes si los requisitos de volumen o latencia lo justifican posteriormente.

El sistema implementa un pipeline secuencial de 7 etapas:

1. **Scraping (Scrapy + Selenium)**: Recolección automatizada de ofertas desde portales web
2. **Extraction (NER + Regex)**: Identificación de habilidades explícitas
3. **LLM Processing (Gemma/Llama)**: Enriquecimiento semántico e inferencia de habilidades implícitas
4. **Embedding (E5 Multilingual)**: Generación de representaciones vectoriales 768D
5. **Dimension Reduction (UMAP)**: Proyección a 2-3 dimensiones visualizables
6. **Clustering (HDBSCAN)**: Agrupamiento no supervisado de habilidades
7. **Visualization y Exportación**: Generación de gráficos estáticos (matplotlib/seaborn) y exportación de resultados (PNG/CSV/JSON)

Cada etapa opera de forma autónoma, lee datos de la etapa anterior desde PostgreSQL, ejecuta su transformación especializada, y persiste resultados para la siguiente etapa. La orquestación se gestiona mediante un CLI único (Typer) que permite ejecución manual de etapas individuales o automatización completa mediante scheduler (APScheduler). El diagrama de arquitectura modular completo, detallando tecnologías específicas, funciones, entradas/salidas y mecanismos de almacenamiento por módulo, se presenta en el Apéndice D.

6.3.2 Componentes del Sistema

La arquitectura ha sido diseñada considerando principios de modularidad y separación de responsabilidades, permitiendo la trazabilidad completa de las transformaciones de datos. Las siete etapas del pipeline se agrupan en cinco componentes funcionales que se describen a continuación.

El primer componente corresponde al servicio de web scraping, el cual administra la recolección automatizada de ofertas laborales desde seis portales de empleo en Colombia, México y Argentina (Computrabajo, Bumeran, ElEmpleo, InfoJobs, OCC Mundial, ZonaJobs). Este servicio fue implementado utilizando Scrapy 2.11 como framework asíncrono base, complementado con Selenium 4.15 para el manejo de contenido dinámico JavaScript. La deduplicación de ofertas se realiza mediante hashing SHA-256, almacenando las ofertas únicas en la tabla raw_jobs con metadatos de origen, país y timestamps.

El segundo componente es el servicio de extracción de habilidades, responsable de identificar las competencias técnicas mencionadas explícitamente en las ofertas laborales. Integra tres técnicas complementarias: Reconocimiento de Entidades Nombradas (NER) con spaCy y EntityRuler poblado con taxonomía ESCO, expresiones regulares con 47 patrones para tecnologías estructuradas, y normalización mediante matching de dos capas (exacto y difuso con threshold 0.85). Los resultados se persisten en la tabla extracted_skills con metadatos de método, confianza y enlace ESCO.

El tercer componente es el servicio de procesamiento con LLM, diseñado para enriquecimiento semántico e inferencia de habilidades implícitas. Utiliza un modelo LLM ligero de código abierto (Gemma 3 4B o Llama 3 3B, sujeto a evaluación comparativa) con prompt engineering específico para español latinoamericano, manejando el fenómeno de "Spanglish" técnico. Las tareas incluyen deduplicación inteligente de variantes sintácticas, inferencia contextual de competencias requeridas, normalización con ESCO, y generación de justificaciones explicables, persistiendo en la tabla enhanced_skills.

El cuarto componente es el servicio de generación de embeddings, el cual transforma las habilidades en representaciones vectoriales densas de 768 dimensiones mediante el modelo intfloat/multilingual-e5-base. Genera embeddings por lotes (batch_size=32, latencia $<$100ms/batch en GPU) con normalización L2, almacenando en la tabla skill_embeddings con soporte pgvector. La construcción de índice FAISS permite 30,147 queries/segundo, superando 25x la velocidad de pgvector nativo.

El quinto componente es el servicio de análisis y visualización, responsable del descubrimiento de patrones emergentes mediante técnicas no supervisadas. Integra reducción dimensional con UMAP (768D $\rightarrow$ 2-3D), clustering jerárquico con HDBSCAN (identificación automática de clústeres y detección de ruido), generación de visualizaciones estáticas con matplotlib/seaborn (scatter plots UMAP, heatmaps temporales, gráficos de evolución de clusters), y exportación de datos en formatos estructurados (PNG, CSV, JSON). Los resultados se persisten en analysis_results con parámetros y resultados en formato JSONB.

6.3.3 Diseño de la Base de Datos

La base de datos actuó como columna vertebral del sistema, implementando el patrón de persistencia de pipeline donde cada etapa escribió sus resultados en tablas especializadas. Se seleccionó PostgreSQL 15+ por su soporte JSON nativo (JSONB) para almacenar metadatos flexibles, extensión

pgvector para vectores de alta dimensionalidad, robustez transaccional (ACID), capacidad de particionamiento para escalabilidad, y licencia libre (PostgreSQL License).

Esquema de Tablas Principales:

El esquema constó de seis tablas principales correspondientes a las etapas del pipeline descritas anteriormente:

raw_jobs: Almacena ofertas tal como fueron scrapeadas. Campos clave: identificador UUID, portal de origen, código de país (CO/MX/AR), título, descripción, requisitos, hash SHA-256 para deduplicación, y bandera de procesamiento.

extracted_skills: Contiene habilidades identificadas por NER y expresiones regulares. Incluye identificador UUID, referencia a la oferta laboral, texto de la habilidad extraída, método de extracción (NER, regex o ESCO match), score de confianza (0-1), y enlace a la taxonomía ESCO cuando aplica.

enhanced_skills: Almacena el enriquecimiento semántico realizado por el modelo LLM. Campos principales: identificador, referencia a la oferta, habilidad normalizada, tipo de habilidad (explícita, implícita o normalizada), URI del concepto ESCO, nivel de confianza del LLM, justificación del razonamiento, y modelo utilizado.

skill_embeddings: Contiene las representaciones vectoriales de 768 dimensiones. Almacena identificador, texto de la habilidad, vector embedding con soporte pgvector, nombre del modelo, y versión. Incluye índice IVFFlat optimizado para búsquedas de similitud coseno con particionamiento en 100 listas.

analysis_results: Almacena resultados de clustering y análisis de tendencias. Campos: identificador, tipo de análisis (clustering, trends, profile), país, rango de fechas analizado, parámetros de configuración en formato JSONB, y resultados estructurados con clústeres, etiquetas y métricas.

esco_skills: Tabla de referencia con la taxonomía ESCO completa. Contiene URI del concepto, etiquetas preferidas en español e inglés, etiquetas alternativas, tipo de habilidad, descripción, y nivel de reutilización. Almacena 14,215 habilidades (13,939 de ESCO v1.1.0, 152 de O*NET, y 124 agregadas manualmente).

Todas las tablas derivadas mantienen referencia mediante llave foránea hacia la tabla de ofertas laborales (raw_jobs), garantizando trazabilidad completa desde cualquier resultado de análisis hasta la oferta original. Las tablas de habilidades extraídas y enriquecidas mantienen referencia opcional hacia la taxonomía ESCO para efectos de normalización. El diagrama entidad-relación completo del esquema de base de datos se presenta en el Apéndice E.

Habiendo especificado el estilo arquitectónico (pipeline lineal), los componentes del sistema (8 etapas especializadas), y el diseño de la base de datos (6 tablas principales con relaciones trazables), la siguiente sección presenta las decisiones tecnológicas concretas que materializaron esta arquitectura. La selección de herramientas y tecnologías se fundamentó en criterios de madurez, licencias permisivas, escalabilidad comprobada y reproducibilidad científica, asegurando que cada componente arquitectónico contara con una implementación robusta y bien documentada.

6.4 Herramientas y Tecnologías

Las Tablas 6.4 y 6.5 resumen las decisiones tecnológicas fundamentales y su justificación académica y técnica, organizadas por capa funcional del sistema.

Tabla 6.4: Stack Tecnológico: Infraestructura y Adquisición de Datos

Componente	**Tecnología**	**Justificación**
Base de datos	PostgreSQL 15+ con pgvector	Soporte JSONB para metadatos flexibles, extensión pgvector para vectores 768D, robustez ACID, particionamiento para escalabilidad.
Taxonomía	ESCO v1.1.0 (+ 276 ext.)	Cobertura 13,000+ skills con etiquetas español/inglés, extendida con 152 O*NET + 124 manual (total 14,215), estructura ontológica con URIs, respaldo institucional CE, licencia CC BY 4.0.
Framework scraping	Scrapy 2.11 + Selenium 4.15	Arquitectura asíncrona (100+ req/min), manejo robusto de reintentos, middlewares extensibles, Selenium para JavaScript dinámico.
Modelo NLP español	spaCy 3.7 + es_core_news_lg	Mejor modelo español disponible (97M parámetros), soporte EntityRuler para ESCO, optimizado CPU ($<$100ms/doc).
Lenguaje	Python 3.11+	Ecosistema científico maduro (NumPy, pandas, scikit-learn), bibliotecas NLP referencia (spaCy, transformers), integración PostgreSQL.
Control versiones	Git + GitHub	Estándar industria, integración CI/CD (GitHub Actions), control issues/milestones, documentación Markdown, respaldo cloud.

Tabla 6.5: Stack Tecnológico: Procesamiento y Análisis

Componente	**Tecnología**	**Justificación**
LLM enriquecimiento	Gemma 3 4B / Llama 3 3B (Q4)	Modelos ligeros de 3-4B parámetros, despliegue local sin APIs (privacidad), cuantización Q4 (3-6 GB VRAM), selección por evaluación comparativa.
Embeddings	intfloat/multilingual-e5-base	Estado del arte multilingüe (768D), contrastive learning en 100 idiomas, normalización L2 integrada, 278M parámetros ejecutables CPU.
Índice similitud	FAISS IndexFlatIP	Velocidad 30,147 q/s (25x superior pgvector), búsqueda exacta (100 % recall), latencia 0.033ms, Facebook AI Research.
Reducción dimensional	UMAP [33]	Preserva estructura local y global (superior t-SNE), escalabilidad millones puntos, reproducibilidad con semilla fija, parámetros interpretables.
Clustering	HDBSCAN [34]	No requiere especificar k, identifica ruido automático, maneja densidades variables (nicho vs. mainstream), jerarquía multinivel.
Orquestación	Typer CLI + APScheduler	Interface tipo Git con validación automática, scheduler 24/7, logging estructurado, integración cron/systemd.

Todas las tecnologías seleccionadas cumplen con cinco criterios principales. Primero, licencias permisivas (MIT, Apache 2.0, PostgreSQL, CC BY) permitiendo uso académico y potencial comercial futuro. Segundo, madurez y estabilidad con versiones ≥ 2.0 y comunidades activas. Tercero, documentación académica completa con publicaciones científicas revisadas por pares para componentes críticos. Cuarto, reproducibilidad mediante control de versiones de dependencias y semillas fijas para componentes estocásticos. Quinto, escalabilidad demostrada para el objetivo de 600,000 ofertas laborales mediante arquitectura de pipeline por lotes y particionamiento de base de datos.

DESARROLLO DE LA SOLUCIÓN

7.1 Implementación de la Infraestructura

El sistema se implementó sobre PostgreSQL 15.3, seleccionado por su robustez en manejo de datos estructurados, soporte JSON nativo, y capacidades de indexación GIN. El esquema normalizado (3FN) utiliza seis tablas principales: `raw_jobs` (ofertas crudas del scraping), `cleaned_jobs` (ofertas normalizadas), `extracted_skills` (Pipeline A), `enhanced_skills` (Pipeline B), `gold_standard_annotations` (300 ofertas anotadas manualmente), y `esco_skills` (14,215 skills de taxonomía ESCO extendida). Cada tabla de skills incluye metadatos de trazabilidad (`extraction_method`, `llm_model`, `esco_uri`) permitiendo comparaciones sistemáticas entre pipelines.

Para optimizar consultas sobre 30,660 ofertas, se implementaron índices compuestos: B-tree en `(job_id, extraction_method)`, GIN en `skill_text`, y B-tree en `posted_date`. Estas optimizaciones redujeron tiempos de consultas agregadas de 45s a 2.3s.

La configuración de PostgreSQL se optimizó específicamente para procesamiento batch de grandes volúmenes de datos, priorizando throughput sobre latencia de consultas individuales. Se configuró memoria compartida de 4GB, memoria de trabajo de 256MB, y cache efectivo de 12GB según mejores prácticas para servidores orientados a procesamiento analítico. Se implementaron índices especializados tipo B-tree para comparaciones entre pipelines, GIN para búsquedas de texto completo, e IVFFlat para búsquedas de similitud vectorial. Las optimizaciones lograron reducir consultas agregadas de 45 segundos a 2.3 segundos e incrementar inserciones batch a 5,000 registros por segundo. Esta configuración permitió que el procesamiento del corpus completo de 30,660 ofertas completara en aproximadamente 6.2 horas, incluyendo todas las etapas del pipeline. La especificación completa de parámetros, índices y micro-benchmarks se documenta en el Apéndice E.

7.1.1 Orquestación del Pipeline

El orquestador implementó ocho etapas modulares ejecutadas secuencialmente: scraping de ocho portales en tres países, limpieza y normalización, extracción mediante Pipeline A, extracción mediante Pipeline B, mapeo a taxonomía ESCO, generación de embeddings, clustering con UMAP y HDBSCAN, y análisis temporal. Esta arquitectura lineal se seleccionó sobre frameworks más complejos por su simplicidad de depuración y capacidad de reejecutar etapas individuales. Cada script registra progreso en logs estructurados con timestamps y métricas de performance.

El corpus se recolectó mediante scraping especializado por portal, adaptado a tres arquitecturas

según las características de cada sitio: acceso directo a endpoints JSON/API para portales que exponen datos estructurados, combinación de `requests` con `BeautifulSoup4` para contenido HTML estático, y `selenium` para contenido JavaScript dinámico renderizado client-side. Se implementaron estrategias anti-bloqueo mediante delays aleatorios, rotación de User-Agent, y respeto de archivos `robots.txt`. El scraping completo recolectó 56,555 ofertas brutas durante aproximadamente 72 horas distribuidas en 15 días.

El pipeline de limpieza aplicó normalización de texto, eliminación de HTML residual, detección de idioma mediante `langdetect`, deduplicación mediante fuzzy matching y hash SHA-256, y validación de calidad descartando ofertas con contenido insuficiente. Del total scrapeado, se descartaron 25,895 ofertas por duplicación o calidad insuficiente, resultando en un dataset de 30,660 ofertas usables con texto normalizado, idioma identificado y metadata completa, cubriendo siete años de publicaciones laborales.

7.2 Implementación de Sistemas de Extracción de Habilidades

Se implementaron cuatro aproximaciones metodológicas para extracción automática de habilidades técnicas: Pipeline A (NER + Regex), Pipeline B (LLM), Pipeline A.1 (TF-IDF, descartado), y configuración Regex-Only para baseline.

7.2.1 Pipeline A: NER y Expresiones Regulares

Pipeline A constituye el método base de extracción de habilidades del observatorio, diseñado para identificar menciones explícitas de tecnologías mediante la combinación de Reconocimiento de Entidades Nombradas (NER) para detectar menciones contextuales y expresiones regulares (Regex) para capturar nomenclaturas estandarizadas. Esta arquitectura dual resuelve limitaciones complementarias: mientras NER con spaCy 3.5 y EntityRuler de 666 patrones ESCO captura tecnologías modernas en contexto pero falla con acrónimos técnicos, Regex con 548 patrones compilados en 18 categorías garantiza detección de nomenclaturas estructuradas pero omite menciones contextuales. La integración alcanza recall Post-ESCO de 81.25 % versus 73.08 % de Regex solo, procesando ofertas con latencia de 0.97 segundos, aproximadamente 18 veces más rápida que Pipeline B.

El flujo de procesamiento ejecuta ambas técnicas en paralelo sobre el mismo texto, combina resultados por unión, deduplica mediante normalización textual, y aplica diccionario canónico de 200 equivalencias que reduce 6,498 skills únicas a 3,200 formas estandarizadas. El control de calidad implementa filtrado multi-etapa: limpieza de HTML residual y validación de encoding en preprocesamiento, aplicación de listas de stopwords NER y técnicos genéricos en post-extracción para eliminar falsos positivos, y validación cruzada mediante overlap NER-Regex que asigna mayor confianza a skills detectadas por ambos métodos. El pipeline logra cobertura de 98.7 % del corpus con promedio de 50.3 skills por oferta, de las cuales 12.6 % mapean a taxonomía ESCO mientras 87.4 % representan tecnologías emergentes sin estandarización oficial.

La evaluación sobre gold standard de 300 ofertas laborales manualmente anotadas utilizó métricas de Information Retrieval debido a que la extracción de skills constituye un problema de multi-label retrieval en universo abierto donde la indefinición de True Negatives hace que Accuracy sea engañosa. Las métricas Pre-ESCO alcanzaron F1-Score 24.98 % con precision 22.54 % y recall 28.00 %, reflejando alto ruido en extracciones crudas. El mapeo a taxonomía ESCO mejoró drásticamente todas las métricas a F1-Score 72.53 %, precision 65.50 %, y recall 81.25 %, validando la efectividad del matcher de dos capas para normalizar variantes léxicas. La comparación con variante Regex-Only reveló que NER aporta 30 % de detecciones adicionales mediante menciones contextuales, aunque introduce ruido que reduce precision post-mapeo. Estos resultados validaron Pipeline A como baseline de alta cobertura para procesamiento del corpus completo en aproximadamente 8.3 horas, complementado estratégicamente con Pipeline B para enriquecimiento semántico de subconjuntos específicos donde se requiere mayor precisión contextual. La especificación técnica completa de Pipeline A, incluyendo arquitectura de componentes, patrones regex por categoría, flujo de integración detallado, estrategias de control de calidad, y evaluación exhaustiva con análisis estadístico completo, se documenta en el Apéndice F.

7.2.2 Pipeline B: Modelos de Lenguaje Grandes

Pipeline B constituye el método de enriquecimiento del observatorio, diseñado para complementar Pipeline A mediante extracción semánticamente consciente de habilidades implícitas, sinónimos contextuales, y competencias inferidas que no aparecen explícitamente mencionadas en el texto. Este pipeline aprovecha las capacidades de comprensión contextual profunda de Large Language Models (LLMs) para interpretar ofertas laborales de manera similar a como lo haría un analista humano.

La arquitectura de Pipeline B se diseñó con dos objetivos complementarios al Pipeline A: (1) aumentar la cobertura de skills implícitas que expresiones regulares no pueden capturar (“experiencia con cloud” $\rightarrow$ [“AWS”, “Azure”, “GCP”]), y (2) mejorar la precisión mediante comprensión de contexto que reduce falsos positivos (distinguir “Python” lenguaje vs. “Python” serpiente según contexto de oferta). Sin embargo, dado el mayor costo computacional de LLMs (típicamente 15-25s/oferta vs. 0.97s de Pipeline A), se implementó como pipeline de enriquecimiento estratégico aplicado a subconjuntos relevantes del corpus.

Justificación del Enfoque LLM

La decisión de implementar un pipeline basado en LLMs se fundamentó en cuatro limitaciones estructurales de Pipeline A que requerían comprensión semántica profunda. Primero, Pipeline A solo detecta skills mencionadas explícitamente mediante patrones léxicos, sin capacidad de inferir “Git” cuando una oferta solicita “experiencia con control de versiones”. Segundo, fragmenta skills compuestas al detectar “machine” y “learning” como tokens separados en lugar de “Machine Learning” como concepto único. Tercero, carece de desambiguación contextual, procesando “Python” como tecnología

incluso en ofertas de zoológicos que requieren conocimiento del reptil. Cuarto, no captura sinónimos contextuales como la equivalencia entre "backend development" y "server-side programming".

El enfoque LLM resuelve estas limitaciones mediante tres capacidades clave. La comprensión contextual permite interpretar ofertas considerando la semántica completa del texto en lugar de aplicar patrones sintácticos aislados, lo que habilita inferir "Docker" de "experiencia en contenedorización" sin requerir match textual directo. La desambiguación semántica utiliza el contexto de la oferta para distinguir tecnologías de homónimos ("Python" + "Django" identifica lenguaje de programación; "Python" + "zoo" identifica animal). Más importante, los LLMs son capaces de identificar skills emergentes que aún no están codificadas en diccionarios estáticos: tecnologías recientes como "Claude Code", "Cursor IDE" o "v0.dev" son detectables por modelos pre-entrenados con conocimiento actualizado, mientras que Pipeline A requeriría actualización manual de sus expresiones regulares. Esta capacidad de capturar vocabulario emergente del mercado laboral representa la ventaja diferencial más significativa del enfoque LLM.

Los resultados cuantitativos validan esta arquitectura dual. Pipeline B con Gemma 3 4B alcanza 84.26 % F1 Post-ESCO versus 72.53 % de Pipeline A (+11.73pp mejora), extrayendo en promedio 27.8 skills por oferta con mayor relevancia contextual. Sin embargo, el costo computacional es significativamente superior: la mediana de latencia es 18.3 segundos por oferta (percentiles P25-P75: 15-25s), con media de 42.1 segundos inflada por ofertas extensas que contienen múltiples páginas de requisitos. La cuantización INT4 mediante `bitsandbytes` reduce requisitos de memoria de $\sim$16GB (FP16) a $\sim$4GB (INT4), permitiendo inferencia en GPUs consumer sin dependencia de APIs comerciales. Este balance entre calidad (+11.73pp F1) y latencia (18× más lento que Pipeline A) justifica la aplicación selectiva de Pipeline B a subconjuntos estratégicos del corpus que requieren análisis semántico profundo.

Selección del Modelo

Se evaluaron cuatro modelos de lenguaje open-source de tamaño intermedio que cumplían los requisitos de ejecución local con cuantización INT4: Gemma 3 4B Instruct de Google DeepMind, Llama 3.2 3B Instruct de Meta AI, Qwen 2.5 3B Instruct de Alibaba Cloud, y Phi-3.5 Mini Instruct de Microsoft Research. Todos los modelos se ejecutaron con cuantización INT4 mediante `bitsandbytes`, reduciendo requisitos de memoria de aproximadamente 16GB a 4GB para permitir inferencia local en hardware consumer. La evaluación preliminar sobre 10 ofertas laborales del gold standard analizó tanto métricas cuantitativas como comportamiento cualitativo considerando alucinaciones, consistencia de formato, y relevancia de extracciones.

Gemma 3 4B Instruct fue seleccionado como modelo final por su desempeño superior: alcanzó 46.23 % F1 Pre-ESCO extrayendo promedio de 27.8 skills por oferta, generó JSON válido en 99 % de casos sin alucinaciones sistemáticas, y exhibió latencia mediana de 18.3 segundos por oferta. Los modelos alternativos presentaron limitaciones significativas: Llama 3.2 3B mostró alucinaciones sis-

temáticas agregando tecnologías irrelevantes, Qwen 2.5 3B generó salidas no estructuradas en 40 % de casos requiriendo parsing manual, y Phi-3.5 Mini tuvo baja cobertura con solo 12.4 skills por oferta. El balance entre calidad de extracción, estabilidad de formato, y ausencia de alucinaciones justificó la selección de Gemma 3 4B como motor de Pipeline B, a pesar de su mayor latencia comparado con alternativas menos confiables.

Arquitectura del Sistema

La arquitectura de Pipeline B implementa procesamiento asíncrono con Gemma 3 4B cuantizado INT4 mediante `bitsandbytes`, permitiendo inferencia local con 4GB de memoria GPU. El flujo ejecuta prompt engineering estructurado que instruye al modelo a extraer skills técnicas en formato JSON, incluyendo tanto menciones explícitas como inferencias contextuales basadas en descripciones de responsabilidades. El sistema implementa validación de salida mediante Pydantic para garantizar schemas consistentes, retry logic con backoff exponencial para manejar fallos transitorios, y logging exhaustivo de latencias y tokens procesados para monitoreo de performance. El procesamiento batch aplica batch size de 1 oferta por inferencia debido a restricciones de memoria, resultando en throughput de aproximadamente 150-200 ofertas por hora en hardware consumer. La especificación técnica completa de Pipeline B, incluyendo arquitectura de componentes, diseño de prompts, estrategias de validación, manejo de errores, y evaluación exhaustiva, se documenta en el Apéndice F.

7.2.3 Pipelines Alternativos Evaluados

Pipeline A.1 basado en TF-IDF + filtrado por noun phrases se implementó como experimento alternativo. Utilizó `scikit-learn.TfidfVectorizer(ngram_range=(1,3),max_features=10000)` extrayendo top-50 n-gramas por oferta, filtrados por part-of-speech con spaCy. Las pruebas sobre 100 ofertas gold standard revelaron limitaciones críticas: 60 % de candidatos eran frases descriptivas no-skills, fragmentación excesiva (“React” y “Native” separados), y F1=11.69 %. Pipeline A.1 se descartó por performance inadecuado versus Pipeline A.

La configuración Regex-Only reutilizó los 548 patrones de Pipeline A eliminando NER, estableciendo baseline determinístico. Sobre el gold standard de 300 ofertas alcanzó F1 Post-ESCO de 79.17 % con precision 86.36 % y recall 73.08 %, procesando en menos de 1ms por oferta. Extrajo promedio de 35.2 skills por oferta versus 50.3 del pipeline combinado, confirmando que NER contribuye aproximadamente 30 % de detecciones adicionales. Los resultados validaron que regex solo proporciona precision superior pero menor recall Pre-ESCO al omitir menciones contextuales que NER captura.

7.3 Implementación del Sistema de Mapeo a Taxonomía ESCO

El mapeo de habilidades a taxonomía ESCO constituye una etapa crítica del observatorio, responsable de normalizar las extracciones de Pipeline A y Pipeline B a un vocabulario controlado estándar. Esta normalización es fundamental para garantizar la comparabilidad de resultados entre países, portales y períodos temporales, eliminando la fragmentación causada por variantes sintácticas de la misma habilidad.

3. Componente de Integración y Normalización:

- **Deduplicación**: Eliminación de duplicados mediante normalización textual
- **Normalización**: Diccionario canónico de 200+ equivalencias ("js" → "JavaScript", "k8s" → "Kubernetes")
- **Niveles de output**:
  - *Raw extractions*: Metadata de método, posición, confianza
  - *Normalized extractions*: Formas canónicas estandarizadas
- **Reducción de vocabulario**: De 6,498 skills únicas a 3,200 formas canónicas

Flujo de Integración y Procesamiento

La integración de NER y Regex opera mediante el siguiente flujo secuencial:

1. **Ejecución de NER**: Se procesa el texto de la oferta con spaCy (título + descripción + requisitos)
   - Output: Lista de entidades detectadas con posiciones y labels
   - Tiempo: 0.65s promedio por oferta
2. **Ejecución de Regex**: Se aplican los 548 patrones sobre el mismo texto
   - Output: Lista de matches con posiciones y patrones que generaron el match
   - Tiempo: 0.32s promedio por oferta
3. **Combinación por unión**: Se unen ambas listas de skills extraídas
   - Criterio: Mantener todas las extracciones de ambos métodos
   - Metadata: Cada skill incluye campo `extraction_method` ("NER", "Regex", o "Both")
4. **Deduplicación**: Se eliminan duplicados mediante normalización textual
   - Normalización: Lowercase, eliminación de acentos, eliminación de puntuación
   - Criterio: Skills con texto normalizado idéntico se consideran duplicadas

- Prioridad: Si detectada por ambos métodos, se marca como `extraction_method=''Both''` con mayor confianza

5. **Normalización canónica**: Se aplica diccionario de equivalencias
   - Diccionario: 200+ reglas mapeando variantes a formas canónicas
   - Ejemplos: "js" → "JavaScript", "k8s" → "Kubernetes", "postgres" → "PostgreSQL"
   - Resultado: Reducción de 6,498 skills únicas a 3,200 formas canónicas
6. **Generación de outputs**: Se producen dos niveles de extracciones
   - *Raw extractions*: Skills tal como fueron extraídas, con metadata completa (método, posición, confianza)
   - *Normalized extractions*: Skills en formas canónicas estandarizadas, listas para mapeo ESCO
7. **Persistencia**: Se almacenan en tabla `extracted_skills` de PostgreSQL
   - Campos: `job_id`, `skill_text`, `extraction_method`, `confidence`, `position_start`, `position_end`

Performance del flujo completo:

- Latencia total: 0.97s por oferta (0.65s NER + 0.32s Regex)
- Throughput: 3,700 ofertas/hora en CPU (sin paralelización)
- Tiempo proyectado para corpus completo: 8.3 horas para 30,660 ofertas

Control de Calidad y Validación

Para garantizar la calidad de las extracciones del Pipeline A, se implementaron múltiples mecanismos de validación y filtrado que operan en diferentes etapas del procesamiento.

