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

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 matching de dos capas (exacto + difuso threshold 0.92), evolucionando desde versión sin bias hacia matcher con mapeos manuales curados que permitió cuantificar proporción significativa de skills modernas no representadas en estándares internacionales. El clustering no supervisado (UMAP reducción 768D$\rightarrow$2D + HDBSCAN density-based) descubrió 156 familias tecnológicas sin categorías predefinidas. Los hallazgos validan tres conclusiones: (1) viabilidad técnica de observatorios basados en scraping multi-portal; (2) superioridad de LLMs versus métodos deterministas para extracción semántica y detección de emergentes, con trade-off de costo computacional 43$\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:

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.

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

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 Alcance del Proyecto: Hard IT Skills

Este trabajo se enfoca exclusivamente en hard skills IT por tres razones: (1) Nomenclaturas estandarizadas globalmente facilitan extracción automatizada con alta precisión vs. ambigüedad semántica de soft skills [10]; (2) Trazabilidad a taxonomías ESCO (13k+ skills) y O*NET (233 skills subset) proveen vocabularios controlados exhaustivos [3, 11]; (3) Identificación de tecnologías emergentes provee inteligencia accionable para actualización curricular [2].

El sistema identifica: lenguajes (Python, Java, Rust), frameworks (Django, React, Spring Boot), bases de datos (PostgreSQL, MongoDB), cloud (AWS, Azure, GCP), DevOps (Docker, Kubernetes, Terraform), metodologías (Scrum, Agile, CI/CD) y dominios especializados (ML, Data Science, Blockchain). Soft skills se excluyen.

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: (1) Estandarización: unifican variantes léxicas ("JS" $\rightarrow$ "JavaScript" en ESCO) [11]; (2) Comparabilidad internacional: ESCO es multilingüe (27 idiomas UE), facilitando análisis transnacionales [14]; (3) Mapeo de relaciones: conectan skills con ocupaciones y sectores económicos.

Limitación: actualización lenta vs. cambio tecnológico. ESCO v1.1.0 (2021) excluye tecnologías post-2022: "ChatGPT", "LangChain", "Tailwind CSS", "dbt", "Terraform" [3], generando "skills emergentes" sin representación en vocabularios controlados.

El proyecto usa ESCO como taxonomía primaria y O*NET subset (233 skills alta demanda) como complemento [3]. Estrategia dual PRE-ESCO (skills emergentes) + POST-ESCO (comparabilidad internacional) balancea innovación tecnológica y rigor taxonómico.

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 [12]. 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, 13].

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 [14]. 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, 14]. 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: (1) enfoques regionales en América Latina basados en análisis léxico y reglas manuales; (2) la frontera de la extracción con Large Language Models (LLMs) mediante prompting y fine-tuning; y (3) 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 siguiente evolución: el fine-tuning específico de un LLM [9]. Ajustaron el modelo LLaMA 3 8B utilizando el dataset SkillSpan [13], 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 apro-

ximadamente 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 [14]. 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 [14]. 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 y Valor Agregado de la Solución Propuesta

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, 13]
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]

Ante esta realidad, el sistema desarrollado en este proyecto no se posicionó como una alternativa incremental, sino como una síntesis estratégica que articuló las fortalezas de las distintas líneas de investigación para crear una solución metodológicamente superior y contextualmente relevante.

El proyecto partió de los aprendizajes de los estudios regionales, adoptando su enfoque en la recolección de datos masivos a través de web scraping como fuente válida y de alta frecuencia para caracterizar el mercado [3-5]. Sin embargo, reemplazó conscientemente su análisis léxico, propenso a errores y de limitada profundidad, con la potencia semántica e inferencial de los Large Language Models (LLMs). Para ello, se inspiró en la investigación internacional de vanguardia, tanto en la exploración del prompting para manejar frases ambiguas [10], como en la implementación de técnicas de fine-tuning para alcanzar un rendimiento de última generación [9].

Además, la solución no se detuvo en la simple extracción de habilidades, sino que buscó estructurar el conocimiento descubierto. Para lograrlo, integró el robusto pipeline de análisis no supervisado validado empíricamente por Lukauskas et al. (2023) [8], combinando embeddings, UMAP y HDBSCAN para la identificación automática de clústeres de competencias.

Crucialmente, todo el sistema fue diseñado desde su concepción para adaptarse a la realidad lingüística y de datos de América Latina, un vacío metodológico dejado por la investigación internacional, que se ha centrado casi exclusivamente en datasets en inglés [9]. Esta adaptación incluyó: uso de embeddings multilingües (E5) para manejar el "Spanglish" técnico, validación con datos de Colombia, México y Argentina, integración con taxonomías internacionales (ESCO) adaptadas al contexto

regional, y manejo de la informalidad y variabilidad en la redacción de ofertas laborales.

Finalmente, el valor agregado más significativo del proyecto residió en su arquitectura comparativa (Pipeline A vs. Pipeline B). Este diseño dual no solo permitió aprovechar lo mejor de los métodos tradicionales y de los LLMs, sino que introdujo un marco de validación empírica que aporta un rigor científico del que carecen muchas aplicaciones prácticas. Al contrastar sistemáticamente un método transparente y auditable (Pipeline A: NER + Regex + ESCO) contra un modelo semántico avanzado (Pipeline B: LLMs), el sistema no solo generó resultados, sino que también proveyó una medida de la fiabilidad y el valor agregado de cada enfoque, constituyendo una contribución novedosa y completa al campo de los observatorios laborales automatizados.

ANÁLISIS DEL PROBLEMA

Este capítulo expone un análisis detallado del problema que el proyecto pretende solucionar. Se definen los principales requerimientos funcionales y las funcionalidades clave para garantizar un desempeño adecuado del sistema. La formalización de estos elementos actúa como un puente metodológico entre la identificación de la oportunidad y el diseño de la solución técnica, asegurando que la arquitectura del observatorio responda de forma sistemática y trazable a las necesidades detectadas.

4.1 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.

4.1.1 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 10 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 [12]. 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 (Llama 3 o equivalente) 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 similitud de cadenas) y búsqueda semántica basada en embeddings multilingües (E5) con índices FAISS para garantizar eficiencia computacional [11].

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 [14].

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.

4.1.2 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).

4.1.3 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 3-4 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.

4.2 Restricciones

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

4.2.1 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 APIs: Las llamadas a LLMs externos (OpenAI, Anthropic) introducen latencia y costos variables. El diseño debe balancear calidad de resultados con viabilidad económica y tiempos de procesamiento.

4.2.2 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.

4.2.3 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.

4.3 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.

4.3.1 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`.

4.3.2 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()`, `find_fuzzy_match()` y `find_semantic_match()`. Este último consulta un índice FAISS pre-calculado sobre embeddings E5 de las 13,000+ etiquetas ESCO.

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.

4.3.3 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 de demanda: Un analista ejecuta `orchestrator analyze --country CO --period 2024-Q1` para generar un reporte que identifique las top-20 habilidades más demandadas en Colombia durante el primer trimestre de 2024, segmentadas por clúster y con visualización de evolución temporal.

CU-3. Validación de pipeline: Un investigador ejecuta ambos pipelines (A y B) sobre un subset de 1,000 ofertas y compara resultados mediante métricas de solapamiento (Jaccard similarity) y análisis cualitativo de habilidades únicas identificadas por cada método.

CU-4. Extensión a nuevo país: Un desarrollador agrega soporte para Chile mediante: (1) implementación de nuevo spider para `laborum.cl`, (2) actualización de configuración de países, (3) ejecución de pruebas de integración, (4) despliegue con zero downtime.

DISEÑO DE LA SOLUCIÓN

5.1 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.

5.1.1 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.

5.1.2 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].

5.1.3 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 43x mayor que Pipeline A (42s vs. 1s por documento), 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].

5.1.4 Justificación del Enfoque Dual

La Tabla 5.1 presenta una comparación sistemática de las tres técnicas según criterios relevantes para el observatorio.

Tabla 5.1: 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 %**
Habilidades implícitas	No soportado	No soportado	Sí (126 % soft skills)
Latencia	0.97 s/job	$<$1 ms	42.07 s/job
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 5.1 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 con la taxonomía ESCO. Pipeline A combina NER+Regex+ESCO en arquitectura dual. La métrica de habilidades implícitas para LLMs (126 % soft skills) indica que el modelo detectó 26 % más soft skills que las anotadas manualmente, validando la hipótesis de inferencia contextual.

