

**Grupo 8**

## **Observatorio de demanda laboral en América Latina**

Nicolas Francisco Camacho Alarcón

Alejandro Pinzón Fajardo

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



**CIS2025CP08**

**Observatorio de demanda laboral en América Latina**

**Autor(es):**

Nicolas Francisco Camacho Alarcón  
Alejandro Pinzón Fajardo

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

**Director**

Ing. Luis Gabriel Moreno Sandoval

**Jurados del Proyecto de Grado**

Ing. {{Nombre Jurado 1}}  
Ing. {{Nombre Jurado 2}}

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

**PONTIFICIA UNIVERSIDAD JAVERIANA  
FACULTAD DE INGENIERÍA  
CARRERA DE INGENIERÍA DE SISTEMAS**

**Rector de la Pontificia Universidad Javeriana**

Luis Fernando Múnera Congote, S.J.

**Decano de la Facultad de Ingeniería**

Ing. Diego Alejandro Patiño Guevara

**Director de Carrera de Ingeniería de Sistemas**

Ing. Carlos Andrés Parra Acevedo

**Director del Departamento de Ingeniería de Sistemas**

Ing. César Julio Bustacara Medina

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

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

## AGRADECIMIENTOS

### Nicolás Francisco Camacho Alarcón

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

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

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

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

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

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

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

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

objetivo.

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

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

### **Alejandro Pinzón Fajardo**

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

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

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

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

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

A todos ellos, mi gratitud más sincera.

# CONTENIDO

|          |                                                                          |          |
|----------|--------------------------------------------------------------------------|----------|
| <b>1</b> | <b>INTRODUCCIÓN</b>                                                      | <b>1</b> |
| <b>2</b> | <b>DESCRIPCIÓN GENERAL</b>                                               | <b>3</b> |
| 2.1      | Oportunidad y problema . . . . .                                         | 3        |
| 2.1.1    | Contexto del problema . . . . .                                          | 3        |
| 2.1.2    | Formulación del problema . . . . .                                       | 4        |
| 2.1.3    | Propuesta de solución . . . . .                                          | 4        |
| 2.1.4    | Justificación de la solución . . . . .                                   | 5        |
| 2.2      | Descripción del proyecto . . . . .                                       | 6        |
| 2.2.1    | Objetivo general . . . . .                                               | 6        |
| 2.2.2    | Objetivos específicos . . . . .                                          | 6        |
| 2.2.3    | Entregables, estándares y justificación . . . . .                        | 7        |
| <b>3</b> | <b>CONTEXTO DEL PROYECTO</b>                                             | <b>8</b> |
| 3.1      | Antecedentes Conceptuales . . . . .                                      | 8        |
| 3.2      | Conceptos Fundamentales del Dominio . . . . .                            | 8        |
| 3.2.1    | Ofertas Laborales y Avisos de Empleo . . . . .                           | 8        |
| 3.2.2    | Habilidades, Competencias y Skills . . . . .                             | 8        |
| 3.2.3    | Tipología de Habilidades: Hard Skills vs. Soft Skills . . . . .          | 9        |
| 3.2.4    | Hard Skills IT en el Análisis Automatizado . . . . .                     | 9        |
| 3.2.5    | Taxonomías de Habilidades y Ocupaciones . . . . .                        | 9        |
| 3.2.6    | Web Scraping y Adquisición de Datos . . . . .                            | 10       |
| 3.2.7    | Procesamiento de Lenguaje Natural (NLP) . . . . .                        | 10       |
| 3.2.8    | Large Language Models (LLMs) . . . . .                                   | 11       |
| 3.2.9    | Embeddings Semánticos y Representación Vectorial . . . . .               | 11       |
| 3.2.10   | Análisis No Supervisado: UMAP y HDBSCAN . . . . .                        | 11       |
| 3.2.11   | Taxonomías Estandarizadas: ESCO e ISCO . . . . .                         | 12       |
| 3.3      | Análisis del Contexto . . . . .                                          | 12       |
| 3.3.1    | Enfoques Regionales: Caracterización del Mercado con Métodos Léxicos . . | 12       |
| 3.3.2    | La Frontera de la Extracción: El Uso de Large Language Models . . . . .  | 13       |

|          |                                                                   |           |
|----------|-------------------------------------------------------------------|-----------|
| 3.3.3    | Pipelines Semánticos y Descubrimiento No Supervisado . . . . .    | 14        |
| 3.3.4    | Análisis Comparativo del Estado del Arte . . . . .                | 15        |
| 3.3.5    | Identificación de Brechas en la Literatura . . . . .              | 16        |
| <b>4</b> | <b>METODOLOGÍA</b>                                                | <b>18</b> |
| 4.1      | Diagrama de Flujo Metodológico . . . . .                          | 18        |
| <b>5</b> | <b>ANÁLISIS DEL PROBLEMA</b>                                      | <b>20</b> |
| 5.1      | Requerimientos del sistema . . . . .                              | 20        |
| 5.1.1    | Requerimientos funcionales . . . . .                              | 20        |
| 5.1.2    | Requerimientos no funcionales . . . . .                           | 21        |
| 5.1.3    | Requerimientos de datos . . . . .                                 | 21        |
| 5.2      | Restricciones . . . . .                                           | 21        |
| 5.2.1    | Restricciones técnicas . . . . .                                  | 21        |
| 5.2.2    | Restricciones de datos . . . . .                                  | 22        |
| 5.2.3    | Restricciones metodológicas . . . . .                             | 22        |
| 5.3      | Especificación funcional . . . . .                                | 22        |
| 5.3.1    | Arquitectura de pipeline de 7 etapas . . . . .                    | 23        |
| 5.3.2    | Interfaces críticas . . . . .                                     | 23        |
| 5.3.3    | Casos de uso principales . . . . .                                | 24        |
| <b>6</b> | <b>DISEÑO DE LA SOLUCIÓN</b>                                      | <b>25</b> |
| 6.1      | Antecedentes Teóricos . . . . .                                   | 25        |
| 6.1.1    | Reconocimiento de Entidades Nombradas (NER) . . . . .             | 25        |
| 6.1.2    | Extracción basada en Expresiones Regulares . . . . .              | 26        |
| 6.1.3    | Extracción basada en Modelos de Lenguaje Grandes (LLMs) . . . . . | 26        |
| 6.1.4    | Justificación del Enfoque Dual . . . . .                          | 26        |
| 6.1.5    | Modelos de Lenguaje Grandes: Selección y Evaluación . . . . .     | 28        |
| 6.1.6    | Embeddings Semánticos y Búsqueda de Similitud . . . . .           | 31        |
| 6.1.7    | Técnicas de Clustering No Supervisado . . . . .                   | 31        |
| 6.1.8    | Taxonomías de Habilidades Laborales . . . . .                     | 32        |
| 6.2      | Pruebas de Modelos . . . . .                                      | 34        |
| 6.2.1    | Conjunto de Datos . . . . .                                       | 34        |
| 6.2.2    | Construcción del Conjunto de Datos . . . . .                      | 35        |
| 6.2.3    | Validación de Técnicas de Extracción (Pipeline A) . . . . .       | 35        |
| 6.2.4    | Evaluación del Sistema Completo con Gold Standard . . . . .       | 36        |
| 6.3      | Arquitectura . . . . .                                            | 38        |
| 6.3.1    | Modelo de Vistas Arquitectónicas . . . . .                        | 39        |
| 6.3.2    | Vista Lógica: Componentes y Patrones Arquitectónicos . . . . .    | 39        |

|          |                                                                               |           |
|----------|-------------------------------------------------------------------------------|-----------|
| 6.3.3    | Vista Física: Infraestructura y Despliegue . . . . .                          | 42        |
| 6.3.4    | Vista de Procesos: Ejecución y Concurrencia . . . . .                         | 45        |
| 6.3.5    | Diseño de la Base de Datos . . . . .                                          | 47        |
| <b>7</b> | <b>DESARROLLO DE LA SOLUCIÓN</b>                                              | <b>50</b> |
| 7.1      | Implementación de la Infraestructura . . . . .                                | 50        |
| 7.1.1    | Orquestación del Pipeline . . . . .                                           | 50        |
| 7.2      | Implementación de Sistemas de Extracción de Habilidades . . . . .             | 51        |
| 7.2.1    | Pipeline A: NER y Expresiones Regulares . . . . .                             | 51        |
| 7.2.2    | Pipeline B: Modelos de Lenguaje Grandes . . . . .                             | 52        |
| 7.2.3    | Pipelines Alternativos Evaluados . . . . .                                    | 55        |
| 7.3      | Implementación del Sistema de Mapeo a Taxonomía ESCO . . . . .                | 55        |
| 7.3.1    | Arquitectura ESCOMatcher3Layers . . . . .                                     | 56        |
| 7.3.2    | Layer 1 y 2: Matching Exacto y Fuzzy . . . . .                                | 56        |
| 7.3.3    | Layer 3: Embeddings Semánticos (Deshabilitado) . . . . .                      | 56        |
| 7.4      | Implementación del Sistema de Clustering de Habilidades . . . . .             | 58        |
| 7.4.1    | Justificación del Enfoque de Clustering No Supervisado . . . . .              | 58        |
| 7.4.2    | Generación de Embeddings y Reducción UMAP . . . . .                           | 59        |
| 7.4.3    | Clustering HDBSCAN y Optimización . . . . .                                   | 60        |
| 7.4.4    | Comparación Pre-ESCO vs Post-ESCO . . . . .                                   | 60        |
| 7.4.5    | Análisis Temporal . . . . .                                                   | 60        |
| 7.4.6    | Experimentación de Hiperparámetros y Trade-off Interpretabilidad vs. Métricas | 61        |
| 7.5      | Creación del Gold Standard y Sistema de Evaluación . . . . .                  | 61        |
| 7.5.1    | Selección y Anotación del Gold Standard . . . . .                             | 61        |
| 7.5.2    | Sistema de Evaluación Dual: Pre-ESCO y Post-ESCO . . . . .                    | 64        |
| 7.6      | Implementación del Frontend Interactivo . . . . .                             | 64        |
| 7.6.1    | Dashboard Principal . . . . .                                                 | 64        |
| 7.6.2    | Módulo de Exploración de Habilidades . . . . .                                | 65        |
| 7.6.3    | Sistema de Consulta de Ofertas Laborales . . . . .                            | 66        |
| 7.6.4    | Panel de Administración . . . . .                                             | 67        |
| <b>8</b> | <b>RESULTADOS</b>                                                             | <b>69</b> |
| 8.1      | Evaluación Comparativa de Pipelines de Extracción . . . . .                   | 69        |
| 8.1.1    | Evaluación Pre-ESCO: Capacidad de Extracción Pura . . . . .                   | 69        |
| 8.1.2    | Evaluación Post-ESCO: Capacidad de Estandarización . . . . .                  | 70        |
| 8.1.3    | Análisis del Pipeline Ganador y Trade-offs . . . . .                          | 70        |
| 8.2      | Análisis del Mercado Laboral Tecnológico Latinoamericano . . . . .            | 71        |
| 8.2.1    | Resultados de Configuraciones de Clustering . . . . .                         | 71        |

|                                                                    |                                                                       |           |
|--------------------------------------------------------------------|-----------------------------------------------------------------------|-----------|
| 8.2.2                                                              | Distribución de Skills y Dominios Tecnológicos . . . . .              | 75        |
| 8.2.3                                                              | Cobertura ESCO y Skills Emergentes . . . . .                          | 77        |
| 8.2.4                                                              | Limitaciones del Análisis Temporal . . . . .                          | 79        |
| <b>9</b>                                                           | <b>CONCLUSIONES Y TRABAJO FUTURO</b>                                  | <b>80</b> |
| 9.1                                                                | Hallazgos Principales . . . . .                                       | 80        |
| 9.1.1                                                              | Superioridad de Modelos de Lenguaje Grandes . . . . .                 | 80        |
| 9.1.2                                                              | Detección de Habilidades Emergentes . . . . .                         | 80        |
| 9.1.3                                                              | Comprensión Contextual y Extracción Semántica . . . . .               | 81        |
| 9.2                                                                | Contribuciones del Trabajo . . . . .                                  | 81        |
| 9.2.1                                                              | Contribuciones Metodológicas . . . . .                                | 81        |
| 9.2.2                                                              | Contribuciones Técnicas . . . . .                                     | 81        |
| 9.2.3                                                              | Contribuciones Empíricas . . . . .                                    | 82        |
| 9.2.4                                                              | Contribuciones Prácticas . . . . .                                    | 82        |
| 9.3                                                                | Limitaciones Identificadas . . . . .                                  | 83        |
| 9.3.1                                                              | Limitaciones del Sistema . . . . .                                    | 83        |
| 9.3.2                                                              | Limitaciones del Dataset . . . . .                                    | 83        |
| 9.3.3                                                              | Limitaciones de Evaluación . . . . .                                  | 84        |
| 9.4                                                                | Lecciones Aprendidas . . . . .                                        | 84        |
| 9.4.1                                                              | Iteración Sistemática y Evaluación Dual . . . . .                     | 84        |
| 9.4.2                                                              | Eficiencia de Modelos Pequeños y Limitaciones de Taxonomías . . . . . | 84        |
| 9.4.3                                                              | Valor del <i>Gold Standard</i> y Comparación Multi-Modelo . . . . .   | 85        |
| 9.4.4                                                              | Decisiones Arquitecturales y Tecnológicas . . . . .                   | 85        |
| 9.5                                                                | Trabajo Futuro . . . . .                                              | 85        |
| 9.5.1                                                              | Extensiones de Corto Plazo . . . . .                                  | 85        |
| 9.5.2                                                              | Desarrollo de Mediano Plazo . . . . .                                 | 86        |
| 9.5.3                                                              | Proyectos de Largo Plazo . . . . .                                    | 86        |
| 9.6                                                                | Reflexión Final . . . . .                                             | 86        |
| <b>REFERENCIAS</b>                                                 |                                                                       | <b>88</b> |
| <b>Apéndices</b>                                                   |                                                                       | <b>92</b> |
| Apéndice A: Requerimientos y Especificación Funcional              | . . . . .                                                             | 93        |
| Apéndice B: Información Adicional sobre Arquitectura y Metodología | . . . . .                                                             | 99        |
| Apéndice C: Información Adicional sobre Base de Datos              | . . . . .                                                             | 104       |
| Apéndice D: Prompt de Pipeline B                                   | . . . . .                                                             | 118       |
| Apéndice E: Implementación Detallada de Pipelines                  | . . . . .                                                             | 124       |
| Apéndice F: Ejemplos de Anotaciones del Gold Standard              | . . . . .                                                             | 134       |
| Apéndice G: Capturas de Pantalla Adicionales del Frontend          | . . . . .                                                             | 138       |