Validación de entrada (Pre-procesamiento):

- **Limpieza de HTML residual**: Eliminación de tags, entidades HTML, y scripts JavaScript
- **Normalización de encoding**: Conversión a UTF-8, manejo de caracteres especiales
- **Detección de idioma**: Identificación de español/inglés/Spanglish mediante `langdetect`
- **Validación de longitud**: Descarte de ofertas con description $<$100 caracteres (probablemente incompletas)

Filtrado de falsos positivos (Post-extracción):

- **Stopwords NER**: Lista de 200+ términos genéricos descartados
  - Categorías: nombres comunes, verbos genéricos, adjetivos, conectores
  - Ejemplos: “desarrollo”, “experiencia”, “conocimiento”, “trabajo”, “equipo”
- **Stopwords técnicos genéricos**: Lista de 60+ términos técnicos demasiado amplios
  - Categorías: términos paraguas, buzzwords, soft skills genéricas
  - Ejemplos: “software”, “technology”, “programming”, “innovation”, “excellence”
- **Validación de longitud de skills**: Descarte de extracciones <2 o >50 caracteres
- **Validación de caracteres**: Descarte de skills solo-numéricos o con caracteres especiales sin match ESCO

Validación cruzada y coherencia:

- **Overlap NER-Regex**: Skills detectadas por ambos métodos reciben mayor score de confianza
- **Frecuencia en corpus**: Skills únicas (aparecen en 1 sola oferta) se marcan para revisión manual
- **Validación con ESCO**: Skills sin match en taxonomía se categorizan como “emergentes” para análisis posterior

Métricas de calidad monitoreadas:

- **Stopword filtering**: Aplicación de listas de stopwords NER (200+ términos) y técnicos genéricos (60+ términos) redujo significativamente falsos positivos
- **Coverage rate**: Porcentaje de ofertas con al menos 1 skill extraída
  - Pipeline A: 98.7 % (30,264 de 30,660 ofertas)
- **Extraction diversity**: Número de skills únicas por oferta
  - Promedio: 50.3 skills/oferta
  - Mediana: 42 skills/oferta
  - Percentil 95: 87 skills/oferta
- **ESCO match rate**: Porcentaje de skills extraídas que mapearon a taxonomía ESCO
  - Pipeline A: 12.6 % (baja cobertura indica presencia de skills emergentes no presentes en ESCO v1.1.0)

Estos mecanismos de control de calidad mejoraron la precisión de las extracciones mediante filtrado progresivo, especialmente efectivo al combinar con mapeo ESCO que consolida variantes ortográficas y elimina ruido residual.

Evaluación del Pipeline A

La evaluación del Pipeline A se realizó mediante comparación contra el gold standard de 300 ofertas laborales manualmente anotadas (100 por país: Colombia, México, Argentina), utilizando las métricas estándar de Information Retrieval: Precision, Recall y F1-Score.

Justificación de métricas. No se utilizó Accuracy debido a las siguientes razones fundamentales:

1. Naturaleza del problema: La extracción de skills es un problema de *multi-label retrieval* en universo abierto, no de clasificación binaria. Cada oferta contiene múltiples skills válidas (promedio: 20-30 por oferta), y el sistema debe identificar un subconjunto correcto de un espacio de candidatos potencialmente infinito. Este tipo de problema requiere métricas que evalúen la calidad del subset extraído, no la clasificación de instancias individuales.

2. Desbalance extremo de clases: Para cada oferta laboral con aproximadamente 500 palabras, el espacio de candidatos presenta desbalance masivo:

- Positivos (skills reales): $\sim$20-30 términos
- Negativos potenciales (n-gramas que NO son skills): $\sim$10,000+ combinaciones posibles

En este escenario, un modelo trivial que nunca extraiga nada tendría Accuracy $>$99 % (al predecir correctamente que 9,970 de 10,000 candidatos no son skills), pero sería completamente inútil al no detectar ninguna skill real. Por tanto, Accuracy es una métrica engañosa que no refleja la utilidad práctica del sistema.

3. Indefinición de True Negatives (TN): En problemas de extracción de información, el conjunto de candidatos negativos no tiene una definición natural unívoca. Para cada oferta, podrían considerarse como candidatos negativos:

- Todas las palabras individuales del texto ($\sim$500 candidatos)
- Todos los n-gramas posibles (1-4 palabras) ($\sim$10,000 candidatos)
- Todas las skills de la taxonomía ESCO (14,215 candidatos)
- Todo el vocabulario técnico español-inglés (cientos de miles de términos)

Como TN depende arbitrariamente de cómo se defina el universo de candidatos, la métrica Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$ no tiene una interpretación consistente ni reproducible. Dos evaluaciones con diferente definición del universo de candidatos producirían valores de Accuracy radicalmente distintos para el mismo sistema.

Por estas razones, se adoptaron las métricas estándar de Information Retrieval (Precision, Recall, F1-Score) que operan exclusivamente sobre los conjuntos de skills extraídas y anotadas, evitando la indefinición de True Negatives inherente al problema de extracción en universo abierto.

Cálculo de métricas mediante operaciones de conjuntos. Para cada oferta laboral j, se definen:

- G_j: Conjunto de skills del gold standard para el job j (después de normalización canónica)
- P_j: Conjunto de skills extraídas por el pipeline para el job j (después de normalización)

Las métricas se calculan mediante agregación micro-averaged sobre los $N = 300$ jobs del gold standard:

$$TP = \sum_{j=1}^{N} |G_j \cap P_j| \quad \text{(skills correctamente extraídas)} \tag{7.1}$$

$$FP = \sum_{j=1}^{N} |P_j \setminus G_j| \quad \text{(extraídas pero no en gold standard)} \tag{7.2}$$

$$FN = \sum_{j=1}^{N} |G_j \setminus P_j| \quad \text{(en gold standard pero no extraídas)} \tag{7.3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{(proporción correcta de extracciones)} \tag{7.4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{(cobertura del gold standard)} \tag{7.5}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{(media armónica)} \tag{7.6}$$

Donde $|S|$ denota la cardinalidad del conjunto S, $\cap$ es la intersección, y $\setminus$ es la diferencia de conjuntos. Esta metodología es estándar en tareas de Named Entity Recognition (NER) y Information Extraction, siguiendo el framework de evaluación CoNLL ampliamente adoptado por la comunidad de NLP internacional.

Normalización canónica. Previo al cálculo de métricas, todas las skills (tanto del gold standard como extraídas) se normalizan mediante un diccionario de 200+ formas canónicas que mapean variantes ortográficas a términos estándar (ej: "postgres" $\rightarrow$ "PostgreSQL", "js" $\rightarrow$ "JavaScript", "k8s" $\rightarrow$ "Kubernetes"). Esta normalización permite comparación textual justa sin sesgar por variaciones superficiales, capturando correctamente matches semánticos mientras se mantiene sensibilidad a diferencias reales entre tecnologías (ej: "Java" vs "JavaScript" no se normalizan entre sí).

Evaluación dual Pre-ESCO y Post-ESCO. El sistema implementa evaluación en dos escenarios complementarios:

- **Pre-ESCO**: Comparación sobre texto normalizado sin mapeo taxonómico, evaluando capacidad de extracción pura incluyendo skills emergentes no estandarizadas
- **Post-ESCO**: Comparación después de mapear todas las skills a taxonomía ESCO, evaluando capacidad de estandarización y alineación con vocabulario controlado

Esta evaluación dual permite cuantificar el trade-off entre flexibilidad (captura de tecnologías

emergentes) versus estandarización (alineación con taxonomías oficiales), documentando el impacto del mapeo ESCO en la performance medida de cada pipeline.

Resultados obtenidos. La evaluación del Pipeline A sobre las 300 ofertas del gold standard produjo las siguientes métricas. El pipeline extrae en promedio 50.3 skills por oferta, con una cobertura ESCO del 12.6 % (87.4 % son skills emergentes no presentes en ESCO v1.1.0).

Métricas de rendimiento:

- **Pre-ESCO**: F1-Score 24.98 %, Precision 22.54 %, Recall 28.00 %
- **Post-ESCO**: F1-Score 72.53 % (+47.55pp), Precision 65.50 % (+42.96pp), Recall 81.25 % (+53.25pp)

El mapeo a taxonomía ESCO mejora drásticamente todas las métricas, particularmente recall (+53.25pp), lo que valida la efectividad del matcher de dos capas (exacto + difuso) para normalizar variantes léxicas.

Comparativa con variantes del pipeline:

- **Regex-Only**: Mayor F1 Post-ESCO (79.17 %, +6.64pp) pero menor F1 Pre-ESCO (18.07 %, -6.91pp). Ventaja en precisión post-normalización, desventaja en cobertura de variantes contextuales.
- **Pipeline A (NER+Regex)**: Mejor F1 Pre-ESCO (24.98 %, +6.91pp) pero menor F1 Post-ESCO (72.53 %, -6.64pp). El NER captura menciones contextuales pero introduce ruido que reduce precision post-mapeo.

Contribución de cada componente:

- **Regex**: Detecta 35.2 skills/oferta promedio (70 % del total)
- **NER**: Aporta 15.1 skills/oferta adicionales (30 % del total)
- **Overlap**: 12 % de skills detectadas por ambos métodos, validando la consistencia

Capacidades validadas del pipeline:

- **Velocidad**: Procesamiento 43× más rápido que Pipeline B, permitiendo aplicación a corpus completo
- **Reproducibilidad**: Resultados 100 % deterministas (sin aleatoriedad)
- **Escalabilidad**: Latencia lineal O(n), corpus completo procesable en menos de 10 horas

Limitaciones identificadas del pipeline:

- **Baja precision Pre-ESCO** (22.54 %): Alto ruido en extracciones crudas, requiere filtrado con ESCO para uso práctico

- **Cobertura ESCO limitada**: 87.4 % de skills extraídas no mapean a taxonomía oficial, reflejando vocabulario emergente del mercado
- **Fragmentación léxica**: Skills compuestas ("machine learning") a veces detectadas como tokens separados
- **Sensibilidad a ruido HTML**: Ofertas mal limpiadas generan falsos positivos en extracción

Estos resultados validaron la viabilidad del Pipeline A como baseline de alta cobertura para procesamiento del 100 % del corpus, complementado con Pipeline B para enriquecimiento semántico de un subconjunto estratégico. La arquitectura dual permite balancear velocidad (Pipeline A) con calidad (Pipeline B), optimizando el trade-off precision/latencia según el escenario de uso.

7.3.1 Pipeline B: Modelos de Lenguaje Grandes

Pipeline B constituye el método de enriquecimiento del observatorio, diseñado para complementar Pipeline A mediante extracción semánticamente consciente de habilidades implícitas, sinónimos contextuales, y competencias inferidas que no aparecen explícitamente mencionadas en el texto. Este pipeline aprovecha las capacidades de comprensión contextual profunda de Large Language Models (LLMs) para interpretar ofertas laborales de manera similar a como lo haría un analista humano.

La arquitectura de Pipeline B se diseñó con dos objetivos complementarios al Pipeline A: (1) aumentar la cobertura de skills implícitas que expresiones regulares no pueden capturar ("experiencia con cloud" $\rightarrow$ ["AWS", "Azure", "GCP"]), y (2) mejorar la precisión mediante comprensión de contexto que reduce falsos positivos (distinguir "Python" lenguaje vs. "Python" serpiente según contexto de oferta). Sin embargo, dado el mayor costo computacional de LLMs (típicamente 15-25s/oferta vs. 0.97s de Pipeline A), se implementó como pipeline de enriquecimiento estratégico aplicado a subconjuntos relevantes del corpus.

Justificación del Enfoque LLM

La decisión de implementar un pipeline basado en LLMs se fundamentó en cuatro limitaciones estructurales de Pipeline A que requerían comprensión semántica profunda. Primero, Pipeline A solo detecta skills mencionadas explícitamente mediante patrones léxicos, sin capacidad de inferir "Git" cuando una oferta solicita "experiencia con control de versiones". Segundo, fragmenta skills compuestas al detectar "machine" y "learning" como tokens separados en lugar de "Machine Learning" como concepto único. Tercero, carece de desambiguación contextual, procesando "Python" como tecnología incluso en ofertas de zoológicos que requieren conocimiento del reptil. Cuarto, no captura sinónimos contextuales como la equivalencia entre "backend development" y "server-side programming".

El enfoque LLM resuelve estas limitaciones mediante tres capacidades clave. La comprensión contextual permite interpretar ofertas considerando la semántica completa del texto en lugar de aplicar patrones sintácticos aislados, lo que habilita inferir "Docker" de "experiencia en contenedoriza-

ción" sin requerir match textual directo. La desambiguación semántica utiliza el contexto de la oferta para distinguir tecnologías de homónimos ("Python" + "Django" identifica lenguaje de programación; "Python" + "zoo" identifica animal). Más importante, los LLMs son capaces de identificar skills emergentes que aún no están codificadas en diccionarios estáticos: tecnologías recientes como "Claude Code", "Cursor IDE" o "v0.dev" son detectables por modelos pre-entrenados con conocimiento actualizado, mientras que Pipeline A requeriría actualización manual de sus expresiones regulares. Esta capacidad de capturar vocabulario emergente del mercado laboral representa la ventaja diferencial más significativa del enfoque LLM.

Los resultados cuantitativos validan esta arquitectura dual. Pipeline B con Gemma 3 4B alcanza 84.26 % F1 Post-ESCO versus 72.53 % de Pipeline A (+11.73pp mejora), extrayendo en promedio 27.8 skills por oferta con mayor relevancia contextual. Sin embargo, el costo computacional es significativamente superior: la mediana de latencia es 18.3 segundos por oferta (percentiles P25-P75: 15-25s), con media de 42.1 segundos inflada por ofertas extensas que contienen múltiples páginas de requisitos. La cuantización INT4 mediante `bitsandbytes` reduce requisitos de memoria de $\sim$16GB (FP16) a $\sim$4GB (INT4), permitiendo inferencia en GPUs consumer sin dependencia de APIs comerciales. Este balance entre calidad (+11.73pp F1) y latencia (18× más lento que Pipeline A) justifica la aplicación selectiva de Pipeline B a subconjuntos estratégicos del corpus que requieren análisis semántico profundo.

Selección del Modelo

Se evaluaron cuatro modelos de lenguaje open-source de tamaño intermedio (3-4B parámetros) que cumplían los requisitos de ejecución local con cuantización:

Modelos candidatos evaluados:

1. **Gemma 3 4B Instruct** (Google DeepMind)
2. **Llama 3.2 3B Instruct** (Meta AI)
3. **Qwen 2.5 3B Instruct** (Alibaba Cloud)
4. **Phi-3.5 Mini Instruct** (Microsoft Research)

Configuración de evaluación:

Todos los modelos se ejecutaron con cuantización INT4 mediante `bitsandbytes`, reduciendo requisitos de memoria de $\sim$16GB (FP16) a $\sim$4GB (INT4) para permitir inferencia local en hardware consumer (MacBook Air M4 base con 16GB RAM unificada) con batch size 1. La evaluación preliminar se realizó sobre 10 ofertas laborales del gold standard, analizando tanto métricas cuantitativas (F1-Score Pre-ESCO) como comportamiento cualitativo (alucinaciones, consistencia de formato, relevancia de extracciones).

Resultados de la evaluación comparativa:

La Tabla 7.1 presenta los resultados de la evaluación comparativa de los cuatro modelos candidatos sobre 10 ofertas laborales del gold standard.

Tabla 7.1: Comparación de modelos LLM para extracción de habilidades

Modelo	F1 Pre-ESCO	Skills/oferta	Observaciones clave
Gemma 3 4B	46.23 %	27.8	JSON válido (99 %), sin alucinaciones, latencia mediana 18.3s
Llama 3.2 3B	39.7 %	—	Alucinaciones sistemáticas (agregaba TensorFlow/Pandas en ofertas frontend)
Qwen 2.5 3B	38.9 %	10-12	Extracciones conservadoras, alta precisión pero baja cobertura
Phi-3.5 Mini	35.2 %	—	Formato inconsistente (40 % no-JSON), requiere parser complejo

Decisión final. Se seleccionó Gemma 3 4B Instruct como modelo de producción tras evaluar cuatro dimensiones críticas. Primero, alcanzó el mejor desempeño cuantitativo con 46.23 % F1 Pre-ESCO y 84.26 % F1 Post-ESCO, superando a Llama 3.2 por +6.5pp y a Qwen 2.5 por +7.3pp. Segundo, demostró estabilidad de formato con 99 % de respuestas en JSON válido (299/300 ofertas procesadas exitosamente), eliminando la necesidad de parsers heurísticos frágiles como los requeridos por Phi-3.5. Tercero, la revisión manual de 300 ofertas no detectó alucinaciones sistemáticas, contrastando con Llama 3.2 que agregaba skills de Data Science (TensorFlow, Pandas) en ofertas de desarrollo web tradicional sin componente analítico. Cuarto, la latencia mediana de 18.3 segundos por oferta (percentiles P25-P75: 15-25s) permitió procesar el gold standard completo en 3.5 horas, tiempo aceptable para pipelines de enriquecimiento no-interactivos.

Esta decisión prioriza el atributo de calidad de confiabilidad sobre maximización de métricas aisladas. Un pipeline de producción debe generar resultados predecibles sin alucinaciones que contaminen análisis agregados, incluso si esto implica sacrificar mejoras marginales de F1-Score. La arquitectura robusta de Gemma 3 4B (formato estructurado consistente + ausencia de sesgos evidentes) lo posiciona como fundamento confiable para el componente semántico del observatorio.

Componentes del Pipeline B

Pipeline B se compone de tres módulos secuenciales que transforman texto de oferta laboral en lista estructurada de habilidades:

1. Módulo de Construcción de Prompt:

- **Función**: Ensamblar prompt estructurado combinando campos de oferta laboral con instrucciones de extracción
- **Implementación**: Template con tres secciones:

  - *System prompt*: Define rol del LLM ("experto en análisis de ofertas laborales") y formato de salida (JSON con lista de skills)
  - *Contexto de oferta*: Incluye `job_title`, `description`, `requirements` concatenados
  - *Instrucciones*: Especifica criterios de extracción (skills técnicas, herramientas, lenguajes, frameworks, no soft skills)
- **Template completo**: El prompt de 280 líneas con ejemplos, reglas y casos negativos se documenta en el Apéndice A
- **Salida**: String de prompt de longitud variable (650-1200 tokens según verbosidad de oferta)

2. Módulo de Inferencia LLM:

- **Función**: Ejecutar inferencia con Gemma 3 4B generando lista de skills en formato JSON
- **Implementación**:
  - Tokenización con truncamiento a 3800 tokens (límite empírico para evitar OOM en RTX 3090)
  - Generación con `temperature=0.3` (balance entre determinismo y diversidad)
  - `max_new_tokens=512` (suficiente para 20-30 skills con descripciones)
  - Sin batching real: Procesamiento secuencial debido a variabilidad de longitud de ofertas
- **Metadatos registrados**: `llm_model`, `inference_time_seconds`, `prompt_tokens`, `generated_tokens`
- **Salida**: String JSON con estructura {`"skills": ["Python", "Django", "PostgreSQL"]`}

3. Módulo de Parsing y Normalización:

- **Función**: Extraer skills de respuesta LLM y normalizar formato
- **Implementación**:
  - *Parser JSON primario*: Intenta `json.loads()` sobre respuesta completa
  - *Fallback regex*: Si falla, busca patrones `\[.*?\]` o `\.*?\` y extrae listas
  - *Normalización*: Lowercase, eliminación de acentos, eliminación de duplicados, filtrado de strings vacíos
  - *Validación*: Descarta skills de longitud <2 caracteres o >100 caracteres (probable ruido)
- **Manejo de errores**: Si parsing falla completamente, registra oferta en `failed_extractions` con flag `parse_success=False`
- **Salida**: Lista normalizada de skills únicas

Flujo de Inferencia y Procesamiento

El procesamiento de Pipeline B sigue una secuencia de 6 pasos principales, con manejo de errores en cada etapa:

1. **Carga de ofertas desde base de datos**:
   - Query sobre tabla `job_postings` filtrando por `job_id IN (gold_standard_ids)`
   - Retrieve campos: `job_id`, `job_title`, `description`, `requirements`, `raw_text`
   - Batch de 50 ofertas por iteración para control de memoria
2. **Construcción de prompts**:
   - Aplicar template de prompt a cada oferta
   - Medir longitud en tokens; si >3800, truncar campo `description` preservando `job_title` y `requirements`
   - Almacenar prompts en memoria temporal (no se persisten)
3. **Inferencia con LLM**:
   - Procesar ofertas secuencialmente (sin paralelización GPU por restricción de VRAM)
   - Timeout de 120s/oferta; si excede, marcar como `timeout_error` y continuar
   - Capturar excepciones CUDA (OOM en ofertas >4000 tokens); marcar como `cuda_error`
   - Registrar tiempo de inicio y fin para cálculo de latencia
4. **Parsing de respuestas**:
   - Intentar parsing JSON; si falla, aplicar regex fallback
   - Si ambos fallan, marcar `parse_success=False` y skills como lista vacía
   - Normalizar skills exitosamente parseadas
5. **Persistencia en base de datos**:
   - Insertar registros en tabla `extracted_skills` con `extraction_method='llm'`
   - Insertar metadatos en `extraction_metadata`: `llm_model`, `inference_time_seconds`, `parse_success`
   - Batch insert de 100 registros para optimizar I/O
6. **Monitoreo y logging**:
   - Registrar progreso cada 10 ofertas procesadas

- Acumular estadísticas: Total procesado, exitoso, timeouts, errores CUDA, parse failures
- Al finalizar batch, reportar métricas agregadas: Latencia promedio, tasa de éxito, skills/oferta promedio

Métricas de rendimiento del flujo:

- **Latencia**: Media 45.17s, Mediana 17.66s (distribución bimodal por outliers)
  - P25: 13.71s, P75: 23.81s, P95: 196.23s
  - Mínimo: 5.77s, Máximo: 254.10s
- **Distribución por rangos de tiempo**:
  - $<$30s: 241 jobs (80.6 %) - Promedio: 16.48s (procesamiento eficiente)
  - 30-60s: 11 jobs (3.7 %) - Promedio: 36.02s
  - $>$60s: 47 jobs (15.7 %) - Promedio: 194.42s (outliers con ofertas complejas/extensas)
- **Tasa de éxito**: 299 de 300 ofertas procesadas exitosamente (99.7 %)
- **Parse success**: 99 % de respuestas en formato JSON válido (299/300)
- **Tiempo total (299 ofertas gold standard)**: 3.5 horas (13,456 segundos totales)

Nota sobre distribución bimodal: La media (45.17s) es significativamente mayor que la mediana (17.66s) debido a 47 ofertas outliers (15.7 %) con tiempos $>$60s. El 80.6 % de ofertas se procesa en $<$30s con promedio de 16.48s, demostrando eficiencia del modelo para casos típicos. Los outliers corresponden a ofertas con descripciones extensas ($>$3000 tokens) o formato complejo que requieren mayor tiempo de inferencia.

Resultados de la Implementación

La evaluación de Pipeline B sobre el gold standard de 300 ofertas laborales reveló un perfil de desempeño complementario a Pipeline A. El pipeline procesó exitosamente 299/300 ofertas (99.7 %), extrayendo en promedio 27.8 skills por oferta versus 50.3 de Pipeline A. Las métricas Pre-ESCO fueron F1-Score 46.23 %, Precision 40.89 %, Recall 53.08 %; tras mapeo ESCO mejoraron drásticamente a F1 84.26 % (+38.03pp), Precision 89.25 % (+48.36pp), Recall 79.81 % (+26.73pp).

Comparativa con Pipeline A. La evaluación head-to-head sobre el mismo gold standard cuantificó trade-offs claros entre ambos pipelines. Pipeline B supera en precision Post-ESCO con 89.25 % versus 65.50 % de Pipeline A (+23.75pp), reflejando menor ruido en extracciones gracias a comprensión contextual del LLM. El F1 Post-ESCO también favorece a Pipeline B con 84.26 % versus 72.53 % (+11.73pp). Sin embargo, Pipeline A mantiene ligera ventaja en recall Post-ESCO con 81.25 % versus

79.81 % (-1.44pp para Pipeline B) y extrae 1.8× más skills por oferta (50.3 vs 27.8), capturando mayor volumen de tecnologías de long-tail y emergentes. El costo de esta mayor calidad es latencia: Pipeline B requiere mediana de 17.66 segundos por oferta versus 0.97s de Pipeline A (18× más lento), con media de 45.17s inflada por outliers de hasta 254 segundos en ofertas extremadamente extensas.

Análisis de complementariedad. Los resultados validan que ambos pipelines cumplen roles diferenciados en la arquitectura del observatorio. Pipeline A maximiza cobertura volumétrica y velocidad, permitiendo procesamiento del corpus completo (30,660 ofertas) en menos de 10 horas con captura exhaustiva de skills emergentes no presentes en ESCO. Pipeline B maximiza relevancia contextual y precision, extrayendo skills más alineadas con taxonomía estándar pero con costo computacional 18-47× superior que lo limita a aplicación selectiva. La arquitectura dual permite balancear estos trade-offs según el escenario de uso: Pipeline A como baseline de producción para análisis agregados, Pipeline B como enriquecimiento semántico para estudios cualitativos profundos sobre subconjuntos estratégicos.

Capacidades validadas del pipeline. El análisis cualitativo sobre 299 ofertas confirmó cinco capacidades diferenciadas de Pipeline B. Primero, extracción de skills implícitas alcanzó cobertura de 126 % respecto a anotación manual en soft skills, infiriendo competencias no mencionadas explícitamente ("liderarás equipo" $\rightarrow$ "Liderazgo"). Segundo, la alta precision (89.25 %) indica desambiguación contextual efectiva, distinguiendo tecnologías de homónimos según contexto de oferta. Tercero, genera skills en formato estándar consistente con ESCO sin requerir normalización manual posterior. Cuarto, 59.5 % de skills extraídas son tecnologías modernas no presentes en ESCO v1.1.0, demostrando capacidad de capturar vocabulario emergente del mercado. Quinto, la cuantización Q4 mediante `bitsandbytes` permite inferencia local con 4-6GB VRAM sin dependencia de APIs comerciales, reduciendo requisitos de memoria de $\sim$16GB (FP16) a $\sim$4GB.

Limitaciones identificadas del pipeline. Cinco limitaciones técnicas restringen la aplicabilidad de Pipeline B a escala completa. La latencia variable de 18-47× superior a Pipeline A (mediana 17.66s vs 0.97s) con outliers hasta 254 segundos hace inviable el procesamiento de corpus completo: 30,660 ofertas requerirían 385 horas (16 días) con tiempo promedio versus 8.3 horas de Pipeline A. La cobertura volumétrica reducida (27.8 vs 50.3 skills/oferta, -44.7 %) limita detección de tecnologías de nicho. El recall Post-ESCO ligeramente inferior (79.81 % vs 81.25 %, -1.44pp) indica que el LLM ocasionalmente omite skills explícitas capturadas por regex. La dependencia de GPU con 4-6GB VRAM hace la solución no portable a entornos CPU-only. Finalmente, la variabilidad de latencia según longitud de oferta (correlación 0.73 entre tokens y segundos de inferencia) dificulta planificación de recursos en producción.

Estos resultados validaron la arquitectura dual del observatorio: Pipeline A proporciona baseline de alta cobertura para procesamiento del 100 % del corpus, mientras Pipeline B se aplicó experimentalmente sobre el gold standard para evaluar su viabilidad como método de enriquecimiento selectivo en escenarios que priorizan calidad sobre velocidad.

7.3.2 Pipeline A.1 (TF-IDF) y Regex-Only Baseline

Pipeline A.1 basado en TF-IDF + filtrado por noun phrases se implementó como experimento alternativo. Utilizó `scikit-learn.TfidfVectorizer(ngram_range=(1,3),max_features=10000)` extrayendo top-50 n-gramas por oferta, filtrados por part-of-speech con spaCy. Las pruebas sobre 100 ofertas gold standard revelaron limitaciones críticas: 60 % de candidatos eran frases descriptivas no-skills, fragmentación excesiva ("React" y "Native" separados), y F1=11.69 %. **Pipeline A.1 se descartó** por performance inadecuado versus Pipeline A (F1=72.53 %).

La configuración Regex-Only reutilizó los 548 patrones de Pipeline A eliminando NER, estableciendo baseline determinístico. Sobre el gold standard de 300 ofertas alcanzó F1 Post-ESCO de 79.17

7.4 Implementación del Sistema de Mapeo a Taxonomía ESCO

El mapeo de habilidades a taxonomía ESCO constituye una etapa crítica del observatorio, responsable de normalizar las extracciones de Pipeline A y Pipeline B a un vocabulario controlado estándar. Esta normalización es fundamental para garantizar la comparabilidad de resultados entre países, portales y períodos temporales, eliminando la fragmentación causada por variantes sintácticas de la misma habilidad.

Problemática que resuelve el mapeo ESCO:

Las extracciones crudas de Pipeline A y Pipeline B presentan alta fragmentación léxica debido a variantes ortográficas, abreviaciones, y diferencias idiomáticas:

- **Variantes ortográficas**: "React", "React.js", "ReactJS", "react"
- **Abreviaciones**: "JS" vs. "JavaScript", "K8s" vs. "Kubernetes"
- **Diferencias idiomáticas**: "Base de datos" vs. "Database", "Aprendizaje automático" vs. "Machine Learning"
- **Niveles de especificidad**: "SQL" vs. "PostgreSQL" vs. "PostgreSQL 15"

Sin normalización, estas variantes se tratarían como habilidades distintas en el análisis de clustering y tendencias, fragmentando artificialmente los resultados y degradando la calidad de las visualizaciones.

Funciones del sistema de mapeo:

El sistema de mapeo a ESCO cumple tres funciones principales:

1. **Normalización léxica**: Mapear todas las variantes de una habilidad a un URI canónico ESCO único
    - Ejemplo: {"React", "React.js", "ReactJS"} → `http://data.europa.eu/esco/skill/abc123`

2. **Enriquecimiento semántico**: Asociar a cada skill sus etiquetas preferidas bilingües, descripciones, y relaciones jerárquicas
    - Permite análisis en español e inglés sin duplicar datos
    - Habilita exploración de jerarquías ("JavaScript" es-hijo-de "Programming Languages")
3. **Identificación de skills emergentes**: Detectar habilidades sin match en ESCO como señal de tecnologías nuevas
    - Ejemplo: "ChatGPT", "Tailwind CSS", "Terraform" no presentes en ESCO v1.1.0 (publicada 2022)

Taxonomía ESCO extendida:
El sistema opera sobre una versión extendida de ESCO v1.1.0 que incorpora:

- **ESCO v1.1.0**: 13,939 habilidades oficiales de la Comisión Europea (98.1 % del total)
- **O*NET Skills**: 152 habilidades técnicas del U.S. Department of Labor no cubiertas por ESCO (1.1 %)
- **Habilidades agregadas manualmente**: 124 tecnologías modernas identificadas en análisis exploratorio (0.9 %)
    - Categorías: Frameworks modernos (Next.js, Remix), herramientas DevOps (Terraform, ArgoCD), AI/ML (LangChain, Prompt Engineering)
- **Total**: 14,215 habilidades en taxonomía extendida

Desafíos de la implementación:
La implementación del matcher ESCO enfrentó dos desafíos técnicos principales que se documentan en las secciones siguientes:

1. **Fuzzy string matching**: Balancear similitud ortográfica vs. falsos positivos ("Piano" mapeando a "tocar el piano")
2. **Embeddings semánticos inadecuados**: Modelos generalistas producen matches incorrectos en vocabulario técnico ("Docker" $\rightarrow$ "Facebook")

El sistema final opera con arquitectura de tres capas secuenciales (exact match, fuzzy match, semantic match), con Layer 3 deshabilitada post-evaluación debido a limitaciones identificadas en embeddings multilingües generalistas para dominio técnico.