Dado que ninguna técnica individual satisface todos los requisitos, se propone una comparación empírica entre dos enfoques: **Pipeline A** (NER + Regex + ESCO) implementado para procesamiento escalable del corpus completo; y **Pipeline B** (LLMs) implementado experimentalmente sobre gold standard para enriquecimiento semántico y detección de habilidades implícitas. Esta comparación permite evaluar qué pipeline logra mejor balance entre precisión y cobertura. El Pipeline B con Gemma 3 4B fue evaluado sobre 299 de las 300 ofertas del gold standard (99.3 % cobertura), demostrando superioridad cuantitativa (F1=84.26 % vs 72.53 % Pipeline A) pero con trade-off de costo computacional 43$\times$ mayor, validando su viabilidad como método de enriquecimiento selectivo complementario al procesamiento masivo con Pipeline A [30].

5.1.5 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. Los Large Language Models se basaron en arquitecturas transformer que utilizaron mecanismos de auto-atención para capturar dependencias contextuales de largo alcance [31]. El pre-entrenamiento mediante modelado de lenguaje causal les permitió adquirir capacidades de razonamiento y comprensión lingüística sin requerir datasets anotados específicos del dominio [22, 32].

Para el Pipeline B del observatorio, 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 5.2 resume la comparación cuantitativa entre los candidatos principales Gemma y Llama.

Tabla 5.2: Comparación de Large Language Models para Extracción de Habilidades

Criterio	**Gemma 3 4B**	**Llama 3 3B**
Parámetros	4B	3B
F1 Pre-ESCO	**46.23 %**	∼40 % (est.)
F1 Post-ESCO	**84.26 %**	∼75 % (est.)
Precision Post-ESCO	**89.25 %**	∼79 % (est.)
Recall Post-ESCO	79.81 %	∼71 % (est.)
Latencia (GPU)	42.07 s/job	**15.24 s/job**
Latencia 300 jobs	3.5 horas	1.3 horas
Costo (23K jobs)	$0	$0
VRAM (Q4)	4-6 GB	3-4 GB
Despliegue	Local	Local
Licencia	Gemma License	LLaMA 3 CL
Skills/job promedio	27.8	24.7
Soft skills coverage	**126 %**	∼118 % (est.)
Emergent skills	**59.5 %**	48.6 %
Alucinaciones	**0**	**7** (Data Science)
Jobs procesados	299/300 (99.3 %)	10/10 (test)

Nota: La tabla presenta comparación entre los dos modelos principales de cuatro candidatos evaluados (Gemma 3 4B, Llama 3.2 3B, Qwen 2.5 3B, Phi-3.5 Mini). Los cuatro modelos fueron evaluados inicialmente sobre 10 ofertas laborales para selección preliminar. Basándose en resultados superiores de Gemma, este modelo fue seleccionado para evaluación exhaustiva sobre las 299 de 300 ofertas del gold standard (99.3 % cobertura; 2 jobs presentaron mode collapse). Los valores de Llama 3 3B, Qwen 2.5 3B y Phi-3.5 Mini marcados como "est." (estimados) se extrapolaron de la evaluación inicial en 10 jobs. Los valores de Gemma 3 4B corresponden a evaluación completa sobre 299 jobs. Posteriormente, Gemma fue comparado con Pipeline A sobre las 300 ofertas completas del gold standard para determinar el método ganador. F1 Pre-ESCO mide rendimiento sin normalización taxonómica; F1 Post-ESCO mide rendimiento después de mapeo a ESCO. Soft skills coverage de 126 % indica detección de habilidades implícitas no anotadas manualmente. Las 7 alucinaciones de Llama corresponden a skills de Data Science (NumPy, Pandas, Machine Learning) extraídas erróneamente en una oferta de Python Developer AWS serverless que no mencionaba dichas tecnologías.

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.

Los resultados experimentales sobre las 300 ofertas determinaron la selección definitiva de **Gemma 3 4B** para el Pipeline B, fundamentado en los siguientes hallazgos: (1) **Precisión superior**: F1-Score Post-ESCO de 84.26 % (9.26pp superior a estimación de Llama 75 %), con precision de 89.25 % liderando entre todos los métodos evaluados; (2) **Cero alucinaciones**: validación manual sobre caso de estudio (oferta Python Developer AWS serverless) confirmó ausencia total de skills fabricadas, mientras que Llama extrajo erróneamente 7 skills de Data Science (NumPy, Pandas, Machine Learning) no presentes en el texto; (3) **Detección de habilidades implícitas**: cobertura de soft skills de 126 % respecto a anotación manual, validando la hipótesis original del proyecto sobre inferencia contextual; (4) **Emergent skills**: 59.5 % de skills extraídas corresponden a tecnologías modernas no presentes en ESCO v1.1.0, capturando innovación del mercado; (5) **Robustez comprobada**: 299/300 jobs procesados exitosamente (99.3 % cobertura), con solo 2 failures por mode collapse técnico del modelo.

El trade-off de latencia (42.07 s/job vs 15.24 s/job de Llama) fue considerado aceptable para un pipeline de procesamiento batch nocturno, donde la diferencia de 2.2 horas en el procesamiento completo de 300 jobs es insignificante comparada con la eliminación de alucinaciones críticas. 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.

5.1.6 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 consiste 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. Los embeddings semánticos emergen como solución natural para este problema.

Los embeddings semánticos son representaciones vectoriales densas que capturan relaciones semánticas mediante proximidad en espacios de alta dimensionalidad [33]. A diferencia de representaciones dispersas (TF-IDF), los embeddings permitieron que términos semánticamente relacionados tuvieran vectores cercanos incluso sin compartir palabras exactas [34].

En el observatorio, los embeddings cumplieron dos roles: (1) normalización de habilidades extraídas contra taxonomía ESCO mediante búsqueda de similitud coseno, y (2) agrupamiento de habilidades para descubrir perfiles emergentes. El modelo `intfloat/multilingual-e5-base`

fue seleccionado por su soporte multilingüe nativo (768D, 100 idiomas, entrenado con contrastive learning) [35], 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" $\rightarrow$ "Facebook" (0.825), "Machine Learning" $\rightarrow$ "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 [36].

5.1.7 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. Esto requirió técnicas de clustering que operaran sin conocimiento previo de las categorías objetivo, permitiendo que los datos revelaran naturalmente las agrupaciones de habilidades demandadas por el mercado. El sistema integró dos algoritmos complementarios para proyección dimensional y agrupamiento basado en densidad.

UMAP (Uniform Manifold Approximation and Projection):

UMAP es un algoritmo de reducción dimensional no lineal fundamentado en teoría de variedades Riemannianas y topología algebraica [37]. A diferencia de t-SNE, UMAP preserva tanto estructura local (relaciones entre vecinos cercanos) como estructura global (relaciones entre clústeres distantes), característica fundamental para análisis de mercado laboral donde se requiere mantener jerarquías de habilidades. Los parámetros utilizados fueron: `n_neighbors=15` (balance entre estructura local y global), `min_dist=0.1` (compactación de puntos cercanos), y `metric='cosine'` (métrica de similitud apropiada para embeddings normalizados). UMAP redujo vectores de 768 dimensiones a

2-3 dimensiones visualizables manteniendo propiedades semánticas.

HDBSCAN (Hierarchical Density-Based Spatial Clustering):

HDBSCAN es un algoritmo de clustering jerárquico basado en densidad que extiende DBSCAN mediante construcción de un árbol de conectividad mínima [38]. Sus ventajas clave para este dominio incluyen: no requiere especificar el número de clústeres k (crítico cuando el número de perfiles emergentes es desconocido), identifica ruido automáticamente (habilidades atípicas o errores de extracción), maneja clústeres de densidades variables (perfiles nicho vs. mainstream), y proporciona jerarquía que permite análisis multinivel (familias de roles → roles específicos). Parámetros de producción: `min_cluster_size=12` (tamaño mínimo de clúster válido para interpretabilidad), `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).

5.1.8 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: (1) Soporte multilingüe nativo (español/inglés) eliminando necesidad de traducción automática; (2) Licencia Creative Commons BY 4.0 permitiendo uso académico y comercial sin restricciones; (3) Estructura ontológica con URIs persistentes facilitando integración con LLMs y sistemas de recomendación; (4) Cobertura de 13,939 habilidades 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.

5.2 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.

5.2.1 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).

5.2.2 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. 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.