## 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 siete 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 276 habilidades técnicas modernas (152 O\*NET + 124 curadas manualmente) mediante matcher conservador de dos capas (exacto + difuso threshold 0.92), alcanzando 12.6 % de cobertura. Experimentos con matcher enhanced aumentaron cobertura a 25 % pero no se implementó en producción para mantener neutralidad entre pipelines, evidenciando que 70-87 % de habilidades detectadas son emergentes no presentes en taxonomías estándar (Next.js, Tailwind CSS, Terraform, Bun), validando necesidad de captura automática de vocabulario IT actual. El clustering no supervisado (UMAP reducción 768D a 2D + HDBSCAN density-based) identificó entre 34 y 53 familias tecnológicas en configuraciones optimizadas, sin categorías predefinidas. Los hallazgos validan tres conclusiones: primero, la viabilidad técnica de observatorios basados en scraping multi-portal; segundo, la superioridad de LLMs versus métodos deterministas para extracción semántica y detección de emergentes, con trade-off de costo computacional hasta  $43\times$  mayor; y por último, la 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 siete 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 276 habilidades técnicas modernas (152 O\*NET + 124 curadas manualmente) mediante matching de dos capas (exacto + difuso), y aplicó clustering no supervisado con UMAP y HDBSCAN identificando entre 34 y 53 familias tecnológicas en configuraciones optimizadas, 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 (hasta 43× mayor latencia). Tercero, identificación de limitaciones críticas de taxonomías internacionales para vocabulario IT moderno, evidenciando mediante experimentación con matcher evolutivo qué proporción significativa de extracciones corresponde 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 atravesó, durante la última década, una coyuntura compleja definida por la convergencia de una acelerada transformación digital y la persistencia de desafíos estructurales, entre ellos, la elevada informalidad laboral y las brechas de capital humano [3]. 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 [3].

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 mediante web scraping [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 [5, 6]. 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” constituyía 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, 7].

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, extrae de forma sistemática y ética decenas de miles de ofertas laborales de portales de empleo clave en la región. El núcleo del sistema fue su arquitectura de extracción dual, compuesta por dos pipelines paralelos.

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

El segundo, Pipeline B (basado en LLMs), empleó Large Language Models (LLMs) como Gemma 3 4B 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 [8, 9].

Posteriormente, un módulo de mapeo de dos capas normalizó las habilidades extraídas por ambos pipelines contra la taxonomía ESCO extendida con 276 habilidades técnicas modernas (152 de O\*NET + 124 curadas manualmente). La primera capa realizó una coincidencia léxica exacta, mientras que la segunda aplicó matching difuso con threshold de 0.92, inspirado en arquitecturas como ESCOX [10]. Finalmente, un módulo de análisis no supervisado aplicó una secuencia metodológica de embeddings con E5, 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 [7].

#### 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 [9], 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 adaptación regional. El matcher de dos capas (exacto + difuso) permitió procesar grandes volúmenes de datos mante-

niendo alta precisión en la normalización de habilidades contra ESCO [10]. 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 [8], 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 [8], y la capacidad estructuradora del clustering semántico [7]. 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 y visualizaciones. Operativamente, se procesaron más de 30,000 ofertas laborales, garantizando calidad, trazabilidad y reproducibilidad.

### 2.2.1 Objetivo general

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

### 2.2.2 Objetivos específicos

- Construir un estado del arte exhaustivo para comparar trabajos existentes en el ámbito de observatorios laborales automatizados y técnicas de procesamiento de lenguaje natural en español.
- Diseñar una arquitectura modular, escalable y reutilizable para el observatorio laboral automatizado, fundamentada en las mejores prácticas identificadas en el estado del arte.
- Implementar e integrar técnicas de inteligencia artificial para la identificación, normalización y agrupación semántica de habilidades tecnológicas en ofertas laborales en español.
- Validar el desempeño y la robustez de la arquitectura y los modelos propuestos mediante métricas cuantitativas y estudios empíricos.

### 2.2.3 Entregables, estándares y justificación

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

Tabla 2.1: Entregables, Estándares y Justificación Técnica

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

## CONTEXTO DEL PROYECTO

### 3.1 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 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.

#### 3.2.1 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 (empleo.com, computrabajo.com, LinkedIn) operan como observatorios del mercado laboral LATAM, proveyendo texto no estructurado procesable mediante NLP.

#### 3.2.2 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 [10]. 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”) [9]. Las skills operan como proxy de demanda laboral: frecuencia refleja intensidad de demanda, co-ocurrencia revela ecosistemas tecnológicos [7].

### **3.2.3 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].

### **3.2.4 Hard Skills IT en el Análisis Automatizado**

Este trabajo se enfoca exclusivamente en hard skills IT por tres razones. Primero, las nomenclaturas estandarizadas globalmente facilitan extracción automatizada con alta precisión versus la ambigüedad semántica de soft skills [9]. Segundo, la trazabilidad a taxonomías ESCO (13,939 skills) y O\*NET (152 skills modernas) provee vocabularios controlados exhaustivos [3, 10]. Tercero, la identificación de tecnologías emergentes provee inteligencia accionable para actualización curricular [3].

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

### **3.2.5 Taxonomías de Habilidades y Ocupaciones**

Una taxonomía de habilidades es un sistema jerárquico que organiza skills en categorías, establece relaciones semánticas y provee identificadores únicos. Cumplen tres funciones principales. Primero, la estandarización mediante la unificación de variantes léxicas (“JS” a “JavaScript” en ESCO) [10]. Segundo, la comparabilidad internacional ya que ESCO es multilingüe (27 idiomas UE), facilitando análisis transnacionales [11]. Tercero, el mapeo de relaciones al conectar skills con ocupaciones y sectores económicos.

Una limitación fundamental compartida por las taxonomías es su lentitud de actualización frente al cambio tecnológico acelerado. Por ejemplo, ESCO v1.1.0 (2021) no incluye tecnologías que sur-

gieron después de 2022, como “ChatGPT”, “LangChain”, “Tailwind CSS”, “dbt” o “Terraform” [3], generando una categoría de “skills emergentes” sin representación en vocabularios controlados. Esta brecha temporal entre los ciclos de actualización taxonómica (cada 2-3 años) y la aparición de nuevas tecnologías (en cuestión de semanas o meses) representa un desafío metodológico significativo para los sistemas de análisis automatizado del mercado laboral tecnológico.

### **3.2.6 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.

### **3.2.7 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 textual involucra operaciones de normalización y estructuración. Herramientas como spaCy realizan automáticamente tokenización (segmentación del texto en unidades mínimas o “tokens”: palabras, signos de puntuación) [9] y análisis morfológico, transformando cadenas continuas en secuencias discretas procesables por algoritmos de NLP.

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 [7], 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 [8], permitiendo reconocer habilidades en contextos gramaticales complejos.

### 3.2.8 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 [8].

A través del Prompt Engineering, es posible guiar a los LLMs para realizar tareas de enriquecimiento semántico: distinción entre habilidades explícitas e implícitas [9], 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) [9] y fine-tuning (re-entrenamiento sobre datasets específicos) [8, 13].

### 3.2.9 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 [11]. 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 (“Span-glish”), 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, 11]. Modelos populares incluyen E5-large, Sentence Transformers basados en BERT y multilingual-e5-base.

### 3.2.10 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 [7].

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 [7]. Esta secuencia metodológica permite la identificación automática de “ecosistemas de habilidades” y perfiles laborales emergentes.

### 3.2.11 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 [10]. ISCO-08 (International Standard Classification of Occupations) es un estándar internacional de la OIT para clasificar ocupaciones, proporcionando un marco común para comparar datos ocupacionales entre países. Adicionalmente, se toma como referencia O\*NET, un portal del Departamento de Trabajo de EE. UU. que reporta habilidades de alta demanda [3].

## 3.3 Análisis del Contexto

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

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

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

### 3.3.1 Enfoques Regionales: Caracterización del Mercado con Métodos Léxicos

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

El estudio más completo fue el de Rubio Arrubla (2025) para el mercado colombiano [3]. Este trabajo construyó una base de datos masiva mediante web scraping del portal empleo.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, Campos et al. (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) [9]. 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 %) [9].

Tomando estas limitaciones como punto de partida, Herandi et al. (2024) representaron la siguiente evolución: el fine-tuning específico de un LLM [8]. 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 [8]. 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 [8]. 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 [9], y las metodologías de fine-tuning, aunque superiores, estaban limitadas por la barrera del idioma de los datos de entrenamiento [8].

### 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 [7]. 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 [7].

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 [7]. El gran aporte fue proporcionar validación empírica para la secuencia completa Regex a Embeddings BERT seguido de UMAP y finalizar con 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) [10]. 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 [10].

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 %) [10]. 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 [10].

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 [11]. 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 [11]. 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 [7], así como herramientas prácticas para estandarización semántica [10]. No obstante, estas capacidades no se habían integrado en una solución única que también incorporara la potencia inferencial de los LLMs de última generación [8]. Más importante aún, ninguna de estas arquitecturas avanzadas había sido desarrollada, adaptada o validada para el contexto específico del mercado laboral en América Latina y las particularidades lingüísticas del español en la región.

### 3.3.4 Análisis Comparativo del Estado del Arte

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

Tabla 3.2: Comparación de enfoques en el estado del arte

| Enfoque                         | Ventajas                                                                        | Limitaciones                                                           | Refs.   |
|---------------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------|---------|
| Enfoques Regionales (Léxicos)   | Validación de web scraping; Datos de alta frecuencia; Contexto local            | Dependencia léxica; Escalabilidad limitada; No captura semántica       | [3-5]   |
| LLMs Prompting                  | Flexibilidad; Sin entrenamiento; Captura contexto complejo                      | Inconsistencia de salida; Alucinaciones; F1 bajo (17-27 %)             | [9]     |
| LLMs Fine-tuned                 | SOTA en F1 (64.8 %); Salidas estructuradas; Auditabilidad                       | Requiere datasets anotados; Solo en inglés; Costoso computacionalmente | [8, 13] |
| Pipelines Semánticos            | Descubrimiento no supervisado; Identificación de perfiles; Metodología validada | No incluye LLMs; Limitado a extracción regex inicial                   | [7]     |
| Herramientas de Estandarización | Open-source; Integración ESCO/ISCO; Fácil de usar                               | Precisión limitada; No captura skills emergentes                       | [10]    |

### 3.3.5 Identificación de Brechas en la Literatura

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

La brecha geográfica y lingüística: Los estudios de vanguardia con LLMs se desarrollaron exclusivamente sobre datasets en inglés [8, 13], dejando un vacío metodológico para el análisis del mercado laboral en español latinoamericano. Las particularidades lingüísticas del “Spanglish” técnico característico de la región no han sido abordadas sistemáticamente.

En segundo lugar, se puede ver la brecha metodológica en la extracción: Los enfoques regionales validaron la recolección masiva de datos mediante web scraping [3-5], pero se limitaron a análisis léxico sin capacidad semántica. Inversamente, los estudios internacionales con LLMs demostraron potencia inferencial [8, 9] pero no fueron aplicados al contexto latinoamericano. Crucialmente, no existen estudios que comparen sistemáticamente métodos tradicionales versus LLMs en español o en mercados laborales de América Latina.

En cuanto a la fragmentación arquitectónica, las tres líneas de investigación operan de forma aislada. Los pipelines semánticos no integran LLMs [7], los estudios de LLMs no implementan análisis no supervisado completo [8], y los enfoques regionales no incorporan embeddings ni clustering [3].

También se encontró la ausencia de validación comparativa ya que ningún estudio contrasta sistemáticamente métodos tradicionales (NER + Regex) contra LLMs en el mismo corpus, impidiendo cuantificar el valor agregado de cada enfoque. Los estudios evalúan técnicas de forma aislada sin

proveer marco de validación empírica cruzada.

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

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

## METODOLOGÍA

El proyecto combinó CRISP-DM con un esquema modular iterativo para manejar, a la vez, la complejidad analítica de la extracción de habilidades y la construcción técnica de un sistema por componentes. CRISP-DM aportó la estructura del ciclo de vida, mientras que la modularidad permitió desarrollar, probar y mejorar cada módulo antes de integrarlo al pipeline completo.

El proceso inició con la clarificación del problema: la ausencia de herramientas automatizadas capaces de identificar habilidades técnicas emergentes en el mercado laboral latinoamericano. Mediante revisión documental, análisis de vacantes y estudio del estado del arte, se definieron objetivos, métricas y criterios de éxito que guiaron el diseño metodológico y el alcance del sistema.

Luego se realizó la caracterización de fuentes de datos, identificando variabilidad entre portales, mezcla frecuente de español e inglés y falta de estandarización. Esto justificó la creación de un conjunto de referencia anotado y la adopción de ESCO como taxonomía base para la normalización de habilidades.

La preparación de datos se desarrolló de forma iterativa, mediante módulos de scraping, limpieza, normalización y deduplicación sometidos a ciclos de inspección manual, pruebas controladas y verificación de coherencia. En paralelo, se construyó un gold standard de 300 vacantes para evaluar formalmente los modelos.