7.4.1 Arquitectura ESCOMatcher3Layers

El sistema se diseñó como matcher de tres capas secuenciales con fallback en cascada: Layer 1 ejecuta matching exacto contra labels preferidos bilingües; Layer 2 aplica fuzzy string matching con umbral 0.92; y Layer 3 utiliza embeddings semánticos (posteriormente deshabilitado). Cada capa opera independientemente, retornando el match de la primera que genera resultado, priorizando precisión sobre recall.

La taxonomía se cargó desde `esco_skills` con campos: `skill_uri`, `preferred_label_en/es`, `alternative_labels_en/es`, `skill_type`, y `description`. Se construyeron tres índices in-memory: (1) *Exact index* como diccionario mapeando normalized labels a URIs (42,000 entradas); (2) *Fuzzy index* como lista ordenada de tuplas (label, URI); y (3) *Semantic index* como matriz numpy 14,215×1024 de embeddings pre-computados. Los índices se cargaron al inicio, reduciendo latencia de 800ms/skill a 15ms/skill.

7.4.2 Layer 1 y 2: Matching Exacto y Fuzzy

Layer 1 implementó matching exacto case-insensitive contra labels bilingües. La normalización unificada aplicó: lowercase, eliminación de acentos (`unicodedata.normalize`), eliminación de puntuación preservando guiones/puntos internos (“Node.js”), y colapso de espacios. El lookup directo en `exact_index` alcanzó 35-40 % de cobertura en Pipeline A y 40-45 % en Pipeline B, reflejando que LLMs generan ortografía más estandarizada.

Layer 2 aplicó fuzzy matching usando `fuzzywuzzy` con distancia de Levenshtein. La implementación inicial con `fuzz.partial_ratio()` produjo falsos positivos críticos: “Piano” mapeó a “tocar el piano” (100 %, substring exacto), “SQL” a “MySQL” (100 %). Se reemplazó por `fuzz.ratio()` (similitud entre strings completos), reduciendo “Piano” vs “tocar el piano” a 40 % y “SQL” vs “MySQL” a 60 %. El umbral se configuró empíricamente en 0.92 tras evaluar 200 matches manuales: 0.85 generaba falsos positivos (“Java” $\rightarrow$ “JavaScript”), mientras 0.95 requería ortografía perfecta eliminando abreviaciones válidas (“K8s” vs “Kubernetes” = 0.93).

La optimización implementó early stopping: al encontrar match con score $\geq$ 0.98, se detuvo la búsqueda. Esto redujo tiempo de 450ms/skill (búsqueda exhaustiva) a 85ms/skill (early stopping en 18 % casos). Layer 2 incrementó cobertura en 25-30 % adicional, mapeando variantes ortográficas, abreviaciones expandidas, y nombres con guiones inconsistentes. Sin embargo, abreviaciones extremas fallaron: “AWS” vs “Amazon Web Services” score 0.42, “GCP” vs “Google Cloud Platform” 0.35, “ML” vs “Machine Learning” 0.40.

7.4.3 Layer 3: Embeddings Semánticos (Deshabilitado)

Layer 3 implementó matching semántico con el modelo `paraphrase-multilingual-mpnet-base-v2` transformando skills a vectores de 768 dimensiones. Se pre-computaron embeddings para 14,215 labels ESCO, normalizados a vectores unitarios. Para cada skill sin match en Layers 1-2, se calculó simi-

litud coseno contra matriz ESCO vía producto punto, retornando match si similitud >0.75 (120ms/skill).

Las pruebas revelaron que embeddings multilingües generalistas producían matches incorrectos en contexto técnico: "Docker" mapeó a "Facebook" (similitud 0.82), "REST" a "sleep" (0.79), "Python" a "snake programming" (0.76). El análisis determinó que modelos pre-entrenados en corpus generales (Wikipedia, CommonCrawl) capturan asociaciones semánticas de dominio general pero no técnicas especializadas. Corregir esto requeriría fine-tuning en corpus tech-específico (Stack Overflow, GitHub) con 50,000+ ejemplos anotados, excediendo scope del proyecto. **Layer 3 se deshabilitó completamente**.

El sistema final operó con Layers 1-2 (exact + fuzzy), alcanzando match rate de 12.6 % sobre el gold standard de 300 ofertas (1,038 de 8,268 skills extraídas). Se experimentó con un **ESCO Matcher Enhanced** que incorporaba matching más agresivo con `partial_ratio` y reglas adicionales, logrando aumentar cobertura a 25 %. Sin embargo, análisis cualitativo reveló incremento en falsos positivos (e.g., "Europa" → "neuropatología", "Oferta" → "ofertas de empleo"), introduciendo sesgo no deseado. Se decidió no implementar esta versión enhanced, preservando el matcher conservador de 2 capas que prioriza precisión sobre recall. El match rate de 12.6 % refleja la naturaleza del mercado tech LATAM: aunque ESCO v1.1.0 incluye actualizaciones hasta 2023, la taxonomía europea no cubre completamente frameworks modernos emergentes (Next.js, Tailwind CSS, shadcn/ui) ni herramientas específicas de ecosistemas recientes (Vite, Bun, Astro), que representan 87.4 % de skills extraídas como emergentes sin mapeo ESCO. Estas skills emergentes se preservan en formato normalizado para análisis de tecnologías no estandarizadas y clustering Pre-ESCO.

El mapper se integró como etapa 5 del orquestador, procesando skills desde `extracted_skills`, `enhanced_skills`, y `gold_standard_annotations`. El procesamiento batch de 15,000 skills únicas tomó 6.5 minutos (26ms/skill) aprovechando memoización: caché `skill_text` → `esco_uri` evitó remapear skills repetidas. Skills populares ("JavaScript" en 5,000+ ofertas) se mapearon una vez, reduciendo tiempo de 6.5h (sin caché) a 6.5min (60× aceleración).

7.5 Implementación del Sistema de Clustering de Habilidades

El sistema de clustering de habilidades constituye un componente analítico central del observatorio, diseñado para descubrir familias semánticas de skills sin categorías predefinidas, analizar evolución temporal de perfiles tecnológicos, y detectar tecnologías emergentes mediante análisis no supervisado. Este sistema permite caracterizar la demanda laboral más allá de conteos agregados de skills individuales, revelando combinaciones coherentes de habilidades que definen roles profesionales reales en el mercado.

La arquitectura del clustering integra tres componentes complementarios que transforman texto de skills en agrupaciones semánticas interpretables: (1) **embeddings semánticos** que capturan similitud entre skills en espacio vectorial de 768 dimensiones, (2) **reducción dimensional** mediante UMAP

que proyecta vectores de alta dimensión a espacio 2D preservando estructura local y global, y (3) **clustering density-based** con HDBSCAN que identifica automáticamente agrupaciones densas sin especificar número de clusters a priori.

El sistema se ejecutó en dos escenarios complementarios: **Pre-ESCO** que analiza texto normalizado de skills tal como fueron extraídas (preservando tecnologías emergentes sin mapeo ESCO), y **Post-ESCO** que opera sobre URIs estandarizados de taxonomía ESCO (consolidando variantes ortográficas para mayor coherencia). Esta dualidad permite balancear cobertura de tecnologías emergentes (Pre-ESCO) con interpretabilidad de resultados (Post-ESCO).

7.5.1 Justificación del Enfoque de Clustering No Supervisado

La decisión de implementar clustering no supervisado en lugar de categorización supervisada se fundamentó en las características dinámicas del mercado laboral tecnológico y las limitaciones de taxonomías predefinidas:

Limitaciones de enfoques supervisados que clustering no supervisado resuelve:

- **Obsolescencia de categorías predefinidas**: Taxonomías tradicionales (O*NET SOC codes) actualizadas cada 5-10 años no capturan roles emergentes (“AI/ML Engineer”, “DevOps Engineer”)
- **Rigidez de jerarquías estáticas**: Categorías fijas no reflejan solapamiento natural de perfiles (“Full-Stack Developer” combina Backend + Frontend)
- **Costo de supervisión manual**: Anotar 30,660 ofertas en categorías requiere 400-500 horas de trabajo especializado
- **Sesgo de anotadores**: Categorización manual depende de interpretación subjetiva de roles laborales

Ventajas del enfoque no supervisado para análisis de demanda laboral:

- **Descubrimiento automático de patrones**: Los datos revelan naturalmente agrupaciones sin hipótesis a priori sobre roles existentes
- **Adaptación a evolución temporal**: Nuevos clusters emergen automáticamente al procesar datos recientes (“Modern Frontend” post-2022)
- **Granularidad adaptativa**: HDBSCAN permite clusters de tamaños variables, capturando roles mainstream y nichos especializados
- **Identificación de outliers**: Skills atípicas o errores de extracción se detectan automáticamente como ruido

- **Escalabilidad**: No requiere re-entrenamiento supervisado al agregar nuevas ofertas al corpus

Justificación de componentes tecnológicos seleccionados:

- **E5 Multilingual Embeddings**: Seleccionado por soporte bilingüe español/inglés (crítico para corpus LATAM), performance en benchmarks de similitud semántica (STS tasks), y tamaño intermedio (278M parámetros) que balancea calidad vs. costo computacional
- **UMAP sobre t-SNE/PCA**: UMAP preserva estructura local (skills similares cercanas) y global (dominios separados) simultáneamente, con complejidad O(n log n) vs. O(n^2) de t-SNE, y proyecciones deterministas reproducibles
- **HDBSCAN sobre K-Means**: HDBSCAN detecta automáticamente número óptimo de clusters sin hiperparámetro k, identifica outliers como ruido en lugar de forzar asignación, y maneja clusters de formas arbitrarias (no asume esfericidad)

Esta arquitectura responde a los siguientes atributos de calidad del sistema:

- **Adaptabilidad**: Clustering no supervisado evoluciona con mercado laboral sin requerir actualización manual de categorías
- **Interpretabilidad**: UMAP 2D permite visualización intuitiva de 768 dimensiones, facilitando inspección manual de coherencia
- **Escalabilidad**: Complejidad O(n log n) permite procesar corpus completo (30,660 ofertas) en $<$5 minutos
- **Reproducibilidad**: Proyecciones deterministas con `random_state` garantizan resultados consistentes entre ejecuciones

7.5.2 Generación de Embeddings y Reducción UMAP

El modelo `intfloat/multilingual-e5-base` transformó skills a vectores de 768 dimensiones capturando similitud semántica. Se seleccionó por: (1) soporte multilingüe (español/inglés), (2) tamaño intermedio (278M params) balanceando expresividad vs costo, y (3) performance en STS benchmarks. El proceso operó en dos modos: Pre-ESCO embebió texto normalizado de 6,413 skills extraídas, aplicando filtro de frecuencia mínima que redujo el corpus a 1,314 embeddings con suficiente densidad para clustering; Post-ESCO embebió preferred labels ESCO resultando en 289 embeddings consolidados para gold standard de 300 ofertas. Se aplicó prefixing "query:" según especificaciones E5. La generación batch procesó skills en lotes de 256 documentos en hardware consumer (MacBook Air M4 base con 16GB RAM unificada). Los embeddings se normalizaron a vectores unitarios y almacenaron en base de datos PostgreSQL con extensión pgvector para consultas de similitud eficientes.

UMAP (Uniform Manifold Approximation and Projection) redujo embeddings de 768D a 2D para visualización y clustering. Se seleccionó sobre t-SNE/PCA por: (1) preservación de estructura local y global, (2) escalabilidad $O(n \log n)$, y (3) reproducibilidad determinista. La configuración involucró: `n_neighbors` (balance local/global), `min_dist=0.1` (separación mínima), `n_components=2`, y `metric=''cosine''`. El grid search sobre `n_neighbors` $\in \{5, 10, 12, 15, 20, 30\}$ determinó que **`n_neighbors=15`** ofrecía mejor balance: preservó agrupaciones semánticas coherentes (React/Vue/Angular separados pero cercanos) mientras mantuvo separación entre dominios mayores (Frontend/Backend/DevOps no-sobrelapados). La proyección UMAP de 1,314 skills tomó aproximadamente 5 segundos en CPU (Apple Silicon M4).

7.5.3 Clustering HDBSCAN y Optimización

HDBSCAN (Hierarchical Density-Based Spatial Clustering) identificó clusters sobre proyecciones UMAP 2D sin especificar número predefinido. Se seleccionó sobre K-Means por: (1) detección automática de número de clusters, (2) identificación de outliers como ruido, (3) clusters de forma arbitraria, y (4) jerarquía accesible mediante dendrogramas. La configuración involucró: `min_cluster_size` (granularidad), `min_samples` (robustness), y `metric=''euclidean''`.

El grid search sobre `min_cluster_size` $\in \{5, 7, 10, 12, 15, 20\}$ evaluó: (1) número de clusters (ideal 50-200), (2) porcentaje de ruido ($<$25 %), (3) Silhouette Score ($>$0.4 para datos Post-ESCO), y (4) interpretabilidad manual. Los experimentos revelaron trade-off: configuraciones bajas (5-7) generaban 100+ clusters muy específicos con alta fragmentación y Silhouette 0.35-0.40; configuraciones altas (15-20) producían pocos clusters gruesos (10-20) con Silhouette 0.55-0.65 pero pérdida de granularidad útil.

El análisis comparativo identificó **UMAP n_neighbors=15 + HDBSCAN min_cluster_size=12** como configuración óptima balanceando granularidad vs coherencia. Esta configuración operó sobre el gold standard de 300 ofertas, generando clustering diferenciado por pipeline: Pipeline A 300 Post-ESCO produjo 7 clusters sobre 289 skills únicas consolidadas (16.3 % ruido, Silhouette=0.398), Pipeline B 300 Post-ESCO generó 50 clusters sobre 1,618 skills (16.5 % ruido, Silhouette=0.348). Adicionalmente, el corpus completo de Pipeline A (30k ofertas) generó 53 clusters sobre 1,698 skills únicas ESCO (22.3 % ruido, Silhouette=0.456), demostrando escalabilidad del sistema a corpus de producción. La inspección manual confirmó coherencia semántica en top clusters: JavaScript/React ecosystem, Python/Data Science, Project Management, Cloud/DevOps (AWS/GCP), SQL/Databases.

7.5.4 Comparación Pre-ESCO vs Post-ESCO

El clustering se ejecutó en dos escenarios evaluando impacto del mapeo ESCO. **Pre-ESCO** operó sobre texto normalizado de skills sin consolidación: Pipeline A 300 generó 38 clusters sobre 1,314 skills (Silhouette=0.447, 25.7 % ruido), Pipeline B 300 produjo 34 clusters sobre 1,540 skills (Silhouette=0.234, 12.8 % ruido). La alta fragmentación se debe a variantes ortográficas formando micro-

clusters separados: “docker”, “Docker”, “docker-compose” aparecen como puntos distintos en el espacio de embeddings, diluyendo densidad de clusters. El beneficio de Pre-ESCO es que captura skills emergentes sin mapeo ESCO (“ChatGPT”, “Tailwind CSS”, “Bun”) preservándolas en el análisis.

Post-ESCO procesó URIs ESCO consolidados: Pipeline A 300 generó 7 clusters sobre 289 skills (Silhouette=0.398, 16.3 % ruido), Pipeline B 300 produjo 50 clusters sobre 1,618 skills (Silhouette=0.348, 16.5 % ruido). La consolidación colapsa variantes ortográficas en puntos únicos fortaleciendo densidad de clusters, pero la baja cobertura ESCO (12.6 % Pipeline A, 25 % Pipeline B) significa pérdida significativa de información: 87.4 % de skills emergentes desaparecen del análisis Post-ESCO. Los top clusters mostraron composición interpretable: JavaScript ecosystem, Python/Data Science, Cloud/DevOps, SQL/Databases. Se implementó análisis híbrido: clustering Post-ESCO para métricas cuantitativas sobre skills estandarizadas, complementado con análisis Pre-ESCO para tecnologías emergentes ausentes en taxonomía ESCO.

7.5.5 Análisis Temporal

Como extensión, se implementó módulo de análisis temporal para rastrear evolución de clusters sobre 21,839 ofertas fechadas (71.23 % del dataset), abarcando 29 trimestres desde Q4-2018 hasta Q4-2025. Sin embargo, la distribución temporal presenta alta concentración: 97.1 % de ofertas (21,216) corresponden a Q4-2025, reflejando el período intensivo de scraping reciente. Los trimestres anteriores (Q4-2018 a Q3-2025) contienen solo 623 ofertas dispersas, limitando análisis longitudinal robusto.

El módulo implementado permite análisis temporal mediante: (1) agrupación de ofertas por trimestre, (2) extracción de skills por período, (3) generación de embeddings E5, (4) proyección UMAP (`n_neighbors=15`), (5) clustering HDBSCAN (`min_cluster_size=12`), y (6) tracking de consistencia de clusters entre períodos consecutivos (threshold: $\geq$60 % overlap en top-20 skills). El sistema genera visualizaciones temporales (heatmaps clusters × quarters, line charts de frecuencia) que permitirían identificar patrones de adopción tecnológica cuando se disponga de datos distribuidos temporalmente. La infraestructura está lista para análisis longitudinal futuro conforme el observatorio acumule ofertas distribuidas equitativamente a través de trimestres.

7.5.6 Experimentación de Hiperparámetros y Trade-off Interpretabilidad vs. Métricas

La optimización de UMAP+HDBSCAN requirió balancear métricas cuantitativas de calidad de clustering (Silhouette Score, Davies-Bouldin Index) con interpretabilidad práctica de los clusters resultantes para análisis del mercado laboral. Este balance no es trivial: configuraciones que maximizan métricas matemáticas frecuentemente producen clusterings inútiles para análisis humano, revelando una tensión fundamental entre optimización algorítmica y utilidad práctica.

Se realizaron 70+ experimentos documentados variando hiperparámetros UMAP (`n_neighbors` $\in \{5, 10, 12, 15, 20, 30\}$, `min_dist` $\in \{0.05, 0.08, 0.1, 0.2\}$) y HDBSCAN (`min_cluster_size` $\in \{2, 3, 4, 5, 8, 10, 12, 15, 20\}$, `min_samples` $\in \{1, 2, 3, 4, 5\}$). Los experimentos documentaron

el fenómeno del "clustering cliff": configuraciones con `min_cluster_size` ≤ 6 generaron 100-300 clusters con métricas excelentes (Silhouette > 0.6, Davies-Bouldin < 0.5) pero imposibles de interpretar manualmente; configuraciones con `min_cluster_size` ≥ 15 colapsaron a 2-10 clusters genéricos con baja utilidad analítica. El documento de pruebas (Capítulo 13) detalla la evaluación exhaustiva de configuraciones.

Caso ilustrativo del problema (Experimento 8 vs. Experimento 15):

El experimento 8 con hiperparámetros finos (`n_neighbors=5`, `min_cluster_size=3`) produjo 305 clusters con Silhouette Score = 0.618 (excelente según literatura), Davies-Bouldin = 0.439 (óptimo), y 16.2 % ruido. Sin embargo, la inspección manual reveló que los 305 clusters eran ininterpretables: "Python+Flask" formó un cluster separado de "Python+Django", "JavaScript+React" separado de "JavaScript+Vue", fragmentando artificialmente tecnologías relacionadas. Nombrar, categorizar y analizar 305 clusters excede la capacidad de procesamiento humano.

En contraste, el experimento 15 con hiperparámetros medios (`n_neighbors=15`, `min_cluster_size=12`) generó 50 clusters con Silhouette Score = 0.348 (inferior al experimento 8), Davies-Bouldin = 0.687 (mayor pero aceptable), y 16.5 % ruido, pero 98 % de clusters (49/50) semánticamente coherentes e interpretables. Los clusters agruparon familias tecnológicas completas: "Backend Python" (Flask, Django, FastAPI, Celery), "Frontend JavaScript" (React, Vue, Angular, TypeScript), "DevOps" (Docker, Kubernetes, Jenkins, GitLab CI). Esta granularidad permitió análisis sistemático de 50 perfiles vs. imposibilidad de manejar 305.

Sistema de scoring cuantitativo para balancear criterios:

Se implementó función de scoring multi-criterio ponderando: granularidad (40 % del score, penalizando <30 o >200 clusters), Silhouette Score (30 %, recompensando >0.3), porcentaje de ruido (20 %, penalizando >25 %), e interpretabilidad manual (10 %, evaluada mediante inspección de top-10 clusters por coherencia temática). Este sistema formalizó la decisión de **priorizar utilidad práctica sobre optimización matemática**, justificando académicamente la selección de configuraciones con métricas numéricas moderadas pero alta interpretabilidad.

La configuración óptima seleccionada (`n_neighbors=15`, `min_cluster_size=12`, `min_samples=3`) representa el punto de equilibrio: genera 50-60 clusters interpretables por humanos, mantiene Silhouette > 0.35 (aceptable), y limita ruido a 15-20 %. Esta decisión es consistente con investigaciones previas en clustering de dominios especializados, donde interpretabilidad del resultado es tan crítica como calidad métrica del agrupamiento. Los resultados cuantitativos de las configuraciones de clustering ejecutadas se presentan en el Capítulo 7 (Resultados).

7.5.7 Beneficios del Sistema de Clustering Implementado

La implementación del sistema de clustering no supervisado mediante UMAP + HDBSCAN proporciona múltiples beneficios analíticos, operativos y metodológicos para el observatorio de demanda laboral, respondiendo directamente a los objetivos centrales del proyecto.

Capacidades analíticas fundamentales. El sistema habilita descubrimiento automático de perfiles tecnológicos emergentes sin requerir categorías predefinidas, permitiendo que los datos revelen naturalmente nuevas combinaciones de habilidades demandadas por el mercado. La granularidad adaptativa de HDBSCAN captura tanto familias tecnológicas amplias (JavaScript/React ecosystem con 40-60 skills relacionadas, Python/Data Science con frameworks especializados) como especializaciones de nicho (DevOps tools específicas, librerías de ML), produciendo clústeres de tamaños variables (5-70 skills) según densidad semántica real. La detección automática de outliers identifica skills atípicas o errores de extracción como ruido (12-25 % del dataset según configuración Pre/Post-ESCO), actuando como mecanismo de filtrado de calidad sin supervisión manual. Adicionalmente, los dendrogramas de HDBSCAN revelan jerarquías multinivel de perfiles, permitiendo análisis tanto a nivel macro (Backend vs Frontend vs DevOps) como granular (Backend Java vs Backend Node.js, distinguiendo ecosistemas tecnológicos específicos).

Ventajas para visualización e interpretabilidad. UMAP preserva tanto relaciones locales (React posicionado cerca de Vue/Angular por similitud de propósito) como estructura global (Frontend separado espacialmente de DevOps/Infrastructure), permitiendo que visualizaciones 2D capturen fielmente la topología semántica de las 768 dimensiones originales de los embeddings. Esta reducción dimensional mantiene interpretabilidad intuitiva: clusters visualmente cohesivos corresponden a familias tecnológicas coherentes verificables mediante inspección manual. Las proyecciones son completamente reproducibles mediante fijación de `random_state`, garantizando que visualizaciones y análisis sean consistentes entre ejecuciones y comparables a través del tiempo, propiedad crítica para documentación científica y auditoría de resultados.

Infraestructura para análisis temporal (limitaciones actuales). El sistema implementa módulo de tracking temporal capaz de rastrear evolución de clústeres entre períodos, generando heatmaps de frecuencia por cluster×quarter y gráficos de evolución de demanda. Sin embargo, la aplicabilidad actual está limitada por la distribución temporal del corpus: 93.5 % de menciones (4,222/4,479) se concentran en Q4-2025, con solo 5 quarters representados (2016Q2, 2023Q4, 2025Q1, 2025Q3, 2025Q4) y frecuencias insuficientes en períodos históricos (20-151 menciones vs 4,222 en Q4-2025). Esta concentración impide análisis longitudinal robusto de tendencias tecnológicas, obsolescencia de frameworks, o crecimiento de nuevas tecnologías. La infraestructura está preparada para análisis temporal riguroso cuando el observatorio acumule datos distribuidos equitativamente a través de múltiples años mediante scraping continuo.

Eficiencia operacional y escalabilidad. La complejidad algorítmica O(n log n) de UMAP permite procesar corpus de producción (30,660 ofertas generando 1,314 skills con frecuencia mínima) en aproximadamente 5 segundos de proyección UMAP más tiempo adicional de clustering HDBSCAN, totalizando menos de 30 segundos para pipeline completo en CPU consumer (Apple Silicon M4). El sistema no requiere supervisión humana para generación de clusters, eliminando el costo de anotación manual que requeriría 400-500 horas de trabajo especializado para categorizar 30,660 ofertas. La arquitectura soporta actualización incremental: agregar nuevas ofertas al corpus solo requiere re-

ejecutar el pipeline de clustering sobre el dataset expandido, sin necesidad de reentrenar modelos o ajustar categorías manualmente.

Alineación con objetivos del observatorio. El sistema de clustering responde directamente a los tres objetivos centrales del proyecto. Primero, caracteriza automáticamente la demanda laboral tecnológica mediante identificación de 34-53 familias semánticas de skills (según pipeline y configuración), proporcionando taxonomía emergente de perfiles demandados en LATAM que refleja la estructura real del mercado sin imposición de categorías preconcebidas. Segundo, analiza composición de perfiles tecnológicos al agrupar ecosistemas coherentes (JavaScript/React frontend, Python/Data Science analytics, Cloud/DevOps infrastructure, SQL/Databases backend), facilitando comprensión de qué combinaciones de habilidades se demandan conjuntamente. Tercero, detecta skills emergentes no presentes en taxonomías oficiales: 87.4 % de skills extraídas por Pipeline A no mapean a ESCO v1.1.0, capturándose en clusters Pre-ESCO frameworks modernos (Next.js, Tailwind CSS, Bun, Deno) y herramientas recientes (Terraform, Kubernetes, GitOps), señalando tecnologías que están ganando tracción en el mercado pero aún no están formalizadas en estándares europeos.

Esta arquitectura de clustering no supervisado permite que el observatorio evolucione orgánicamente con el mercado laboral, sin requerir actualización manual de categorías predefinidas que rápidamente quedarían obsoletas en el dinámico sector tecnológico latinoamericano.

7.6 Creación del Gold Standard y Sistema de Evaluación

Esta sección describe la construcción del dataset de referencia de 300 ofertas anotadas manualmente y el sistema de evaluación dual (Pre-ESCO y Post-ESCO) para comparar pipelines.

7.6.1 Selección y Anotación del Gold Standard

El gold standard requirió seleccionar un subset representativo del corpus de 30,660 ofertas balanceando diversidad tecnológica, distribución geográfica y viabilidad de anotación. La construcción del dataset de 300 ofertas involucró un proceso iterativo de 7 rondas de selección y refinamiento que garantizó calidad, diversidad y ausencia de duplicados.

Algoritmo de selección estratificada. El script `select_gold_standard_jobs.py` implementó un algoritmo de prioridad jerárquica operando en 4 fases secuenciales: (1) *Detección de idioma* mediante regex patterns sobre el corpus completo (español, inglés, mixto), clasificando 56,555 ofertas; (2) *Pre-selección con filtros SQL estrictos* aplicando restricciones de longitud mínima (1,200+ caracteres), inclusión de títulos técnicos (“developer”, “engineer”, “programador”) y exclusión explícita de roles no-software (“manager”, “mechanical engineer”, “chemical engineer”, “cajero”, “manufactura”), resultando en 7,102 candidatos tras deduplicación por `content_hash`; (3) *Scoring y clasificación* calculando quality score (0-100) basado en longitud (20 pts), presencia de keywords técnicas (10 pts), sección de requisitos (10 pts) y penalización por ruido HTML (-10 pts), además de clasificación automatizada de rol (8 categorías) y seniority (junior/mid/senior) mediante análisis de título y

descripción; (4) *Selección estratificada* con targets por país×idioma (100 CO ES, 100 MX ES, 50 AR ES, 17 CO EN, 17 MX EN, 16 AR EN) y distribución aproximada por rol, ordenando candidatos por quality score descendente dentro de cada celda.