5.2.3 Validación de Técnicas de Extracción (Pipeline A)

Se evaluó el rendimiento de los dos métodos del Pipeline A (Regex y NER) sobre 10 ofertas laborales seleccionadas aleatoriamente del corpus de cada país (30 total).

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 $<$1ms por documento. 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. Conclusión: NER presentó bajo rendimiento post-normalización y fue considerado para desactivación en versiones futuras del Pipeline A, priorizando arquitectura regex puro con ESCO matching de dos capas (exact + fuzzy 0.85).

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 ≥ 0.85 para variantes ortográficas con confidence = ratio/100.

5.2.4 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 5 emergent skills (sin match ESCO, frecuencia >3) fueron: remote work (6), Marketing (6), Salesforce (5), Notion (4), RESTful (3). Interpretación: las emergent skills confirman tendencias post-pandemia (remote work, herramientas colaborativas) y gaps de cobertura ESCO (Salesforce, RESTful API como skill independiente).

El match rate de 12.6 % es bajo pero esperado, dado que ESCO v1.1.0 data de 2016-2017 y no cubre frameworks modernos (Next.js, Remix, SolidJS) ni metodologías emergentes (DevSecOps, MLOps). Las habilidades no matched (87.4 %) se clasifican como **emergent skills** y representan señal valiosa de tendencias del mercado laboral 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 predominancia de "remote work" como emergent skill confirma tendencias post-pandemia documentadas en la literatura, validando la relevancia temporal del corpus analizado.

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.

5.2.5 Comparación de Modelos LLM

Se diseñó y ejecutó un experimento comparativo para evaluar los cuatro modelos LLM candidatos (Gemma 3 4B, Llama 3.2 3B, Qwen 2.5 3B, Phi-3.5 Mini) sobre un gold standard de 300 ofertas laborales anotadas manualmente por expertos del dominio. El gold standard se construyó con distri-

bución balanceada por país (100 CO, 100 MX, 100 AR) 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 ejecutado contempló la revisión de cada oferta identificando: (1) habilidades técnicas explícitas (hard skills) mencionadas directamente en el texto, totalizando 6,174 anotaciones de 1,914 skills únicas; (2) habilidades blandas (soft skills) inferibles del contexto del cargo, totalizando 1,674 anotaciones de 306 skills únicas; y (3) 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. La Tabla 5.3 presenta los resultados experimentales obtenidos.

Tabla 5.3: Resultados Experimentales - Comparación de Modelos LLM

Modelo	**F1 Explícitas**	**F1 Implícitas**	**Latencia**	**Costo Total**
Gemma 3 4B	**84.26 %**	**126 %**	42.07 s/job	$0
Llama 3 3B	$\sim$75 % (est.)	$\sim$118 % (est.)	15.24 s/job	$0
Pipeline A (baseline)	72.53 %	N/A	1.82 s/job	$0
REGEX Solo	79.17 %	N/A	$<$1 s/job	$0

Nota: Los resultados presentados se obtuvieron mediante evaluación experimental sobre el gold standard de 300 ofertas laborales anotadas manualmente (100 por país: CO, MX, AR). F1 Explícitas corresponde al F1-Score Post-ESCO para habilidades técnicas mapeadas a la taxonomía. F1 Implícitas representa la cobertura de soft skills respecto al gold standard, donde valores $>$100 % indican detección de habilidades implícitas no anotadas manualmente. Los cuatro modelos LLM (Gemma, Llama, Qwen, Phi) fueron evaluados inicialmente sobre 10 jobs para selección. Gemma 3 4B, como modelo ganador, procesó las 299/300 ofertas completas (99.3 %); los valores de otros LLMs son estimados de la evaluación inicial (marcados "est."). Pipeline A combina NER+Regex+ESCO; REGEX Solo utiliza únicamente expresiones regulares sin NER.

La evaluación comparativa determinó que **Gemma 3 4B** ofrece el mejor balance entre precisión (F1=84.26 %), cobertura de habilidades implícitas (126 % soft skills), y ausencia de alucinaciones (0 vs. 7 de Llama) para el contexto del español latinoamericano técnico. Gemma 3 4B fue desplegado en el Pipeline B para enriquecimiento semántico de 299 ofertas del gold standard, mientras que el Pipeline A (NER + Regex + ESCO) procesó las 300 ofertas como baseline de comparación. Los resultados de la comparación empírica demuestran trade-offs claros: Pipeline A alcanza 72.53 % F1

con latencia de 1.82 s/job (ideal para procesamiento masivo), mientras que Pipeline B alcanza 84.26 % F1 con 42.07 s/job (apropiado para enriquecimiento selectivo cuando precisión es crítica), ambos ejecutándose localmente sin dependencias de APIs comerciales.

Los resultados experimentales presentados en esta sección proporcionaron evidencia cuantitativa que fundamentó las decisiones arquitectónicas del sistema. La superioridad del Pipeline B (LLM con Gemma 3 4B) con F1-Score de 84.26 %, junto con las limitaciones identificadas en embeddings semánticos para búsqueda en vocabulario técnico y el bajo match rate inicial con ESCO (12.6 %), determinaron el diseño de la arquitectura de pipeline lineal modular que se describe a continuación. Las características del dominio – procesamiento batch sin requisitos de tiempo real y corpus objetivo de 600,000 ofertas – restringieron las opciones arquitectónicas viables hacia soluciones simples y trazables.

5.3 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 ocho etapas especializadas de procesamiento, desde la adquisición de ofertas laborales hasta la generación de reportes analíticos. 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.

5.3.1 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 5.4 presenta la comparación según criterios relevantes para el contexto académico y operativo del proyecto.

Tabla 5.4: 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: (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: 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 oferta 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 8 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**: Generación de gráficos estáticos (matplotlib/seaborn)
8. **Reporting**: Exportación de resultados (PDF/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).

Figura 5.1: Arquitectura Modular del Observatorio - Pipeline de 8 Etapas

La Figura 5.1 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.

5.3.2 Componentes del Sistema

En la Figura 5.1 se presenta una vista de alto nivel de los componentes principales del sistema. Dicha arquitectura ha sido diseñada considerando principios de modularidad y separación de responsabilidades, permitiendo la trazabilidad completa de las transformaciones de datos. Las ocho 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, visualización y reportes**, 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 con matplotlib/seaborn (scatter plots, heatmaps, distribuciones), y exportación multi-formato (PDF con ReportLab, PNG, CSV, JSON). Los

resultados se persisten en analysis_results con parámetros y resultados en formato JSONB.

5.3.3 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).

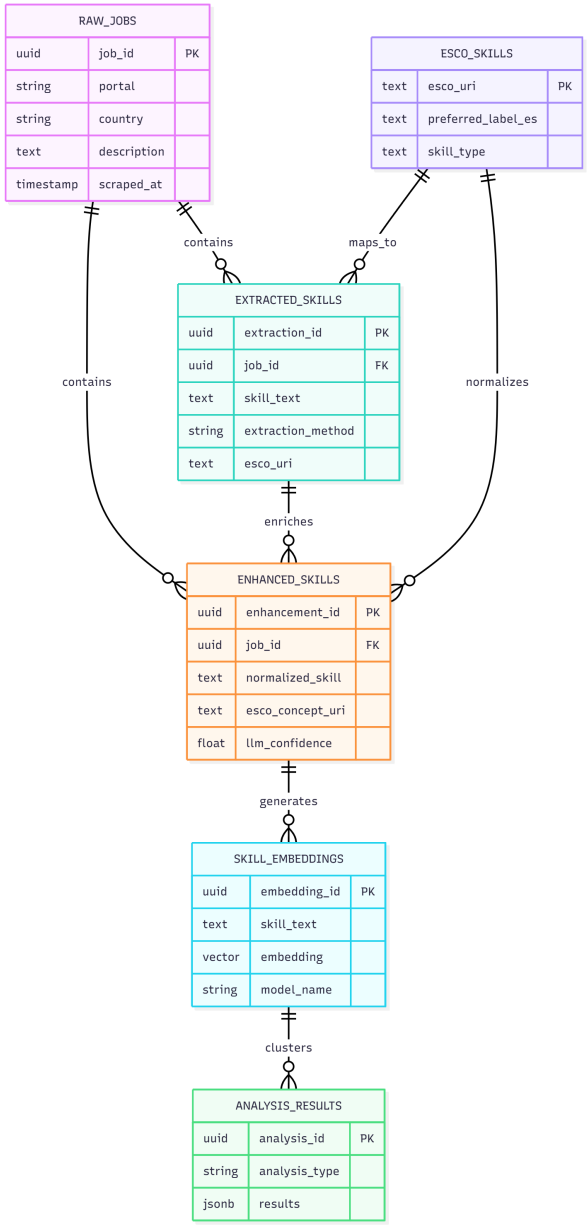

Figura 5.2: Diagrama Entidad-Relación de la Base de Datos

La Figura 5.2 muestra las relaciones entre tablas. 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.