El modelado siguió la misma lógica incremental, con dos pipelines complementarios, el primero sinedo técnicas tradicionales basadas en reglas, regex y reconocimiento de entidades y el segundo se establecio como modelos de lenguaje con mapeo semántico hacia ESCO. Cada iteración se validó contra el gold standard, permitiendo mejorar precisión, cobertura y coherencia en la extracción.

La evaluación del sistema combinó métricas cuantitativas, revisión cualitativa y análisis de robustez operativa, lo que abrió nuevos ciclos de refinamiento y fortaleció la calidad global del pipeline.

Finalmente, todos los componentes se integraron en una herramienta operativa, con ejecución automática, CLI y documentación completa, validada en un entorno real para asegurar estabilidad, trazabilidad y operación recurrente sin supervisión.

### 4.1 Diagrama de Flujo Metodológico

El flujo metodológico del proyecto integró las seis fases de CRISP-DM (Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, Deployment) con las siete etapas del pipeline de software mediante ciclos iterativos de refinamiento. Cada fase de CRISP-DM se tradujo en actividades concretas de desarrollo de software, estableciendo un proceso sistemático que equilibró

diseño metodológico y evolución técnica. Los resultados de la fase de Evaluación retroalimentaron continuamente la fase de Modelado, generando versiones mejoradas de los pipelines de extracción mediante ajuste de parámetros, refinamiento de reglas lingüísticas y optimización de prompts para LLMs.

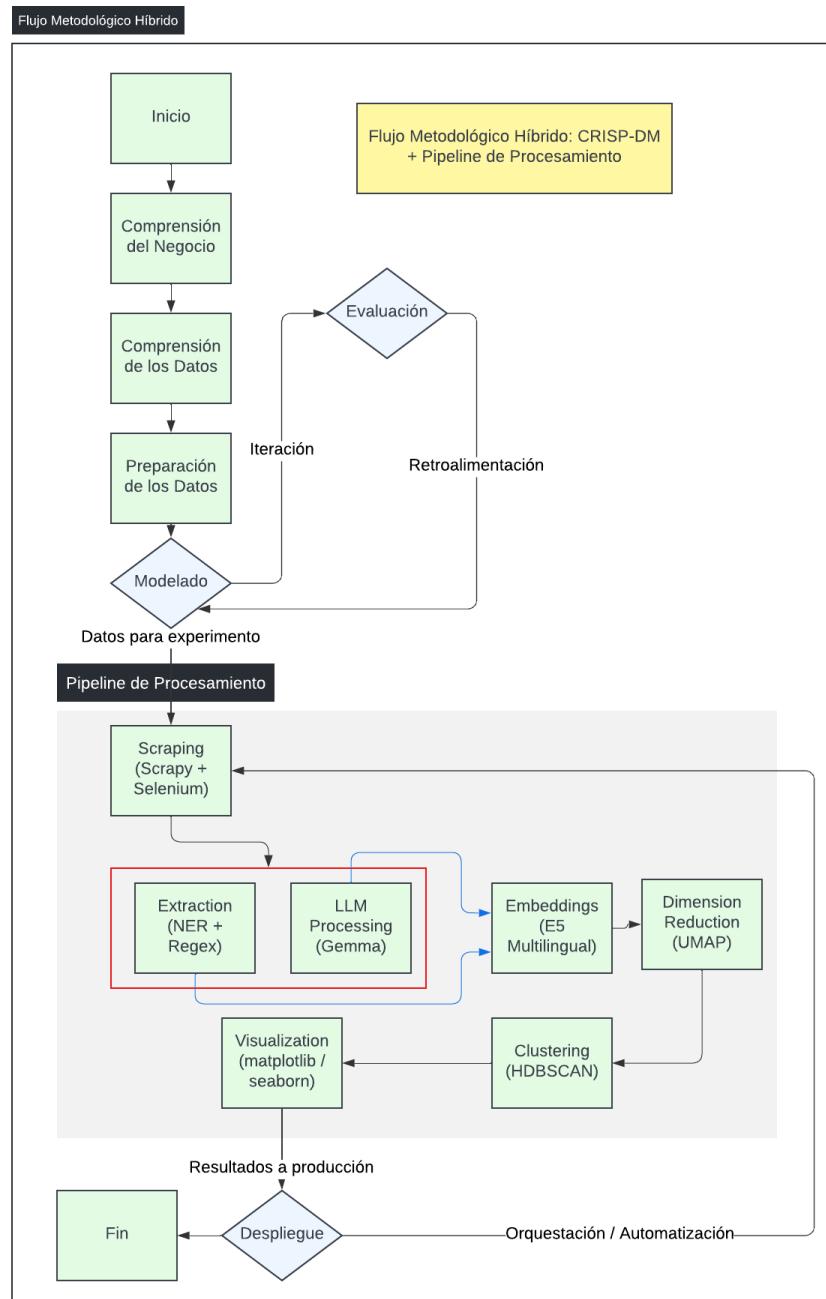


Figura 4.1: Flujo Metodológico del Proyecto: Integración de CRISP-DM con Pipeline de 7 Etapas

## ANÁLISIS DEL PROBLEMA

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

### 5.1 Requerimientos del sistema

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

#### 5.1.1 Requerimientos funcionales

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

- **Adquisición automatizada de datos:** Recolección de ofertas laborales desde portales de empleo en Colombia, México y Argentina mediante web scraping que soporte tanto contenido estático (HTML directo) como dinámico (JavaScript rendering) [12].
- **Procesamiento de lenguaje natural:** Normalización y limpieza de texto adaptada al español latinoamericano técnico, incluyendo tokenización y manejo de “Spanglish” [3].
- **Extracción dual de habilidades:** Pipeline A con NER y regex para habilidades explícitas; Pipeline B con LLMs para inferencia de habilidades implícitas.
- **Normalización taxonómica:** Mapeo de habilidades extraídas contra taxonomía ESCO mediante matching exacto y difuso (fuzzywuzzy  $\geq 0.92$ ).
- **Análisis no supervisado:** Generación de embeddings semánticos (768D), reducción dimensional (UMAP), y clustering jerárquico (HDBSCAN) para descubrir perfiles emergentes.
- **Trazabilidad completa:** Registro de metadatos de procesamiento (timestamps, versiones de modelos, parámetros) para reproducibilidad científica.

### 5.1.2 Requerimientos no funcionales

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

- **Escalabilidad:** Capacidad de procesar más de 30,000 ofertas con latencias aceptables (extracción <30 seg/oferta, clustering <4 horas).
- **Multilingüismo:** Soporte nativo para español, inglés y mezcla técnica (“Spanglish”) en todos los modelos de NLP.
- **Reproducibilidad científica:** Semillas aleatorias fijas, versionado de modelos, y persistencia de datasets intermedios.
- **Portabilidad y mantenibilidad:** Contenedorización Docker, adherencia a PEP 8, documentación técnica, y arquitectura modular extensible.

### 5.1.3 Requerimientos de datos

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

- **Cobertura geográfica:** Ofertas de Colombia, México y Argentina con al menos 2 portales por país.
- **Volumen y calidad:** Más de 30,000 ofertas con descripción >100 caracteres y al menos 2 habilidades identificables, tras deduplicación y filtrado de calidad.
- **Metadatos temporales:** Fecha de publicación, recolección y expiración para análisis de evolución temporal.
- **Representatividad sectorial:** Aunque enfocado en tecnología, captura de múltiples sectores (TI, finanzas, manufactura, salud) para identificar demanda transversal de competencias digitales.

## 5.2 Restricciones

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

### 5.2.1 Restricciones técnicas

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

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

### 5.2.2 Restricciones de datos

Las características del dominio de ofertas laborales introducen restricciones significativas:

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

### 5.2.3 Restricciones metodológicas

El contexto de español latinoamericano técnico presenta desafíos metodológicos únicos:

- **Ausencia de ground truth:** No existe dataset etiquetado de referencia para habilidades en ofertas laborales en español latinoamericano, dificultando evaluación cuantitativa rigurosa.
- **Sesgo de fuente:** Portales de empleo excluyen ofertas en sitios corporativos directos, redes sociales o canales informales, introduciendo sesgo de formalidad y tamaño de empresa.

## 5.3 Especificación funcional

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

### 5.3.1 Arquitectura de pipeline de 7 etapas

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

1. **Scraping:** Recolecta ofertas laborales de portales mediante Scrapy + Selenium → raw\_jobs
2. **Normalización:** Limpia y estandariza texto UTF-8 → description\_clean, combined\_text
3. **Extracción Pipeline A:** Aplica NER + Regex → extracted\_skills
4. **Extracción Pipeline B:** Aplica LLM con prompt engineering → enhanced\_skills
5. **Mapeo ESCO:** Normaliza contra taxonomía (exact + fuzzy) → esco\_uri, esco\_preferred\_label
6. **Embeddings:** Genera vectores 768D con E5 Multilingual → skill\_embeddings
7. **Clustering:** UMAP (768D→2-3D) + HDBSCAN → analysis\_results

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

### 5.3.2 Interfaces críticas

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

- **Scraper-Database:** JobPostgresPipeline implementa batch inserts optimizados con deduplicación por hash SHA-256 del contenido, almacenando ofertas en raw\_jobs.
- **Extractor-ESCO:** ESCOMatcher3Layers implementa estrategia de matching de 3 capas (exact, fuzzy, semantic) contra taxonomía ESCO extendida, con caché de resultados para optimizar procesamiento batch.
- **LLM-Processor:** LLMHandler abstrae llamadas a modelos locales (llama.cpp, transformers) o APIs remotas (OpenAI) con interfaz unificada, soportando múltiples backends sin modificar código de negocio.
- **Orquestador-Pipeline:** MasterController coordina scheduler, pipeline automator y health monitoring, con control de estado persistente para reinicio tras fallos.

### 5.3.3 Casos de uso principales

El observatorio soporta cuatro casos de uso fundamentales:

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

*Nota: La especificación técnica detallada de los requerimientos, restricciones y casos de uso presentados en este capítulo se encuentra en el Apéndice A. Los detalles de arquitectura física, lógica, despliegue e interfaces se presentan en el capítulo de Desarrollo.*

## DISEÑO DE LA SOLUCIÓN

### 6.1 Antecedentes Teóricos

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

#### 6.1.1 Reconocimiento de Entidades Nombradas (NER)

El Reconocimiento de Entidades Nombradas es una tarea fundamental de NLP que consiste en localizar y clasificar entidades en texto dentro de categorías predefinidas [14]. Los sistemas NER modernos se basan en modelos de aprendizaje supervisado, particularmente arquitecturas basadas en transformers [15]. 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 [16]. 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 [17].

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 [18]. Los resultados experimentales revelaron que NER en este contexto específico presentó desempeño mixto: si bien aporta cobertura adicional de skills contextuales (mejora de +6.91pp F1 Pre-ESCO), introduce ruido que degrada precisión Post-ESCO (-6.64pp versus configuración Regex-Only). A pesar de este trade-off, NER se mantuvo en Pipeline A dado que incrementa recall detectando menciones contextuales que patrones deterministas omiten, como se documenta en la sección de pruebas de modelos.

### 6.1.2 Extracción basada en Expresiones Regulares

Las expresiones regulares (regex) son patrones de búsqueda basados en lenguajes formales que permiten identificar secuencias de caracteres con estructuras específicas [19]. 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 [20].

### 6.1.3 Extracción basada en Modelos de Lenguaje Grandes (LLMs)

Los Large Language Models representaron un cambio de paradigma en NLP, basándose en arquitecturas transformer pre-entrenadas sobre corpus masivos mediante objetivos de modelado de lenguaje [21, 22]. 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 [13, 23, 24].

Sin embargo, presentaron limitaciones significativas: latencia 15-25x mayor que Pipeline A (tiempo típico de procesamiento de 15-25s por documento vs. 1s), no-determinismo con temperatura > 0, alucinaciones que generaron habilidades incorrectas, alto costo computacional (GPU integrado o APIs de pago), y sesgo lingüístico hacia el inglés con rendimiento degradado en español técnico [25, 26].

### 6.1.4 Justificación del Enfoque Dual

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

Tabla 6.1: Comparación de Técnicas de Extracción de Habilidades

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

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

Dado que ninguna técnica individual satisface todos los requisitos, se implementaron dos pipelines en paralelo para comparación empírica: Pipeline A (NER + Regex) diseñado para procesamiento escalable del corpus completo mediante reglas determinísticas; y Pipeline B (LLMs) implementado experimentalmente sobre gold standard para evaluar enriquecimiento semántico y detección de habilidades implícitas. Ambos pipelines procesan las mismas ofertas de forma independiente, permitiendo comparación directa de sus capacidades. La evaluación experimental determinó que Pipeline B supera cuantitativamente a Pipeline A pero con mayor costo computacional, validando la necesidad de arquitectura dual que permita seleccionar el pipeline óptimo según el caso de uso: Pipeline A para análisis masivo y Pipeline B para validación cualitativa o detección de skills emergentes (evaluación comparativa detallada en el capítulo de Resultados) [27].