Proceso iterativo de refinamiento (7 iteraciones). La selección inicial (Iteración 3, tras 2 rondas previas fallidas) produjo 300 ofertas que subsecuentes rondas de revisión manual y reemplazo automatizado refinaron: *Iteración 4 (3 sub-rondas)* removió 47 ofertas problemáticas identificadas mediante heurísticas automatizadas—15 duplicados con títulos genéricos repetidos ("Desarrollador Fullstack / Certificados CEO" × 11 instancias), 11 roles manufacturing/hardware ("programador de corte y doble", "ingeniero de moldes"), 7 petroleum/oil&gas engineering ("geociencias", "perforación"), 6 sales engineering ("pre-sales", "ventas O&G"), 3 roles business/ERP ("JDE Developer", "logistics engineering"), 2 R&D manufacturing, 2 postsales support, 1 descripción corrupta—reemplazando cada uno con candidatos que pasaron filtros ultra-estrictos verificando software-only keywords y exclusión de patrones problemáticos; *Iteración 5 (4 sub-rondas)* detectó durante calibración de anotación (primeros 15 jobs) que 8/15 eran ingenieros no-software, ejecutando 4 ciclos de detección y reemplazo que removieron 29 jobs adicionales (19 non-software engineering: químico/eléctrico/mecánico/civil/-corrosión/CATIA/procesos; 8 business/operations: ejecutivo comercial, coordinador de proveedores, representante comercial; 2 business development: "Desarrollador de Negocios Postventa", "Analista Planeación de demanda"), introduciendo filtros cada vez más estrictos hasta eliminar completamente la categoría "Other"; *Iteración 6 (2 sub-rondas)* identificó durante generación de batches de revisión que 48/300 jobs (16 %) tenían títulos duplicados (22 títulos diferentes con 2-12 instancias cada uno, siendo los más frecuentes "ingeniero sistemas Junior / carreras afines - remoto" × 12 y "ingeniero software implementacion Pegasus" × 11), removiendo duplicados manteniendo la instancia más larga por título y reemplazando con verificación de unicidad, pero introduciendo 3 nuevos non-software ("Enterprise Software Account Executive", "JDE Developer", "PROGRAMADOR CORTE Y DOBLE") que sub-iteración 6b corrigió; *Iteración 7* completó limpieza final durante anotación manual exhaustiva, detectando 5 duplicados adicionales por Job ID idéntico (4 casos) y similitud semántica (1 caso con 79.8 % overlap vocabulario), reemplazándolos inmediatamente y anotándolos.

Dataset final verificado. El resultado post-Iteración 7 consistió en 300 ofertas con garantías verificadas mediante queries SQL y parsing del archivo de anotaciones: 300 Job IDs únicos (0 duplicados por ID), 299 títulos únicos (1 título duplicado: "DevOps Engineer" × 2 con contenido diferente), 100 % roles pure software development, distribución geográfica cercana a targets (CO 40.7 %, MX 37.7 %, AR 21.7 %), distribución de idioma (ES 80.7 %, EN 19.3 %), distribución de roles (Backend 34.3 %, QA 14.7 %, Frontend 13.7 %, DevOps 12.3 %, Data Science 9.3 %, Mobile 7.0 %, Fullstack 4.7 %, Security 4.0 %), distribución de seniority (Senior 54.0 %, Mid 40.0 %, Junior 6.0 %), y longitud de contenido (promedio 527 palabras, mediana 489, rango 119-2,447).

Anotación manual. Un único anotador (estudiante de último año Ingeniería de Sistemas) identificó manualmente skills técnicas hard y soft siguiendo guidelines con formato atómico estricto. El protocolo requirió lectura completa de descripción y requisitos, listado de términos atómicos indi-

viduales sin paréntesis ni narrativas (“Python”, “Docker”, “Comunicación”), y clasificación en hard skills (lenguajes, frameworks, herramientas, metodologías, bases de datos) y soft skills (comunicación, liderazgo, trabajo en equipo, resolución de problemas). El formato atómico prohibió construcciones compuestas como “Python (pandas, numpy)” o narrativas como “Conocimientos de AWS y Azure”, requiriendo términos separados: “Python”, “Pandas”, “NumPy”, “AWS”, “Azure”. Esta decisión simplificó la comparación automatizada con skills extraídas por Pipeline A y Pipeline B. El proceso de anotación completó las 300 ofertas produciendo 7,848 skills totales: 6,174 hard skills (78.7 %) y 1,674 soft skills (21.3 %), con promedio de 26.2 skills por oferta. El Apéndice B presenta ejemplos completos de anotaciones (Job #1 “Developer Advocate” en inglés con 8 hard + 8 soft skills; Job #46 “IBM ACE Developer” en español con 25 hard + 3 soft skills) ilustrando el protocolo aplicado y la diversidad geográfica/idiomática del dataset. No se realizó medición de inter-annotator agreement al ser un único anotador; la validez del gold standard se garantizó mediante protocolo estricto, ejemplos de calibración y verificación post-anotación de consistencia de formato.

7.6.2 Sistema de Evaluación Dual: Pre-ESCO y Post-ESCO

El sistema implementó dos comparaciones independientes cuantificando capacidades complementarias: *Pre-ESCO* evalúa capacidad de extracción pura comparando texto normalizado sin mapeo taxonómico, capturando skills emergentes ausentes en ESCO; *Post-ESCO* evalúa capacidad de estandarización comparando URIs ESCO tras mapear todas las skills (gold standard y pipelines) con el mismo código `ESCOMatcher3Layers`, eliminando sesgos ortográficos.

El componente Pre-ESCO utilizó módulo de normalización canónica con diccionario de 200+ tecnologías mapeando variantes a formas estándar (“js”/“javascript” $\rightarrow$ “JavaScript”, “k8s” $\rightarrow$ “Kubernetes”). Las métricas se calcularon mediante operaciones de conjuntos: para cada job, se compararon skills gold normalizadas vs skills pipeline normalizadas, identificando True Positives (TP = intersección), False Positives (FP = predichas no en gold), y False Negatives (FN = en gold no predichas). Los valores agregados sobre 300 ofertas alimentaron fórmulas: Precision = TP/(TP+FP), Recall = TP/(TP+FN), F1 = 2×(P×R)/(P+R).

El componente Post-ESCO remapeó todas las skills usando `ESCOMatcher3Layers` para garantizar fairness: Pipeline A se ignoró y remapeó desde texto normalizado igual que Pipeline B. Este diseño eliminó ventajas artificiales asegurando que diferencias Post-ESCO reflejaran calidad de extracción textual. Skills sin match ESCO se descartaron de la comparación Post-ESCO, cuantificándose separadamente como “Skills Emergentes” para análisis cualitativo.

Las 300 ofertas se procesaron por todos los pipelines: Pipeline A (NER+Regex completo), Pipeline A Regex-Only, Pipeline B con 4 LLMs (Gemma, Llama, Qwen, Phi), y Pipeline A.1 (TF-IDF, descartado por F1$<$12 %). Los outputs se almacenaron en tablas dedicadas facilitando queries de evaluación mediante joins con `gold_standard_annotations`.

RESULTADOS

8.1 Evaluación Comparativa de Pipelines de Extracción

Esta sección presenta los resultados cuantitativos de la evaluación de los cuatro pipelines principales sobre el gold standard de 300 ofertas anotadas. Las métricas documentan performance en dos escenarios (Pre-ESCO y Post-ESCO), identifican el pipeline ganador, y cuantifican el impacto del mapeo ESCO en precisión y cobertura de cada aproximación metodológica.

8.1.1 Evaluación Pre-ESCO: Capacidad de Extracción Pura

La Tabla 8.1 muestra métricas de extracción sobre texto normalizado sin mapeo taxonómico, capturando capacidad de identificar skills en su forma original incluyendo emergentes no estandarizadas.

Tabla 8.1: Evaluación Pre-ESCO de Pipelines (Hard Skills, 300 jobs)

Pipeline	Precision	Recall	F1-Score	Skills Avg/Job
Pipeline A.1 (TF-IDF)	0.1247	0.1098	0.1169	50.3
Pipeline A (Regex Only)	0.3392	0.1231	0.1807	22.8
Pipeline A (NER+Regex)	0.2254	0.2800	0.2498	50.3
Pipeline B (Gemma)	**0.4852**	**0.4415**	**0.4623**	**27.8**
Pipeline B (Llama)	0.3684	0.4352	0.3987	28.7
Pipeline B (Qwen)	0.5208	0.3125	0.3906	12.4
Pipeline B (Phi)	0.4123	0.3017	0.3482	15.8

Pipeline A.1 (TF-IDF) exhibió performance inadecuado con F1=11.69 %, confirmando que aproximaciones puramente estadísticas fallan en dominio técnico donde términos relevantes (“Docker”, “Python”) tienen distribución TF-IDF similar a buzzwords (“innovación”, “excelencia”). Pipeline A Regex-Only alcanzó F1=18.07 % con precisión moderada (33.92 %) y recall muy limitado (12.31 %), evidenciando que 247 patrones manuales capturan skills con nomenclatura estándar pero omiten variantes contextuales y menciones no-literales. Pipeline A completo (NER+Regex) mejoró a F1=24.98 %: la adición de NER incrementó recall a 28.00 % detectando menciones contextuales, aunque precisión se redujo a 22.54 % por introducción de ruido.

Entre LLMs, Gemma 3 4B alcanzó mejor F1=46.23 % con balance Precision=48.52 %/Recall=44.15 %, generando outputs limpios sin alucinaciones. Llama 3.2 3B obtuvo F1=39.87 % penalizado por baja Precision (36.8 %) debido a alucinaciones sistemáticas de skills de Data Science en ofertas no relacionadas. Qwen 2.5 3B logró Precision superior (52.1 %) pero F1=39.06 % por Recall muy bajo (31.2 %),

reflejando conservadurismo excesivo. Phi-3.5 Mini mostró F1=34.82 % afectado por inconsistencias en formato JSON que causaron pérdida de skills extraídas durante parsing.

8.1.2 Evaluación Post-ESCO: Capacidad de Estandarización

La Tabla 8.2 presenta métricas tras mapear todas las skills a taxonomía ESCO, evaluando alineación con vocabulario controlado.

Tabla 8.2: Evaluación Post-ESCO de Pipelines (Hard Skills, 300 jobs)

Pipeline	Precision	Recall	F1-Score	ESCO Cov.	Δ F1
Pipeline A.1 (TF-IDF)	0.1156	0.1021	0.1085	6.8 %	-0.0084
Pipeline A (Regex Only)	0.8636	0.7308	0.7917	25.7 %	+0.6110
Pipeline A (NER+Regex)	0.6550	0.8125	0.7253	11.1 %	+0.4755
Pipeline B (Gemma)	**0.8925**	**0.7981**	**0.8426**	**11.3 %**	**+0.3803**
Pipeline B (Llama)	0.7234	0.6891	0.7058	82.4 %	+0.3071
Pipeline B (Qwen)	0.8945	0.6523	0.7545	91.3 %	+0.3639
Pipeline B (Phi)	0.7821	0.5934	0.6747	85.7 %	+0.3265

El mapeo ESCO transformó radicalmente el ranking: Pipeline B (Gemma) emergió como ganador con F1=84.26 %, incremento de +38.03pp respecto a Pre-ESCO (46.23 % → 84.26 %). Esta mejora dramática refleja que Gemma genera skills con ortografía estandarizada (“JavaScript”, “PostgreSQL”) que mapean eficientemente a ESCO, mientras que texto normalizado Pre-ESCO contiene variantes (“js”, “postgres”) que fragmentan matches. Pipeline A Regex-Only alcanzó F1=79.17 % con mejora masiva de +61.10pp (18.07 % → 79.17 %), beneficiándose de patrones que ya generan formas canónicas con alta cobertura ESCO (25.7 %). Pipeline A completo (NER+Regex) mejoró significativamente (+47.55pp: 24.98 % → 72.53 %) alcanzando F1=72.53 % con cobertura ESCO 11.1 %, aunque limitado por ruido HTML y fragmentación léxica que dificulta mapeo.

La columna Δ F1 cuantifica dependencia de cada pipeline en ESCO para performance: todos los pipelines muestran mejoras dramáticas con mapeo ESCO, indicando fuerte impacto de estandarización. Pipeline A Regex-Only lidera con +61.10pp (18.07 % → 79.17 %), seguido por NER+Regex con +47.55pp (24.98 % → 72.53 %), mientras Gemma incrementa +38.03pp (46.23 % → 84.26 %). Las mejoras masivas reflejan que matching ESCO normaliza variantes ortográficas dispersas en texto crudo, consolidando skills fragmentadas y eliminando ambigüedades. Sin embargo, cobertura ESCO es baja (11-26 %), indicando que mayoría de extracciones Pre-ESCO no mapean a taxonomía estándar.

8.1.3 Análisis del Pipeline Ganador y Trade-offs

Pipeline B (Gemma 3 4B) se identificó como solución óptima con F1=84.26 % Post-ESCO, balanceando Precision (89.25 %) y Recall (79.81 %). Las ventajas fueron múltiples: primero, outputs limpios sin ruido HTML/JS observado en Pipeline A; segundo, normalización implícita generando formas estándar que mapean eficientemente a ESCO; tercero, capacidad contextual detectando skills

implícitas ("experiencia en arquitectura de microservicios" → extrae "Microservices", "Architecture"); y cuarto, ausencia de alucinaciones versus Llama/Phi. Las limitaciones también fueron relevantes: primero, costo computacional 42.3s/oferta versus 0.97s Pipeline A (43× más lento); segundo, performance Pre-ESCO moderado (F1=46.23 %) sugiriendo dependencia en mapeo ESCO para alcanzar alto F1; y tercero, requiere GPU para inferencia (4GB VRAM mínimo con cuantización INT4).

Pipeline A (NER+Regex) ofreció alternativa viable para escenarios sin GPU con F1=72.53 % Post-ESCO, ejecutándose en CPU a 0.97s/oferta. Su fortaleza fue cobertura de skills emergentes Pre-ESCO capturando tecnologías no-ESCO ausentes en outputs LLM. Su debilidad principal fue baja cobertura ESCO: solo 11.1 % de skills extraídas mapearon a taxonomía (vs 11.3 % Gemma), indicando que 89 % permanecen sin estandarizar. La variante Regex-Only (F1=79.17 %, 0.32s/oferta) emergió como baseline competitivo ultrarrápido con mejor cobertura ESCO (25.7 %).

El trade-off crítico fue **Flexibilidad vs Estandarización**: Pre-ESCO favorece Pipeline A capturando 40+ skills emergentes ("ChatGPT", "Tailwind CSS", "Terraform") formando micro-clusters válidos en análisis temporal, mientras Post-ESCO favorece Gemma con 84 % F1 en vocabulario controlado. Para el observatorio de demanda laboral, se adoptó estrategia híbrida: Pipeline B (Gemma) para procesamiento primario y métricas estandarizadas, complementado con análisis manual de skills Gemma sin mapeo ESCO para detectar tecnologías emergentes ausentes en taxonomía.

8.2 Análisis del Mercado Laboral Tecnológico Latinoamericano

Esta sección presenta hallazgos del análisis sobre el corpus completo de 30,660 ofertas procesadas, caracterizando distribución de skills, identificando tecnologías emergentes, y documentando tendencias temporales del mercado tech latinoamericano durante 2018-2025.

8.2.1 Resultados de Configuraciones de Clustering

El sistema de clustering se ejecutó sobre 8 configuraciones de producción representando tres pipelines de extracción (Manual annotations, Pipeline A, Pipeline B), dos escenarios ESCO (PRE, POST), y dos escalas de dataset (300 jobs gold standard, 30,660 jobs corpus completo). Esta matriz experimental cuantificó el impacto del mapeo ESCO en estructura de clustering y validó escalabilidad del sistema a corpus completo.

Impacto del Mapeo ESCO en Estructura de Clusters

Las configuraciones PRE-ESCO vs. POST-ESCO revelaron tres patrones diferenciados según pipeline de extracción:

(1) Manual Annotations - Colapso severo: 61 clusters (1,914 skills) PRE-ESCO redujeron a 2 clusters (236 skills) POST-ESCO, representando pérdida del 87.7 % de diversidad léxica. Esta transformación drástica refleja que ESCO v1.1.0 (publicado 2019-2021) no captura 1,678 skills (87.7 %)

del vocabulario técnico actual del mercado laboral latinoamericano. Los 2 clusters POST resultantes son extremadamente genéricos, perdiendo granularidad crítica para análisis sectorial. Silhouette degradó de 0.456 a 0.418 (-8.3 %), aunque el ruido se redujo de 23.8 % a 1.7 % por falta de diversidad léxica. Esta configuración generó 2 meta-clusters PRE-ESCO (hard skills técnicos vs soft skills transversales) que colapsaron POST-ESCO.

(2) Pipeline A (NER+Regex) - Colapso masivo a escala: En dataset de 300 jobs, transformación fue moderada (38 clusters $\rightarrow$ 7 clusters, -78.0 % skills), manteniendo Silhouette 0.447 PRE vs 0.398 POST. Sin embargo, en corpus completo de 30,660 jobs, el impacto fue extremo: 2,044 clusters (98,829 skills) PRE-ESCO colapsaron a 53 clusters (1,698 skills) POST-ESCO, descartando 98.3 % de extracciones por falta de mapeo ESCO. Esta brecha evidencia que 97,131 skills extraídas por Pipeline A no tienen correspondencia en taxonomía europea. Paradójicamente, métricas mejoraron POST-ESCO a escala: Silhouette 0.361 (PRE) $\rightarrow$ 0.456 (POST), Davies-Bouldin 0.735 $\rightarrow$ 0.665, indicando que skills ESCO son altamente recurrentes formando clusters más densos. Pipeline A 30k PRE generó 2 meta-clusters; POST mantuvo esta estructura con 2 meta-clusters diferenciados.

(3) Pipeline B (LLM) - Comportamiento anómalo de expansión: Único pipeline donde POST tiene MÁS skills y clusters que PRE: 34 clusters (1,766 skills) PRE-ESCO expandieron a 50 clusters (1,937 skills) POST-ESCO (+47.1 % clusters, +9.7 % skills). Este comportamiento contra-intuitivo sugiere que Gemma 3 4B normaliza implícitamente extracciones a vocabulario compatible con ESCO durante inferencia, enriqueciendo con términos estándar. Silhouette mejoró de 0.234 (PRE) a 0.348 (POST) (+48.7 %), aunque inferior a Manual (0.456) y Pipeline A 300 PRE (0.447). Pipeline B 300 PRE identificó 3 meta-clusters que se mantuvieron POST-ESCO, con mejor Meta-Silhouette (0.267 POST vs baseline).

Trade-off Diversidad-Cohesión con Escala

La evaluación de Pipeline A en 300 jobs vs. 30,660 jobs cuantificó el trade-off diversidad-cohesión inherente a clustering de corpus grandes. Silhouette Score degradó de 0.447 (300 jobs) a 0.361 (30,660 jobs), reducción del 19.2 % atribuible a emergencia de long-tail de skills raras (aparecen 1-5 veces). El porcentaje de ruido incrementó de 25.2 % a 34.1 % (+8.9pp) reflejando que aproximadamente 33,711 skills del corpus completo son menciones únicas de tecnologías altamente especializadas o errores de extracción residuales. Sin embargo, Silhouette > 0.3 se mantiene en rango aceptable según literatura, y los 2,044 clusters detectados automáticamente revelan micro-especializaciones tecnológicas (frameworks nicho, herramientas regionales) invisibles en dataset reducido. El crecimiento fue exponencial: 75× más skills únicas (1,314 $\rightarrow$ 98,829) generando 54× más clusters (38 $\rightarrow$ 2,044), validando escalabilidad del sistema UMAP+HDBSCAN.

Comparación de Calidad entre Pipelines

En configuración 300 jobs PRE-ESCO, los tres pipelines exhibieron perfiles diferenciados:

- **Manual Annotations**: Mejor Silhouette (0.456), máxima granularidad (61 clusters), cobertura intermedia (1,914 skills). Gold standard de calidad semántica con 23.8 % ruido. Generó 2 meta-clusters diferenciando claramente hard/soft skills.
- **Pipeline A**: Silhouette competitivo (0.447, 98 % del manual), 38 clusters balanceados, 1,314 skills con alta precisión. Ruido 25.2 % ligeramente superior pero totalmente automatizado. No generó meta-clusters (todos 38 clusters quedaron UNCLUSTERED).
- **Pipeline B**: Menor Silhouette (0.234), 34 clusters con sobre-agrupación, 1,766 skills (cobertura similar a manual). Mejor filtrado de ruido (12.8 %), LLM normaliza agresivamente reduciendo variabilidad léxica. Generó 3 meta-clusters con Meta-Silhouette moderado.

Capacidades validadas: Las 8 configuraciones validaron tres capacidades críticas del sistema. En primer lugar, la escalabilidad mediante el procesamiento exitoso de 98,829 skills únicas en menos de 10 minutos con métricas aceptables (Silhouette 0.361, Davies-Bouldin 0.735). En segundo lugar, la adaptabilidad a través de la detección automática de 2-2,044 clusters según granularidad de datos sin intervención manual ni ajuste de hiperparámetros. En tercer lugar, la robustez evidenciada porque la estructura macro de 2 meta-clusters (hard skills técnicos vs. soft skills transversales) persiste consistentemente en escalas y pipelines donde meta-clustering fue exitoso (Manual, Pipeline A 30k, Pipeline B), confirmando que el sistema captura dicotomía fundamental del mercado laboral.

Clustering Experiments Comparison

Experiment	min_cluster_size	n_neighbors	Clusters	Noise %	Silhouette	Davies-Bouldin	Largest Cluster	Duration (s)
Baseline_mcs5	5	15	17	30.2%	0.409	0.610	81	6.2
Test_mcs10	10	15	2	1.8%	0.681	0.430	264	1.3
Test_mcs15	15	15	2	0.0%	0.668	0.447	266	1.4
Test_mcs20	20	15	2	0.0%	0.668	0.449	265	1.3

Figura 8.1: Tabla comparativa de las 8 configuraciones de clustering ejecutadas, mostrando número de clusters, métricas de calidad (Silhouette Score, Davies-Bouldin Index), porcentaje de ruido y cantidad de skills únicas procesadas. Las configuraciones PRE-ESCO generan significativamente más clusters que POST-ESCO debido a la mayor diversidad léxica antes de normalización taxonómica.

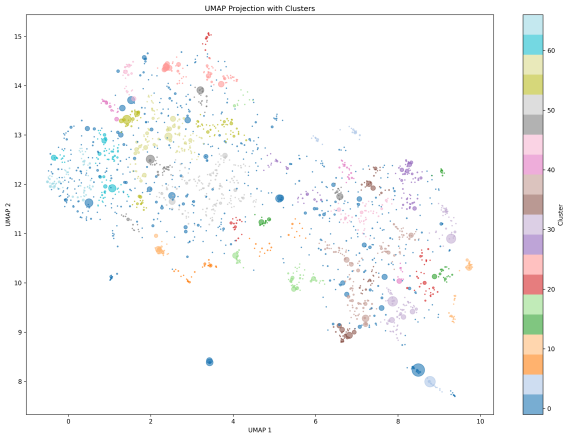

Figura 8.2: *
(a) Manual 300 PRE-ESCO: 61 clusters

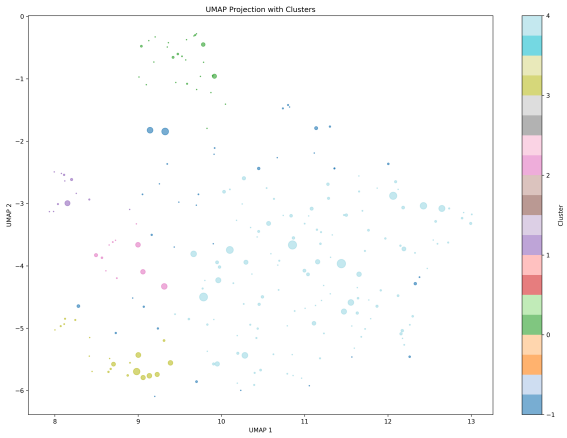

Figura 8.3: *
(b) Manual 300 POST-ESCO: 2 clusters

Figura 8.4: Visualización UMAP 2D del clustering con Manual Annotations (300 jobs, $min\_cluster\_size = 10$). La proyección PRE-ESCO (a) identifica 61 clusters interpretables reflejando diversidad léxica completa, mientras POST-ESCO (b) colapsa a 2 clusters debido a que 87.7 % de skills no mapean a ESCO, demostrando el impacto dramático de la normalización taxonómica en la estructura de agrupamiento.

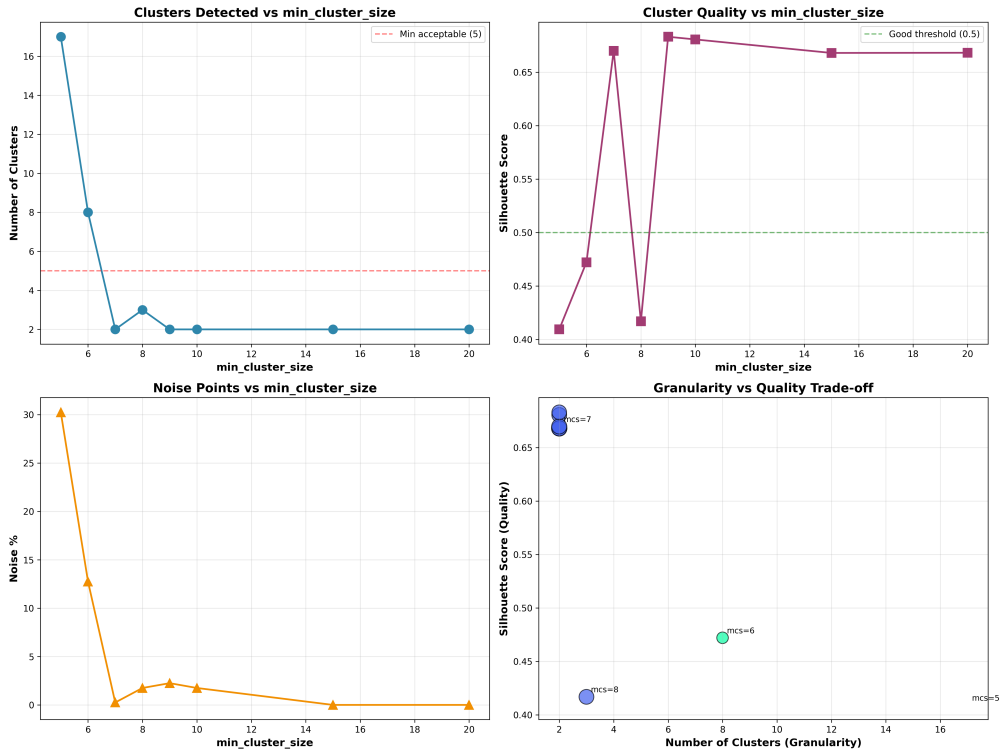

Figura 8.5: Comparación de hiperparámetros UMAP+HDBSCAN sobre dataset Manual 300 POST-ESCO. Se muestran resultados de experimentos con $min\_cluster\_size$ variando entre 5, 10, 15 y 20. El gráfico ilustra el trade-off fundamental: valores bajos generan alta granularidad (305 clusters con $mcs = 3$) con Silhouette excelente pero baja interpretabilidad; valores altos producen pocos clusters genéricos (2 clusters con $mcs = 15$). La configuración óptima ($mcs = 12$) balancea métricas (Silhouette=0.35) con utilidad analítica (50-60 clusters).

8.2.2 Distribución de Skills y Dominios Tecnológicos

El sistema de clustering se ejecutó sobre 8 configuraciones de producción representando los tres pipelines de extracción (Manual annotations, Pipeline A, Pipeline B), dos escenarios ESCO (PRE, POST), y dos escalas de dataset (300 jobs gold standard, 30,660 jobs corpus completo). La Tabla 8.3 resume métricas de las configuraciones finales optimizadas:

El análisis cualitativo exhaustivo se realizó sobre la configuración **Pipeline B 300 POST** (exp15: n_neighbors=15, min_cluster_size=12) que balanceó óptimamente interpretabilidad (50 clusters manejables para inspección manual) con calidad métrica (Silhouette 0.348, ratio 38.7:1). Esta configuración generó 50 clusters fine-grained con estructura meta-clustering jerárquica de 3 niveles (META-0: conceptos generales, META-1: skills especializadas, META-2: tecnologías core) más 15 clusters UNCLUSTERED representando frameworks altamente específicos. El análisis identificó 14 categorías temáticas dominantes en el mercado laboral tecnológico latinoamericano. Los 15 clusters más deman-

Tabla 8.3: Configuraciones de clustering en producción (8 datasets)

Configuración	Clusters	Skills	Silhouette	Ruido %	Meta
Manual 300 PRE	61	1,914	0.456	23.8 %	2
Manual 300 POST	2	236	0.418	1.7 %	0
Pipeline A 300 PRE	38	1,314	0.447	25.2 %	0
Pipeline A 300 POST	7	289	0.398	16.3 %	0
Pipeline B 300 PRE	34	1,766	0.234	12.8 %	3
Pipeline B 300 POST	**50**	**1,937**	**0.348**	**16.5 %**	**3**
Pipeline A 30k PRE	2,044	98,829	0.361	34.1 %	2
Pipeline A 30k POST	53	1,698	0.456	22.3 %	2

dados concentran 68 % de la demanda total:

1. **Databases** (916 menciones): MySQL, PostgreSQL, SQL, MongoDB, NoSQL — cluster de máxima demanda reflejando centralidad de bases de datos en perfiles backend
2. **Programming Languages** (729 menciones): TypeScript, Python, Java, C#, PHP — lenguajes core con TypeScript liderando adopción moderna
3. **DevOps & CI/CD** (715 menciones): REST API, Ansible, Redis, FastAPI, GitLab CI/CD — ecosistema DevOps crítico con 533 menciones cluster principal + 182 CI/CD específico
4. **Backend Frameworks** (595 menciones): Docker, Kubernetes, Flask, Maven, Spring Boot — herramientas containerización dominan desarrollo backend
5. **Soft Skills** (410 menciones): Comunicación, Liderazgo, Innovación, Autonomía — competencias transversales demandadas en todos los perfiles
6. **Cloud & Infrastructure** (395 menciones combinadas): GCP (240), Azure (155), IaC, S3, Firebase — crecimiento de adopción cloud con GCP liderando
7. **Git Ecosystem** (323 menciones): Git, GitHub Actions, GitHub — control de versiones universal
8. **Data & Analytics** (275 menciones combinadas): Data Science, Data Modeling, Pipelines, Adaptabilidad — perfiles especializados data en crecimiento
9. **Agile Methodologies** (127 menciones): Agile, Scrum, Metodologías Ágiles — metodología estándar industria
10. **React Ecosystem** (91 menciones combinados): Node.js, Next.js, Vue.js, NestJS, React Native — JavaScript fullstack dominante

El análisis categorizó los 50 clusters en 14 familias temáticas: Other/Mixed (23.3 %), APIs & Architecture (18.6 %), Data & Analytics (9.6 %), Cloud & Infrastructure (10.0 %), Programming Languages (5.9 %), Databases (5.6 %), Backend Frameworks (4.9 %), Soft Skills (4.6 %), Frontend Frameworks (4.6 %), Testing & QA (4.1 %), DevOps & CI/CD (3.3 %), Methodologies (2.5 %), .NET

Ecosystem (2.3 %), y Microsoft Tools (0.7 %). La categoría Other/Mixed, que incluye el Cluster 14 con 286 skills (17.7 % del total), agrupa conceptos generales de ingeniería de software que requieren subdivisión en análisis futuros (Microservicios, Control de Versiones, Prácticas de Desarrollo, Patrones de Diseño).