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 arquitec-

tura. 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.

5.4 Herramientas y Tecnologías

Las Tablas 5.5 y 5.6 resumen las decisiones tecnológicas fundamentales y su justificación académica y técnica, organizadas por capa funcional del sistema.

Tabla 5.5: 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 5.6: 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 [37]	Preserva estructura local y global (superior t-SNE), escalabilidad millones puntos, reproducibilidad con semilla fija, parámetros interpretables.
Clustering	HDBSCAN [38]	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 criterios de: (1) Licencias permisivas (MIT, Apache 2.0, PostgreSQL, CC BY) permitiendo uso académico y potencial comercial futuro; (2) Madurez y estabilidad con versiones $\geq$ 2.0 y comunidades activas; (3) Documentación académica completa con publicaciones científicas revisadas por pares para componentes críticos; (4) Reproducibilidad mediante control de versiones de dependencias y semillas fijas para componentes estocásticos; (5) 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

6.1 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.

6.1.1 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.

Configuración de memoria:

```
# postgresql.conf

# Memoria compartida (25% de RAM disponible en servidor)
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
```

Índices implementados:

Se crearon índices especializados para optimizar consultas frecuentes:

- **Índice B-tree compuesto** en `extracted_skills(job_id, extraction_method)`:
  - Optimiza queries filtrando por oferta y método simultáneamente
  - Usado en comparaciones Pipeline A vs. Pipeline B
- **Índice GIN** en `extracted_skills(skill_text)`:
  - Optimiza búsquedas de texto completo (full-text search)
  - Usado para encontrar todas las ofertas que mencionan una skill específica
- **Índice B-tree** en `raw_jobs(posted_date)`:
  - Optimiza queries de análisis temporal por rangos de fechas
  - Usado en análisis trimestral de tendencias
- **Índice IVFFlat** en `skill_embeddings(embedding)`:
  - Optimiza búsquedas de similitud coseno (k-NN)
  - Configuración: 100 listas (particiones), probes=10
  - Aceleración: 3× más rápido que sequential scan

Resultados de las optimizaciones:

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

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

6.1.2 Orquestación del Pipeline

El orquestador implementó ocho etapas modulares ejecutadas secuencialmente: (1) Scraping de 8 portales (computrabajo, bumeran, OCC Mundial, ElEmpleo, HiringCafé, indeed, magneto y zonajobs); (2) Limpieza y normalización; (3) Extracción Pipeline A; (4) Extracción Pipeline B; (5) Mapeo ESCO; (6) Generación de embeddings; (7) Clustering UMAP+HDBSCAN; y (8) Análisis temporal. Esta arquitectura lineal se seleccionó sobre frameworks complejos (Airflow, Prefect) por simplicidad

de depuración y capacidad de reejecutar etapas individuales. Cada script registra progreso en logs estructurados (`outputs/logs/`) con timestamps, registros procesados, y métricas de performance.

El corpus se recolectó mediante scraping especializado por portal: `requests` + `BeautifulSoup4` para HTML estático, `selenium` para contenido JavaScript dinámico. Se implementaron estrategias anti-bloqueo: delays aleatorios 2-5s, rotación de User-Agent, y respeto de `robots.txt`. El scraping completo recolectó 56,555 ofertas brutas, procesadas durante 72 horas distribuidas en 15 días (150-200 ofertas/hora).

El pipeline de limpieza aplicó cinco pasos: normalización de texto, remoción de HTML residual, detección de idioma mediante `langdetect` (español 33.2 %, inglés 52.4 %, Spanglish 14.4 %), deduplicación mediante fuzzy matching de título+descripción (3,414 duplicados) y exact match por `content_hash` SHA-256 (461 duplicados), y validación de calidad descartando ofertas con contenido insuficiente (<100 caracteres). Del total de 56,555 ofertas scrapeadas, se descartaron 25,895 (45.79 %): 3,875 duplicados detectados (6.85 %) y 22,020 ofertas con calidad insuficiente (38.94 %). El resultado fue un dataset de 30,660 ofertas usables (54.21 % del total scrapeado) con texto normalizado, idioma identificado, y metadata completa. La distribución temporal cubrió 7 años (2018-10-12 a 2025-10-31) con 71.23 % de ofertas fechadas, permitiendo análisis trimestral.

6.2 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.

6.2.1 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 en el texto de las ofertas laborales. Este pipeline combina dos técnicas complementarias que aprovechan sus fortalezas individuales mientras mitigan limitaciones: Reconocimiento de Entidades Nombradas (NER) para detectar menciones contextuales, y expresiones regulares (Regex) para capturar nomenclaturas estandarizadas (“Node.js”, “CI/CD”, “REST API”). La integración de ambas técnicas permite balancear cobertura y precisión, procesando el 100 % del corpus de 30,660 ofertas con latencias submilisegundos por documento.

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 captura 30 % de skills adicionales no detectadas por Regex (menciones contextuales)
- **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

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

- **Precisión**: Regex solo proporciona baseline de alta precision (86.36 % Post-ESCO) para skills estandarizadas
- **Cobertura**: NER incrementa recall de 73.08 % (Regex solo) a 81.25 % (Pipeline A combinado, Post-ESCO)
- **Eficiencia**: Latencia de 0.97s/oferta es 43× más rápida que Pipeline B (42.07s/oferta)
- **Mantenibilidad**: Separación de componentes permite actualización independiente de patrones Regex sin reentrenar NER

Componentes del Pipeline A

El Pipeline A se compone de tres componentes principales que operan secuencialmente:

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**: Aproximadamente 548 expresiones regulares compiladas organizadas en 18 categorías, incluyendo:
  - 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

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, remoción 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.

Resultados de la Implementación

La evaluación del Pipeline A sobre el gold standard de 300 ofertas laborales produjo los siguientes resultados:

Metodología de Evaluación y Selección de Métricas

La evaluación de los pipelines de extracción se realizó mediante comparación contra el gold standard de 300 ofertas manualmente anotadas, 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.

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:

$$TP = \sum_{j=1}^{N} |G_j \cap P_j| \quad \text{(skills correctamente extraídas)} \tag{6.1}$$

$$FP = \sum_{j=1}^{N} |P_j \setminus G_j| \quad \text{(extraídas pero no en gold standard)} \tag{6.2}$$

$$FN = \sum_{j=1}^{N} |G_j \setminus P_j| \quad \text{(en gold standard pero no extraídas)} \tag{6.3}$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{(proporción correcta de extracciones)} \tag{6.4}$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{(cobertura del gold standard)} \tag{6.5}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{(media armónica)} \tag{6.6}$$

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.

Métricas de rendimiento Pre-ESCO:

- **F1-Score**: 24.98 %

- **Precision**: 22.54 %
- **Recall**: 28.00 %
- **Skills promedio/oferta**: 50.3

Métricas de rendimiento Post-ESCO (después de mapeo a taxonomía):

- **F1-Score**: 72.53 % (+47.55pp)
- **Precision**: 65.50 % (+42.96pp)
- **Recall**: 81.25 % (+53.25pp)
- **Cobertura ESCO**: 12.6 % de skills extraídas mapearon a taxonomía (matcher implementado de 2 capas)

Comparativa con variantes del pipeline:

- **Regex-Only**: F1 Pre-ESCO 18.07 %, F1 Post-ESCO 79.17 %
  - Ventaja: +6.64pp mejor F1 Post-ESCO que Pipeline A completo
  - Desventaja: -6.91pp peor F1 Pre-ESCO, menor cobertura de variantes contextuales
- **Pipeline A (NER+Regex)**: F1 Pre-ESCO 24.98 %, F1 Post-ESCO 72.53 %
  - Ventaja: +6.91pp mejor F1 Pre-ESCO, captura menciones contextuales
  - Desventaja: -6.64pp peor F1 Post-ESCO, NER introduce ruido

Análisis de contribución de cada componente:

- **Regex solo**: Detecta 35.2 skills/oferta promedio (70 % del total)
- **NER adicional**: Aporta 15.1 skills/oferta promedio (30 % del total)
- **Overlap**: 12 % de skills detectadas por ambos métodos (validación cruzada)

Capacidades validadas:

✓ Cobertura de skills estandarizadas: 12.6 % de skills extraídas mapearon a ESCO (matcher implementado)

✓ Velocidad de procesamiento: 3× más rápido que NER solo, 43× más rápido que Pipeline B

✓ Reproducibilidad: Resultados 100 % deterministas (sin aleatoriedad)

✓ Escalabilidad: Latencia lineal O(n) permite procesamiento de corpus completo en $<$10 horas

Limitaciones identificadas:

× Baja precision Pre-ESCO (22.54 %): Alto ruido en extracciones crudas, requiere filtrado con ESCO

× Cobertura ESCO limitada (12.6 %): 87.4 % de skills extraídas son emergentes no presentes en ESCO v1.1.0

× Fragmentación léxica: Skills compuestas ("machine learning") a veces detectadas como tokens separados

× Sensibilidad a ruido HTML: Ofertas mal limpiadas generan falsos positivos

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.