### 6.1.5 Modelos de Lenguaje Grandes: Selección y Evaluación

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

#### **Justificación de la Selección:**

La selección de modelos LLM se fundamentó en cuatro criterios técnicos principales. Primero, se priorizaron modelos open-source dada la naturaleza académica del proyecto y la necesidad de reproducibilidad científica. Segundo, el estado del arte en extracción de habilidades con LLMs estableció a Llama 3 como referente base, con estudios demostrando efectividad de LLMs fine-tuned alcanzando F1-Score de 64.8 % sobre SkillSpan dataset [8] y validaciones de Llama-3 8B para contextos multilingües europeos [11]. Adicionalmente, se documentó que el prompting con LLMs requiere prompt engineering cuidadoso para tareas de extracción estructurada [9]. Tercero, el rango de parámetros 3-4B se determinó mediante restricción de hardware disponible: el procesamiento local con GPU integrado (MacBook Air M4 con 16 GB de memoria unificada y aceleración Metal) permite ejecutar modelos de hasta 4.3B parámetros con cuantización Q4\_K\_M (4 bits por peso), requiriendo aproximadamente 2.4-2.8 GB de memoria unificada según la fórmula:

$$\text{Memoria} \approx \frac{\text{Parámetros} \times 4 \text{ bits}}{8 \text{ bits/byte}} + \text{overhead KV-cache} \quad (6.1)$$

donde el overhead del KV-cache representa típicamente 15-20 % adicional. Modelos de 7B+ parámetros excederían los 5 GB de memoria unificada con cuantización Q4, comprometiendo estabilidad del sistema durante inferencia. Cuarto, se consultaron rankings especializados (Hugging Face Open LLM Leaderboard filtrado por español, LM Arena) para identificar candidatos con desempeño reportado en tareas de NLP multilingüe.

#### **Candidatos Evaluados:**

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. Dado que el Pipeline B fue utilizado únicamente para comparación experimental contra el gold standard, los modelos se ejecutaron localmente con GPU integrado (aceleración Metal en Apple Silicon) para validar su viabilidad técnica en contextos académicos con recursos limitados.

Gemma 3 4B es un modelo ligero desarrollado por Google basado en la arquitectura Gemini, con 4 mil millones de parámetros optimizado para despliegue eficiente en hardware limitado. El modelo permite ejecución con GPU integrado mediante cuantización Q4 (tamaño compacto), cuenta con licencia permisiva Gemma para uso académico y comercial, ofrece tokenización eficiente heredada de la familia Gemini, y provee soporte multilingüe con cobertura de español latinoamericano. Sin embargo, presenta 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 3.2 3B es la versión compacta de la familia LLaMA 3 de Meta AI, entrenado sobre un corpus masivo multilingüe con enfoque en eficiencia y 3 mil millones de parámetros que representan el balance extremo entre rendimiento y recursos computacionales. El modelo requiere memoria mínima (3-4 GB con cuantización Q4), implementa arquitectura optimizada con Grouped-Query Attention para inferencia rápida, ofrece tokenización eficiente de la familia LLaMA 3, cuenta con licencia LLaMA 3 Community License permisiva para proyectos académicos, y proporciona latencia reducida ideal para procesamiento batch. No obstante, presenta 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 es parte de la familia Qwen (Qianwen) de Alibaba Cloud optimizada para multilingüismo con énfasis en idiomas asiáticos y europeos, con 3 mil millones de parámetros que destacan por su arquitectura de atención eficiente. El modelo ofrece precisión superior en tareas de extracción estructurada, tokenización eficiente multilingüe, requisitos moderados de memoria (3-4 GB con Q4), y licencia Apache 2.0 permisiva. Por otro lado, muestra conservadurismo excesivo con recall bajo en contextos ambiguos, menor cobertura de vocabulario técnico latinoamericano, y tendencia a omitir skills implícitas.

Phi-3.5 Mini de Microsoft Research es un modelo ultra-compacto de 3.8 mil millones de parámetros entrenado con datos sintéticos de alta calidad. El modelo implementa arquitectura extremadamente eficiente con baja latencia, demuestra capacidad de razonamiento avanzado para su tamaño, provee soporte multilingüe, y cuenta con licencia MIT. Sin embargo, presenta 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.

#### **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. 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 del gold standard (cuyo protocolo de construcción se describe detalladamente en el capítulo de Desarrollo, sección Protocolo de Anotación Manual) para validación exhaustiva.

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 6.2 resume la comparación cuantitativa entre los cuatro modelos evaluados.

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

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

*Nota: Los cuatro modelos fueron evaluados comparativamente sobre 10 ofertas laborales para selección inicial. Gemma 3 4B demostró mejor rendimiento en la evaluación preliminar y fue seleccionado para procesar el conjunto completo de 300 ofertas del gold standard, logrando 299/300 ofertas procesadas exitosamente (99.3 % cobertura; 2 jobs con mode collapse). Llama, Qwen y Phi procesaron solo las 10 ofertas iniciales, siendo descartados por menor rendimiento y problemas técnicos. F1 Pre-ESCO mide rendimiento antes de normalización taxonómica. Alucinaciones (DS) indica skills de Data Science extraídas erróneamente. JSON válido mide porcentaje de respuestas con formato estructurado correcto.*

Los resultados experimentales determinaron la selección definitiva de Gemma 3 4B para el Pipeline B, fundamentado en cinco hallazgos principales: precisión superior en la evaluación de 10 ofertas preliminares, ausencia de alucinaciones sistemáticas detectadas en otros modelos, cobertura efectiva de habilidades implícitas validada en las 300 ofertas completas, captura de tecnologías emergentes no presentes en ESCO, y robustez operacional con tasa de éxito superior al 99 %. La validación manual confirmó que Gemma genera outputs limpios sin fabricar skills, mientras que alternativas como Llama 3.2 presentaron alucinaciones sistemáticas en ofertas técnicas (métricas cuantitativas detalladas en el capítulo de Resultados).

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

### 6.1.6 Embeddings Semánticos y Búsqueda de Similitud

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

En el observatorio, los embeddings semánticos se utilizan para el agrupamiento de habilidades y descubrimiento de perfiles emergentes mediante clustering. El modelo `intfloat/multilingual-e5-base` fue seleccionado por su soporte multilingüe nativo (768D, 100 idiomas, entrenado con contrastive learning) [28], normalización L2 integrada, tamaño compacto (278M parámetros ejecutables en CPU <100ms/batch), y licencia MIT.

#### Intento de Uso para Matching ESCO:

Inicialmente se intentó utilizar embeddings semánticos para normalización de habilidades extraídas contra taxonomía ESCO mediante búsqueda de similitud coseno (Layer 3 del matcher). Sin embargo, la evaluación experimental con 15 habilidades técnicas agregadas manualmente a ESCO reveló limitaciones críticas del modelo E5 para vocabulario técnico especializado. 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 de mapeos incorrectos: “React” se relacionó con “neoplasia” (0.828), “Docker” con “Facebook” (0.825), y “Machine Learning” con “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 ( $\geq 0.92$ ) excluyen matches exactos. Con base en esta evidencia, se tomó la decisión de deshabilitar la capa de búsqueda semántica (Layer 3), operando el sistema de matching ESCO con estrategia de dos capas: Layer 1 (Exact Match) mediante búsqueda SQL con confidence = 1.00, y Layer 2 (Fuzzy Match) con threshold = 0.92 para capturar variantes ortográficas. Los embeddings E5 se mantienen únicamente para clustering de habilidades, donde han demostrado efectividad en capturar relaciones semánticas para agrupamiento no supervisado.

### 6.1.7 Técnicas de Clustering No Supervisado

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

de ofertas laborales ha sido validada en estudios previos mediante combinación de técnicas de NLP y clustering [7].

#### **Configuración de UMAP:**

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

#### **Configuración de HDBSCAN:**

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

### **6.1.8 Taxonomías de Habilidades Laborales**

Si bien las técnicas de extracción, embeddings y clustering constituyeron el núcleo analítico del observatorio, todas estas operaciones dependieron de un componente fundamental: una taxonomía de referencia que permitiera normalizar las habilidades extraídas del texto crudo y mapearlas a conceptos estandarizados. Sin esta normalización, habilidades equivalentes como “React”, “React.js” y “ReactJS” hubieran sido tratadas como entidades distintas, fragmentando el análisis y degradando la calidad del clustering. La selección de la taxonomía apropiada requirió balancear cobertura, actualización, soporte multilingüe y accesibilidad.

Se evaluaron tres alternativas principales para la normalización de habilidades. La primera, O\*NET (Occupational Information Network), es la taxonomía del Departamento de Trabajo de Estados Unidos que contiene más de 1,000 ocupaciones y 20,000 habilidades organizadas jerárquicamente con actualización periódica y datos salariales asociados. Esta taxonomía ha sido ampliamente utilizada en investigación de mercado laboral y extracción de habilidades [13]. Sin embargo, presenta limitaciones significativas para el contexto latinoamericano: enfoque centrado en el mercado estadounidense, escasa cobertura de tecnologías emergentes populares en la región, y ausencia de soporte multilingüe nativo que requeriría traducción automática con riesgo de pérdida semántica.

La segunda alternativa, ESCO (European Skills, Competences, Qualifications and Occupations), es la taxonomía oficial de la Unión Europea que provee más de 13,000 habilidades, 3,000 ocupaciones, y soporte para 27 idiomas incluyendo español e inglés. ESCO ha sido adoptada extensivamente en sistemas de extracción de habilidades mediante LLMs y grafos de conocimiento [10, 29], demostrando efectividad para matching de vacantes en contextos multilingües. La taxonomía ofrece etiquetas nativas en español que eliminan necesidad de traducción, cobertura amplia de habilidades tecnológicas organizadas mediante estructura ontológica con URIs únicos, y respaldo institucional de la Comisión Europea.

sión Europea que garantiza mantenimiento a largo plazo. La versión v1.1.0 utilizada en este proyecto, publicada en 2021-2022, contiene 13,939 conceptos de habilidades con relaciones semánticas que facilitan navegación y expansión taxonómica. No obstante, ESCO presenta actualización menos frecuente que el ritmo del mercado tecnológico, resultando en menor cobertura de frameworks JavaScript modernos y librerías emergentes que aparecen constantemente en ofertas latinoamericanas.

La tercera categoría evaluada corresponde a taxonomías propietarias como LinkedIn Skills y Burning Glass Technologies, que se actualizan con mayor frecuencia capturando tecnologías emergentes casi en tiempo real. Sin embargo, estas alternativas presentan limitaciones críticas para investigación académica: acceso restringido mediante APIs de pago con costos prohibitivos para proyectos sin financiamiento, licencias restrictivas que impiden publicación de resultados derivados, y falta de transparencia metodológica sobre criterios de inclusión y relaciones entre conceptos.

Tras evaluar estas alternativas, se seleccionó ESCO v1.1.0 como base taxonómica fundamentado en cuatro razones principales que priorizan reproducibilidad y accesibilidad. Primero, el soporte multilingüe nativo en español e inglés eliminó la necesidad de traducción automática que hubiera introducido errores y ambigüedad semántica en vocabulario técnico especializado. Segundo, la licencia Creative Commons BY 4.0 permitió uso académico y potencial comercial futuro sin restricciones legales ni costos de licenciamiento. Tercero, la estructura ontológica con URIs persistentes (formato [http://data.europa.eu/esco/skill/...](http://data.europa.eu/esco/skill/)) facilitó la integración con sistemas de recomendación, LLMs mediante prompt engineering, y exportación de resultados en formatos estandarizados como RDF y JSON-LD. Cuarto, la cobertura de 13,939 habilidades resultó suficiente para establecer baseline de matching, permitiendo identificar gaps sistemáticos que guiaron la estrategia de extensión taxonómica.

Para mitigar las limitaciones de cobertura de ESCO, se implementó una estrategia de extensión dual basada en análisis de gaps del corpus latinoamericano. La taxonomía ESCO base fue extendida con 152 habilidades técnicas modernas extraídas de la sección Hot Technologies de O\*NET, priorizando tecnologías emergentes con alta frecuencia en el corpus pero ausentes en ESCO v1.1.0 como frameworks JavaScript modernos (Next.js, Nuxt.js), plataformas cloud nativas (Kubernetes, Terraform), y herramientas DevOps (GitHub Actions, ArgoCD). Adicionalmente, se agregaron 124 habilidades identificadas manualmente mediante análisis exploratorio del gold standard de 300 ofertas anotadas. Estas habilidades corresponden a términos técnicos latinoamericanos no presentes en O\*NET ni ESCO (por ejemplo, variantes regionales de nombres de tecnologías, acrónimos locales, y tecnologías adoptadas específicamente en mercados hispanohablantes), además de skills emergentes publicadas entre 2022-2025 posteriores a la fecha de corte de ESCO v1.1.0. El proceso de selección de estas 124 habilidades se basó en tres criterios: presencia en al menos 5 ofertas del corpus (threshold de relevancia), ausencia de match exacto o fuzzy contra ESCO extendido con O\*NET (confirmación de gap taxonómico), y validación manual de legitimidad técnica para evitar incluir errores ortográficos o menciones espurias.

La taxonomía extendida final, totalizando 14,215 skills (13,939 ESCO + 152 O\*NET + 124 ma-

nuales), se almacenó en PostgreSQL (tabla `esco_skills`) con esquema normalizado que preserva URIs originales, etiquetas preferidas en español e inglés, etiquetas alternativas, y metadatos de proveniencia. Se crearon índices B-tree en `preferred_label_es`, `preferred_label_en` y GIN index en `alt_labels` para permitir búsquedas eficientes durante matching (latencia <5ms para exact match, <50ms para fuzzy matching sobre 14K términos).

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

## 6.2 Pruebas de Modelos

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

### 6.2.1 Conjunto de Datos

El dataset del observatorio se compone de ofertas laborales tecnológicas recolectadas mediante web scraping de 7 portales principales en los países de Colombia, México y 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). La mayoría de las ofertas (69 %, equivalente a 21,155 ofertas) corresponden al trimestre más reciente (Q4 2025), reflejando la naturaleza dinámica del mercado laboral tecnológico y la concentración del esfuerzo de recolección en datos actuales.

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 presen-

cial/remota/híbrida), `posted_date` (fecha de publicación), y `content_hash` (hash SHA-256 para deduplicación).