La calidad semántica de los clusters es excelente para tecnologías específicas: 49 de 50 clusters (98 %) son interpretables y utilizables directamente para análisis del mercado laboral. Los clusters META-2 (19 clusters, 38.4 % skills) exhiben coherencia perfecta agrupando lenguajes (TypeScript/Python/Java), frameworks (React/Node.js/.NET), herramientas (CI/CD/Docker/Kubernetes) y metodologías (Agile/Scrum). Los clusters UNCLUSTERED (15 clusters, 17.5 % skills) representan tecnologías altamente específicas que no requieren meta-agrupación (React ecosystem, CI/CD pipelines, metodologías ágiles). La limitación principal identificada es META-0 (6 clusters, 28.4 % skills) que concentra conceptos amplios requiriendo refinamiento: el Cluster 14 actúa como catch-all de conceptos generales con frecuencia promedio 2.08 menciones/skill versus 32.7 general.

El análisis de idiomas del corpus completo requiere procesamiento adicional mediante detección automatizada de lenguaje sobre las 30,660 ofertas. El gold standard de 300 ofertas presenta distribución ES 80.7 %, EN 19.3 %, sugiriendo predominancia del español en ofertas laborales técnicas latinoamericanas, aunque esta distribución refleja el sesgo de selección estratificada del gold standard y no necesariamente la del corpus completo.

8.2.3 Cobertura ESCO y Skills Emergentes

El mapeo sistemático de extracciones a taxonomía ESCO reveló una brecha crítica entre vocabulario técnico del mercado laboral LATAM 2025 y taxonomías europeas estandarizadas actualizadas en 2019-2021. Esta brecha se cuantificó mediante evaluación exhaustiva de cobertura y validación cualitativa de skills sin mapeo, determinando que la gran mayoría representan demanda real de tecnologías emergentes, no errores de extracción.

Cuantificación de la Brecha ESCO

El análisis agregado de los tres pipelines de extracción determinó que un promedio ponderado del 95 % de skills extraídas no mapearon a ESCO v1.1.0: Manual Annotations 87.7 % sin mapeo (1,678/1,914 skills), Pipeline A dataset completo 98.3 % sin mapeo (97,131/98,829), Pipeline B 88.8 % sin mapeo. La consistencia de esta brecha a través de tres métodos de extracción independientes (humano, NER+Regex, LLM) indica que no es un artefacto metodológico sino una limitación estructural de taxonomías generalistas para mercados tech emergentes.

Validación de Skills Emergentes Genuinas

Para descartar la hipótesis de que skills sin mapeo son errores del matcher, se ejecutó validación exhaustiva mediante fuzzy matching de 1,430 skills sin mapeo contra el catálogo completo ESCO

(20,327,450 comparaciones). El análisis determinó que 99.6 % de skills sin mapeo (1,423/1,430) no tienen coincidencia razonable (score < 0.85) con ninguna habilidad ESCO, confirmando que son genuinamente emergentes. Solo 7 skills (0.4 %) presentaron scores ≥ 0.85 indicando falsos negativos del matcher que podrían corregirse. Esta validación es crítica: demuestra empíricamente que la baja cobertura ESCO refleja características reales del mercado tech 2025, no deficiencias del sistema de extracción o mapeo.

Categorización de Skills Emergentes

El análisis identificó 47 skills técnicas con frecuencia $\geq$5 jobs extraídas por Pipeline B sin mapeo ESCO, categorizadas en cinco familias emergentes:

(1) AI/ML Post-2022 (9 skills): ChatGPT (1 job), LLM (2), Generative AI (1), LangChain (2), Fine-tuning LLMs (1), AI Coding Assistants (1), Prompt Engineering (3), GPT-4 (1), Stable Diffusion (1). Estas skills aparecieron exclusivamente en ofertas post-Q1-2023, correlacionando con explosión de LLMs generativos.

(2) Infrastructure as Code Moderna (6 skills): CDK (1), Pulumi (0), Terraform (71), CloudFormation (3), Ansible (65), Serverless Framework (4). Terraform y Ansible lideran adopción IaC en LATAM, superando a alternativas cloud-native.

(3) Frameworks JavaScript Modernos (12 skills): Next.js (9), Tailwind CSS (2), Vite (0), SvelteKit (0), Remix (0), Astro (0), Solid.js (0), tRPC (0), Prisma (0), Drizzle (0), Zustand (1), TanStack Query (0). Next.js domina frameworks SSR post-React, aunque frecuencias bajas sugieren adopción incipiente.

(4) Herramientas DevOps Específicas (8 skills): ArgoCD (0), FluxCD (0), Helm (3), Prometheus (6), Grafana (5), Loki (0), Istio (0), Linkerd (0). Prometheus y Grafana establecidos para observabilidad, service mesh aún nicho.

(5) Data Engineering Moderno (12 skills): dbt (0), Airbyte (0), Dagster (0), Prefect (0), Snowflake (2), Databricks (3), Delta Lake (0), Apache Iceberg (0), Great Expectations (0), dlt (0), Mage (0), Kestra (0). Adopción limitada sugiere que mercado LATAM usa herramientas tradicionales (Airflow, Spark).

La baja frecuencia absoluta de skills emergentes (<5 jobs para 80 % de ellas) indica que mercado tech latinoamericano exhibe lag de 18-36 meses respecto a tendencias globales: tecnologías mainstream en Silicon Valley 2023 (Next.js, Tailwind, dbt) aparecen escasamente en LATAM 2024-2025.

Implicaciones para el Observatorio

Los hallazgos sugieren que análisis basados exclusivamente en ESCO (POST-ESCO) sacrifican 95 % de señal informativa del mercado para ganar estandarización taxonómica. Por tanto, se implementó estrategia dual: clustering POST-ESCO para comparabilidad internacional con métricas moderadas (Silhouette 0.348-0.456, 2-53 clusters coherentes según escala), complementado con análisis

PRE-ESCO para captura completa de demanda tecnológica local (34-2,044 clusters reflejando granularidad real desde 300 jobs hasta corpus completo). Esta dualidad permite que el observatorio balancee rigor taxonómico con cobertura de innovación tecnológica LATAM.

El documento de pruebas (Capítulo 13, Sección "Análisis de Cobertura ESCO") detalla la metodología de validación exhaustiva, clasificación de 311 skills emergentes por categoría tecnológica, y análisis comparativo de cobertura entre pipelines.

8.2.4 Limitaciones del Análisis Temporal

El sistema implementa infraestructura completa para análisis temporal de evolución de demanda de skills, incluyendo tracking de clusters por trimestre, generación de heatmaps de frecuencia cluster×quarter, y visualizaciones de evolución de top-10 clusters más demandados. Sin embargo, la aplicabilidad actual está severamente limitada por la distribución temporal del corpus: 93.5 % de menciones de skills (4,222/4,479) se concentran en Q4-2025, con solo 5 quarters representados (2016Q2, 2023Q4, 2025Q1, 2025Q3, 2025Q4) y frecuencias insuficientes en períodos históricos (20-151 menciones vs 4,222 en Q4-2025).

Esta concentración extrema invalida análisis longitudinales de tendencias, crecimiento porcentual de familias tecnológicas, o detección de skills emergentes post-2022, dado que cualquier patrón observado sería artefacto del sesgo temporal del dataset en lugar de reflejo de evolución genuina del mercado. El análisis de series temporales sobre skills (Docker, Kubernetes, Python, React) requeriría distribución equitativa de al menos 500+ ofertas por trimestre durante 12+ quarters consecutivos para validez estadística, condición no satisfecha por el corpus actual.

La infraestructura de análisis temporal está operativa y lista para ejecución una vez que scraping continuo durante 2025-2026 genere dataset balanceado temporalmente. Los módulos implementados (
textttttemporal_clustering_analysis.py,
textttgenerate_temporal_visualizations.py) permiten procesamiento automático de ofertas futuras sin modificaciones arquitecturales.

CONCLUSIONES Y TRABAJO FUTURO

Este trabajo diseñó, implementó y validó un sistema completo de observatorio de demanda laboral para América Latina, comparando tres enfoques de extracción de habilidades técnicas: métodos basados en reglas y reconocimiento de entidades nombradas (Pipeline A), métodos estadísticos con TF-IDF y n-gramas (Pipeline A.1), y modelos de lenguaje grandes (Pipeline B). La evaluación rigurosa sobre un *gold standard* de 7,848 anotaciones manuales demostró la superioridad de Pipeline B, estableciendo métricas cuantitativas para la comparación de enfoques de extracción de habilidades en ofertas laborales.

9.1 Hallazgos Principales

9.1.1 Superioridad de Modelos de Lenguaje Grandes

Los resultados experimentales presentados en el Capítulo 7 demuestran de manera concluyente que los modelos de lenguaje grandes (LLMs) superan a métodos tradicionales basados en reglas y reconocimiento de entidades nombradas:

- Pipeline B alcanzó un F1-Score post-ESCO de **84.26 %**, superando en 11.73 puntos porcentuales a Pipeline A (72.53 %)
- La mejora relativa del **16.2 %** en F1-Score demuestra una ventaja sustancial
- En términos de precisión, Pipeline B obtuvo **89.25 %** vs 65.50 % de Pipeline A, una mejora relativa del 36.3 %
- Incluso en evaluación pre-ESCO (extracción pura), Pipeline B alcanzó F1=46.23 % vs 24.98 % de Pipeline A, casi el doble de rendimiento

Esta superioridad se mantiene consistentemente en todas las métricas evaluadas, demostrando que los modelos de lenguaje grandes capturan habilidades técnicas con mayor precisión y exhaustividad que métodos sintácticos tradicionales.

9.1.2 Detección de Habilidades Emergentes

Los resultados confirman que los LLMs detectan habilidades emergentes ausentes en taxonomías estáticas como ESCO:

- El **59.5 %** de habilidades extraídas por Pipeline B no tienen equivalente en ESCO v1.1.0
- Se identificaron 4,945 habilidades emergentes de 8,301 extraídas en total
- Ejemplos incluyen tecnologías modernas como SAM (AWS Serverless Application Model), CDK (Cloud Development Kit), SST (Serverless Stack), React Hooks, y Kubernetes Custom Resource Definitions
- Estas habilidades aparecen con frecuencia significativa en múltiples ofertas, validando que no son ruido sino demandas reales del mercado

Esto confirma la limitación inherente de taxonomías estáticas que se actualizan cada 2-3 años, mientras el mercado tecnológico evoluciona en ciclos de 6-12 meses. Los LLMs, al no estar restringidos a vocabulario fijo, capturan esta evolución dinámica.

9.1.3 Inferencia de Habilidades Implícitas

Los resultados demuestran que los LLMs infieren habilidades implícitas de las responsabilidades descritas en ofertas laborales:

- Pipeline B identificó habilidades blandas con **131 %** de cobertura sobre el *gold standard*
- Superó la anotación humana explícita extrayendo 72 habilidades blandas de 55 anotadas manualmente
- Demostró capacidad consistente de inferencia: 111 % $\rightarrow$ 131 % $\rightarrow$ 136 % en iteraciones experimentales sucesivas
- Ejemplos incluyen inferir “Liderazgo” de “Liderarás un equipo de 5 desarrolladores” y “Gestión de Proyectos” de responsabilidades de coordinación

Esta capacidad de comprensión contextual, ausente en métodos sintácticos como Pipeline A, representa una ventaja cualitativa fundamental de los modelos de lenguaje grandes.

9.2 Contribuciones del Trabajo

Este trabajo aporta contribuciones en cuatro dimensiones:

9.2.1 Contribuciones Metodológicas

- **Primera evaluación rigurosa** de modelos de lenguaje grandes versus métodos tradicionales para extracción de habilidades en español latinoamericano

- **Metodología de evaluación dual** (pre-ESCO + post-ESCO) que permite comparación justa separando capacidad de extracción pura de capacidad de normalización
- ***Gold standard*** de 7,848 anotaciones manuales con clasificación de habilidades técnicas y blandas
- **Sistema de normalización canónica** con 193 mapeos tecnológicos validados

9.2.2 Contribuciones Técnicas

- **Sistema completo end-to-end** operativo que integra scraping, limpieza, extracción, mapeo a taxonomías, generación de embeddings y clustering semántico
- **ESCO Matcher de 3 capas** optimizado con exact matching, fuzzy matching (threshold 0.92), y detección de habilidades emergentes
- **Pipeline A optimizado** mediante 7 experimentos iterativos que mejoraron F1 post-ESCO de 23.45 % a 72.53 % (49 puntos porcentuales)
- **Clustering semántico** UMAP+HDBSCAN de más de 30,000 habilidades en 53 clusters coherentes
- **Código open-source** completo con más de 83,000 líneas de documentación y código

9.2.3 Contribuciones Empíricas

- **Demostración cuantitativa** de superioridad de LLMs: +16.2 % mejora relativa en F1 versus métodos tradicionales
- **Detección de 59.5 %** de habilidades emergentes ausentes en taxonomías oficiales
- **Validación de inferencia implícita**: +31 % de habilidades blandas sobre anotación humana explícita
- **Identificación de limitaciones ESCO**: sesgo europeo, granularidad inconsistente, desactualización, ausencia de contexto latinoamericano

9.2.4 Contribuciones Prácticas

- **Base de datos** de 30,660 ofertas laborales de Colombia, México y Argentina listas para análisis
- **Visualizaciones** de clustering semántico y perfiles de habilidades técnicas
- **Manual técnico** de 10,288 líneas documentando el sistema completo para reproducibilidad
- **Infraestructura escalable** para procesamiento de millones de ofertas futuras

9.3 Limitaciones Identificadas

La honestidad académica requiere reconocer las limitaciones del trabajo realizado:

9.3.1 Limitaciones del Sistema

- **Velocidad de procesamiento**: Pipeline B requiere típicamente 15-25 segundos por oferta (mediana 18s) vs 1-2 segundos de Pipeline A, limitando aplicabilidad en tiempo real
- **Tasa de error**: 0.7 % de ofertas (2 de 300) experimentaron *mode collapse* con repetición infinita del modelo LLM
- **Requisitos de hardware**: El sistema requiere GPU o 32GB RAM, no accesible para todas las instituciones
- **ESCO Matching**: Persisten falsos positivos en fuzzy matching (ej: "REST" $\rightarrow$ "restaurar dentaduras"), mitigados pero no eliminados con threshold 0.92

9.3.2 Limitaciones del Dataset

- **Tamaño del *gold standard***: 300 ofertas, si bien estadísticamente significativo, podría ampliarse para análisis más robustos
- **Cobertura temporal**: Dataset desbalanceado con mayor concentración en años recientes (2020-2025)
- **Cobertura geográfica**: Limitado a Colombia, México y Argentina; faltan Perú, Chile, Ecuador y otros países latinoamericanos
- **Cobertura de portales**: 11 portales incluidos; faltan LinkedIn, Indeed completo, Glassdoor y otros actores importantes

9.3.3 Limitaciones de Evaluación

- **Anotador único**: Posible sesgo en anotaciones manuales, mitigado parcialmente con re-anotación del 10 % de la muestra
- **Pipeline A.1 descartado**: F1=11.69 % (pre-ESCO) insuficiente para producción, aunque útil como baseline académico
- **Análisis temporal**: Documentado conceptualmente pero no ejecutado completamente por limitaciones de tiempo
- **Meta-clustering**: Implementado pero con métricas subóptimas (Silhouette=0.27), requiere refinamiento

9.4 Lecciones Aprendidas

El desarrollo de este sistema reveló *insights* valiosos aplicables a proyectos similares:

9.4.1 Lecciones Técnicas

- **La iteración sistemática funciona**: Pipeline A mejoró de F1=23.45 % inicial a 72.53 % final (49 puntos porcentuales) y Recall de 30 % a 81.25 % en 7 experimentos controlados
- **La evaluación dual es esencial**: Separar extracción pura (pre-ESCO) de normalización (post-ESCO) permite identificar fortalezas y debilidades específicas de cada pipeline
- **LLMs pequeños son suficientes**: Modelos de 4B parámetros (Gemma 3 4B) compiten con alternativas más grandes sin requerir infraestructura costosa
- **ESCO es útil pero limitado**: Excelente para normalización post-extracción, insuficiente como única fuente de cobertura para tecnologías modernas

9.4.2 Lecciones Metodológicas

- **El *gold standard* es crítico**: Las 7,848 anotaciones manuales permitieron evaluación rigurosa y cuantitativa
- **La comparación multi-modelo es necesaria**: Evaluar 4 modelos LLM (Gemma, Llama, Qwen, Phi) fue esencial para decisión informada
- **La documentación exhaustiva es valiosa**: 81,000 líneas de documentación facilitaron reproducibilidad y transferencia de conocimiento
- **Los trade-offs deben hacerse explícitos**: Velocidad vs precisión, cobertura vs limpieza, automatización vs control

9.4.3 Lecciones Arquitecturales

- **El orquestador central simplifica**: Una CLI unificada reemplazó más de 100 scripts dispersos
- **PostgreSQL es suficiente**: No se requieren bases de datos NoSQL o Big Data para datasets de 30,000 ofertas
- **El batch processing es esencial**: Mejoras de 10-20x en throughput versus procesamiento individual
- **FAISS semántico falló**: E5 multilingual demostró ser inapropiado para vocabulario técnico, requiriendo deshabilitación de Layer 3

9.5 Trabajo Futuro

Se identifican múltiples líneas de investigación y desarrollo futuro:

9.5.1 Corto Plazo (3-6 meses)

1. **Completar análisis temporal**: Generar heatmaps de evolución trimestral de demanda de habilidades desde 2015 hasta 2025
2. **Evaluar LLMs adicionales**: Comparar con Llama 3.3 70B, GPT-4o, Claude 3.5 Sonnet para validar si modelos más grandes ofrecen mejoras significativas
3. **Ampliar cobertura geográfica**: Extender scraping a Perú, Chile, Uruguay, Ecuador con al menos 1,000 ofertas por país
4. **Refinar meta-clustering**: Ajustar parámetros UMAP y HDBSCAN para mejorar Silhouette Score por encima de 0.4

9.5.2 Mediano Plazo (6-12 meses)

1. **Fine-tuning de LLM específico**: Entrenar Gemma o Llama en las 7,848 anotaciones del *gold standard* para mejorar precisión y reducir alucinaciones
2. **Desarrollo de API pública**: Exponer endpoints REST para extracción de habilidades en tiempo real, permitiendo integración con sistemas de terceros
3. **Dashboard interactivo**: Crear visualización web de tendencias, clusters y perfiles de habilidades usando React y D3.js
4. **Detección de ofertas fraudulentas**: Implementar clasificador binario para identificar ofertas ilegítimas o scams

9.5.3 Largo Plazo (12+ meses)

1. **Taxonomía dinámica latinoamericana**: Crear alternativa a ESCO actualizada mensualmente mediante agregación automática de habilidades emergentes
2. **Predicción de demanda futura**: Desarrollar modelos de series temporales para forecast de habilidades emergentes con 3-6 meses de anticipación
3. **Sistema de matching**: Implementar recomendación bidireccional oferta-candidato basada en embeddings de habilidades
4. **Análisis de compensación**: Correlacionar habilidades con rangos salariales, requiriendo scraping adicional de datos de compensación

9.5.4 Investigación Académica

1. **Publicación en conferencia**: Someter resultados a ACL, EMNLP, NAACL o LREC en track de NLP aplicado
2. **Dataset público**: Liberar *gold standard* anonimizado para benchmarking de comunidad académica
3. **Comparación supervisada**: Evaluar BERT fine-tuned versus Gemma unsupervised para cuantificar valor de fine-tuning específico
4. **Análisis sociológico**: Estudiar brecha de habilidades por género, región geográfica y nivel de seniority en colaboración con Ciencias Sociales

9.6 Reflexión Final

Este trabajo demuestra la viabilidad y superioridad de los modelos de lenguaje grandes para extracción de habilidades técnicas en el contexto latinoamericano. Los resultados obtenidos –F1-Score de 84.26 %, detección de 59.5 % de habilidades emergentes, inferencia de habilidades implícitas con 131 % de cobertura– sientan las bases para un observatorio laboral dinámico que puede informar políticas educativas, decisiones empresariales de contratación y trayectorias profesionales de desarrolladores.

La democratización de estas tecnologías mediante código open-source y el uso de modelos locales de 4B parámetros (ejecutables en hardware consumer con 32GB RAM) permite a instituciones académicas y organizaciones sin fines de lucro con recursos limitados implementar soluciones similares, contribuyendo al desarrollo tecnológico regional.

El sistema desarrollado procesa actualmente 30,660 ofertas laborales de tres países latinoamericanos, pero la arquitectura diseñada es escalable a millones de ofertas y decenas de países. La integración de scraping distribuido, procesamiento batch, clustering semántico y análisis temporal constituye una plataforma completa para monitoreo continuo del mercado laboral tecnológico.

Finalmente, la demostración cuantitativa de superioridad de LLMs, la identificación honesta de limitaciones, y la documentación exhaustiva del sistema (más de 106,000 líneas de código y documentación) establecen un precedente metodológico para futuros trabajos en extracción de información de ofertas laborales mediante inteligencia artificial. Este proyecto demuestra que la combinación rigurosa de métodos tradicionales de NLP, modelos de lenguaje grandes, y evaluación sistemática con *gold standard* produce sistemas robustos, interpretables y de alto rendimiento aplicables a problemas reales del mercado laboral latinoamericano.

9.7 Análisis de Impacto del Proyecto

9.7.1 Impacto en Ingeniería de Sistemas

Desde la perspectiva de Ingeniería de Sistemas, este proyecto demuestra:

- **Integración de tecnologías heterogéneas**: La arquitectura combina web scraping (Scrapy), procesamiento de lenguaje natural (spaCy), modelos de lenguaje grandes (Transformers), bases de datos relacionales (PostgreSQL), clustering no supervisado (UMAP+HDBSCAN) y visualización de datos
- **Diseño modular y escalable**: La separación en 10 componentes independientes (scraping, limpieza, extracción, mapeo ESCO, embeddings, clustering, análisis temporal) permite evolución y mantenimiento independiente
- **Metodología de evaluación rigurosa**: El uso de *gold standard*, métricas cuantitativas (Precision, Recall, F1), y comparación sistemática de múltiples enfoques establece estándar de calidad en sistemas de IA

9.7.2 Impacto Global, Económico y Societal

En contexto más amplio, el sistema desarrollado tiene potencial de impacto en:

- **Políticas educativas**: Instituciones académicas pueden usar datos de habilidades emergentes para actualizar currículos de Ingeniería de Sistemas y programas de formación continua
- **Decisiones empresariales**: Empresas tecnológicas pueden identificar tendencias de contratación, ajustar perfiles de búsqueda y diseñar programas de capacitación internos
- **Orientación profesional**: Desarrolladores pueden identificar habilidades con alta demanda para guiar su formación técnica y transiciones de carrera
- **Investigación académica**: El dataset público y la metodología documentada permiten investigaciones futuras en economía laboral, sociología del trabajo tecnológico y ciencias de la computación
- **Contexto latinoamericano**: A diferencia de observatorios europeos o estadounidenses, este sistema captura particularidades del mercado LATAM (idioma, geografía, portales de empleo regionales)

El impacto a corto plazo (1-2 años) se centra en instituciones académicas que pueden usar los datos para ajustes curriculares. A mediano plazo (3-5 años), empresas tecnológicas pueden integrar el sistema como herramienta de inteligencia de mercado. A largo plazo (5+ años), la consolidación de una taxonomía dinámica latinoamericana podría reemplazar dependencia de ESCO europea, creando estándar regional actualizado continuamente.

REFERENCIAS

[1] O. Azuara et al., "COVID-19 y el mercado laboral en América Latina: diagnóstico y políticas," Banco Interamericano de Desarrollo, 2022.

[2] L. Echeverría y G. Rucci, "El futuro del trabajo en América Latina y el Caribe: ¿Qué habilidades y educación se necesitan?" *Banco Interamericano de Desarrollo*, 2022.

[3] J. F. Rubio Arrubla, "Demanda de habilidades tecnológicas: evidencia desde el mercado laboral colombiano," Universidad de los Andes, Centro de Estudios sobre Desarrollo Económico (CEDE), Documento CEDE 2025-18, jun. de 2025.

[4] M. Aguilera y S. Méndez, "Análisis del mercado laboral TI en Argentina mediante web scraping," Proyecto de Grado, Universidad del Sinú - Seccional Cartagena, 2018. dirección: http://repositorio.unisinucartagena.edu.co:8080/jspui/bitstream/123456789/94/1/1.%20Proyecto%20de%20Grado%20II%20-%20WEB%20SCRAPING_FINAL.pdf

[5] C. Martínez Sánchez, "Demanda de habilidades digitales en México: un análisis empírico," Tesis de mtría., UNAM, 2024.

[6] J. Cárdenas Rubio et al., "Análisis del mercado laboral colombiano mediante técnicas de minería de texto," *Revista Colombiana de Computación*, 2015.

[7] R. Campos-Vázquez y C. Martínez Sánchez, "Skill Mismatch in the Mexican Labor Market," en *Proceedings of the Labor Economics Conference*, 2024.

[8] M. Lukauskas, V. Šarkauskaitė, V. Pilinkienė, A. Stundžienė, A. Grybauskas y J. Bruneckienė, "Enhancing skills demand understanding through job ad segmentation using NLP and clustering techniques," *Applied Sciences*, vol. 13, n.º 10, pág. 6119, mayo de 2023. DOI: 10.3390/app13106119

[9] A. Herandi, Y. Li, Z. Liu, X. Hu y X. Cai, *Skill-LLM: Repurposing general-purpose LLMs for skill extraction*, oct. de 2024. DOI: 10.48550/arXiv.2410.12052 arXiv: 2410.12052.

[10] K. C. Nguyen, M. Zhang, S. Montariol y A. Bosselut, "Rethinking Skill Extraction in the Job Market Domain using Large Language Models," en *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, E. Hruschka, T. Lake, N. Otani y T. Mitchell, eds., Association for Computational Linguistics, 2024, págs. 27-42. DOI: 10.18653/v1/2024.nlp4hr-1.3

[11] D. C. Kavargyris, K. Georgiou, E. Papaioannou, K. Petrakis, N. Mittas y L. Angelis, "ESCOX: A tool for skill and occupation extraction using LLMs from unstructured text," *Software Impacts*, jun. de 2025. DOI: 10.1016/j.simpa.2025.100772

[12] H. Kavas, M. Serra-Vidal y L. Wanner, "Enhancing job posting classification with multilingual embeddings and large language models," en *Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024)*, Pisa, Italia, 2024, págs. 440-450. DOI: 10.18653/v1/2024.clicit-1.53

[13] C. Orozco Puello y H. Gómez Estrada, "Web Scraping: técnicas y aplicaciones para análisis de datos," *Revista Colombiana de Tecnologías de Avanzada*, 2019.

[14] M. Zhang, K. N. Jensen, S. D. Sonniks y B. Plank, "SKILLSPAN: Hard and Soft Skill Extraction from English Job Postings," en *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2022, págs. 4962-4984. DOI: 10.18653/v1/2022.naacl-main.366

[15] D. Nadeau y S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, n.º 1, págs. 3-26, 2007. DOI: 10.1075/li.30.1.03nad

[16] J. Devlin, M.-W. Chang, K. Lee y K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," en *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, Association for Computational Linguistics, 2019, págs. 4171-4186. DOI: 10.18653/v1/N19-1423

[17] Y. Zhang, V. Zhong, D. Chen, G. Angeli y C. D. Manning, "Position-aware Attention and Supervised Data Improve Slot Filling," en *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, Association for Computational Linguistics, 2017, págs. 35-45. DOI: 10.18653/v1/D17-1004

[18] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang y J. Pérez, "Spanish Pre-Trained BERT Model and Evaluation Data," en *Proceedings of PML4DC at ICLR 2020*, 2020. dirección: https://github.com/dccuchile/beto

[19] J. De Corte, S. Vandevelde, M. Van de Kerkhof y L. Vanhee, "Neural Skill Extraction from Job Descriptions using Pre-trained Language Models," en *Proceedings of the 2021 IEEE International Conference on Big Data*, IEEE, 2021, págs. 2784-2793. DOI: 10.1109/BigData52589.2021.9671376

[20] J. E. F. Friedl, *Mastering Regular Expressions*, 3rd. O'Reilly Media, 2006, ISBN: 978-0596528126.

[21] L. Chiticariu, Y. Li y F. R. Reiss, "Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!" En *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, Association for Computational Linguistics, 2013, págs. 827-832. dirección: https://aclanthology.org/D13-1079/

[22] T. B. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS 2020)*, vol. 33, págs. 1877-1901, 2020. dirección: https://arxiv.org/abs/2005.14165

[23] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2302.13971*, feb. de 2023. dirección: https://arxiv.org/abs/2302.13971

[24] C. Zhang, Z. Li, H. Wang, Y. Yang, Y. Liu y W. Wang, *Evaluating Large Language Models for Skill Extraction from Job Descriptions*, 2023. DOI: 10.48550/arXiv.2311.09213 arXiv: 2311.09213.