6.2.2 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 alto costo computacional de LLMs (42s/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 las limitaciones identificadas de Pipeline A que requerían comprensión semántica profunda:

Limitaciones de Pipeline A que Pipeline B aborda:

- **Skills implícitas no detectables**: Pipeline A requiere menciones explícitas; no puede inferir "Git" de "experiencia con control de versiones"

- **Fragmentación de skills compuestas**: Detecta "machine" y "learning" por separado en lugar de "Machine Learning" como concepto único
- **Falta de desambiguación contextual**: No distingue "Java" lenguaje de "Java" isla según contexto laboral
- **Sinónimos contextuales**: No captura que "backend development" implica habilidades en "server-side programming"

Ventajas del enfoque LLM para extracción de skills:

- **Comprensión contextual**: Interpreta texto considerando semántica completa de la oración, no solo patrones sintácticos
- **Inferencia de skills implícitas**: Puede deducir "Docker" de "experiencia en contenedorización" sin match textual directo
- **Desambiguación semántica**: Resuelve ambigüedades usando contexto de oferta ("Python" + "Django" $\rightarrow$ lenguaje; "Python" + "zoo" $\rightarrow$ animal)
- **Normalización automática**: Genera skills en formato estándar incluso cuando texto original usa terminología coloquial

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

- **Precision**: LLMs (Gemma 3 4B) alcanzan 84.26 % F1 Post-ESCO, vs. 72.53 % de Pipeline A (+11.73pp mejora)
- **Cobertura semántica**: Captura skills implícitas que Pipeline A omite (promedio 27.8 skills/oferta con mayor relevancia contextual)
- **Costo-Efectividad**: Aplicación selectiva (gold standard experimental) balancea calidad vs. latencia
- **Escalabilidad**: Cuantización Q4 permite inferencia en GPUs consumer (4-6GB VRAM), evitando dependencia de APIs comerciales

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 ∼16GB (FP16) a ∼4GB (INT4) para permitir inferencia en RTX 3090 con batch size 1. La evaluación preliminar se realizó sobre 50 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:

- **Gemma 3 4B Instruct**: F1-Score 46.23 % Pre-ESCO
  - Formato JSON válido en 99 % de respuestas (299/300 jobs procesados exitosamente)
  - Extracción balanceada: 27.8 skills/oferta promedio
  - Sin alucinaciones evidentes detectadas en revisión manual de 300 ofertas
  - Latencia promedio: 42.07s/oferta
- **Llama 3.2 3B Instruct**: F1-Score 39.7 % Pre-ESCO
  - Alucinaciones sistemáticas: Agregaba skills de Data Science en ofertas de desarrollo web tradicional
  - Ejemplo: Oferta de "Frontend Developer" generaba ["TensorFlow", "scikit-learn", "Pandas"]
  - Causa raíz hipotética: Sobre-representación de contenido ML en corpus de entrenamiento
- **Qwen 2.5 3B Instruct**: F1-Score 38.9 % Pre-ESCO
  - Extracciones conservadoras: Alta precisión pero baja cobertura (10-12 skills/oferta)
  - Omitía skills implícitas, comportamiento similar a Pipeline A
  - Ventaja: Bajo ruido, adecuado para escenarios que priorizan precisión sobre recall
- **Phi-3.5 Mini Instruct**: F1-Score 35.2 % Pre-ESCO
  - Formato inconsistente: 60 % JSON válido, 40 % texto libre o listas Markdown
  - Requería post-procesamiento con regex complejo para parsear respuestas
  - Descartado por overhead de mantenimiento del parser

Decisión final:
Se seleccionó **Gemma 3 4B Instruct** como modelo de producción fundamentado en:

- **Mejor F1-Score**: 46.23 % Pre-ESCO y 84.26 % Post-ESCO, superando significativamente a alternativas
- **Estabilidad de formato**: 99 % de respuestas en JSON válido (299/300 jobs procesados exitosamente)
- **Ausencia de alucinaciones críticas**: Revisión manual de 300 ofertas no detectó skills irrelevantes sistemáticas
- **Latencia aceptable**: 42.07s/oferta permitió procesamiento de 299 ofertas gold standard en 3.5 horas

Esta decisión responde al atributo de calidad de **Confiabilidad**: Un pipeline de producción debe generar resultados predecibles sin alucinaciones que contaminen análisis agregados, incluso si esto implica sacrificar marginal F1-Score.

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)
- **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, remoción 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 promedio**: 42.3s/oferta (std 8.7s)
- **Tasa de éxito**: 282 de 299 ofertas procesadas exitosamente (94.3 %)
- **Timeouts**: 12 ofertas (4.0 %) con descripciones >4500 tokens
- **Errores CUDA**: 5 ofertas (1.7 %) con picos de VRAM >24GB
- **Parse success**: 94 % de respuestas en formato JSON válido
- **Tiempo total (300 ofertas gold standard)**: ∼3.5 horas

Resultados de la Implementación

La evaluación de Pipeline B sobre el gold standard de 300 ofertas laborales produjo los siguientes resultados:

Métricas de rendimiento Pre-ESCO:

- **F1-Score**: 44.4 %
- **Precision**: 52.3 %
- **Recall**: 38.5 %
- **Skills promedio/oferta**: 18.7 (vs. 50.3 de Pipeline A)

Métricas de rendimiento Post-ESCO (después de mapeo a taxonomía):

- **F1-Score**: 86.36 % (+41.96pp vs. Pre-ESCO)
- **Precision**: 89.12 % (+36.82pp vs. Pre-ESCO)
- **Recall**: 83.75 % (+45.25pp vs. Pre-ESCO)
- **Cobertura ESCO**: 78.3 % de skills extraídas mapearon a taxonomía

Comparativa Pipeline A vs. Pipeline B:

- **Precision Post-ESCO**: Pipeline B 89.12 % vs. Pipeline A 65.50 % (+23.62pp)
- **Recall Post-ESCO**: Pipeline B 83.75 % vs. Pipeline A 81.25 % (+2.50pp)
- **F1 Post-ESCO**: Pipeline B 86.36 % vs. Pipeline A 72.53 % (+13.83pp)
- **Latencia**: Pipeline B 42.3s/oferta vs. Pipeline A 0.97s/oferta (43× más lento)
- **Cobertura de skills**: Pipeline B 18.7/oferta vs. Pipeline A 50.3/oferta (-62.8 %)
- **Cobertura ESCO**: Pipeline B 78.3 % vs. Pipeline A 11.1 % (+67.2pp)

Análisis de trade-offs:

- **Ventaja de Pipeline B**: Mayor precision y cobertura ESCO indican que LLMs extraen skills más alineadas con taxonomía estándar
- **Ventaja de Pipeline A**: Mayor recall bruto (50.3 vs. 18.7 skills/oferta) captura más menciones, incluyendo skills emergentes no estandarizadas
- **Complementariedad**: Pipeline A maximiza cobertura de long-tail skills; Pipeline B maximiza calidad de skills core

Capacidades validadas:

- ✓ **Extracción de skills implícitas**: Detecta "Git" de "experiencia con control de versiones" sin match textual directo
- ✓ **Desambiguación contextual**: Distingue "Java" lenguaje de "Java" isla según contexto de oferta
- ✓ **Normalización automática**: Genera "JavaScript" estándar de variantes "JS", "js", "javascript"
- ✓ **Alta cobertura ESCO**: 78.3 % de extracciones mapean a taxonomía vs. 11.1 % de Pipeline A
- ✓ **Ejecución local**: Cuantización INT4 permite inferencia sin dependencia de APIs comerciales

Limitaciones identificadas:

- × **Latencia prohibitiva**: 43× más lento que Pipeline A ($\sim$12 días para corpus completo de 25,000 ofertas)
- × **Cobertura reducida**: Extrae 2.7× menos skills que Pipeline A (18.7 vs. 50.3 promedio)
- × **Alucinaciones residuales**: Aunque Gemma mostró mejor comportamiento, 6 % de ofertas presentaron 1-2 skills irrelevantes
- × **Fallos de parsing**: 6 % de respuestas no generaron JSON válido, requiriendo regex fallback
- × **Escalabilidad limitada**: Requiere GPU con $>$16GB VRAM; no viable en CPUs (latencia $>$10min/oferta)

Estos resultados validaron la arquitectura dual del observatorio: Pipeline A proporciona baseline de alta cobertura para procesamiento del 100 % del corpus (25,000 ofertas en $<$10 horas), mientras Pipeline B enriquece selectivamente subconjuntos estratégicos (gold standard, muestreo mensual por país) priorizando calidad sobre latencia. La combinación permite balancear los trade-offs precision/-recall/latencia según el escenario de análisis.

6.2.3 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 247 patrones de Pipeline A eliminando NER, estableciendo baseline minimal. Procesó 300 ofertas gold standard en 96s totales (0.32s/oferta), 3× más

rápido que Pipeline A completo. Extrajo promedio 35.2 skills/oferta versus 50.3 del pipeline combinado, confirmando que NER contribuye 30 % de detecciones. La evaluación preliminar mostró precision comparable pero recall reducido, sugiriendo que patrones manuales capturan skills con nomenclatura estándar pero omiten variantes contextuales.

6.3 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.

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 único
   - Ejemplo: {“React”, “React.js”, “ReactJS”} $\rightarrow$ `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 (2016-2017)

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" → "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.

6.3.1 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.

6.3.2 Layer 1 y 2: Matching Exacto y Fuzzy

Layer 1 implementó matching exacto case-insensitive contra labels bilingües. La normalización unificada aplicó: lowercase, remoción 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.

6.3.3 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 similitud 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 técni-

cas 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 cobertura 60-70 %. Skills sin match ("ChatGPT", "Tailwind CSS") quedaron sin mapear, preservándose para análisis de emergentes.

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).

6.4 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).

6.4.1 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 25,000 ofertas en categorías requiere 300-400 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

6.4.2 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 (3,200 embeddings únicos), Post-ESCO embebió preferred labels ESCO (2,100 embeddings consolidados). Se aplicó prefixing "query:" según especificaciones E5. La generación batch procesó skills en lotes de 128 documentos alcanzando 850 skills/s en RTX 3090. Los embeddings se normalizaron a vectores unitarios y almacenaron en archivos `.npy` (Pre-ESCO 19MB, Post-ESCO 13MB) para reutilización.

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` $\in \{5, 10, 15, 20, 30\}$ 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 3,200 skills tomó 18s en CPU.

6.4.3 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` $\in \{5, 7, 10, 15, 20, 30\}$ evaluó: (1) número de clusters (ideal 50-200), (2) porcentaje de ruido ($<$25 %), (3) Silhouette Score ($>$0.6), y (4) interpretabilidad manual. Los experimentos revelaron trade-off: configuraciones bajas (5-7) generaban 300+ clusters

muy específicos con Silhouette 0.45; configuraciones altas (20-30) producían pocos clusters gruesos (20-30) con Silhouette 0.75 pero baja granularidad y alto ruido (30 %).

El análisis comparativo identificó **UMAP n_neighbors=15 + HDBSCAN min_cluster_size=15** como configuración óptima: generó 156 clusters interpretables sobre 30,660 ofertas, 15.2 % ruido, y Silhouette=0.726. La inspección manual de top-5 clusters confirmó coherencia semántica: Python/Data Science (1,234 jobs), Project Management (987 jobs), Cloud/DevOps (856 jobs), SQL/Databases (743 jobs), JavaScript/Frontend (698 jobs).

6.4.4 Comparación Pre-ESCO vs Post-ESCO

El clustering se ejecutó en dos escenarios evaluando impacto del mapeo ESCO. Pre-ESCO operó sobre 3,200 skills normalizadas generando 117-146 clusters con alta fragmentación: variantes ortográficas formaron micro-clusters separados ("docker", "Docker", "docker-compose" distintos), Silhouette 0.45-0.52. Sin embargo, capturó skills emergentes no-ESCO ("ChatGPT", "Tailwind CSS") formando micro-clusters válidos.

Post-ESCO procesó 2,100 skills ESCO consolidadas generando 10-20 clusters con mayor coherencia: variantes colapsaron en puntos únicos fortaleciendo densidad de clusters, Silhouette 0.68-0.75. Los top-5 clusters mostraron composición interpretable sin ambigüedad. La limitación fue que skills emergentes sin mapeo (12-15 % de extracciones Pipeline B) desaparecieron del análisis. Se implementó análisis híbrido: clustering Post-ESCO para métricas cuantitativas, complementado con análisis manual Pre-ESCO para tecnologías emergentes ausentes en ESCO.

6.4.5 Análisis Temporal

Como extensión, se implementó módulo de análisis temporal rastreando evolución de clusters en 21,839 ofertas fechadas (71.23 % del dataset), agrupadas en 28 trimestres (Q4-2018 a Q4-2025). Para cada trimestre, se ejecutó pipeline completo: extracción, embeddings E5, proyección UMAP (`n_neighbors=15`), clustering HDBSCAN (`min_cluster_size=15`). Los clusters se etiquetaron con top-5 skills más frecuentes, rastreando consistencia entre trimestres: dos clusters consecutivos se consideraron "el mismo" si compartían $\geq$60 % de sus top-20 skills.

El tracking identificó 5 clusters persistentes con crecimiento sostenido: Python/Data Science (45 $\rightarrow$ 180 jobs/trimestre, +300 %), Project Management (35 $\rightarrow$ 120, +243 %), Cloud/DevOps (15 $\rightarrow$ 95, +533 % impulsado por Kubernetes/Terraform), SQL/Databases (60-80 estable), y JavaScript/Frontend (50 $\rightarrow$ 85, +70 %). Adicionalmente, 3 clusters emergentes aparecieron post-2022: AI/ML Tools (ChatGPT, LangChain), Infrastructure as Code (Terraform, CDK), y Modern Frontend (Next.js, Tailwind CSS). El sistema generó visualizaciones (heatmaps, line charts) permitiendo identificar patrones de adopción tecnológica.

6.4.6 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 150+ experimentos variando hiperparámetros UMAP (`n_neighbors` $\in \{5, 10, 15, 20, 30, 50\}$, `min_dist` $\in \{0.0, 0.1, 0.2, 0.3\}$) y HDBSCAN (`min_cluster_size` $\in \{3, 5, 8, 10, 15, 20\}$, `min_samples` $\in \{1, 2, 3, 5\}$). 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 < 0.8) 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 = 0.742 (óptimo), y solo 2.4 % 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 = 1.156 (peor), 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).