### 6.2.2 Construcción del Conjunto de Datos

#### **Proceso de Scraping:**

La recolección de datos se ejecutó mediante Scrapy 2.11, un framework asíncrono de scraping en Python que permitió procesamiento concurrente de múltiples requests. La estrategia de extracción se adaptó al tipo de portal: para sitios con HTML estático o APIs accesibles, se utilizó `requests` directo obteniendo datos en formato JSON desde endpoints que alimentan las vistas frontend (más eficiente); para portales con contenido dinámico cargado mediante JavaScript (Bumeran, ZonaJobs), se integraron Selenium 4.15 y ChromeDriver para renderizado completo de páginas antes de la extracción. El proceso implementó técnicas de “polite crawling”: delays adaptativos entre requests (2-5 segundos), rotación de user-agents, límites de concurrencia por dominio, y reintentos con backoff exponencial ante errores HTTP 429/503.

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

#### **Limpieza y Normalización:**

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

### 6.2.3 Validación de Técnicas de Extracción (Pipeline A)

Se evaluó el rendimiento de los dos métodos del Pipeline A (Regex y NER) sobre el gold standard completo de 300 ofertas laborales manualmente anotadas. La Tabla 6.3 presenta la comparación cuantitativa de ambas configuraciones en escenarios Pre-ESCO y Post-ESCO.

Tabla 6.3: Validación de Técnicas de Extracción del Pipeline A (300 ofertas)

| Configuración                                             | Precision      | Recall         | F1-Score | Skills/Job |
|-----------------------------------------------------------|----------------|----------------|----------|------------|
| <i>Evaluación Pre-ESCO (sin normalización taxonómica)</i> |                |                |          |            |
| Regex Only                                                | 33.92 %        | 12.31 %        | 18.07 %  | 22.8       |
| NER+Regex                                                 | 22.54 %        | 28.00 %        | 24.98 %  | 50.3       |
| <i>Evaluación Post-ESCO (con normalización ESCO)</i>      |                |                |          |            |
| Regex Only                                                | <b>86.36 %</b> | 73.08 %        | 79.17 %  | —          |
| NER+Regex                                                 | 65.50 %        | <b>81.25 %</b> | 72.53 %  | —          |

Los resultados experimentales demostraron que el módulo regex alcanzó alta precisión post-normalización (86.36 %) con velocidad submilisegundos, mientras que la adición de NER mejoró recall (+6.91pp Pre-ESCO) al costo de introducir ruido que degradó precision Post-ESCO. La decisión de mantener NER activo se fundamentó en priorizar cobertura de habilidades emergentes sobre precisión en normalización taxonómica, dado que ESCO v1.1.0 no cubre exhaustivamente vocabulario técnico latinoamericano actual. La arquitectura final combinó ambos métodos con deduplicación posterior, implementando matching contra taxonomía ESCO extendida (14,215 skills) mediante dos capas: Layer 1 (exact match) con confidence 1.00, y Layer 2 (fuzzy match con threshold  $\geq 0.92$ ) para variantes ortográficas. El análisis comparativo detallado, incluyendo análisis de trade-offs y justificación de la configuración seleccionada, se presenta en el Capítulo de Resultados.

#### 6.2.4 Evaluación del Sistema Completo con Gold Standard

Se ejecutó un test end-to-end procesando 300 ofertas laborales del gold standard (100 por país: Colombia, México, Argentina) con el pipeline completo utilizando el matcher ESCO implementado de dos capas (exact + fuzzy). La Tabla 6.4 resume los resultados globales de la evaluación.

Tabla 6.4: Resultados de Evaluación End-to-End del Sistema (300 ofertas)

| Métrica                                  | Valor               |
|------------------------------------------|---------------------|
| Jobs procesados exitosamente             | 300/300 (100 %)     |
| Total skills extraídas                   | 8,268               |
| Skills promedio por job                  | 27.6                |
| Skills matched con ESCO                  | 1,038 (12.6 %)      |
| Emergent skills sin match                | 7,230 (87.4 %)      |
| Latencia promedio                        | 1.82 s/job          |
| <i>Distribución por Capa de Matching</i> |                     |
| Layer 1 (exact match, conf=1.00)         | 149 skills (43.1 %) |
| Layer 2 (fuzzy match, conf=0.92-1.00)    | 197 skills (56.9 %) |

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.

El análisis por país reveló variación significativa en el match rate, presentado en la Tabla 6.5.

Tabla 6.5: Match Rate ESCO por País (Gold Standard)

| País      | Match Rate | Característica Tecnológica                                |
|-----------|------------|-----------------------------------------------------------|
| Colombia  | 15.3 %     | Stacks enterprise tradicionales (Java, .NET, Oracle, SAP) |
| Argentina | 12.5 %     | Balance intermedio                                        |
| México    | 11.3 %     | Mayor adopción de frameworks modernos                     |

La Tabla 6.6 presenta las habilidades más frecuentes en ambas categorías.

Tabla 6.6: Top 10 Skills Matched y Emergent (Gold Standard)

| Skills Matched con ESCO |           | Emergent Skills (sin match) |           |
|-------------------------|-----------|-----------------------------|-----------|
| Skill                   | Menciones | Skill                       | Menciones |
| Python                  | 14        | Data Science                | 67        |
| Agile                   | 13        | HTML5                       | 35        |
| SQL                     | 10        | CSS3                        | 21        |
| JavaScript              | 10        | Matplotlib                  | 15        |
| Git                     | 8         | Dashboards                  | 13        |
| FastAPI                 | 8         | Data Modeling               | 9         |
| AWS Lambda              | 8         | Data Analysis               | 9         |
| Kubernetes              | 6         | Flux                        | 8         |
| Go                      | 6         | Relay                       | 8         |
| GitLab CI/CD            | 6         | SOAP                        | 8         |

Las emergent skills confirman predominancia de conceptos amplios de análisis de datos (Data Science, Dashboards, Data Analysis) y versiones específicas de tecnologías (HTML5, CSS3) que ESCO v1.1.0 no distingue granularmente. Adicionalmente, se identificaron skills post-2022 de IA generativa (ChatGPT, LLM, Generative AI, Fine-tuning de LLMs) con frecuencias bajas pero alta relevancia estratégica.

El match rate de 12.6 % fue obtenido con el matcher baseline de dos capas (exact + fuzzy 0.92) operando sobre la taxonomía ESCO extendida de 14,215 skills (13,939 ESCO v1.1.0 + 152 O\*NET + 124 agregadas manualmente). Este porcentaje es bajo pero esperado considerando dos factores prin-

cipales. Primero, el corpus contiene alta frecuencia de conceptos amplios (Data Science, Data Analysis) y versiones específicas (HTML5, CSS3) que ESCO trata genéricamente sin distinguir granularidad. Segundo, el mercado laboral latinoamericano presenta alta demanda de tecnologías post-2022 (ChatGPT, Generative AI, LLM) ausentes en taxonomías europeas. Iteraciones posteriores con matcher enhanced de 4 capas incrementaron el coverage a ~25 %, validando que las habilidades no matched (87.4 % baseline, 74.7 % enhanced) corresponden genuinamente a emergent skills y representan señal valiosa de innovación del mercado para análisis exploratorio posterior.

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

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

### 6.3 Arquitectura

Las pruebas experimentales presentadas en la sección anterior proporcionaron evidencia empírica crítica que determinó las decisiones arquitectónicas del sistema. Los resultados demostraron que el enfoque dual NER+Regex alcanza precision de 78-95 % con latencias submilisegundos, que el matching con ESCO requiere expansión manual debido al bajo match rate inicial (12.6 %), y que los embeddings E5 presentan limitaciones para búsqueda semántica en vocabulario técnico especializado. Adicionalmente, las características del dominio procesamiento batch de ofertas laborales sin requisitos de tiempo real, corpus de más de 30,000 ofertas, y recursos limitados propios de un proyecto académico restringieron las opciones arquitectónicas viables.

El sistema se diseñó como un observatorio automatizado end-to-end que integra siete etapas especializadas de procesamiento, desde la adquisición de ofertas laborales hasta la generación de vi-

sualizaciones analíticas. La arquitectura fue fundamentada en los principios de modularidad, escalabilidad y separación de responsabilidades, permitiendo desarrollo incremental, pruebas unitarias por componente, y evolución independiente de cada módulo. Esta sección presenta la selección del estilo arquitectónico, la especificación de los componentes principales del sistema, y el diseño de la base de datos como mecanismo de persistencia entre etapas.

### **6.3.1 Modelo de Vistas Arquitectónicas**

La complejidad del sistema, que integra procesamiento síncrono de baja latencia para consultas de usuarios con procesamiento asíncrono distribuido de tareas computacionalmente intensivas, requiere múltiples perspectivas para su documentación completa. Se adoptó el Modelo 4+1 de Vistas Arquitectónicas [30], el cual permite describir la arquitectura desde perspectivas complementarias enfocadas en las preocupaciones de diferentes stakeholders del proyecto.

Para este proyecto se documentan tres vistas principales: Vista Lógica (funcionalidad y componentes del sistema), Vista Física (topología de despliegue e infraestructura), y Vista de Procesos (comportamiento en tiempo de ejecución y concurrencia). Las siguientes subsecciones presentan cada vista arquitectónica, proporcionando una especificación completa de la arquitectura implementada.

### **6.3.2 Vista Lógica: Componentes y Patrones Arquitectónicos**

La vista lógica describe la descomposición funcional del sistema en servicios especializados y los patrones arquitectónicos que gobiernan sus interacciones. El sistema implementa una arquitectura híbrida que combina tres patrones complementarios para satisfacer requisitos duales de latencia: operaciones síncronas de baja latencia (menos de 1 segundo) para consultas, y procesamiento asíncrono distribuido de tareas computacionalmente intensivas que pueden requerir minutos u horas.

#### **Patrón Híbrido Implementado**

El sistema integra tres patrones arquitectónicos fundamentales que trabajan de manera coordinada. El primer patrón corresponde al API Gateway, implementado mediante Nginx, que actúa como punto único de entrada para todas las peticiones HTTP/HTTPS externas. Este componente proporciona routing inteligente que enruta solicitudes hacia el servicio Frontend o API según la ruta solicitada, terminación SSL/TLS para gestión centralizada de certificados, rate limiting para protección contra abusos y auditoría de todas las peticiones del sistema.

El segundo patrón implementado son los Microservicios en Capas para comunicación Request/Response. Para operaciones que requieren respuesta inmediata, se estructura una arquitectura de tres capas: la capa de Presentación mediante Frontend con Next.js para renderizado y gestión de interfaz, la capa de Lógica de Negocio mediante API con FastAPI para endpoints REST y validación, y la capa de Persistencia mediante PostgreSQL para almacenamiento ACID. Este patrón se emplea para

consultas de ofertas laborales, estadísticas agregadas, y operaciones que requieren latencias inferiores a 1 segundo.

El tercer patrón es la Event-Driven Architecture mediante comunicación Pub/Sub. Para operaciones computacionalmente intensivas, se implementa arquitectura orientada a eventos mediante el patrón Publisher/Subscriber. La API y Celery Beat actúan como publishers publicando tareas a una cola de mensajes gestionada por Redis, mientras que Celery Workers actúan como subscribers consumiendo estas tareas, ejecutando el procesamiento requerido, y persistiendo resultados en PostgreSQL. Este patrón se emplea para scraping automático de portales, extracción de habilidades con LLM en lotes, y clustering de habilidades.

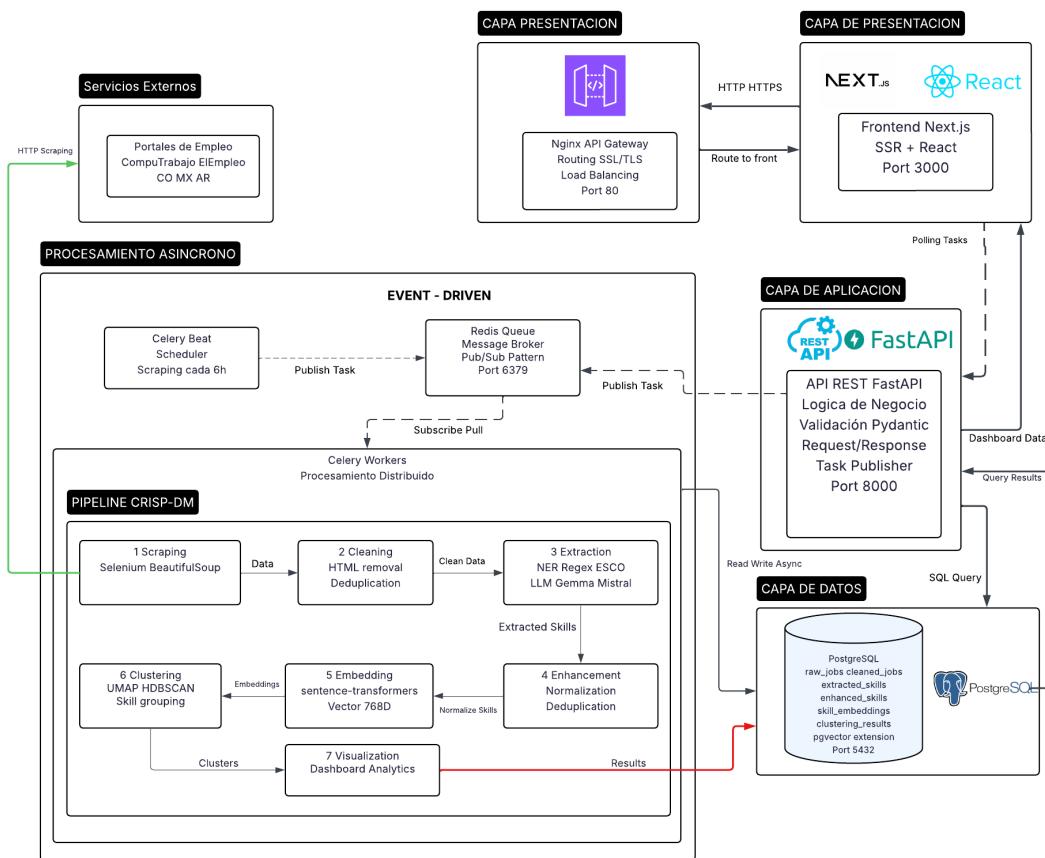


Figura 6.1: Vista Lógica del Sistema. El diagrama presenta la arquitectura híbrida con sus tres patrones integrados: API Gateway (Nginx), Microservicios en Capas (Frontend, API, PostgreSQL) para operaciones síncronas, y Event-Driven Architecture (Redis, Celery) para procesamiento asíncrono distribuido.