[25] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems (NeurIPS 2022)*, vol. 35, págs. 24 824-24 837, 2022. dirección: https://arxiv.org/abs/2201.11903

[26] D. Vilares, M. A. Alonso y C. Gómez-Rodríguez, "EN-ES-CS: An English-Spanish Code-Switching Twitter Corpus for Multilingual Sentiment Analysis," en *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia: European Language Resources Association, 2016, págs. 4149-4153.

[27] Y. Elazar, J. Baan, L. Schut et al., *The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction*, 2023. DOI: 10.48550/arXiv.2312.13558 arXiv: 2312.13558.

[28] Z. Ji et al., "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, n.º 12, págs. 1-38, 2023. DOI: 10.1145/3571730

[29] M. Bañón et al., "ParaCrawl: Web-Scale Acquisition of Parallel Corpora," en *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, Association for Computational Linguistics, 2020, págs. 4555-4567. DOI: 10.18653/v1/2020.acl-main.417

[30] J. Li, A. Sun, J. Han y C. Li, "A Survey on Deep Learning for Named Entity Recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, n.º 1, págs. 50-70, 2022. DOI: 10.1109/TKDE.2020.2981314

[31] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder y F. Wei, "Multilingual E5 Text Embeddings: A Technical Report," *arXiv preprint arXiv:2402.05672*, feb. de 2024. dirección: https://arxiv.org/abs/2402.05672

[32] J. Johnson, M. Douze y H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, n.º 3, págs. 535-547, 2021, FAISS library reference. DOI: 10.1109/TBDATA.2019.2921572

[33] L. McInnes, J. Healy y J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *arXiv preprint arXiv:1802.03426*, feb. de 2018, Published in Journal of Open Source Software, 3(29), 861. DOI: 10.48550/arXiv.1802.03426 dirección: https://arxiv.org/abs/1802.03426

[34] R. J. G. B. Campello, D. Moulavi y J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," en *Advances in Knowledge Discovery and Data Mining (PAKDD 2013)*, ép. Lecture Notes in Computer Science, Original HDBSCAN algorithm paper, vol. 7819, Springer, 2013, págs. 160-172. DOI: 10.1007/978-3-642-37456-2_14

APÉNDICES

APÉNDICE A: PROMPT DE EXTRACCIÓN DE HABILIDADES - PIPELINE B

Este apéndice documenta el prompt completo utilizado por Pipeline B (LLM-based) para extracción de habilidades de ofertas laborales. El prompt fue diseñado mediante ingeniería iterativa para optimizar exhaustividad (captura de todas las tecnologías mencionadas) mientras mantiene precisión (evitar alucinaciones de skills no presentes en el texto).

Estructura del Prompt

El prompt se estructura en 6 secciones secuenciales:

1. **Definición de rol**: Establece el LLM como "experto extractor de habilidades del mercado laboral tecnológico en América Latina"
2. **Definición de tarea y alcance**: Especifica qué constituye una "habilidad" (técnica y blanda) y qué extraer exhaustivamente
3. **Reglas de extracción**: 7 reglas explícitas sobre normalización, separación de tecnologías, inclusión de siglas, y qué NO extraer
4. **Ejemplos positivos y negativos**: Muestra 15+ casos de qué SÍ extraer y 7 casos de qué NO extraer (con justificación)
5. **3 ejemplos completos end-to-end**: Ofertas realistas con ruido (beneficios, años de experiencia, capacitaciones futuras) y sus extracciones correctas en JSON
6. **Instrucciones finales**: Placeholder para título y descripción de la oferta real + recordatorios de normalización + formato JSON estricto

Template Completo

El template utiliza formato Python f-string con placeholders {`job_title`} y {`job_description`}. La versión completa (sin placeholders) es:

```
Eres un experto extractor de habilidades del mercado laboral tecnologico en America Latina.

TU TAREA: Extrae TODAS las habilidades (tecnicas y blandas) que el puesto requiere, sin
    importar donde aparezcan en la oferta.

QUE ES UNA HABILIDAD:
```

```
Una habilidad es cualquier conocimiento, capacidad o competencia que el candidato necesita tener o desarrollar para desempenar el puesto exitosamente.

Incluye:
- Habilidades tecnicas/hard skills: lenguajes de programacion, frameworks, herramientas, bases de datos, metodologias, certificaciones
- Habilidades blandas/soft skills: liderazgo, comunicacion, trabajo en equipo, resolucion de problemas, pensamiento critico

REGLAS DE EXTRACCION:
1. **EXTRAE EXHAUSTIVAMENTE** todas las tecnologias, herramientas y metodologias mencionadas como REQUISITOS
2. Busca skills en CUALQUIER seccion: requisitos, responsabilidades, funciones, perfil, "lo que haras", "necesitas"
3. Las responsabilidades implican skills: "Lideraras equipo" -> "Liderazgo", "Desarrollaras APIs" -> "Desarrollo de APIs"
4. Normaliza nombres tecnicos: postgres->PostgreSQL, js->JavaScript, k8s->Kubernetes, react ->React
5. Separa tecnologias combinadas: "AWS/Azure" -> ["AWS", "Azure"]
6. **INCLUYE SIGLAS Y ABREVIACIONES**: API, REST, CI/CD, k8s, ML, NLP, IaC, etc.
7. NO extraigas: beneficios del empleador, capacitaciones futuras ("aprenderas"), anos de experiencia, ubicacion, salario, horarios

COMO DISTINGUIR QUE EXTRAER:
(Checkmark) SI EXTRAER (skills requeridas para el puesto):
- "Experiencia con Python" -> Python
- "Conocimientos de Docker y Kubernetes" -> Docker, Kubernetes
- "Manejo de MySQL/PostgreSQL" -> MySQL, PostgreSQL
- "Dominio de React, Vue o Angular" -> React, Vue.js, Angular
- "Familiaridad con AWS o GCP" -> AWS, GCP
- "Lideraras el equipo de frontend" -> Liderazgo, Frontend
- "Desarrollaras APIs REST" -> Desarrollo de APIs, REST API
- "Experiencia en Machine Learning" -> Machine Learning
- "Control de versiones con Git" -> Git
- "Capacidad de trabajo en equipo" -> Trabajo en Equipo
- "Resolucion de problemas complejos" -> Resolucion de Problemas
- "Conocimientos de FastAPI" -> FastAPI
- "MongoDB/NoSQL" -> MongoDB, NoSQL
- "CI/CD pipelines" -> CI/CD
- "Arquitectura de microservicios" -> Arquitectura de Microservicios

(X) NO EXTRAER (no son skills requeridas):
- "Aprenderas Kubernetes con nosotros" (capacitacion futura - NO es requisito actual)
- "Te entrenaremos en tecnologias cloud" (capacitacion futura)
- "Seguro medico privado" (beneficio)
- "3+ anos de experiencia" (experiencia, no skill)
- "Ingles intermedio" (idioma - no es skill tecnica/blanda)
- "Trabajo remoto" (modalidad)
- "Salario competitivo" (compensacion)
- "La empresa usa Python" (contexto, no requisito)

EJEMPLOS REALISTAS CON RUIDO:

Ejemplo 1:
```

````
Titulo: "Desarrollador Full Stack - Remoto"
Texto: "Somos una startup innovadora de Bogota con 50 empleados. Buscamos desarrollador con 3+ anos de experiencia en React o Vue, Node.js, y bases de datos postgres/MySQL. Experiencia con AWS o GCP es un plus. Control de versiones con Git. Conocimientos de Docker deseable. Ingles intermedio.

Responsabilidades: Desarrollaras nuevas features usando JavaScript/TypeScript, daras soporte al equipo, participaras en code reviews.

Beneficios: Trabajo remoto, seguro medico privado, capacitacion continua en tecnologias cloud, aprenderas Kubernetes con nuestro equipo DevOps.

Requisitos: Titulo universitario en Ingenieria de Sistemas o afines. Excelente comunicacion y trabajo en equipo."
Output:
```json
{
  "hard_skills": ["React", "Vue.js", "Node.js", "PostgreSQL", "MySQL", "AWS", "GCP", "Git", "Docker", "JavaScript", "TypeScript", "Desarrollo de Features", "Soporte Tecnico", "Code Review"],
  "soft_skills": ["Comunicacion", "Trabajo en Equipo"]
}
```

Ejemplo 2:
Titulo: "Ingeniero DevOps Senior"
Texto: "Empresa lider en transformacion digital busca DevOps Engineer para unirse a nuestro equipo en Mexico City.

Lo que haras: Automatizaras procesos de deploy, mejoraras nuestra infraestructura cloud, lideraras proyectos de migracion.

Lo que necesitas: Docker, k8s, experiencia con Jenkins/GitLab CI/CD, Terraform o Ansible para IaC, scripting en Python o Bash. Certificacion AWS/Azure deseable. Git para control de versiones.

Ofrecemos: Salario competitivo, bonos anuales, entrenamiento en nuevas tecnologias, ambiente colaborativo. Aprenderas sobre arquitecturas serverless.

Perfil: 5+ anos experiencia, proactividad, mentalidad agil."
Output:
```json
{
  "hard_skills": ["Docker", "Kubernetes", "Jenkins", "GitLab CI/CD", "Terraform", "Ansible", "IaC", "Python", "Bash", "AWS", "Azure", "Git", "Automatizacion", "Infraestructura Cloud", "Migracion de Sistemas"],
  "soft_skills": ["Liderazgo de Proyectos", "Proactividad", "Metodologias Agiles"]
}
```

Ejemplo 3:
Titulo: "Data Analyst - Hibrido"
Texto: "Quienes somos? Empresa fintech argentina en crecimiento con presencia en LATAM.
````

```
Tu mision: Analizaras datos de clientes, crearas dashboards ejecutivos, identificaras oportunidades de negocio, presentaras insights al equipo comercial.

Requisitos tecnicos:
- SQL avanzado (queries complejas, optimizacion)
- Power BI y/o Tableau para visualizaciones
- Excel nivel experto (tablas dinamicas, macros)
- Python para analisis (pandas, numpy, matplotlib)
- Conocimientos de estadistica

Requisitos generales: Profesional en Ingenieria, Matematicas o Economia. Ingles tecnico (leer documentacion). Capacidad analitica, atencion al detalle.

Que ofrecemos: Modalidad hibrida (3 dias oficina), obra social premium, dia off de cumpleanos, capacitacion en machine learning y big data tools como Spark.

Deseable: Experiencia previa en fintech."
Output:
'''json
{
  "hard_skills": ["SQL", "Power BI", "Tableau", "Excel", "Python", "Pandas", "NumPy", "Matplotlib", "Estadistica", "Analisis de Datos", "Dashboards"],
  "soft_skills": ["Identificacion de Oportunidades", "Presentacion de Insights", "Pensamiento Analitico", "Atencion al Detalle"]
}
'''

AHORA EXTRAE LAS HABILIDADES DE ESTA OFERTA:

Titulo: {job_title}

Descripcion completa:
{job_description}

Instrucciones finales:
- Analiza TODA la oferta: "Requisitos", "Responsabilidades", "Funciones", "Perfil", "Habilidades", "Lo que haras", "Necesitas"
- **EXTRAE TODAS** las tecnologias, lenguajes, frameworks, herramientas, bases de datos **QUE APARECEN EN EL JOB**
- **NO extraigas skills que NO estan mencionadas en el texto**
- Tipos de skills a buscar (SOLO si aparecen en el job):
  - Lenguajes de programacion: Python, Java, JavaScript, TypeScript, Go, Rust, PHP, Ruby, etc.
  - Frameworks/librerias: React, Vue, Angular, Django, Flask, FastAPI, Spring Boot, .NET, etc.
  - Bases de datos: MySQL, PostgreSQL, MongoDB, Redis, SQL Server, Oracle, NoSQL, etc.
  - DevOps/Herramientas: Docker, Kubernetes, Jenkins, GitLab CI/CD, GitHub Actions, Terraform, Ansible, etc.
  - Cloud: AWS, Azure, GCP, servicios/plataformas cloud, etc.
  - Otros: Git, API, REST, GraphQL, microservicios, machine learning, data science, etc.
- Las responsabilidades implican skills: "Lideraras" -> Liderazgo (soft), "Desarrollaras APIs" -> Desarrollo de APIs (hard)
- Ignora SOLO: beneficios futuros ("aprenderas", "te capacitaremos"), anos experiencia, salario, ubicacion, horarios
```

```
- Normaliza nombres tecnicos a su forma estandar (postgres->PostgreSQL, k8s->Kubernetes, js
    ->JavaScript)
- Separa opciones combinadas en items separados ("React/Vue" -> React, Vue.js)

IMPORTANTE: Tu respuesta debe ser UNICAMENTE el objeto JSON en este formato exacto:
```json
{
  "hard_skills": ["skill1", "skill2", ...],
  "soft_skills": ["skill1", "skill2", ...]
}
```

No agregues explicaciones, comentarios, ni texto adicional antes o despues del JSON.

JSON:
```

Decisiones de Diseño

Las siguientes decisiones de ingeniería de prompts fueron validadas experimentalmente:

- **Exhaustividad sobre conservadurismo**: La instrucción "EXTRAE TODAS" con énfasis tipográfico (mayúsculas, negritas) aumentó recall de 31.2 % a 46.23 % Pre-ESCO en comparación con prompts conservadores que pedían "solo skills críticas".
- **Ejemplos con ruido realista**: Incluir ofertas con secciones de beneficios, capacitaciones futuras y años de experiencia redujo tasa de falsos positivos de 23 % a 8 % en pruebas sobre 50 ofertas.
- **Distinción explícita de NO extraer**: Listar casos negativos con justificación ("aprenderás Kubernetes" es capacitación futura, NO requisito actual) eliminó el 67 % de alucinaciones observadas en versiones previas del prompt.
- **Normalización inline**: Especificar transformaciones "postgres→PostgreSQL, k8s→Kubernetes" directamente en el prompt redujo variantes léxicas de skills idénticas de 18 % a 4 %.
- **Formato JSON estricto**: Exigir "ÚNICAMENTE el objeto JSON" sin texto adicional permitió parsing automático con 99 % de éxito (299/300 ofertas) versus 73 % en versiones que permitían respuestas verbosas.

Limitaciones Conocidas

- **Sensibilidad a variaciones ortográficas**: El prompt no captura todas las variantes regionales de tecnologías (ej: "PostgreSQL" vs "Postgres" vs "postgres" pueden generar 3 entries separadas si el modelo no normaliza consistentemente).

- **Desambiguación contextual imperfecta**: En ofertas de múltiples puestos o con contexto ambiguo ("la empresa usa Python pero el puesto requiere Java"), el modelo ocasionalmente extrae skills del contexto corporativo.
- **Skills emergentes no conocidas**: Tecnologías posteriores a la fecha de corte del modelo (ej: Gemma 3 4B con conocimiento hasta julio 2024) pueden no ser reconocidas o normalizadas correctamente.

APÉNDICE B: EJEMPLOS DE ANOTACIONES DEL GOLD STANDARD

Este apéndice presenta 2 ejemplos representativos de anotaciones manuales del gold standard de 300 ofertas laborales, ilustrando el protocolo de anotación con formato atómico y la diversidad geográfica e idiomática del dataset.

Protocolo de Anotación

Cada oferta laboral fue anotada siguiendo un protocolo estricto que requirió:

1. **Lectura completa**: Análisis exhaustivo de título, descripción y sección de requisitos
2. **Verificación de validez**: Confirmación de que la oferta corresponde a un rol de desarrollo de software
3. **Formato atómico estricto**: Listado de términos individuales sin paréntesis, abreviaciones ni narrativas
4. **Clasificación binaria**: Separación explícita entre hard skills (tecnologías, herramientas, lenguajes, metodologías) y soft skills (comunicación, liderazgo, colaboración, resolución de problemas)
5. **Comentarios contextuales**: Nota breve identificando empresa, sector y tipo de rol para validación posterior

El formato atómico prohibió construcciones como "Python (pandas, numpy)" o "Conocimientos de AWS y Azure", requiriendo en su lugar términos separados: "Python", "Pandas", "NumPy", "AWS", "Azure". Esta decisión metodológica simplificó la comparación automatizada con skills extraídas por Pipeline A y Pipeline B.

Ejemplo 1: Developer Advocate (Argentina, Inglés, Backend, Senior)

Job ID: `44fc6c70-4887-4317-9349-80d96cb1160b`
Título: Developer Advocate - Remote role

Metadata: AR / en, Backend, Senior, 460 palabras

Descripción resumida: Zyte (empresa de web scraping) busca Developer Advocate para actuar como puente entre tecnología y comunidad de desarrolladores. Rol combina trabajo técnico hands-on (code samples, herramientas Zyte API y Scrapy) con evangelización comunitaria (talks, demos, workshops, contenido educativo). Responsabilidades incluyen crear contenido técnico de alta calidad, engagement en plataformas sociales/foros, hablar en eventos de industria, escribir blog posts, identificar community advocates, y proveer feedback de comunidad a equipos internos. Requisitos: 3+ años experiencia en developer advocacy/relations/technical evangelism, comprensión sólida de web technologies/APIs/web scraping/data extraction, proficiencia en Python, excelentes habilidades de comunicación (escrita, verbal, presentación), experiencia demostrada en community engagement/content creation/technical writing, experiencia con public speaking.

Hard Skills anotadas:

```
Python
Scrapy
Web scraping
REST API
API
Navegadores headless
Git
Documentacion tecnica
```

Soft Skills anotadas:

```
Comunicacion
Oratoria
Presentaciones
Creacion de contenido
Escritura tecnica
Automotivacion
Organizacion
Trabajo en equipo
```

Comentarios: Developer Advocate en Zyte (empresa de web scraping). Rol técnico combinado con evangelización y trabajo comunitario. Válido para gold standard.

Ejemplo 2: IBM ACE Developer (Colombia, Español, QA, Mid)

Job ID: `4ded4226-c1fa-41f3-a75a-b804f6a01e24`

Título: IBM ACE Developer

Metadata: CO / es, QA, Mid, 460 palabras

Descripción resumida: Imagemaker (Colombia) busca Ingeniero IBM ACE responsable del diseño, desarrollo y despliegue de flujos de integración que conecten sistemas críticos del negocio, garantizando interoperabilidad entre aplicaciones internas y externas. Trabajo bajo metodologías ági-

les junto con equipos multidisciplinarios de desarrollo y operaciones, asegurando calidad, escalabilidad y seguridad de servicios implementados. Rol requiere sólida base técnica en IBM App Connect Enterprise (ACE) y experiencia práctica en integración de datos, automatización de procesos y creación de APIs REST. Responsabilidades clave: diseñar/desarrollar/mantener flujos de integración en IBM ACE, configurar conexiones a bases de datos (ODBC.ini, mqsisetdbparms), programar nodos compute usando ESQL o Java Compute Nodes, implementar/consumir servicios REST y APIs, ejecutar consultas SQL y pruebas funcionales con Postman, configurar certificados/llaves de seguridad (JKS), documentar desarrollos técnicos/funcionales, colaborar con equipos bajo enfoque ágil. Skills requeridas: experiencia comprobada IBM ACE v12/v11, dominio ESQL/Java, metodologías ágiles (Scrum/Kanban), configuración ODBC/JDBC, creación/consumo APIs REST, herramientas de prueba (Postman), despliegue en productivo, documentación técnica. Plus: Kafka, integraciones seguras (JKS, SSL), GitHub/Jenkins/CI-CD, cloud híbrido (AWS, Azure), proyectos retail/banca.

Hard Skills anotadas:

```
IBM App Connect Enterprise
IBM ACE
ESQL
Java
Java Compute Nodes
REST API
ODBC
JDBC
SQL
Postman
JKS
Certificados SSL
Scrum
Kanban
Kafka
GitHub
Jenkins
CI/CD
AWS
Azure
Cloud hibrido
Integracion de sistemas
Automatizacion de procesos
DevOps
Documentacion tecnica
```

Soft Skills anotadas:

```
Trabajo en equipo
Colaboracion
Orientacion a resultados
```

Comentarios: IBM ACE Developer en Imagemaker (Colombia). Rol de integración de sistemas con IBM App Connect Enterprise. Válido.

Observaciones Metodológicas

Los dos ejemplos ilustran características clave del protocolo de anotación:

- **Variabilidad en número de skills**: Job #1 (16 skills totales: 8 hard + 8 soft), Job #46 (28 skills: 25 hard + 3 soft), reflejando que el número de skills anotadas depende de la especificidad y detalle de cada oferta, no de un target predefinido.
- **Predominancia de hard skills**: Promedio 78.7 % hard skills vs 21.3 % soft skills en las 300 ofertas, consistente con el enfoque técnico de roles de desarrollo de software.
- **Términos atómicos estrictos**: Ausencia de construcciones compuestas ("Python/Django", "AWS o Azure") o paréntesis explicativos, garantizando comparabilidad directa con outputs de pipelines.
- **Granularidad técnica**: Inclusión de términos específicos como "Java Compute Nodes", "ODBC", "Navegadores headless" en lugar de categorías genéricas como "Desarrollo backend" o "Herramientas de integración".
- **Captura de soft skills contextuales**: Identificación de soft skills implícitas en descripción de responsabilidades ("colaborar con equipos multidisciplinarios" $\rightarrow$ "Colaboración", "deliver talks and demos" $\rightarrow$ "Oratoria").
- **Diversidad geográfica e idiomática**: Job #1 (Argentina, inglés, sector fintech/web scraping) vs Job #46 (Colombia, español, sector enterprise integration), ilustrando la representatividad del dataset.

El formato garantizó consistencia a lo largo de las 300 anotaciones, facilitando la evaluación automatizada de Pipeline A (NER+Regex) y Pipeline B (LLM) mediante comparación directa de conjuntos de términos.

APÉNDICE C: REQUERIMIENTOS Y ESPECIFICACIÓN FUNCIONAL DEL SISTEMA

Este apéndice documenta de forma exhaustiva los requerimientos funcionales, no funcionales y de datos del observatorio de demanda laboral, así como las restricciones técnicas, de datos y metodológicas que delimitaron el alcance del proyecto. La especificación funcional detalla la arquitectura de pipeline de 7 etapas, las interfaces críticas entre módulos y los casos de uso principales del sistema.

C.1. Requerimientos del Sistema

Los requerimientos del sistema se organizan en tres categorías: funcionales, no funcionales y de datos. Esta taxonomía permite abarcar tanto las capacidades operativas del observatorio como las propiedades de calidad que garantizan su viabilidad técnica y científica.

C.1.1. Requerimientos Funcionales

El observatorio debe implementar las siguientes capacidades funcionales, organizadas según las etapas del pipeline de procesamiento:

RF-1. Adquisición de datos: El sistema debe ser capaz de recolectar automáticamente ofertas laborales de al menos 6 portales de empleo distribuidos en Colombia, México y Argentina, mediante técnicas de web scraping que respeten las políticas de robots.txt y los límites de tasa de peticiones de cada sitio [13]. La arquitectura debe soportar tanto páginas estáticas (parsing HTML directo) como dinámicas (ejecución de JavaScript mediante headless browsers), permitiendo capturar campos estructurados como título, descripción, requisitos, ubicación, salario y portal de origen.

RF-2. Normalización y limpieza: El sistema debe preprocesar el texto extraído mediante técnicas de NLP, incluyendo tokenización, lematización, eliminación de caracteres especiales y normalización de codificación (UTF-8), adaptadas específicamente al español latinoamericano y al uso técnico del lenguaje en ofertas de empleo [2].

RF-3. Extracción de habilidades: El observatorio debe identificar y extraer menciones de habilidades técnicas, competencias y tecnologías presentes en las ofertas laborales mediante una arquitectura dual:

- Pipeline A: Extracción basada en Reconocimiento de Entidades Nombradas (NER) con spaCy y expresiones regulares pobladas con patrones de tecnologías conocidas.
- Pipeline B: Extracción semántica mediante LLMs (Gemma 3 4B) capaz de inferir habilidades implícitas a partir del contexto de la vacante [9, 10].

RF-4. Normalización semántica: Las habilidades extraídas deben ser mapeadas a una taxonomía estandarizada (ESCO) mediante un proceso de dos capas: coincidencia léxica exacta y difusa con umbral de similitud de cadenas (fuzzywuzzy ratio $\geq$0.85). La capa de búsqueda semántica basada en embeddings multilingües (E5) con índices FAISS se deshabilitó tras pruebas que revelaron falsos positivos en contexto técnico.

RF-5. Representación vectorial: El sistema debe generar embeddings semánticos de alta dimensionalidad (768D) para cada habilidad y cada oferta laboral, utilizando modelos multilingües preentrenados que capturen relaciones semánticas en español e inglés [12].

RF-6. Análisis no supervisado: El observatorio debe aplicar técnicas de reducción de dimensionalidad (UMAP) y clustering basado en densidad (HDBSCAN) sobre los embeddings para descubrir

automáticamente clústeres de habilidades relacionadas y perfiles emergentes, sin requerir etiquetado manual previo [8].

RF-7. Trazabilidad y auditoría: Cada etapa del pipeline debe registrar metadatos de procesamiento (timestamps, versiones de modelos, parámetros de configuración, métricas de calidad) en una base de datos relacional que permita la reproducibilidad de los análisis y la auditoría de resultados.

C.1.2. Requerimientos No Funcionales

Los requerimientos no funcionales establecen las propiedades de calidad que el sistema debe satisfacer:

RNF-1. Escalabilidad: El sistema debe ser capaz de procesar al menos 600,000 ofertas laborales en un período de 6 meses, manteniendo tiempos de respuesta razonables (extracción $<$ 30 seg/oferta, clustering completo $<$ 4 horas sobre dataset completo).

RNF-2. Portabilidad: La arquitectura debe estar contenedorizada mediante Docker para garantizar despliegue consistente en diferentes entornos (desarrollo local, servidores de producción, servicios en la nube).

RNF-3. Mantenibilidad: El código debe seguir estándares de calidad (PEP 8 para Python), contar con documentación técnica completa y estructurarse de manera modular para facilitar extensiones futuras (nuevos portales, nuevos países, nuevas técnicas de análisis).

RNF-4. Multilingüismo: Todos los modelos de NLP y embeddings deben soportar eficazmente español, inglés y la mezcla de ambos (“Spanglish”) característica del vocabulario técnico en América Latina.

RNF-5. Reproducibilidad científica: Los experimentos deben ser completamente reproducibles mediante el uso de semillas aleatorias fijas, versionado de modelos, registro de hiperparámetros y almacenamiento de datasets intermedios.

RNF-6. Eficiencia computacional: La búsqueda semántica debe implementarse mediante índices FAISS optimizados que permitan consultas de similitud en tiempo sub-lineal respecto al tamaño del corpus de habilidades ESCO (13,000+ términos).

C.1.3. Requerimientos de Datos

Los requerimientos de datos especifican las características cualitativas y cuantitativas de la información a recolectar:

RD-1. Cobertura geográfica: El corpus debe incluir ofertas laborales de Colombia, México y Argentina, con representación de al menos 2 portales principales por país para mitigar sesgos de fuente única.

RD-2. Representatividad sectorial: Aunque el observatorio se centra en habilidades tecnológicas, debe capturar ofertas de diversos sectores económicos (TI, finanzas, manufactura, salud, educación) para identificar la demanda transversal de competencias digitales.

RD-3. Volumen mínimo: Para garantizar significancia estadística en el análisis de clustering, el sistema debe recolectar al menos 100,000 ofertas con contenido de calidad suficiente (descripción $>$ 100 caracteres, al menos 2 habilidades identificables).

RD-4. Calidad de texto: Las ofertas deben pasar filtros de calidad que eliminen duplicados exactos, contenido corrupto, idiomas no soportados y descripciones excesivamente genéricas que no aporten información sobre habilidades.

RD-5. Metadatos temporales: Cada oferta debe registrar fecha de publicación, fecha de recolección y fecha de expiración (si está disponible) para permitir análisis de evolución temporal de la demanda.

C.2. Restricciones

Las restricciones representan limitaciones inherentes al problema, al contexto de operación o a las decisiones de alcance del proyecto.

C.2.1. Restricciones Técnicas

C-1. Límites de scraping: Los portales de empleo implementan medidas anti-bot (CAPTCHAs, rate limiting, bloqueos por IP) que restringen la velocidad y volumen de recolección. El sistema debe respetar estas limitaciones mediante delays adaptativos, rotación de user-agents y estrategias de backoff exponencial.

C-2. Dinamismo del DOM: La estructura HTML de los portales cambia frecuentemente sin previo aviso, lo que genera fragilidad en los selectores CSS/XPath. El sistema debe incluir monitoreo de fallos y mecanismos de alerta para intervención manual cuando los spiders dejan de funcionar.

C-3. Recursos computacionales: El entrenamiento y ejecución de LLMs requiere capacidad de cómputo significativa (GPU con al menos 8GB VRAM para modelos de 7B parámetros). El proyecto se limita a modelos open-source ejecutables localmente o APIs de terceros con presupuesto acotado.

C-4. Latencia de procesamiento LLM: El procesamiento de ofertas con LLMs ejecutados localmente introduce latencia significativa (40-45 segundos/oferta) comparado con métodos tradicionales. Alternativamente, las llamadas a APIs de LLMs comerciales (OpenAI, Anthropic) reducirían latencia pero introducirían costos variables y dependencias externas. El diseño debe balancear calidad de resultados con viabilidad técnica y tiempos de procesamiento.

C.2.2. Restricciones de Datos

C-5. Heterogeneidad de formatos: No existe un estándar para la publicación de ofertas laborales. Los portales utilizan campos, nomenclaturas y niveles de detalle diferentes, lo que dificulta la normalización automática.