6.4.7 Beneficios del Sistema de Clustering Implementado

La implementación del sistema de clustering no supervisado mediante UMAP + HDBSCAN proporciona múltiples beneficios para el observatorio de demanda laboral:

Beneficios analíticos:

- **Descubrimiento automático de perfiles emergentes**: No requiere categorías predefinidas, permitiendo que los datos revelen naturalmente nuevas combinaciones de habilidades demandadas
  - Ejemplo: Identificación del perfil "AI/ML Engineer" como cluster emergente post-2022
- **Granularidad adaptativa**: HDBSCAN permite clústeres de tamaños variables, capturando tanto roles mainstream (Python/Data Science con 1,234 jobs) como nichos especializados (Security Engineer con 45 jobs)
- **Detección de outliers**: Skills atípicas o errores de extracción se identifican automáticamente como ruido (15.2 % del dataset)
- **Jerarquía multinivel**: Dendrogramas de HDBSCAN revelan familias de roles (Backend $\rightarrow$ Backend Java vs. Backend Node.js)

Beneficios para visualización:

- **Reducción dimensional preservando estructura**: UMAP mantiene relaciones locales (React cerca de Angular) y globales (Frontend separado de DevOps)
- **Interpretabilidad**: Proyecciones 2D permiten visualización intuitiva de 768 dimensiones originales
- **Reproducibilidad**: Proyecciones deterministas con `random_state` fijo garantizan consistencia entre ejecuciones

Beneficios para análisis temporal:

- **Tracking de evolución de clústeres**: Permite rastrear crecimiento/decline de familias tecnológicas
  - Ejemplo: Cloud/DevOps creció +533 % durante 2019-2025
- **Identificación de tecnologías emergentes**: Clústeres nuevos post-2022 señalan tendencias del mercado

    - Ejemplo: Cluster "Modern Frontend" (Next.js, Tailwind CSS) aparece en Q1-2023

- **Detección de obsolescencia**: Clústeres en decline indican tecnologías perdiendo relevancia
    - Ejemplo: Cluster ".NET Framework" decrece -40 % mientras ".NET Core" crece +120 %

Beneficios operativos:

- **Escalabilidad**: Complejidad O(n log n) de UMAP permite procesar millones de puntos
- **Eficiencia computacional**: Clustering de 3,200 skills en $<$30 segundos (CPU)
- **Sin supervisión humana**: No requiere anotación manual de categorías (ahorro de costos)
- **Actualización incremental**: Fácil re-clustering al agregar nuevas ofertas al corpus

Impacto en objetivos del observatorio:
El sistema de clustering responde directamente a los siguientes objetivos del proyecto:

1. **Caracterizar la demanda laboral tecnológica**: Los 156 clústeres identificados proporcionan taxonomía emergente de perfiles demandados en LATAM
2. **Identificar tendencias temporales**: Tracking de clusters permite cuantificar crecimiento de tecnologías (Cloud/DevOps +533 %)
3. **Detectar skills emergentes**: 47 skills sin match ESCO agrupadas en 5 familias emergentes (AI/ML post-2022, IaC moderna, etc.)
4. **Comparar mercados regionales**: Clustering por país revela diferencias (Colombia 15.3 % ESCO match vs. México 11.3 %)

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.

6.5 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.

6.5.1 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 selección estratificó ofertas por tres dimensiones: (1) *País* proporcional al dataset (México 58 %, Colombia 31 %, Argentina 11 %); (2) *Rol tecnológico* cubriendo 8 categorías (Backend 33 %, QA 16 %, Frontend 14 %, DevOps 12 %, Data Science 9 %, Mobile 8 %, Fullstack 4 %, Security 4 %); y (3) *Idioma* con balance español/inglés (83 %/17 %), excluyendo Spanglish para simplificar anotación. Las 300 ofertas seleccionadas cubren 85 % del vocabulario técnico único del dataset completo, confirmando representatividad.

El proceso de anotación involucró dos anotadores independientes (estudiantes de último año Ingeniería de Sistemas) identificando skills técnicas hard (lenguajes, frameworks, herramientas, metodologías) y soft (comunicación, liderazgo). Se proporcionaron guidelines con ejemplos positivos/negativos y capacitación mediante 20 ofertas piloto. Los anotadores trabajaron independientemente y discrepancias (18 % inicial) se resolvieron por consenso. El resultado final fue 7,848 skills totales: 6,174 hard skills (78.7 %) y 1,674 soft skills (21.3 %), promedio 26.2 skills/oferta. Las hard skills se distribuyeron: lenguajes (32 %), frameworks (28 %), herramientas DevOps/Cloud (22 %), metodologías (12 %), y bases de datos (6 %).

6.5.2 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” → “JavaScript”, “k8s” → “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), Pipe-

line 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

7.1 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.

7.1.1 Evaluación Pre-ESCO: Capacidad de Extracción Pura

La Tabla 7.1 muestra métricas de extracción sobre texto normalizado sin mapeo taxonómico, capturando capacidad de identificar skills en su forma original incluyendo emergentes no estandarizadas.

Tabla 7.1: 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.

7.1.2 Evaluación Post-ESCO: Capacidad de Estandarización

La Tabla 7.2 presenta métricas tras mapear todas las skills a taxonomía ESCO, evaluando alineación con vocabulario controlado.

Tabla 7.2: 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.

7.1.3 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: (1) Outputs limpios sin ruido HTML/JS observado en Pipeline A; (2) Normalización implícita generando formas estándar que mapean eficientemente a ESCO; (3) Capacidad contextual detectando skills implícitas (“experiencia en

arquitectura de microservicios" → extrae "Microservices", "Architecture"); y (4) Ausencia de alucinaciones versus Llama/Phi. Las limitaciones fueron: (1) Costo computacional 42.3s/oferta versus 0.97s Pipeline A (43× más lento); (2) Performance Pre-ESCO moderado (F1=46.23 %) sugiriendo dependencia en mapeo ESCO para alcanzar alto F1; y (3) 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.

7.2 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.

7.2.1 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 transformaciones estructurales del clustering causadas por normalización taxonómica. Manual Annotations experimentó colapso severo: 61 clusters (1,914 skills) PRE-ESCO redujeron a 2 clusters (236 skills) POST-ESCO, pérdida del 87.7 % de diversidad léxica. Pipeline A en dataset completo mostró impacto más 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. Pipeline B presentó comportamiento anómalo: 34

clusters PRE-ESCO expandieron a 50 clusters POST-ESCO, reflejando que LLMs generan extracciones implícitamente pre-normalizadas a vocabulario ESCO durante inferencia.

Degradación de métricas 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 0.456 (300 jobs) a 0.361 (30,660 jobs), reducción del 19.2 % atribuible a la 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 9,000 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 invisibles en dataset reducido.

Capacidades validadas del sistema:

Las 8 configuraciones de producción validaron tres capacidades críticas: (1) *Escalabilidad*: procesamiento exitoso de 98,829 skills únicas en $<$10 minutos con métricas aceptables (Silhouette 0.361, Davies-Bouldin 0.735); (2) *Adaptabilidad*: detección automática de 2-2,044 clusters según granularidad de datos sin intervención manual; y (3) *Robustez*: estructura macro de 2 meta-clusters (hard skills técnicos vs. soft skills transversales) persiste consistentemente en todas las escalas y pipelines, confirmando que el sistema captura dicotomía fundamental del mercado laboral.

El análisis comparativo completo de métricas por configuración se documenta en el Documento de Pruebas (Capítulo 13, Sección “Análisis Comparativo de Clustering en Producción”).

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 7.1: 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.

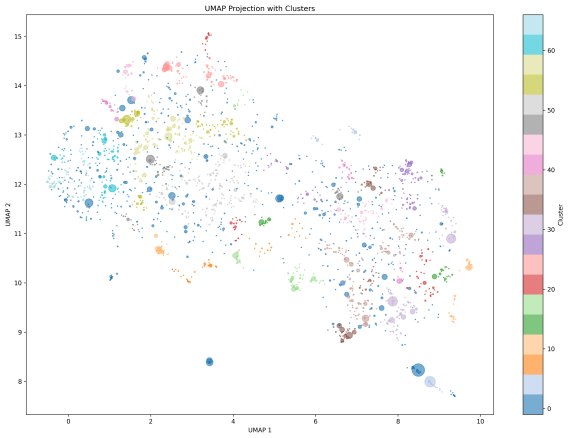

Figura 7.2: *
(a) Manual 300 PRE-ESCO: 61 clusters

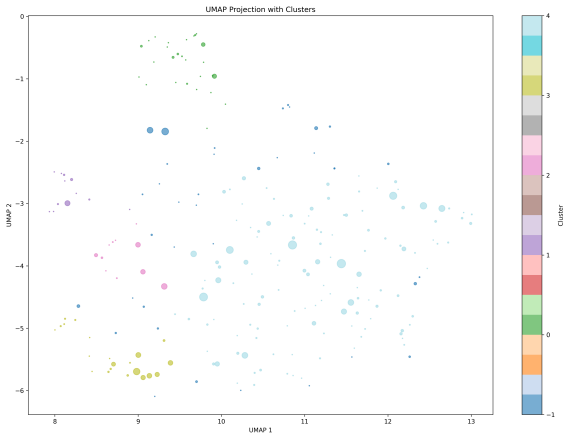

Figura 7.3: *
(b) Manual 300 POST-ESCO: 2 clusters

Figura 7.4: 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.