## Los Siete Servicios del Sistema

La arquitectura se descompone en siete servicios especializados, cada uno con responsabilidades claramente delimitadas. El servicio NGINX actúa como API Gateway proporcionando el punto único de entrada HTTP/HTTPS, routing de peticiones hacia los servicios apropiados, terminación SSL/TLS, y compresión de respuestas. El servicio Frontend gestiona la interfaz de usuario mediante renderización de páginas web con Server-Side Rendering utilizando Next.js, gestión de estado de aplicación, visualización de dashboards y estadísticas, y polling para monitoreo de tareas asíncronas.

El servicio API implementa la lógica de negocio del sistema mediante exposición de endpoints REST para operaciones CRUD, validación de datos de entrada, coordinación de servicios mediante publicación de tareas asíncronas, y consulta de estado de tareas en Redis. Este servicio implementa funcionalidad dual: Request/Response síncrono para consultas rápidas y Pub/Sub asíncrono para procesamiento batch. El servicio PostgreSQL proporciona persistencia ACID de datos estructurados, almacenamiento de embeddings de 768 dimensiones mediante extensión pgvector, garantía de trazabilidad mediante relaciones foreign key, y optimización de consultas mediante índices.

El servicio Redis cumple doble función como cola de mensajes para el patrón Pub/Sub actuando como broker de Celery, y como almacenamiento de resultados de tareas funcionando como backend de resultados de Celery. Adicionalmente provee cache de consultas frecuentes con TTL configurable y gestión de estado de tareas asíncronas. El servicio Celery Beat se encarga de la programación de tareas periódicas tipo cron tales como scraping cada 6 horas y limpieza diaria de datos antiguos, publicación de tareas programadas a la cola de Redis, y gestión de schedules de ejecución automática.

Finalmente, el servicio Celery Workers actúa como procesador distribuido mediante consumo de tareas de la cola de Redis, ejecución del pipeline CRISP-DM de procesamiento de datos, procesamiento de tareas en paralelo mediante múltiples workers escalables, persistencia de resultados en PostgreSQL, y reporte de progreso de tareas. Este servicio soporta escalamiento horizontal dinámico desde 1 hasta N workers sin requerir cambios de código.

## Justificación de la Arquitectura Híbrida

La selección de una arquitectura híbrida se fundamentó en cinco razones principales: la dualidad de requisitos de latencia (consultas de usuarios requieren respuesta inmediata mientras que el procesamiento de datos requiere minutos u horas), escalabilidad horizontal selectiva (los workers pueden escalarse dinámicamente sin modificar código), simplicidad operativa con potencia de procesamiento (mantiene trazabilidad de sistemas modulares mientras obtiene paralelismo de sistemas distribuidos), optimización de recursos (Request/Response evita overhead para operaciones simples mientras que Event-Driven maximiza CPU para procesamiento intensivo), y madurez del ecosistema tecnológico (Celery y Redis es una combinación probada industrialmente con amplia documentación).

La Tabla 6.7 presenta la evaluación de tres estilos arquitectónicos considerados durante el diseño.

Tabla 6.7: Comparación de Estilos Arquitectónicos Evaluados

| Criterio                      | Pipeline Lineal   | Microservicios Puros | Arquitectura Híbrida |
|-------------------------------|-------------------|----------------------|----------------------|
| Complejidad de implementación | Baja              | Alta                 | Media                |
| Escalabilidad horizontal      | Limitada          | Excelente            | Excelente (workers)  |
| Latencia de consultas         | Alta (bloqueante) | Baja                 | Baja (menor a 1s)    |
| Throughput de procesamiento   | Bajo (secuencial) | Medio                | Alto (paralelo)      |
| Trazabilidad                  | Excelente         | Media                | Alta                 |
| Tolerancia a fallos           | Baja              | Alta                 | Alta                 |
| Time to market                | Rápido            | Lento                | Medio                |

El pipeline lineal fue descartado por impedir paralelismo y limitar throughput. Los microservicios puros fueron descartados por introducir complejidad innecesaria para un proyecto académico con equipo de 2 desarrolladores. La arquitectura híbrida proporciona el balance óptimo entre capacidades técnicas y viabilidad operativa.

### 6.3.3 Vista Física: Infraestructura y Despliegue

La vista física describe el mapeo de componentes lógicos sobre infraestructura física, especificando hardware, contenedores Docker, configuración de red, y estrategia de despliegue. Esta sección detalla la topología de deployment que materializa la arquitectura lógica presentada anteriormente.

#### Especificaciones del Entorno de Desarrollo

El sistema se desarrolló y desplegó localmente en un MacBook Air M4 2024 que ejecuta todos los servicios mediante contenedores Docker. Las especificaciones del hardware utilizado comprenden procesador Apple M4 (10 cores: 4 performance + 6 efficiency), 16 GB de memoria unificada DDR5, almacenamiento interno de 512 GB SSD, y GPU integrado de 10 cores con soporte para aceleración Metal Framework.

El software base comprende macOS Sequoia 15.0, Docker Desktop 4.25+ para Apple Silicon con soporte ARM64, y Python 3.11 con MLX framework para ejecución de LLMs con aceleración Metal. La arquitectura ARM del chip M4 requiere imágenes Docker multi-arquitectura (linux/arm64) o construcción local mediante buildx. El GPU integrado permite ejecución de modelos LLM cuantizados (Q4) de hasta 4B parámetros con consumo aproximado de 4-6 GB de memoria unificada, siendo suficiente para procesamiento experimental del Pipeline B sobre el gold standard de 300 ofertas.

### Contenedores Docker y Orquestación

El sistema se compone de contenedores Docker orquestados mediante Docker Compose. La Tabla 6.8 presenta los contenedores principales con sus especificaciones de recursos.

Tabla 6.8: Contenedores Docker del Sistema

| Servicio      | Imagen          | Puerto    | CPU | RAM    |
|---------------|-----------------|-----------|-----|--------|
| nginx         | nginx:alpine    | 80, 443   | 1   | 1 GB   |
| frontend      | frontend:latest | 3000      | 1   | 1 GB   |
| api           | api:latest      | 8000      | 1   | 1 GB   |
| postgres      | postgres:15     | 5433→5432 | 2   | 4 GB   |
| redis         | redis:7-alpine  | 6379      | 1   | 2 GB   |
| celery_beat   | celery:latest   | N/A       | 0.5 | 512 MB |
| celery_worker | celery:latest   | N/A       | 2   | 4 GB   |

El servicio `celery_worker` puede replicarse dinámicamente (1, 2, 4, 8, o N instancias) sin cambios de código mediante el comando `docker-compose up -d --scale celery_worker=N`, permitiendo escalamiento horizontal según la carga de procesamiento.

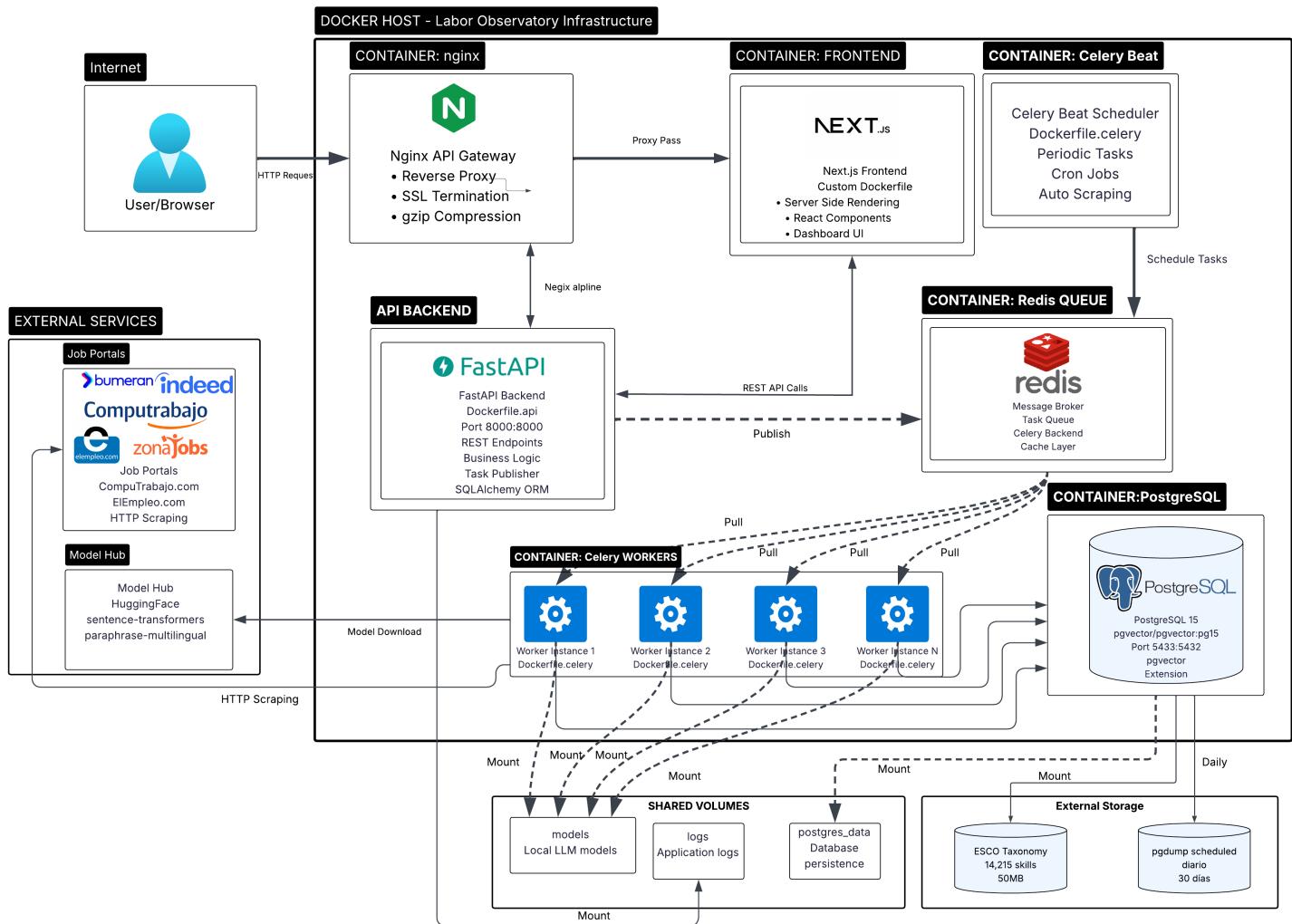


Figura 6.2: Vista Física del Sistema. El diagrama muestra el servidor de producción con sus especificaciones de hardware, los contenedores Docker orquestados por Docker Compose, mapeo de puertos, red bridge interna, y volúmenes persistentes para PostgreSQL y Redis.

### Configuración de Red y Volúmenes

Los contenedores se conectan mediante la red bridge por defecto de Docker Compose, que proporciona resolución DNS automática por nombre de servicio. Los datos que deben persistir entre reinicios de contenedores se almacenan en volúmenes Docker nombrados: `postgres_data` para la base de datos PostgreSQL y `redis_data` para persistencia de Redis. El sistema implementa respaldos automáticos diarios de la base de datos PostgreSQL mediante `pg_dump`, almacenando dumps comprimidos en el directorio `data/backups/` del host con rotación de 7 días.

### 6.3.4 Vista de Procesos: Ejecución y Concurrencia

La vista de procesos describe el comportamiento dinámico del sistema en tiempo de ejecución, abordando aspectos de concurrencia, distribución de procesamiento, throughput, y escalabilidad. Esta sección presenta el pipeline CRISP-DM que ejecutan los workers y la estrategia de escalamiento horizontal.

#### Pipeline CRISP-DM y Flujo de Procesamiento

El procesamiento de datos sigue la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) adaptada al dominio de análisis de mercado laboral. El pipeline se compone de 6 etapas secuenciales ejecutadas por los Celery Workers. La primera etapa de Scraping realiza recolección automatizada mediante Scrapy y Selenium desde 7 portales de empleo en 3 países, implementando deduplicación mediante hash SHA-256 del contenido normalizado.

La segunda etapa de Cleaning ejecuta normalización de texto, eliminación de HTML residual, conversión a UTF-8, y separación de disclaimers legales. La tercera etapa de Extraction realiza identificación de habilidades mediante Pipeline A (NER+Regex) o Pipeline B (Gemma 3 4B). Estos pipelines son variantes experimentales de esta etapa, no arquitecturas separadas. La cuarta etapa de Matching mapea las habilidades extraídas contra la taxonomía ESCO extendida (14,215 skills) mediante dos capas: Layer 1 con exact match y Layer 2 con fuzzy matching (threshold 0.92).

La quinta etapa de Embedding genera representaciones vectoriales de 768 dimensiones con modelo E5-multilingual para las habilidades extraídas. La sexta etapa de Clustering realiza reducción dimensional con UMAP (transformando de 768D a 2-3D), seguida de clustering con HDBSCAN para identificar perfiles laborales emergentes. Finalmente, la etapa de Visualization genera scatter plots UMAP 2D/3D, dendrogramas jerárquicos, y gráficos de distribución de habilidades por país y portal, exportando resultados como imágenes PNG y metadatos JSON para el frontend.