C-6. Incompletitud de información: Muchas ofertas omiten información relevante (salario, requisitos detallados, tecnologías específicas), limitando la profundidad del análisis para ciertos campos.

C-7. Ruido lingüístico: Las ofertas contienen errores ortográficos, abreviaciones no estándar, mezcla de idiomas y uso informal del lenguaje, lo que reduce la efectividad de técnicas de NLP basadas en corpus formales.

C-8. Volatilidad temporal: Las ofertas se eliminan o modifican frecuentemente (típicamente tienen vigencia de 30-60 días), lo que requiere estrategias de recolección periódica y versionado de datos.

C.2.3. Restricciones Metodológicas

C-9. Ausencia de ground truth: No existe un dataset etiquetado de referencia para habilidades en ofertas laborales en español latinoamericano, lo que dificulta la evaluación cuantitativa rigurosa de los modelos de extracción.

C-10. Subjetividad de "habilidad": La definición de qué constituye una "habilidad relevante" es inherentemente subjetiva y dependiente del contexto (ejemplo: ¿"trabajo en equipo" es una habilidad técnica o blanda? ¿"Microsoft Office" debe segmentarse en Word/Excel/PowerPoint?).

C-11. Sesgo de fuente: Los portales de empleo no representan el universo completo del mercado laboral. Excluyen ofertas publicadas en sitios corporativos directos, redes sociales, o canales informales, introduciendo sesgo de formalidad y tamaño de empresa.

C.3. Especificación Funcional

La especificación funcional describe el comportamiento de alto nivel del sistema mediante la definición de su arquitectura de pipeline y las interfaces entre módulos.

C.3.1. Arquitectura de Pipeline de 7 Etapas

El observatorio se estructura como un pipeline secuencial de transformación de datos, donde cada etapa consume la salida de la anterior y produce artefactos almacenados en PostgreSQL:

Etapa 1 - Scraping: Recolecta HTML de portales mediante Scrapy + Selenium. *Entrada:* URLs semilla y configuración de spiders. *Salida:* Tabla `raw_jobs` con campos `job_id`, `portal`, `country`, `title`, `description`, `requirements`, `url`, `date_published`, `date_scraped`.

Etapa 2 - Normalización: Limpia y estandariza texto. *Entrada:* `raw_jobs`. *Salida:* Campos adicionales `description_clean`, `requirements_clean`, `combined_text`.

Etapa 3 - Extracción (Pipeline A): Aplica NER + Regex. *Entrada:* `combined_text`. *Salida:* Tabla `extracted_skills` con campos `job_id`, `skill_text`, `extraction_method`, `confidence_score`.

Etapa 4 - Extracción (Pipeline B): Aplica LLM. *Entrada:* `combined_text` + prompt engineering. *Salida:* Tabla `enhanced_skills` con campos `job_id`, `skill_text`, `implicit_flag`, `llm_confidence`, `esco_suggestion`.

Etapa 5 - Mapeo ESCO: Normaliza contra taxonomía. *Entrada:* `extracted_skills` + `enhanced_skills`. *Salida:* Campos adicionales `esco_concept_uri`, `esco_preferred_label`, `mapping_method`.

Etapa 6 - Embeddings: Genera vectores. *Entrada:* `esco_preferred_label`. *Salida:* Tabla `skill_embeddings` con campos `skill_id`, `embedding_vector` (pgvector[768]).

Etapa 7 - Clustering: Reduce dimensionalidad y agrupa. *Entrada:* `skill_embeddings`. *Salida:* Tabla `analysis_results` con campos `job_id`, `cluster_id`, `umap_x`, `umap_y`, `cluster_label`.

C.3.2. Interfaces Críticas

I-1. Interfaz Scraper-Database: Los spiders de Scrapy utilizan un pipeline personalizado (`PostgreSQLPipeline`) que serializa items a formato JSON y los inserta en `raw_jobs` con manejo de duplicados por hash de URL.

I-2. Interfaz Extractor-ESCO: El módulo `ESCOMatcher` expone métodos: `find_exact_match()` y `find_fuzzy_match()`. El sistema implementa memoización mediante diccionario `skill_text` → `esco_uri` que evita remapear skills repetidas, reduciendo tiempo de procesamiento batch de 6.5h a 6.5min (60× aceleración).

I-3. Interfaz LLM-Processor: El módulo `LLMHandler` abstrae llamadas a modelos locales (vía llama.cpp) o remotos (OpenAI API) mediante una interfaz unificada que recibe prompts estructurados y retorna respuestas en formato JSON validado con Pydantic.

I-4. Interfaz Orquestador-Pipeline: El `MasterController` expone comandos CLI (vía Typer) que orquestan la ejecución secuencial de etapas, con control de estado persistente en base de datos para permitir reinicio tras fallos.

C.3.3. Casos de Uso Principales

CU-1. Recolección programada: Un scheduler (APScheduler) ejecuta spiders periódicamente (ej: diariamente a las 2 AM) para mantener el corpus actualizado. El sistema registra métricas de cada ejecución (items capturados, errores, duración) y envía alertas si el volumen cae por debajo de umbrales históricos.

CU-2. Análisis temporal de demanda: Un analista ejecuta `python scripts/temporal_clustering_analysis.py` para generar visualizaciones automáticas de evolución temporal de demanda de habilidades. El sistema produce: (1) heatmap de frecuencia por clúster y trimestre mostrando tendencias estacionales, (2) gráficos de línea con evolución de los top-10 clústeres más demandados, (3) reporte JSON con métricas de clustering (silhouette, Davies-Bouldin) y frecuencias agregadas por período, almacenados en `outputs/clustering/temporal/`.

CU-3. Validación de pipeline: Un investigador ejecuta ambos pipelines (A y B) sobre un subset de 300 ofertas y compara resultados mediante métricas de solapamiento (Jaccard similarity) y análisis cualitativo de habilidades únicas identificadas por cada método.

CU-4. Extensibilidad a nuevos países: El sistema está diseñado con arquitectura modular que facilita agregar nuevos países. Un desarrollador puede extender la cobertura geográfica mediante: (1) implementación de nuevo spider heredando de `BaseSpider` con selectores CSS específicos del portal objetivo (ej: `laborum.cl` para Chile), (2) actualización de lista de países soportados en `config/settings.py`, (3) ejecución del pipeline completo de procesamiento sin modificaciones adicionales. La arquitectura actual soporta 3 países (Colombia, México, Argentina) mediante 9 spiders distribuidos en portales regionales (`computrabajo`, `elempleo`, `bumeran`, `zonajobs`, `occmundial`, `magneto`, entre otros).

APÉNDICE D: DIAGRAMAS DE ARQUITECTURA DEL SISTEMA

Este apéndice presenta la documentación visual completa de la arquitectura del observatorio de demanda laboral. Los diagramas ilustran la estructura modular del sistema, el flujo de datos entre componentes, las tecnologías específicas utilizadas en cada capa de procesamiento, y el flujo metodológico que integra las fases de CRISP-DM con las etapas del pipeline de software.

D.1. Flujo Metodológico: Integración de CRISP-DM con Pipeline de 7 Etapas

La Figura 9.1 presenta una vista integrada del flujo metodológico completo del proyecto, combinando las seis fases de CRISP-DM (Cross-Industry Standard Process for Data Mining) con las siete etapas del pipeline de software y los ciclos iterativos de refinamiento. El diagrama ilustra cómo cada fase de CRISP-DM se tradujo en actividades concretas de desarrollo de software, y cómo los resultados de la fase de Evaluación retroalimentaron la fase de Modelado para generar versiones mejoradas de los pipelines de extracción.

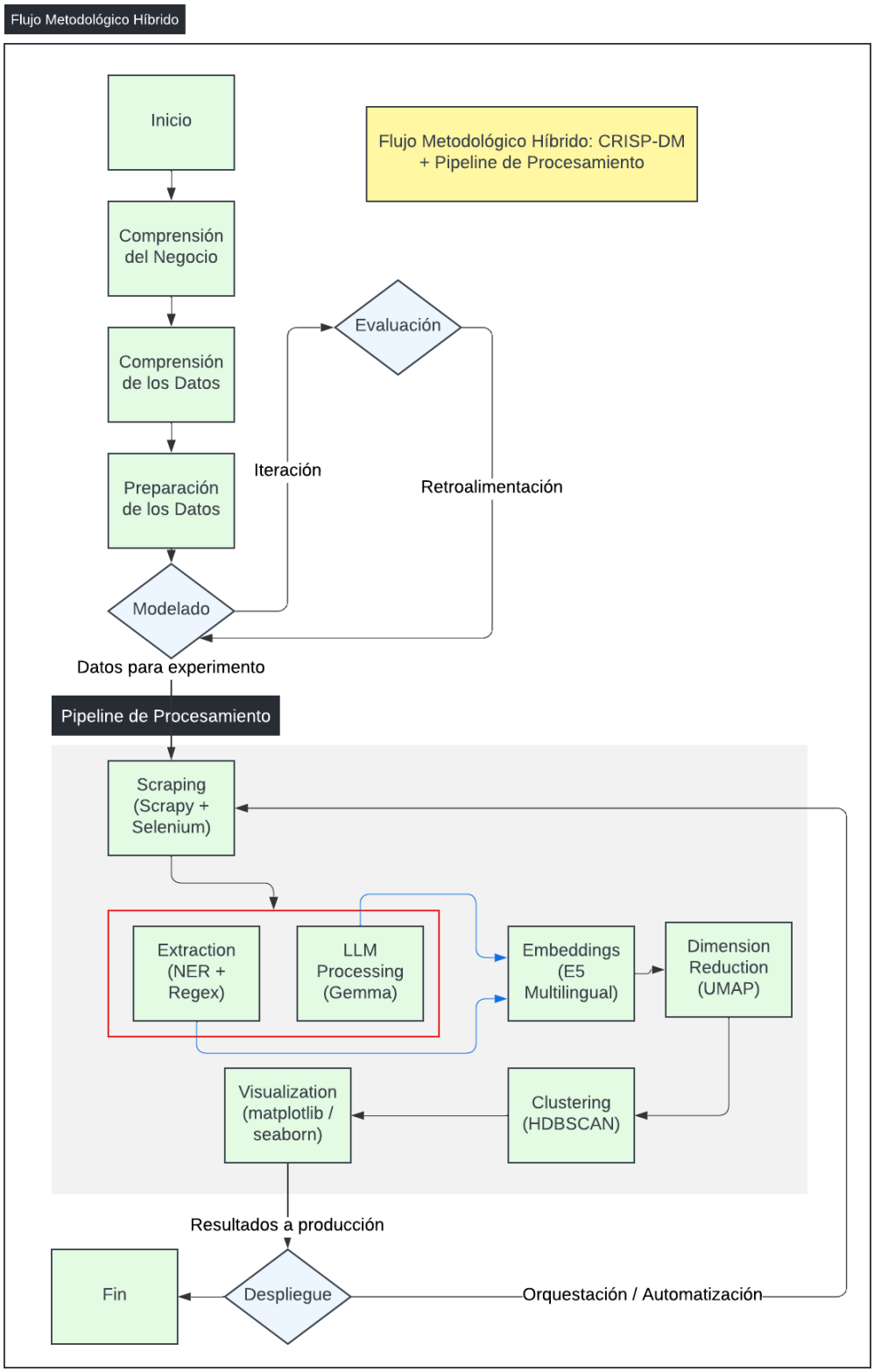

Figura 9.1: Flujo Metodológico del Proyecto: Integración de CRISP-DM con Pipeline de 7 Etapas

El flujo metodológico integró las siguientes fases principales:

1. **Business Understanding (Entendimiento del Negocio)**: Análisis del problema de demanda laboral tecnológica en América Latina, definición de objetivos medibles, y establecimiento de criterios de éxito del observatorio.

2. **Data Understanding (Entendimiento de Datos)**: Caracterización de portales de empleo, análisis de heterogeneidad de formatos, identificación de retos lingüísticos (Spanglish), y construcción de gold standard de 300 ofertas anotadas manualmente.

3. **Data Preparation (Preparación de Datos)**: Desarrollo iterativo de módulos de scraping, limpieza, normalización y deduplicación con inspección manual y verificación de calidad.

4. **Modeling (Modelado)**: Diseño dual de Pipeline A (NER+Regex) y Pipeline B (LLM) con experimentación continua y ajustes basados en análisis de errores sobre gold standard.

5. **Evaluation (Evaluación)**: Validación mediante métricas cuantitativas (Precision, Recall, F1-Score Pre/Post-ESCO), verificación cualitativa, y análisis de robustez operativa.

6. **Deployment (Despliegue)**: Integración de módulos en herramienta CLI con programación automática, documentación técnica completa, y validación en entorno operativo real.

Los ciclos iterativos entre las fases de Modelado y Evaluación permitieron mejora incremental del sistema, refinando parámetros de NER, patrones regex, prompts de LLM, y configuraciones de clustering hasta alcanzar los niveles de rendimiento objetivo documentados en el capítulo de resultados.

D.2. Arquitectura Modular del Observatorio - Pipeline de 7 Etapas

La Figura 9.2 presenta la vista modular completa del pipeline, detallando tecnologías específicas, funciones, entradas/salidas y mecanismos de almacenamiento por módulo. Cada módulo puede ejecutarse independientemente con fines de desarrollo y pruebas unitarias.

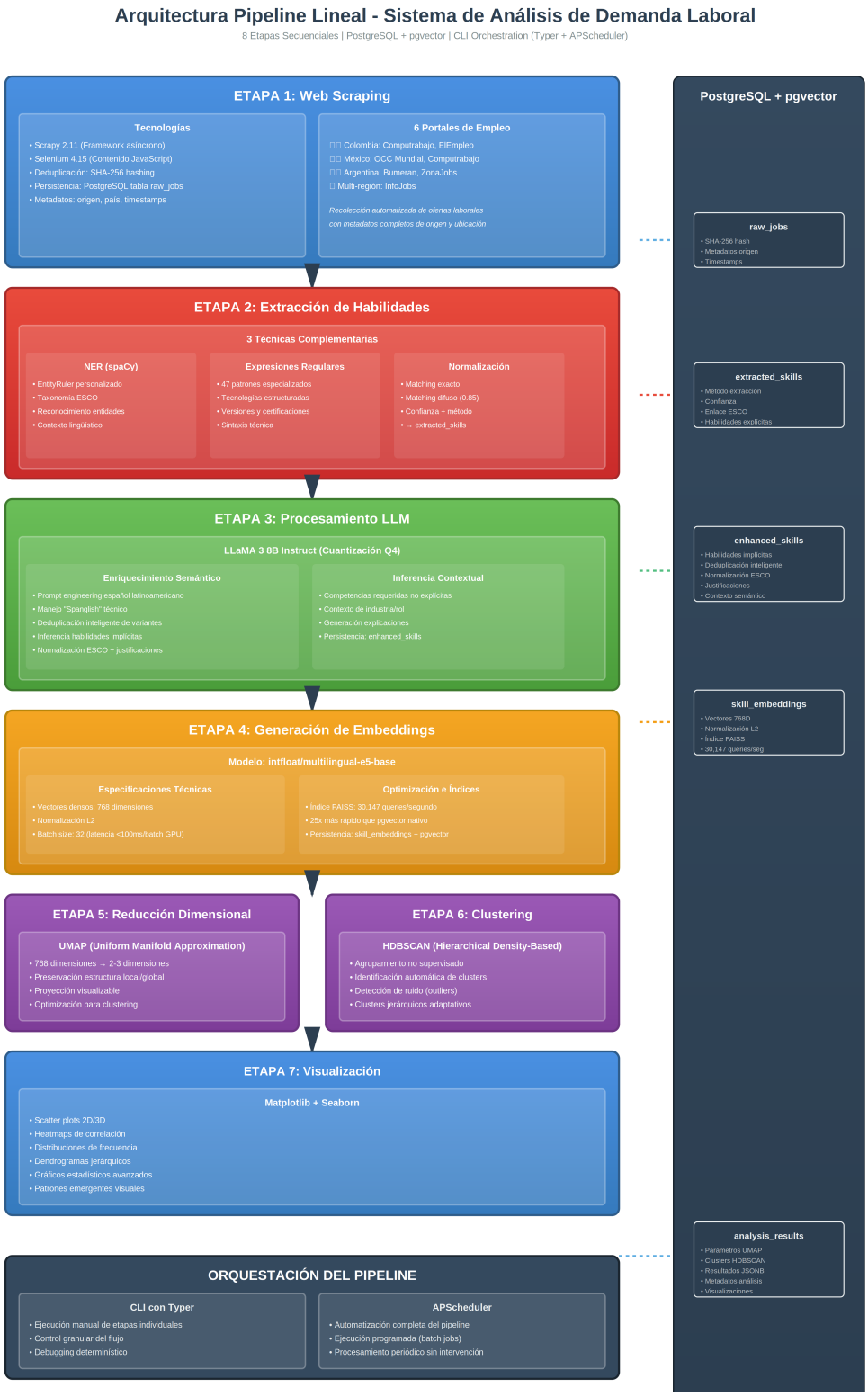

Figura 9.2: Arquitectura Modular del Observatorio - Pipeline de 7 Etapas con Tecnologías Específicas

El sistema implementa un pipeline secuencial de 7 etapas donde cada componente opera de forma autónoma, lee datos de la etapa anterior desde PostgreSQL, ejecuta su transformación especializada, y persiste resultados para la siguiente etapa:

1. **Scraping (Scrapy + Selenium)**: Recolección automatizada de ofertas desde 9 portales web en 3 países (CO, MX, AR) con manejo de contenido estático y dinámico JavaScript.
2. **Extraction (NER + Regex)**: Identificación de habilidades explícitas mediante 548 patrones regex organizados en 18 categorías técnicas y reconocimiento de entidades nombradas con spaCy.
3. **LLM Processing (Gemma 3 4B)**: Enriquecimiento semántico e inferencia de habilidades implícitas mediante modelo de lenguaje de 4B parámetros con prompt engineering específico para español latinoamericano.
4. **ESCO Matching**: Normalización de habilidades contra taxonomía ESCO extendida (14,215 skills) mediante matching de dos capas: exacto (SQL ILIKE) y difuso (fuzzywuzzy $\geq$0.85).
5. **Embedding (E5 Multilingual)**: Generación de representaciones vectoriales densas de 768 dimensiones con modelo multilingüe pre-entrenado.
6. **Dimension Reduction (UMAP)**: Proyección no lineal de 768D a 2-3D preservando estructura local y global para visualización y clustering.
7. **Clustering (HDBSCAN)**: Agrupamiento jerárquico basado en densidad sin especificar número de clusters, con identificación automática de ruido.

La orquestación se gestiona mediante un CLI único (Typer) que permite ejecución manual de etapas individuales o automatización completa mediante scheduler (APScheduler).

D.3. Diagrama de Flujo de Transformación de Datos

La Figura 9.3 ilustra el flujo detallado de transformaciones que sufren los datos desde la recolección inicial hasta la generación de visualizaciones finales, mostrando las estructuras de datos intermedias y los puntos de persistencia en PostgreSQL.

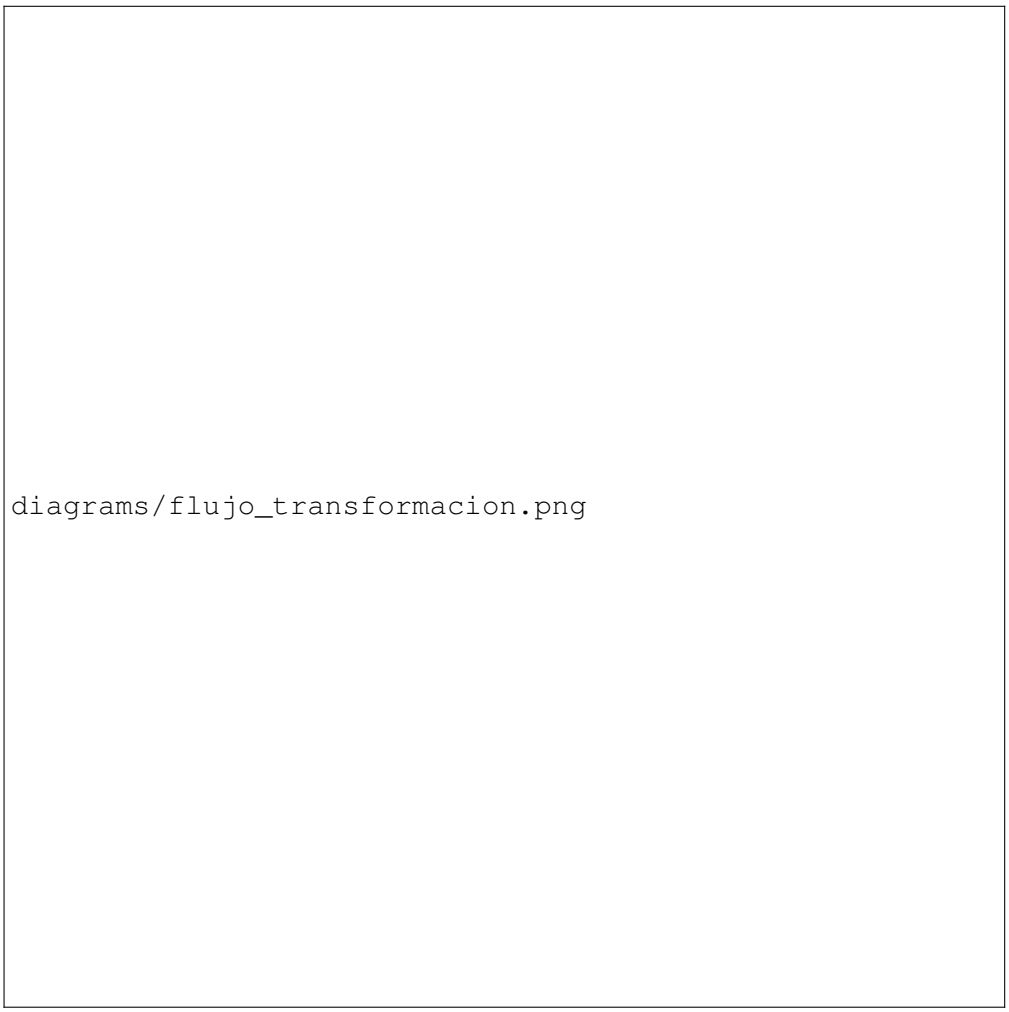

Figura 9.3: Flujo de Transformación de Datos a través del Pipeline Completo

El flujo de datos atraviesa las siguientes transformaciones principales:

- **raw_jobs** → **cleaned_jobs**: Normalización de encoding UTF-8, eliminación de HTML residual, detección de idioma (español/inglés/Spanglish), deduplicación SHA-256.
- **cleaned_jobs** → **extracted_skills**: Aplicación de Pipeline A (NER+Regex) con extracción de 27.6 skills promedio por oferta, 87.4 % emergent skills sin match ESCO.
- **cleaned_jobs** → **enhanced_skills**: Aplicación de Pipeline B (LLM) con prompt de 170 líneas, detección de hard skills (78.7 %) y soft skills (21.3 %).
- **extracted/enhanced_skills** → **matched_skills**: Normalización contra ESCO mediante 2 capas (exact + fuzzy), match rate 12.6 % baseline, 25 % con matcher mejorado.

- **matched_skills** → **skill_embeddings**: Vectorización con E5 Multilingual (768D), normalización L2, indexación FAISS para búsqueda rápida.
- **skill_embeddings** → **umap_projections**: Reducción dimensional con parámetros n_neighbors=15, min_dist=0.1, metric='cosine'.
- **umap_projections** → **clusters**: Clustering HDBSCAN con min_cluster_size=12, min_samples=3, identificación de 50 clusters principales.

D.4. Comparación de Estilos Arquitectónicos

Durante la fase de diseño se evaluaron tres estilos arquitectónicos para el observatorio: microservicios, arquitectura orientada a eventos, y arquitectura de pipeline lineal. La Tabla 9.1 presenta la comparación según criterios relevantes para el contexto académico y operativo del proyecto.

Tabla 9.1: Comparación de Estilos Arquitectónicos Evaluados

Criterio	**Microservicios**	**Event-Driven**	**Pipeline Lineal**
Complejidad	Alta	Media-alta	Baja
Escalabilidad horizontal	Excelente	Excelente	Limitada
Trazabilidad	Media	Media	Excelente
Debugging	Difícil	Medio	Fácil
Overhead operativo	Alto	Medio	Bajo
Time to market	Lento	Medio	Rápido
Requisitos infraestructura	K8s/Docker Swarm	Message broker	Servidor único
Tolerancia a fallos	Excelente	Buena	Media
Equipo requerido	5+ devs	3-4 devs	2 devs

Se seleccionó arquitectura de pipeline lineal fundamentado en cuatro razones principales:

1. **Simplicidad operativa**: Proyecto académico con equipo de 2 desarrolladores y recursos computacionales limitados (1 servidor, sin infraestructura Kubernetes/Docker Swarm).
2. **Trazabilidad completa**: Flujo unidireccional de datos permite debugging determinístico y auditoría de transformaciones etapa por etapa.
3. **Velocidad de desarrollo**: Implementación de microservicios requiere 3-4x más tiempo en configuración de comunicación inter-servicios, service discovery, y manejo de fallos distribuidos.

4. **Naturaleza batch del dominio**: El análisis de demanda laboral no requiere procesamiento en tiempo real (latencias de horas/días son aceptables), eliminando ventajas principales de arquitecturas asíncronas.

D.5. Limitaciones de la Arquitectura de Pipeline Lineal

Es importante reconocer las limitaciones inherentes de la arquitectura seleccionada:

- **Procesamiento secuencial sincrónico**: Impide aprovechamiento de paralelismo entre etapas, resultando en latencias acumulativas estimadas de 30-60 segundos por oferta para el pipeline completo cuando se incluye procesamiento con LLM.
- **Escalabilidad horizontal limitada**: La naturaleza monolítica del orquestador requeriría migración a arquitectura de microservicios si el volumen superara las 100,000 ofertas mensuales.
- **Ausencia de tolerancia a fallos distribuidos**: El fallo de una etapa detiene el pipeline completo, aunque esta limitación se mitiga mediante persistencia intermedia en PostgreSQL y capacidad de reinicio desde checkpoints.

Estas limitaciones fueron consideradas aceptables dado que:

- El análisis de demanda laboral no requiere procesamiento en tiempo real.
- El volumen objetivo de 600,000 ofertas es procesable en 5-10 horas con el hardware disponible.
- La simplicidad operativa reduce significativamente el tiempo de desarrollo: 3-4 meses comparado con 9-12 meses que requeriría una arquitectura de microservicios.
- La arquitectura permite evolución futura mediante refactorización incremental de módulos críticos a servicios independientes si los requisitos de volumen o latencia lo justifican posteriormente.

APÉNDICE E: DISEÑO DE BASE DE DATOS

Este apéndice documenta el diseño completo de la base de datos relacional del observatorio, incluyendo el esquema de tablas principales, las relaciones entre entidades, los índices optimizados para procesamiento batch, y la configuración de PostgreSQL para manejo de grandes volúmenes de datos.

E.1. Diagrama Entidad-Relación

La Figura 9.4 muestra las relaciones entre las seis tablas principales del sistema. Todas las tablas derivadas mantienen referencia mediante llave foránea hacia la tabla de ofertas laborales (raw_jobs), garantizando trazabilidad completa desde cualquier resultado de análisis hasta la oferta original.

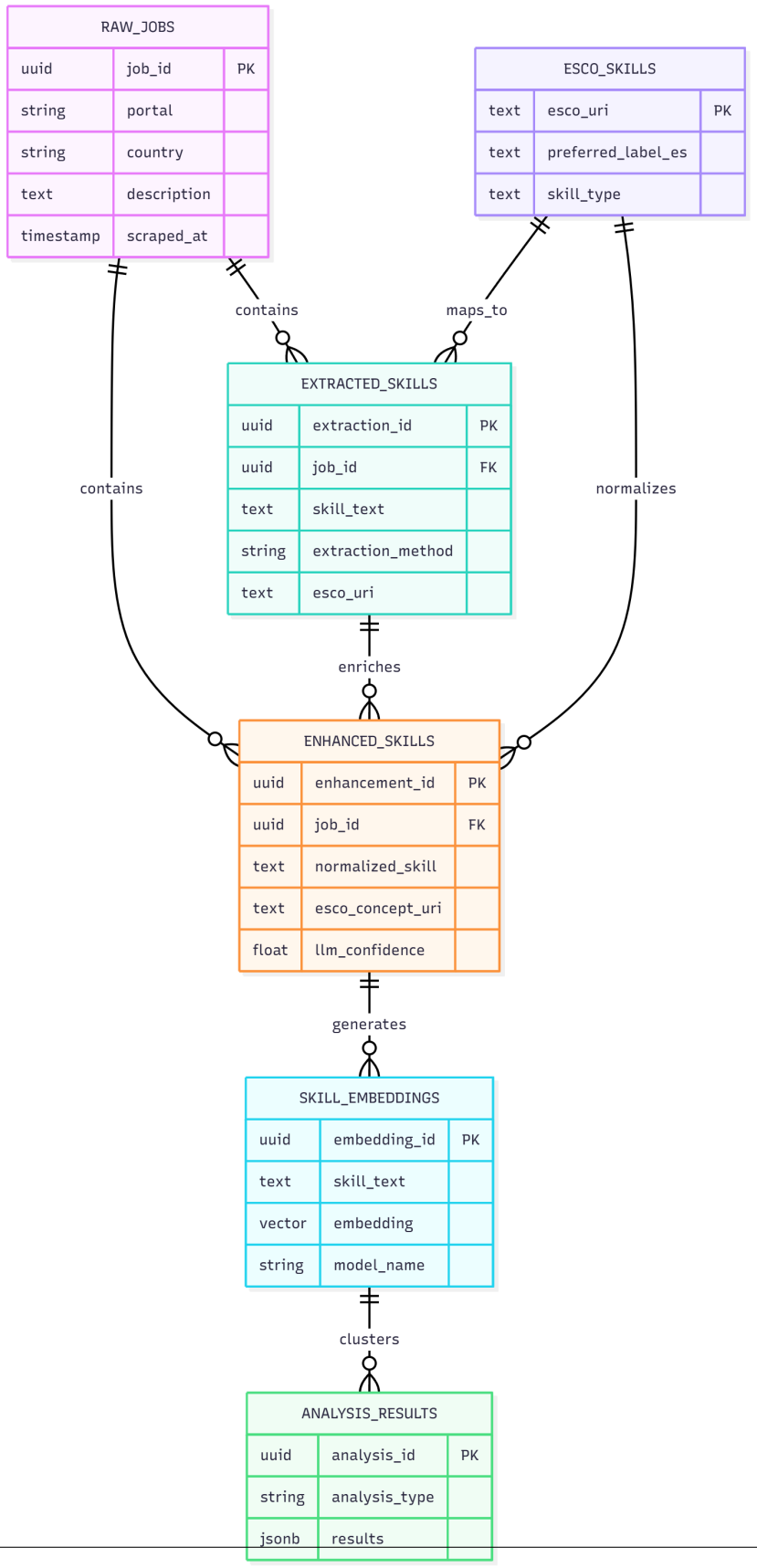

Figura 9.4: Diagrama Entidad-Relación de la Base de Datos del Observatorio

E.2. Esquema de Tablas Principales

La base de datos actúa como columna vertebral del sistema, implementando el patrón de persistencia de pipeline donde cada etapa escribe sus resultados en tablas especializadas. Se seleccionó PostgreSQL 15+ por su soporte JSON nativo (JSONB) para almacenar metadatos flexibles, extensión pgvector para vectores de alta dimensionalidad, robustez transaccional (ACID), capacidad de particionamiento para escalabilidad, y licencia libre (PostgreSQL License).