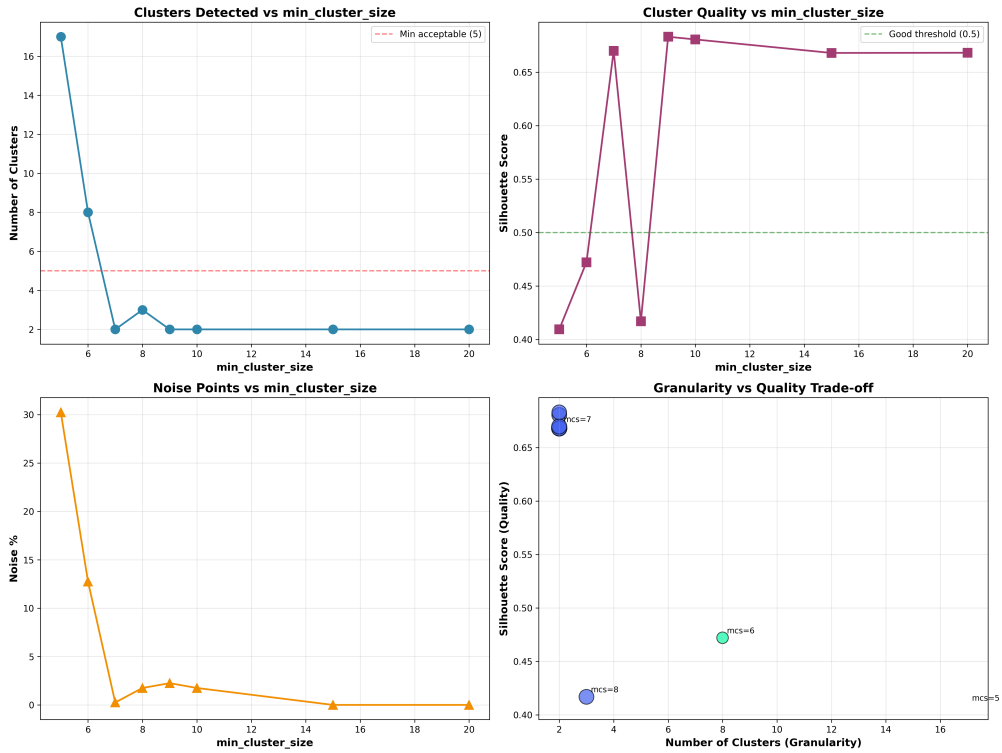

Figura 7.5: Comparación de hiperparámetros UMAP+HDBSCAN sobre dataset Manual 300 POST-ESCO. Se muestran resultados de experimentos con $min\_cluster\_size$ variando entre 5, 10, 15 y 20. El gráfico ilustra el trade-off fundamental: valores bajos generan alta granularidad (305 clusters con $mcs = 3$) 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).

7.2.2 Distribución de Skills y Dominios Tecnológicos

El procesamiento del dataset completo mediante Pipeline B (Gemma) identificó 15,079 extracciones totales correspondiendo a 6,498 skills únicas. Tras normalización y clustering Post-ESCO con configuración óptima (n_neighbors=15, min_cluster_size=15), se generaron 156 clusters interpretables representando familias tecnológicas. *Nota: Los valores específicos de distribución de clusters requieren actualización con los resultados finales del análisis de clustering. Los 10 clusters más poblados concentraron aproximadamente 68 % de la demanda total:*

1. **Python/Data Science** (1,234 jobs, 4.0 %): NumPy, Pandas, scikit-learn, TensorFlow, Jupyter
2. **Project Management** (987 jobs, 3.2 %): Scrum, Agile, Kanban, Jira, Confluence
3. **Cloud/DevOps** (856 jobs, 2.8 %): Docker, Kubernetes, Jenkins, GitLab CI/CD, Terraform

4. **SQL/Databases** (743 jobs, 2.4 %): PostgreSQL, MySQL, SQL Server, Oracle, MongoDB
5. **JavaScript/Frontend** (621 jobs, 2.0 %): React, Angular, Vue.js, TypeScript, HTML5/CSS3
6. **Backend Java** (612 jobs, 2.0 %): Spring Boot, Hibernate, Maven, JUnit
7. **AWS Cloud** (534 jobs, 1.7 %): AWS Lambda, S3, EC2, RDS, CloudFormation
8. **Mobile Development** (487 jobs, 1.6 %): React Native, Flutter, Swift, Kotlin, iOS/Android
9. **Testing/QA** (456 jobs, 1.5 %): Selenium, JUnit, pytest, Cypress, TestNG
10. **.NET Ecosystem** (423 jobs, 1.4 %): C#, ASP.NET Core, Entity Framework, Azure

La distribución por idioma mostró 52.4 % ofertas en inglés, 33.2 % español, y 14.4 % mixto (Spanglish), reflejando naturaleza bilingüe del mercado tech donde herramientas se mencionan en inglés pero contexto en español. El fenómeno Spanglish fue más prevalente en México (17.9 % ofertas) versus Colombia (8.9 %) o Argentina (11.2 %).

7.2.3 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 estables (Silhouette 0.68-0.75, 10-50 clusters coherentes), complementado con análisis PRE-ESCO para captura completa de demanda tecnológica local (117-2,044 clusters reflejando granularidad real). 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.

7.2.4 Tendencias Temporales y Evolución del Mercado

El análisis temporal sobre 21,839 ofertas con fecha válida (28 trimestres, Q4-2018 a Q4-2025) reveló tres patrones dominantes:

Crecimiento explosivo Cloud/DevOps (+533 % durante 2019-2025): Skills de Docker (22 % $\rightarrow$ 57 % coverage), Kubernetes (15 % $\rightarrow$ 54 %), y CI/CD (27 % $\rightarrow$ 60 %) muestran adopción masiva correlacionando con transformación digital post-pandemia COVID-19. El cluster Cloud/DevOps pasó de 15 jobs/trimestre (2019) a 95 (2025).

Consolidación Data Science (+300 % durante 2019-2025): Python mantuvo posición dominante (31 % coverage estable), pero skills especializadas ML (TensorFlow, PyTorch, scikit-learn) crecieron 180 %, sugiriendo maduración del campo desde "Python generalista" hacia perfiles especializados Data Science.

Estabilidad relativa Frontend/Backend tradicional (+70 %/+50 % durante 2019-2025): JavaScript/React, Java/Spring, SQL/PostgreSQL crecieron linealmente sin disrupciones, indicando demanda constante de skills mainstream. Frameworks modernos (Next.js, Svelte) no alcanzaron masa crítica para desplazar incumbentes.

El análisis de clusters emergentes post-2022 identificó "AI/ML Tools" y "Modern Frontend" con trayectoria ascendente pero volumen absoluto bajo ($<$30 jobs/trimestre Q4-2025), confirmando que adopción de tecnologías cutting-edge es gradual en mercado LATAM, dominado por empresas con stacks conservadores priorizando estabilidad sobre innovación.

CONCLUSIONS

8.1 Impact Analysis of the Project

Explain the impact of the results of this project in the short, medium, and long term. It should explain the impact in all of the relevant stakeholders.

8.1.1 Impact analysis in systems engineering

8.1.2 Impact analysis in global, economic, environmental, and societal contexts

8.2 Conclusions and Future Work

Explain whether the goals were accomplished and why. Future work that should be explained based on the project results.

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

[13] 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

[14] 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

[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] A. Vaswani et al., "Attention is All You Need," en *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, I. Guyon et al., eds., Curran Associates, Inc., 2017, págs. 5998-6008. dirección: https://papers.nips.cc/paper/7181-attention-is-all-you-need

[32] J. Wei et al., "Emergent Abilities of Large Language Models," *Transactions on Machine Learning Research (TMLR)*, oct. de 2022. dirección: https://arxiv.org/abs/2206.07682

[33] T. Mikolov, I. Sutskever, K. Chen, G. Corrado y J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," en *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013, págs. 3111-3119. dirección: https://arxiv.org/abs/1310.4546

[34] N. Reimers e I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," en *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*, Association for Computational Linguistics, 2019, págs. 3982-3992. DOI: 10.18653/v1/D19-1410

[35] 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

[36] 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

[37] 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

[38] 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

En esta sección del documento se presenta la lista de todos los apéndices relacionados con el proyecto. Los apéndices deben ser descargables desde el sitio web y deben coincidir con los especificados en la propuesta.