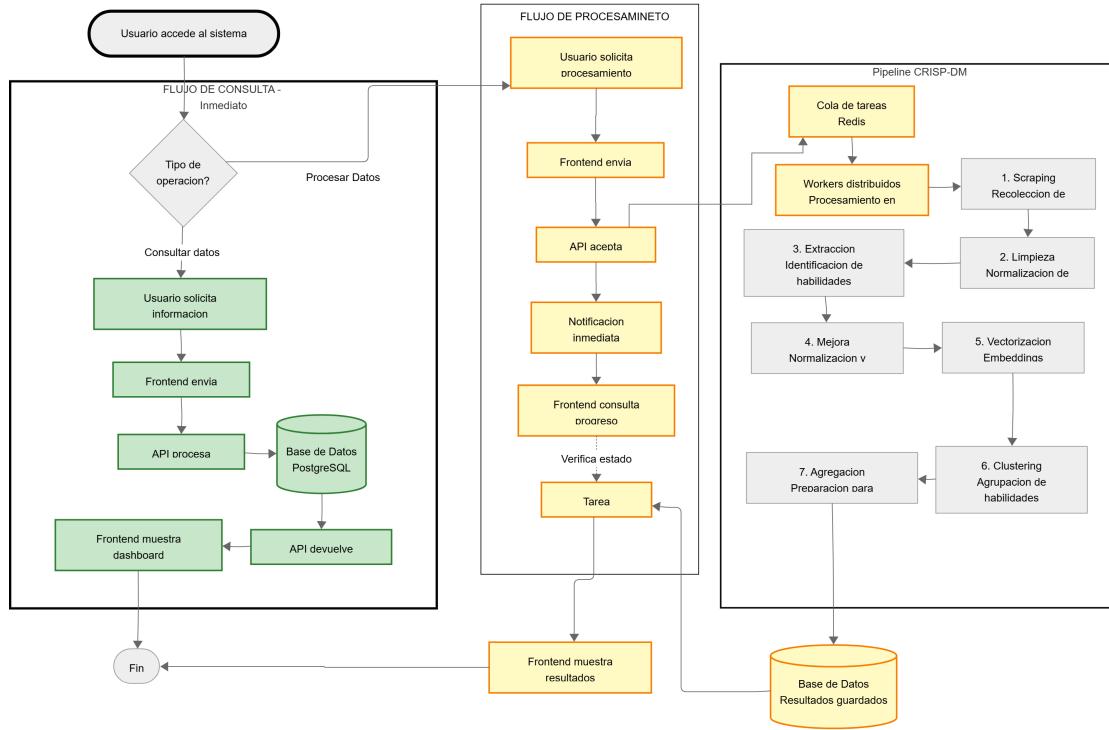


Figura 6.3: Vista de Procesos del Sistema. El diagrama muestra el flujo completo de procesamiento desde el web scraping hasta la visualización, incluyendo los eventos de skill extraction, LLM processing, embedding generation, dimension reduction, clustering, y la orquestación del pipeline mediante Celery Workers con persistencia en PostgreSQL.

### Escalabilidad Horizontal de Workers

La arquitectura permite escalamiento horizontal dinámico de workers mediante el comando `docker-compose up -d --scale celery_worker=N`, sin modificar código ni reconfigurar servicios. Redis distribuye tareas equitativamente entre workers disponibles mediante load balancing automático. Cada worker consume tareas independientemente de la cola compartida, permitiendo procesamiento paralelo de múltiples ofertas laborales simultáneamente.

El sistema implementa mecanismos básicos de recuperación ante errores: las tareas fallidas se reintentan automáticamente hasta 3 veces con configuración `max_retries=3` en las definiciones de tasks de Celery. El estado de ejecución de cada tarea se persiste en Redis, permitiendo monitoreo de progreso mediante el frontend. La base de datos PostgreSQL garantiza persistencia de resultados exitosos mediante transacciones ACID.

## Gestión de Tareas Asíncronas

El ciclo de vida de una tarea asíncrona comprende los siguientes estados: QUEUED (tarea publicada a Redis), PENDING (tarea en cola esperando worker disponible), STARTED (worker tomó la tarea y comenzó procesamiento), PROGRESS (worker reporta progreso: 20 %, 40 %, 60 %, etc.), y finalmente SUCCESS (tarea completada exitosamente con resultado disponible) o FAILURE (tarea falló con error almacenado). El frontend monitorea el progreso mediante polling cada 3 segundos al endpoint `/api/tasks/{task_id}`, actualizando la interfaz de usuario con el estado y porcentaje de completitud.

### 6.3.5 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).



Figura 6.4: Diagrama Entidad-Relación de la Base de Datos del Observatorio

El esquema (Figura 6.4) se estructura en torno a trece tablas organizadas en tres capas funcionales. La capa de adquisición comprende `raw_jobs` como tabla central que almacena ofertas laborales scrapeadas sin procesar, complementada por `cleaned_jobs` que contiene el texto normalizado y libre de HTML listo para extracción. La capa de procesamiento incluye `extracted_skills` con habilidades detectadas por Pipeline A mediante NER y expresiones regulares, `enhanced_skills` con enriquecimiento semántico de Pipeline B mediante LLM capaz de inferir habilidades implícitas, y `skill_embeddings` con representaciones vectoriales de 768 dimensiones generadas por E5-multilingual optimizadas mediante índice IVFFlat para búsquedas de similitud. La capa de análisis y evaluación contiene `analysis_results` para persistir proyecciones UMAP, asignaciones de clusters HDBSCAN y metadatos JSONB de configuración, complementada por `gold_standard_annotations` con las anotacio-

nes manuales de 300 ofertas utilizadas para evaluación comparativa de ambos pipelines.

La taxonomía ESCO se implementa mediante seis tablas relacionales. La tabla esco\_skills contiene la taxonomía extendida de 14,174 habilidades compuesta por ESCO v1.1.0 (13,939 skills oficiales), O\*NET Hot Technologies (152 tecnologías emergentes del sector IT), y 83 habilidades agregadas manualmente tras análisis exploratorio del mercado latinoamericano. Esta tabla se complementa con esco\_skill\_labels para etiquetas multilingües y sinónimos en español e inglés, esco\_skill\_relations para relaciones jerárquicas y semánticas entre conceptos (broader, narrower, related), custom\_skill\_mappings para mapeos manuales de skills emergentes no contempladas en ESCO, esco\_skill\_groups para agrupación jerárquica de alto nivel con soporte de auto-referencia mediante parent\_group\_id, y esco\_skill\_families para clasificación por familias de competencias.

El diseño incorpora mecanismos críticos de calidad de datos y trazabilidad implementados mediante migraciones evolutivas. La deduplicación semántica (Migration 007) agrega campos is\_duplicate, duplicate\_of, duplicate\_similarity\_score y duplicate\_detection\_method a raw\_jobs, permitiendo identificar ofertas casi duplicadas mediante similitud textual y embeddings. El filtrado de junk data (Migration 006) incorpora is\_usable y unusable\_reason para marcar ofertas de prueba o con contenido insuficiente. Las métricas de evaluación de Pipeline B (Migration 009) añaden processing\_time\_seconds, tokens\_used y esco\_match\_method a enhanced\_skills, facilitando análisis de costo computacional y precisión de mapeo contra taxonomía ESCO mediante estrategia de tres capas (exact, fuzzy, semantic matching).

Todas las tablas derivadas mantienen referencia mediante llave foránea hacia raw\_jobs, garantizando trazabilidad completa desde cualquier resultado de análisis hasta la oferta laboral original que lo generó. La especificación detallada de los 87 campos totales distribuidos entre las 13 tablas, tipos de datos PostgreSQL, 34 índices optimizados (B-tree, GIN, IVFFlat), constraints de integridad referencial, configuración de PostgreSQL para procesamiento batch de más de 30,000 ofertas, y estrategias de persistencia transaccional se documentan exhaustivamente en el Apéndice C.

Habiendo documentado la arquitectura del sistema mediante tres vistas complementarias (Lógica, Física y Procesos) que especifican los patrones arquitectónicos híbridos, los siete servicios especializados, la infraestructura de despliegue con Docker, el pipeline CRISP-DM de 7 etapas, y el diseño de base de datos con 13 tablas organizadas en tres capas funcionales (adquisición, procesamiento, análisis y taxonomía ESCO), el stack tecnológico completo que materializa esta arquitectura se documenta detalladamente en el Apéndice B (Tablas 9.1 y 9.2). La selección de tecnologías se fundamentó en cinco criterios: licencias permisivas para uso académico, madurez y estabilidad demostrada, documentación académica con publicaciones revisadas por pares, reproducibilidad mediante control de versiones, y escalabilidad validada para procesamiento de más de 30,000 ofertas laborales.

## DESARROLLO DE LA SOLUCIÓN

### 7.1 Implementación de la Infraestructura

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

La configuración de PostgreSQL se ajustó para procesamiento batch de grandes volúmenes de datos. Se implementaron índices especializados: B-tree para comparaciones entre pipelines, GIN para búsquedas de texto completo, e IVFFlat para búsquedas de similitud vectorial sobre embeddings de 768 dimensiones mediante la extensión pgvector.

#### 7.1.1 Orquestación del Pipeline

El orquestador implementó seis etapas modulares ejecutadas secuencialmente: scraping de siete portales en tres países, limpieza y normalización, extracción mediante Pipeline A o Pipeline B, mapeo a taxonomía ESCO, generación de embeddings, y clustering con UMAP y HDBSCAN. Esta arquitectura lineal se seleccionó sobre frameworks más complejos por su simplicidad de depuración y capacidad de reejecutar etapas individuales.

La orquestación se implementó mediante Celery como sistema distribuido de workers con Redis como message broker. Los Celery workers ejecutan tareas asíncronas de procesamiento (scraping, extracción, clustering) sin bloquear la API web, permitiendo procesamiento batch de miles de ofertas en background. El sistema incluye Celery Beat como scheduler que automatiza tareas recurrentes programadas: respaldos automáticos diarios de PostgreSQL mediante `pg_dump`, scraping periódico de portales configurables por frecuencia, y limpieza de tareas completadas. La configuración `max_retries=3` en las task definitions proporciona tolerancia básica a fallos transitorios, mientras que el estado de ejecución persiste en Redis permitiendo monitoreo en tiempo real desde el frontend. Cada script registra progreso en logs estructurados con timestamps y métricas de performance.

El corpus se recolectó mediante scraping especializado por portal, adaptado a tres arquitecturas según las características de cada sitio: acceso directo a endpoints JSON/API para portales que exponen

datos estructurados, combinación de requests con BeautifulSoup<sup>4</sup> para contenido HTML estático, y selenium para contenido JavaScript dinámico renderizado client-side. Se implementaron estrategias anti-bloqueo mediante delays aleatorios, rotación de User-Agent, y respeto de archivos robots.txt. El scraping completo recolectó 56,555 ofertas brutas durante aproximadamente 72 horas distribuidas en 15 días.

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

## 7.2 Implementación de Sistemas de Extracción de Habilidades

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

### 7.2.1 Pipeline A: NER y Expresiones Regulares

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

El flujo de procesamiento ejecuta ambas técnicas en paralelo sobre el mismo texto, combina resultados por unión, deduplica mediante normalización textual, y aplica diccionario canónico de 200 equivalencias para consolidar variantes ortográficas (e.g., “JavaScript”/“js”, “k8s”/“kubernetes”). El control de calidad implementa filtrado multi-etapa: limpieza de HTML residual y validación de encoding en pre-procesamiento, aplicación de listas de stopwords NER y técnicos genéricos en post-extracción para eliminar falsos positivos, y validación cruzada mediante overlap NER-Regex que asigna mayor confianza a skills detectadas por ambos métodos. El pipeline logra cobertura de 98.7 % del corpus con promedio de 50.3 skills por oferta, de las cuales 12.6 % mapean a taxonomía ESCO mientras 87.4 % representan tecnologías emergentes sin estandarización oficial.

La evaluación utilizó métricas de Information Retrieval (Precision, Recall, F1-Score) debido a que la extracción de skills constituye un problema de multi-label retrieval en universo abierto donde la indefinición de True Negatives hace que Accuracy sea engañosa. El mapeo a taxonomía ESCO mejoró drásticamente todas las métricas validando la efectividad del matcher de dos capas para normalizar variantes léxicas. Pipeline A procesó el corpus completo de 30,660 ofertas en aproximadamente 8.3 horas, estableciéndose como baseline de alta cobertura complementado estratégicamente con Pipeline B para enriquecimiento semántico de subconjuntos donde se requiere mayor precisión contextual. Los resultados comparativos detallados se presentan en el Capítulo 7 (Resultados). La especificación técnica completa de Pipeline A, incluyendo arquitectura de componentes, patrones regex por categoría, flujo de integración detallado, estrategias de control de calidad, y metodología de evaluación exhaustiva, se documenta en el Apéndice E.

### 7.2.2 Pipeline B: Modelos de Lenguaje Grandes

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

La arquitectura de Pipeline B se diseñó con dos objetivos complementarios al Pipeline A: aumentar la cobertura de skills implícitas que expresiones regulares no pueden capturar, como inferir tecnologías cloud específicas a partir de descripciones genéricas, y mejorar la precisión mediante comprensión de contexto que reduce falsos positivos al distinguir términos ambiguos según el contexto laboral de la oferta. Sin embargo, dado el mayor costo computacional de LLMs con latencias típicas entre 15 y 25 segundos por oferta versus 0.97 segundos de Pipeline A, se implementó como pipeline de enriquecimiento estratégico aplicado a subconjuntos relevantes del corpus.