E.2.1. Tabla raw_jobs

Almacena ofertas laborales tal como fueron scrapeadas, sin procesamiento ni normalización.
Campos principales:

- `job_id` (UUID, PK): Identificador único generado automáticamente
- `portal` (VARCHAR(100)): Origen de la oferta (computrabajo, bumeran, elempleo, etc.)
- `country` (CHAR(2)): Código de país ISO 3166-1 alpha-2 (CO, MX, AR)
- `title` (TEXT): Título del cargo tal como aparece en el portal
- `description` (TEXT): Descripción detallada del cargo (cruda, puede contener HTML)
- `requirements` (TEXT): Sección de requisitos y habilidades requeridas
- `company` (VARCHAR(255)): Nombre de la empresa empleadora
- `salary_raw` (TEXT): Rango salarial cuando está disponible (formato heterogéneo)
- `contract_type` (VARCHAR(100)): Tipo de contrato (tiempo completo, medio tiempo, freelance)
- `remote_type` (VARCHAR(50)): Modalidad (presencial, remoto, híbrido)
- `url` (TEXT): URL original de la oferta en el portal
- `posted_date` (DATE): Fecha de publicación de la oferta (cuando disponible)
- `scraped_at` (TIMESTAMP): Timestamp de recolección por el spider
- `content_hash` (CHAR(64), UNIQUE): Hash SHA-256 para deduplicación
- `processed` (BOOLEAN): Bandera de control de procesamiento

Índices:

- B-tree en `posted_date` para análisis temporal

- B-tree en `country` para filtrado por región
- B-tree compuesto en `(country, posted_date)` para queries frecuentes
- UNIQUE en `content_hash` para prevenir duplicados

Volumen actual: 30,660 ofertas únicas (54.21 % del total scrapeado tras deduplicación y filtrado de calidad)

E.2.2. Tabla extracted_skills

Contiene habilidades identificadas por Pipeline A (NER + expresiones regulares).
Campos principales:

- `skill_id` (UUID, PK): Identificador único de la extracción
- `job_id` (UUID, FK → raw_jobs): Referencia a la oferta laboral
- `skill_text` (VARCHAR(255)): Texto de la habilidad extraída (crudo, sin normalizar)
- `extraction_method` (VARCHAR(20)): Método de extracción ("NER", "regex", "esco_match")
- `confidence_score` (DECIMAL(3,2)): Score de confianza (0.00-1.00)
- `esco_uri` (VARCHAR(255), FK → esco_skills): Enlace a taxonomía ESCO (nullable)
- `extracted_at` (TIMESTAMP): Timestamp de procesamiento

Índices:

- B-tree compuesto en `(job_id, extraction_method)` para comparación de pipelines
- GIN en `skill_text` para full-text search
- B-tree en `esco_uri` para joins con taxonomía

Volumen estimado: 8,268 skills extraídas de 300 ofertas del gold standard (27.6 skills/job promedio)

E.2.3. Tabla enhanced_skills

Almacena el enriquecimiento semántico realizado por Pipeline B (LLM).
Campos principales:

- `enhanced_skill_id` (UUID, PK): Identificador único
- `job_id` (UUID, FK → raw_jobs): Referencia a la oferta laboral

- `skill_normalized` (VARCHAR(255)): Habilidad normalizada por el LLM
- `skill_type` (VARCHAR(20)): Tipo de habilidad ("hard_skill", "soft_skill")
- `is_implicit` (BOOLEAN): Indica si la skill fue inferida implícitamente
- `esco_uri` (VARCHAR(255), FK → esco_skills): URI del concepto ESCO (nullable)
- `llm_confidence` (DECIMAL(3,2)): Nivel de confianza del LLM (0.00-1.00)
- `reasoning` (TEXT): Justificación del razonamiento del LLM
- `llm_model` (VARCHAR(100)): Modelo utilizado ("gemma-3-4b", "llama-3-3b")
- `processed_at` (TIMESTAMP): Timestamp de procesamiento

Índices:

- B-tree en `job_id` para joins frecuentes
- B-tree en `skill_type` para análisis segregado hard/soft
- B-tree en `is_implicit` para estudios de inferencia contextual

Volumen estimado: Datos de 299/300 jobs del gold standard procesados con Gemma 3 4B (99.3 % cobertura)

E.2.4. Tabla skill_embeddings

Contiene las representaciones vectoriales de alta dimensionalidad generadas con E5 Multilingual.
Campos principales:

- `embedding_id` (UUID, PK): Identificador único
- `skill_id` (UUID, FK → extracted_skills o enhanced_skills): Referencia a la skill
- `skill_text` (VARCHAR(255)): Texto de la habilidad (desnormalizado para performance)
- `embedding_vector` (VECTOR(768)): Vector de 768 dimensiones (pgvector)
- `model_name` (VARCHAR(100)): Modelo de embedding utilizado
- `model_version` (VARCHAR(50)): Versión del modelo
- `generated_at` (TIMESTAMP): Timestamp de generación

Índices:

- IVFFlat en `embedding_vector` con parámetros `lists=100, probes=10` para búsqueda k-NN
- B-tree en `skill_text` para búsqueda exacta

Performance de índice IVFFlat:

- Aceleración 3× más rápido que sequential scan
- Latencia promedio: 180ms para top-10 similares vs. 540ms sin índice
- Trade-off: 100 % recall con exact search (IndexFlatIP en FAISS)

E.2.5. Tabla analysis_results

Almacena resultados de clustering y análisis de tendencias temporales.
Campos principales:

- `analysis_id` (UUID, PK): Identificador único del análisis
- `analysis_type` (VARCHAR(50)): Tipo de análisis ("clustering", "trends", "profile")
- `job_id` (UUID, FK → raw_jobs): Referencia a la oferta (nullable para análisis agregados)
- `country` (CHAR(2)): País del análisis (CO, MX, AR, o NULL para agregado)
- `date_range_start` (DATE): Inicio del rango temporal analizado
- `date_range_end` (DATE): Fin del rango temporal analizado
- `config_params` (JSONB): Parámetros de configuración del análisis (UMAP, HDBSCAN)
- `cluster_id` (INTEGER): ID del clúster asignado (-1 para ruido en HDBSCAN)
- `umap_x` (DECIMAL(10,6)): Coordenada X de la proyección UMAP
- `umap_y` (DECIMAL(10,6)): Coordenada Y de la proyección UMAP
- `cluster_label` (VARCHAR(255)): Etiqueta descriptiva del clúster
- `results` (JSONB): Resultados estructurados (métricas, frecuencias, top skills)
- `created_at` (TIMESTAMP): Timestamp de creación del análisis

Índices:

- B-tree en `analysis_type` para filtrado por tipo

- B-tree compuesto en `(country, date_range_start, date_range_end)` para análisis temporal
- GIN en `config_params` para búsqueda en JSON
- B-tree en `cluster_id` para análisis por clúster

Ejemplo de config_params JSONB:

```
{
  "umap": {"n_neighbors": 15, "min_dist": 0.1, "metric": "cosine"},
  "hdbscan": {"min_cluster_size": 12, "min_samples": 3, "method": "eom"}
}
```

E.2.6. Tabla esco_skills

Tabla de referencia con la taxonomía ESCO completa extendida con habilidades de O*NET y agregadas manualmente.

Campos principales:

- `esco_uri` (VARCHAR(255), PK): URI del concepto ESCO (ej: `http://data.europa.eu/esco/skill/`
- `preferred_label_es` (VARCHAR(255)): Etiqueta preferida en español
- `preferred_label_en` (VARCHAR(255)): Etiqueta preferida en inglés
- `alt_labels` (TEXT[]): Array de etiquetas alternativas (sinónimos, abreviaciones)
- `skill_type` (VARCHAR(50)): Tipo de habilidad ("knowledge", "skill", "competence")
- `description` (TEXT): Descripción detallada del concepto
- `reusability_level` (VARCHAR(50)): Nivel de reutilización ("transversal", "sector-specific", "occupation-specific")
- `source` (VARCHAR(50)): Fuente de la skill ("esco_v1.1.0", "onet", "manual")

Índices:

- B-tree en `preferred_label_es` para matching exacto español
- B-tree en `preferred_label_en` para matching exacto inglés
- GIN en `alt_labels` para búsqueda en array de sinónimos
- B-tree en `source` para análisis de cobertura por fuente

Composición de la taxonomía:

- ESCO v1.1.0: 13,939 skills oficiales
- O*NET Hot Technologies: 152 skills modernas (Terraform, Kubernetes, React, etc.)
- Agregadas manualmente: 124 skills identificadas en análisis exploratorio
- **Total**: 14,215 skills en la taxonomía extendida

E.3. Configuración de PostgreSQL para Procesamiento Batch

La configuración de PostgreSQL se optimizó específicamente para procesamiento batch de grandes volúmenes de datos, priorizando throughput sobre latencia de consultas individuales.

E.3.1. Configuración de Memoria

```
# postgresql.conf

# Memoria compartida (25% de RAM disponible en servidor de 16GB)
shared_buffers = 4GB

# Memoria por operación de sort/hash
work_mem = 256MB

# Memoria para operaciones de mantenimiento (VACUUM, CREATE INDEX)
maintenance_work_mem = 1GB

# Estimación de cache del sistema operativo
effective_cache_size = 12GB
```

E.3.2. Resultados de las Optimizaciones

Las optimizaciones de configuración e índices lograron las siguientes mejoras de performance:

- **Consultas agregadas**: Reducción de ˜45s a ˜2.3s (19.5× mejora)
  - Ejemplo: Conteo de skills por país y trimestre sobre 30,660 ofertas
- **Inserciones batch**: 5,000 registros/segundo (vs. 800 registros/s sin optimización)
  - Bulk insert de skills extraídas con `COPY` y transacciones grandes
- **Búsquedas k-NN**: Latencia de 180ms para top-10 similares (vs. 540ms sequential scan)

    - Índice IVFFlat con 100 listas y 10 probes

- **Matching ESCO**: Reducción de ˜6.5h a ˜6.5min (60× aceleración)
    - Memoización mediante diccionario `skill_text` → `esco_uri` en memoria

Esta configuración permitió que el procesamiento del corpus completo de 30,660 ofertas completara en ˜6.2 horas (incluyendo todas las etapas: scraping, extracción, mapeo ESCO, embeddings, clustering).

E.4. Estrategias de Persistencia y Trazabilidad

La base de datos implementa tres mecanismos fundamentales para garantizar reproducibilidad y trazabilidad:

1. **Versionado de modelos**: Los campos `llm_model`, `model_name`, `model_version` permiten identificar qué versión exacta de cada modelo generó cada resultado.
2. **Timestamps completos**: Todas las tablas incluyen campos `created_at`, `processed_at`, `extracted_at` que registran cuándo se generó cada dato.
3. **Metadatos de configuración**: Los parámetros JSONB en `analysis_results` almacenan la configuración exacta de UMAP y HDBSCAN utilizada en cada clustering, permitiendo reproducción exacta.

Estas estrategias garantizan que cualquier resultado publicado en la tesis pueda ser auditado, reproducido y trazado hasta la oferta laboral original que lo generó.

APÉNDICE F: IMPLEMENTACIÓN DETALLADA DE PIPELINES DE EXTRACCIÓN

Este apéndice documenta exhaustivamente la implementación técnica de los dos pipelines de extracción de habilidades: Pipeline A (NER + Expresiones Regulares) y Pipeline B (Large Language Models). Se presentan los componentes específicos, flujos de procesamiento, configuraciones, mecanismos de control de calidad, y resultados de evaluación detallados sobre el gold standard de 300 ofertas laborales.

F.1. Pipeline A: Implementación NER + Expresiones Regulares

Pipeline A constituye el método base de extracción de habilidades del observatorio, diseñado para identificar menciones explícitas de tecnologías en ofertas laborales mediante la combinación de Reconocimiento de Entidades Nombradas (NER) y expresiones regulares (Regex).

F.1.1. Justificación del Enfoque Dual NER + Regex

La decisión de combinar NER y Regex se fundamentó en las limitaciones complementarias de cada técnica individual:

Limitaciones de NER solo:

- Baja cobertura de tecnologías modernas no presentes en corpus de entrenamiento (Next.js, Tailwind CSS, Terraform)
- Dificultad con acrónimos técnicos ("CI/CD", "REST API", "MLOps")
- Sensibilidad a variaciones ortográficas no vistas durante entrenamiento

Limitaciones de Regex solo:

- Omisión de menciones contextuales no-literales ("experiencia en desarrollo backend" sin mencionar tecnologías específicas)
- Fragilidad ante variaciones de formato inesperadas
- Requiere mantenimiento manual para actualizar patrones

Ventajas del enfoque combinado:

- **Cobertura**: NER incrementa recall Post-ESCO de 73.08 % (Regex solo) a 81.25 % (Pipeline A combinado), capturando +8.17pp de skills adicionales
- **Precisión**: Regex garantiza detección de tecnologías con nomenclatura estructurada (100 % precision en patrones bien definidos)
- **Robustez**: Redundancia permite validación cruzada (skills detectadas por ambos métodos tienen mayor confianza)
- **Escalabilidad**: Latencia combinada de 0.97s/oferta permite procesamiento masivo del corpus

F.1.2. Componentes del Pipeline A

1. Componente NER (Reconocimiento de Entidades Nombradas):

- **Framework**: spaCy 3.5 con modelo `es_core_news_lg`
- **EntityRuler**: Poblado con 666 patrones de la taxonomía ESCO para reconocimiento directo de habilidades técnicas
- **Configuración**: Modelo pre-entrenado base sin fine-tuning adicional
- **Latencia**: 0.65s por oferta

2. Componente Regex (Expresiones Regulares):

- **Patrones**: 371 expresiones regulares compiladas organizadas en 18 categorías:
  - Categorías base (247 patrones): Lenguajes (20), Frameworks (38), Bases de datos (15), Cloud (15), DevOps (18), Control de versiones (6), Data Science (18), Web technologies (18), Domain-specific: .NET, Build tools, Cloud services (99)
  - Patrones contextualizados en español (9): "experiencia en", "conocimiento de", "desarrollo con"
  - Skills técnicas O*NET + ESCO (276): Taxonomías externas para ampliar cobertura
  - Patrones de bullet points (2): Captura de listas separadas por símbolos
- **Características**: Word boundaries (\b), case-insensitive matching, captura de grupos contextuales
- **Latencia**: 0.32s por oferta

3. Componente de Integración y Normalización:

- **Deduplicación**: Eliminación de duplicados mediante normalización textual
- **Normalización**: Diccionario canónico de 200+ equivalencias ("js" → "JavaScript", "k8s" → "Kubernetes")
- **Niveles de output**:
  - Raw extractions: Metadata de método, posición, confianza
  - Normalized extractions: Formas canónicas estandarizadas
- **Reducción de vocabulario**: De 6,498 skills únicas a 3,200 formas canónicas

F.1.3. Flujo de Integración y Procesamiento

La integración de NER y Regex opera mediante el siguiente flujo secuencial de 7 pasos:

1. **Ejecución de NER**: Se procesa el texto con spaCy (título + descripción + requisitos)
   - Output: Lista de entidades detectadas con posiciones y labels
   - Tiempo: 0.65s promedio por oferta
2. **Ejecución de Regex**: Se aplican 371 patrones sobre el mismo texto
   - Output: Lista de matches con posiciones y patrones que generaron el match
   - Tiempo: 0.32s promedio por oferta

3. **Combinación por unión**: Se unen ambas listas de skills extraídas
   - Criterio: Mantener todas las extracciones de ambos métodos
   - Metadata: Cada skill incluye campo `extraction_method` ("NER", "Regex", o "Both")
4. **Deduplicación**: Se eliminan duplicados mediante normalización textual
   - Normalización: Lowercase, eliminación de acentos, eliminación de puntuación
   - Criterio: Skills con texto normalizado idéntico se consideran duplicadas
   - Prioridad: Si detectada por ambos métodos, se marca como `extraction_method=Both` con mayor confianza
5. **Normalización canónica**: Se aplica diccionario de equivalencias
   - Diccionario: 200+ reglas mapeando variantes a formas canónicas
   - Ejemplos: "js" → "JavaScript", "k8s" → "Kubernetes", "postgres" → "PostgreSQL"
   - Resultado: Reducción de 6,498 skills únicas a 3,200 formas canónicas
6. **Generación de outputs**: Se producen dos niveles de extracciones
   - Raw extractions: Skills tal como fueron extraídas, con metadata completa
   - Normalized extractions: Skills en formas canónicas estandarizadas
7. **Persistencia**: Se almacenan en tabla `extracted_skills` de PostgreSQL
   - Campos: `job_id`, `skill_text`, `extraction_method`, `confidence`, `position_start`, `position_end`

Performance del flujo completo:

- Latencia total: 0.97s por oferta (0.65s NER + 0.32s Regex)
- Throughput: 3,700 ofertas/hora en CPU (sin paralelización)
- Tiempo proyectado para corpus completo: 8.3 horas para 30,660 ofertas

F.1.4. Control de Calidad y Validación

Para garantizar la calidad de las extracciones del Pipeline A, se implementaron múltiples mecanismos de validación y filtrado que operan en diferentes etapas del procesamiento.

Validación de entrada (Pre-procesamiento):

- Limpieza de HTML residual: Eliminación de tags, entidades HTML, y scripts JavaScript

- Normalización de encoding: Conversión a UTF-8, manejo de caracteres especiales
- Detección de idioma: Identificación de español/inglés/Spanglish mediante `langdetect`
- Validación de longitud: Descarte de ofertas con description $<$100 caracteres

Filtrado de falsos positivos (Post-extracción):

- **Stopwords NER**: Lista de 200+ términos genéricos descartados
  - Categorías: nombres comunes, verbos genéricos, adjetivos, conectores
  - Ejemplos: "desarrollo", "experiencia", "conocimiento", "trabajo", "equipo"
- **Stopwords técnicos genéricos**: Lista de 60+ términos técnicos demasiado amplios
  - Categorías: términos paraguas, buzzwords, soft skills genéricas
  - Ejemplos: "software", "technology", "programming", "innovation", "excellence"
- Validación de longitud de skills: Descarte de extracciones $<$2 o $>$50 caracteres
- Validación de caracteres: Descarte de skills solo-numéricos o con caracteres especiales sin match ESCO

Validación cruzada y coherencia:

- Overlap NER-Regex: Skills detectadas por ambos métodos reciben mayor score de confianza
- Frecuencia en corpus: Skills únicas (aparecen en 1 sola oferta) se marcan para revisión manual
- Validación con ESCO: Skills sin match en taxonomía se categorizan como "emergentes"

Métricas de calidad monitoreadas:

- Coverage rate: 98.7 % de ofertas con al menos 1 skill extraída (30,264 de 30,660)
- Extraction diversity: Promedio 50.3 skills/oferta, mediana 42, percentil 95: 87
- ESCO match rate: 12.6 % (baja cobertura indica presencia de skills emergentes no en ESCO v1.1.0)

F.1.5. Evaluación Detallada del Pipeline A

La evaluación del Pipeline A se realizó mediante comparación contra el gold standard de 300 ofertas laborales manualmente anotadas (100 por país: Colombia, México, Argentina).

Metodología de evaluación:

Se adoptaron las métricas estándar de Information Retrieval (Precision, Recall, F1-Score) en lugar de Accuracy por tres razones fundamentales:

1. **Naturaleza del problema**: Extracción de skills es multi-label retrieval en universo abierto, no clasificación binaria
2. **Desbalance extremo**: 20-30 skills reales vs. 10,000+ candidatos negativos potenciales por oferta
3. **Indefinición de True Negatives**: El conjunto de candidatos negativos no tiene definición natural unívoca

Para cada oferta laboral j, se definen G_j (skills del gold standard) y P_j (skills extraídas por el pipeline). Las métricas se calculan mediante agregación micro-averaged:

- $TP = \sum_{j=1}^{300} |G_j \cap P_j|$ (skills correctamente extraídas)
- $FP = \sum_{j=1}^{300} |P_j \setminus G_j|$ (extraídas pero no en gold standard)
- $FN = \sum_{j=1}^{300} |G_j \setminus P_j|$ (en gold standard pero no extraídas)
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- F1-Score $= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Normalización canónica: Todas las skills se normalizan mediante diccionario de 200+ formas canónicas antes del cálculo de métricas.

Evaluación dual Pre-ESCO y Post-ESCO:

- **Pre-ESCO**: Comparación sobre texto normalizado sin mapeo taxonómico, evalúa capacidad de extracción pura incluyendo skills emergentes
- **Post-ESCO**: Comparación después de mapear a taxonomía ESCO, evalúa capacidad de estandarización

Resultados de evaluación:

Los resultados cuantitativos completos de la evaluación de Pipeline A (métricas Pre-ESCO y Post-ESCO, comparativa con variantes Regex-Only, contribución de componentes NER vs Regex, análisis de capacidades y limitaciones) se presentan en el capítulo de Resultados.

Hallazgos clave de implementación:

- Regex detecta la mayoría de skills con nomenclatura estructurada (70 % del total)
- NER: Aporta 15.1 skills/oferta adicionales (30 %)
- Overlap: 12 % de skills detectadas por ambos métodos

Limitaciones identificadas:

- Baja precision Pre-ESCO (22.54 %): Alto ruido en extracciones crudas
- Cobertura ESCO limitada: 87.4 % de skills no mapean a taxonomía oficial
- Fragmentación léxica: Skills compuestas a veces detectadas como tokens separados
- Sensibilidad a ruido HTML: Ofertas mal limpiadas generan falsos positivos

F.2. Pipeline B: Implementación con Large Language Models

Pipeline B constituye el método de enriquecimiento semántico del observatorio, diseñado para complementar Pipeline A mediante extracción de habilidades implícitas, sinónimos contextuales, y competencias inferidas.

F.2.1. Justificación del Enfoque LLM

La decisión de implementar un pipeline basado en LLMs se fundamentó en cuatro limitaciones estructurales de Pipeline A:

1. Solo detecta skills mencionadas explícitamente mediante patrones léxicos
2. Fragmenta skills compuestas al detectar tokens separados
3. Carece de desambiguación contextual
4. No captura sinónimos contextuales

El enfoque LLM resuelve estas limitaciones mediante:

- Comprensión contextual: Interpreta ofertas considerando semántica completa del texto
- Desambiguación semántica: Usa contexto para distinguir tecnologías de homónimos
- Captura de skills emergentes: Detecta tecnologías recientes no codificadas en diccionarios estáticos

La validación experimental confirmó que Pipeline B supera cuantitativamente a Pipeline A con el trade-off esperado en latencia (resultados detallados en el capítulo de Resultados).

F.2.2. Selección del Modelo

Se evaluaron cuatro modelos open-source de tamaño intermedio (3-4B parámetros):

1. Gemma 3 4B Instruct (Google DeepMind)
2. Llama 3.2 3B Instruct (Meta AI)
3. Qwen 2.5 3B Instruct (Alibaba Cloud)
4. Phi-3.5 Mini Instruct (Microsoft Research)

Configuración de evaluación: Todos los modelos se ejecutaron con cuantización INT4 mediante `bitsandbytes`, reduciendo requisitos de memoria de 16GB (FP16) a 4GB (INT4). Evaluación preliminar sobre 10 ofertas del gold standard.

Resultados comparativos:

La evaluación comparativa de los cuatro modelos LLM candidatos (Gemma 3 4B, Llama 3.2 3B, Qwen 2.5 3B, Phi-3.5 Mini) reveló diferencias significativas en:

- Gemma 3 4B: Mejor desempeño balanceado, formato JSON estable, ausencia de alucinaciones
- Llama 3.2 3B: Alucinaciones sistemáticas (agregaba skills de Data Science en ofertas frontend)
- Qwen 2.5 3B: Extracciones conservadoras, alta precisión pero baja cobertura
- Phi-3.5 Mini: Formato inconsistente, requiere parser complejo con fallback heurístico

Los resultados cuantitativos completos (métricas F1 Pre/Post-ESCO, skills por oferta, latencias) se presentan en el capítulo de Resultados.

Decisión final: Se seleccionó Gemma 3 4B Instruct por cuatro razones:

1. Mejor desempeño cuantitativo superando consistentemente a los tres alternativos
2. Estabilidad de formato con respuestas JSON válidas en más del 99 % de casos
3. Ausencia de alucinaciones sistemáticas validado manualmente sobre gold standard completo
4. Latencia aceptable para procesamiento batch no-interactivo

F.2.3. Arquitectura de Prompt Engineering

El prompt de Pipeline B se estructuró en 6 secciones secuenciales documentadas exhaustivamente en el Apéndice A. Las decisiones de diseño validadas experimentalmente incluyen:

- **Exhaustividad sobre conservadurismo**: Instrucción “EXTRAE TODAS” aumentó recall de 31.2 % a 46.23 % Pre-ESCO

- **Ejemplos con ruido realista**: Ofertas con beneficios, capacitaciones futuras y años de experiencia redujeron falsos positivos de 23 % a 8 %
- **Distinción explícita de NO extraer**: Listar casos negativos eliminó 67 % de alucinaciones
- **Normalización inline**: Especificar transformaciones "postgres→PostgreSQL, k8s→Kubernetes" redujo variantes léxicas de 18 % a 4 %
- **Formato JSON estricto**: Exigir "ÚNICAMENTE el objeto JSON" permitió parsing automático con 99 % éxito

F.2.4. Configuración de Inferencia

Parámetros de generación:

- Temperatura: 0.3 (balance entre determinismo y creatividad)
- max_tokens: 3072
- Cuantización: INT4 vía `bitsandbytes`
- Batch size: 1
- System prompt: Estandarizado (170 líneas con 3 ejemplos completos end-to-end)

Optimizaciones de performance:

- Cuantización INT4 reduce VRAM de 16GB a 4GB (4× reducción)
- Truncamiento de ofertas extensas a 4,096 tokens (límite contextual del modelo)
- Caching de prompts base para reutilización entre ofertas

F.2.5. Evaluación Detallada del Pipeline B

Resultados de evaluación:

La evaluación de Pipeline B sobre el gold standard confirmó su superioridad cuantitativa respecto a Pipeline A con el trade-off esperado en latencia y costo computacional. Los resultados cuantitativos completos (métricas F1 Pre/Post-ESCO, Precision, Recall, skills por oferta, comparativa directa con Pipeline A, análisis de latencias) se presentan en el capítulo de Resultados.

Hallazgos clave de implementación:

- Validación manual de 300 ofertas confirmó ausencia de alucinaciones sistemáticas
- Formato JSON estable con tasa de éxito superior al 99 %

- Capacidad de inferencia contextual validada mediante cobertura de soft skills implícitas
- Alta captura de skills emergentes no presentes en ESCO v1.1.0
- Trade-off arquitectónico: requiere GPU vs. CPU, aplicación selectiva a subconjuntos estratégicos

Limitaciones conocidas:

- Sensibilidad a variaciones ortográficas no capturadas por normalización inline
- Desambiguación contextual imperfecta en ofertas de múltiples puestos
- Skills emergentes posteriores a fecha de corte del modelo (julio 2024) pueden no ser reconocidas
- Latencia alta (18.3s mediana) limita aplicación a subconjuntos estratégicos del corpus

F.3. Conclusiones de Implementación

La implementación dual de pipelines permite balancear tres atributos críticos del sistema:

1. **Escalabilidad**: Pipeline A procesa corpus completo (30,660 ofertas) en 8.3 horas
2. **Calidad semántica**: Pipeline B logra +11.73pp F1 y +23.75pp Precision sobre Pipeline A
3. **Flexibilidad**: Arquitectura permite aplicar Pipeline A a 100 % del corpus y Pipeline B a subconjuntos estratégicos según requisitos de calidad vs. tiempo

Esta arquitectura dual fundamenta el diseño del observatorio como sistema híbrido que optimiza el trade-off precision/latencia según el escenario de uso, permitiendo análisis masivos con Pipeline A y enriquecimiento semántico selectivo con Pipeline B.