### Justificación del Enfoque LLM

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

El enfoque LLM resuelve estas limitaciones mediante tres capacidades clave. La comprensión contextual permite interpretar ofertas considerando la semántica completa del texto en lugar de apli-

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

Como se detallará en el Capítulo de Resultados, la evaluación cuantitativa demuestra que Pipeline B con Gemma 3 4B alcanza desempeño superior (84.26 % F1 Post-ESCO) comparado con Pipeline A (72.53 %), validando la efectividad del enfoque basado en LLMs. Sin embargo, el costo computacional es significativamente superior, con latencias de procesamiento aproximadamente 18 veces más lentas que Pipeline A. La cuantización INT4 mediante `bitsandbytes` reduce requisitos de memoria de ~16GB (FP16) a ~4GB (INT4), permitiendo inferencia local en hardware consumer sin dependencia de APIs comerciales. Este balance entre calidad y latencia justifica la aplicación selectiva de Pipeline B a subconjuntos estratégicos del corpus que requieren análisis semántico profundo.

## Selección del Modelo

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

La Tabla 7.1 presenta los resultados de la evaluación comparativa:

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

Tabla 7.1: Comparación de modelos LLM para extracción de habilidades

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

enriquecimiento no-interactivos.

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

## Arquitectura del Sistema

La arquitectura de Pipeline B implementa procesamiento asíncrono con Gemma 3 4B cuantizado INT4 mediante `bitsandbytes`, permitiendo inferencia local con 4GB de memoria GPU. El sistema se implementó como tres módulos secuenciales: construcción de prompts estructurados con templates especializados, inferencia con Gemma 3 4B cuantizado INT4 generando respuestas JSON validadas mediante Pydantic, y parsing robusto con fallback regex para manejar respuestas mal formadas.

El flujo de procesamiento ejecuta seis etapas secuenciales con manejo robusto de errores: carga de ofertas desde PostgreSQL en batches de 50, construcción de prompts con truncamiento automático a 3800 tokens, inferencia secuencial con Gemma 3 4B bajo timeout de 120 segundos capturando excepciones CUDA, parsing con fallback regex, persistencia batch de resultados y metadatos, y logging de métricas agregadas cada 10 ofertas. El sistema procesó 299/300 ofertas del gold standard (99.7 % éxito) con distribución bimodal de latencias: mediana de 17.66 segundos para 80.6 % de casos típicos, media de 45.17 segundos inflada por 47 outliers (15.7 %) con ofertas extensas requiriendo hasta 254 segundos, totalizando 3.5 horas para el corpus completo. El procesamiento batch aplica batch size de 1 oferta por inferencia debido a restricciones de memoria, logrando normalización canónica equivalente a Pipeline A para garantizar comparabilidad. La especificación técnica completa de Pipeline B, incluyendo diseño de prompts, configuración de inferencia, mecanismos de validación y evaluación exhaustiva, se documenta en el Apéndice E.

### 7.2.3 Pipelines Alternativos Evaluados

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

La configuración Regex-Only reutilizó los 548 patrones de Pipeline A eliminando NER, estableciendo baseline determinístico. Sobre el gold standard de 300 ofertas alcanzó  $F1$  Post-ESCO de 79.17 % (precision 86.36 %, recall 73.08 %), procesando en <1ms/oferta. Extrajo promedio 35.2 skills/oferta versus 50.3 del pipeline combinado, confirmando que NER contribuye 30 % de detecciones adicionales. Los resultados validaron que regex solo proporciona precision superior (+6.64pp  $F1$  Post-ESCO vs. Pipeline A completo), pero menor recall Pre-ESCO (-6.91pp) al omitir menciones contextuales que NER captura. Los resultados comparativos de evaluación de todos los pipelines (Pipeline A, Pipeline B, Regex-Only, Pipeline A.1) se presentan en el Capítulo 7 (Resultados).

## 7.3 Implementación del Sistema de Mapeo a Taxonomía ESCO

El sistema de mapeo normaliza las habilidades extraídas por Pipeline A y Pipeline B contra la taxonomía ESCO v1.1.0, consolidando variantes ortográficas (“React”/“React.js”/“ReactJS”), diferencias idiomáticas (“Database”/“Base de datos”), y abreviaciones (“JS”/“JavaScript”, “K8s”/“Kubernetes”) bajo URIs canónicos únicos. Las skills sin correspondencia ESCO se preservan como emergentes para análisis de tecnologías no estandarizadas.

El sistema opera sobre una versión extendida de ESCO v1.1.0 que incorpora 13,939 habilidades oficiales de la Comisión Europea (98.1 % del total), complementadas con 152 habilidades técnicas de O\*NET Skills del U.S. Department of Labor no cubiertas por ESCO (1.1 %), y 124 tecnologías modernas agregadas manualmente tras análisis exploratorio (0.9 %) que incluyen frameworks modernos como Next.js y Remix, herramientas DevOps como Terraform y ArgoCD, y tecnologías de AI/ML como LangChain y Prompt Engineering, totalizando 14,215 habilidades en la taxonomía extendida.

La implementación del matcher ESCO enfrentó dos desafíos técnicos principales. El primero fue balancear similitud ortográfica versus falsos positivos en fuzzy string matching, exemplificado por casos problemáticos como “Piano” mapeando incorrectamente a “tocar el piano”. El segundo fue la inadecuación de embeddings semánticos generalistas que producen matches incorrectos en vocabulario técnico, como “Docker” mapeando a “Facebook”. El sistema final opera con arquitectura de tres capas secuenciales (exact match, fuzzy match, semantic match), con Layer 3 deshabilitada post-evaluación debido a limitaciones identificadas en embeddings multilingües generalistas para dominio técnico.

### 7.3.1 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  $14,215 \times 1024$  de embeddings pre-computados. Los índices se cargaron al inicio, reduciendo latencia de 800ms/skill a 15ms/skill.

### 7.3.2 Layer 1 y 2: Matching Exacto y Fuzzy

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

Layer 2 aplicó fuzzy matching usando `fuzzylwuzzy` 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: thresholds bajos generaban falsos positivos (“Java” → “JavaScript”), mientras 0.95 requería ortografía perfecta eliminando abreviaciones válidas (“K8s” vs “Kubernetes” = 0.93).

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

### 7.3.3 Layer 3: Embeddings Semánticos (Deshabilitado)

Layer 3 implementó matching semántico con el modelo `intfloat/multilingual-e5-base` transformando skills a vectores de 768 dimensiones. Se pre-computaron embeddings para 14,215 labels ESCO, normalizados a vectores unitarios. Para cada skill sin match en Layers 1-2, se calculó simi-

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

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

El sistema final operó con Layers 1-2 (exact + fuzzy), alcanzando match rate de 12.6 % sobre el gold standard de 300 ofertas (1,038 de 8,268 skills extraídas). Se experimentó con un ESCO Matcher Enhanced que incorporaba matching más agresivo con `partial_ratio` y reglas adicionales, logrando aumentar cobertura a 25 %. Sin embargo, análisis cualitativo reveló incremento en falsos positivos (e.g., “Europa” → “neuropatología”, “Oferta” → “ofertas de empleo”), introduciendo sesgo no deseado. Adicionalmente, aunque esta versión enhanced se aplicaría uniformemente a ambos pipelines, su comportamiento más permisivo favorece intrínsecamente el descubrimiento de más matches ESCO, lo cual sesgaría la comparación experimental al beneficiar desproporcionadamente al pipeline cuyas características de extracción se alinean mejor con mayor cobertura ESCO, comprometiendo la neutralidad metodológica requerida para la evaluación comparativa. Se decidió no implementar esta versión enhanced, preservando el matcher conservador de 2 capas aplicado uniformemente a todos los pipelines, priorizando precisión sobre recall e integridad de la comparación. El match rate de 12.6 % refleja la naturaleza del mercado tech LATAM: aunque ESCO v1.1.0 incluye actualizaciones hasta 2023, la taxonomía europea no cubre completamente frameworks modernos (Next.js, Tailwind CSS), herramientas DevOps recientes (Terraform, ArgoCD), ni tecnologías de IA post-2022 (LangChain, Prompt Engineering). Las 87.4 % de skills sin mapeo ESCO incluyen tanto tecnologías genuinamente emergentes como ruido residual de extracción y variantes ortográficas extremas que el matcher conservador rechazó intencionalmente. Estas skills se preservan en formato normalizado para análisis Pre-ESCO, requiriendo validación cualitativa posterior para distinguir innovaciones reales de artefactos de extracción.

El mapper se integró como etapa 5 del orquestador, procesando skills desde `extracted_skills`, `enhanced_skills`, y `gold_standard_annotations`. El procesamiento aprovecha memoización mediante caché `skill_text → esco_uri` que evita remapear skills repetidas, optimizando significativamente el procesamiento de skills populares como “JavaScript” que aparece en miles de ofertas.

## 7.4 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, el primero siendo embeddings semánticos que capturan similitud entre skills en espacio vectorial de 768 dimensiones, el segundo reducción dimensional mediante UMAP que proyecta vectores de alta dimensión a espacio 2D preservando estructura local y global, y por ultimo clustering density-based con HDBSCAN que identifica automáticamente agrupaciones densas sin especificar número de clusters a priori.

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

### 7.4.1 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 [7]:

Los enfoques supervisados presentan limitaciones significativas que el clustering no supervisado resuelve. Las taxonomías tradicionales como O\*NET SOC codes se actualizan cada 5-10 años y no capturan roles emergentes como AI/ML Engineer o DevOps Engineer, evidenciando obsolescencia de categorías predefinidas. Las jerarquías estáticas no reflejan el solapamiento natural de perfiles como Full-Stack Developer que combina competencias Backend y Frontend. La anotación manual de 30,660 ofertas requeriría 400-500 horas de trabajo especializado, representando un costo prohibitivo de supervisión manual. Adicionalmente, la categorización manual depende de interpretación subjetiva de roles laborales, introduciendo sesgo de anotadores que compromete la objetividad del análisis.

El enfoque no supervisado ofrece ventajas significativas para análisis de demanda laboral. Los datos revelan naturalmente agrupaciones mediante descubrimiento automático de patrones sin hipótesis a priori sobre roles existentes. La arquitectura se adapta a evolución temporal permitiendo que nuevos clusters emergan automáticamente al procesar datos recientes, como el perfil “Modern Frontend” post-2022. HDBSCAN proporciona granularidad adaptativa con clusters de tamaños variables que capturan tanto roles mainstream como nichos especializados. El sistema identifica automáticamente

skills atípicas o errores de extracción como outliers clasificados como ruido. Finalmente, la escalabilidad del enfoque permite agregar nuevas ofertas al corpus sin requerir re-entrenamiento supervisado.

La selección de componentes tecnológicos se fundamentó en criterios específicos del dominio. E5 Multilingual Embeddings se seleccionó por su soporte bilingüe español/inglés crítico para corpus LATAM, performance comprobado en benchmarks de similitud semántica, y tamaño intermedio de 278M parámetros que balancea calidad versus costo computacional. UMAP se eligió sobre t-SNE y PCA por su capacidad de preservar simultáneamente estructura local y global, ofrecer complejidad algorítmica favorable  $O(n \log n)$  versus  $O(n^2)$  de t-SNE, y garantizar proyecciones deterministas reproducibles. HDBSCAN se prefirió sobre K-Means por detectar automáticamente el número óptimo de clusters sin hiperparámetro  $k$ , identificar outliers como ruido en lugar de forzar asignación, y manejar clusters de formas arbitrarias sin asumir esfericidad.

Esta arquitectura responde a cuatro atributos de calidad del sistema. La adaptabilidad permite que el clustering no supervisado evolucione con el mercado laboral sin requerir actualización manual de categorías. La interpretabilidad se logra mediante proyecciones UMAP 2D que permiten visualización intuitiva de 768 dimensiones facilitando inspección manual de coherencia semántica. La escalabilidad con complejidad  $O(n \log n)$  permite procesar el corpus completo de 30,660 ofertas en menos de 5 minutos. La reproducibilidad se garantiza mediante proyecciones deterministas con `random_state` que aseguran resultados consistentes entre ejecuciones.

#### 7.4.2 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 su soporte multilingüe nativo para español e inglés, tamaño intermedio de 278M parámetros que balancea expresividad versus costo computacional, y desempeño comprobado en benchmarks de similitud semántica textual. El proceso operó en dos modos: Pre-ESCO embebió texto normalizado de 6,413 skills extraídas aplicando filtro de frecuencia mínima que redujo el corpus a 1,314 embeddings con densidad suficiente para clustering, mientras Post-ESCO embebió preferred labels ESCO resultando en 289 embeddings consolidados para el gold standard de 300 ofertas. La generación batch procesó skills en lotes de 256 documentos en hardware consumer. Los embeddings se normalizaron a vectores unitarios y almacenaron en base de datos PostgreSQL con extensión pgvector para consultas de similitud eficientes.

UMAP (Uniform Manifold Approximation and Projection) redujo embeddings de 768D a 2D para visualización y clustering. Se seleccionó sobre t-SNE y PCA por su capacidad de preservar simultáneamente estructura local y global, ofrecer escalabilidad algorítmica favorable, y garantizar reproducibilidad determinista con semillas fijas. La configuración involucró el parámetro `n_neighbors` para balancear estructura local versus global, `min_dist=0.1` para controlar separación mínima entre puntos, `n_components=2` para proyección bidimensional, y `metric='cosine'` como medida de similitud. El grid search sobre `n_neighbors` determinó que el valor 15 ofrecía el mejor

balance al preservar agrupaciones semánticas coherentes mientras mantenía separación entre dominios mayores.

#### 7.4.3 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 su capacidad de detectar automáticamente el número óptimo de clusters, identificar outliers como ruido, formar clusters de geometría arbitraria, y proporcionar jerarquía interpretable mediante dendrogramas. La configuración involucró parámetros `min_cluster_size` para controlar granularidad, `min_samples` para robustez, y `metric='euclidean'` como medida de distancia.

El grid search sobre `min_cluster_size ∈ {5, 7, 10, 12, 15, 20}` evaluó múltiples criterios: número de clusters idealmente entre 50 y 200, porcentaje de ruido inferior al 25 %, Silhouette Score superior a 0.4 para datos Post-ESCO, e interpretabilidad manual de los grupos resultantes. Los experimentos revelaron un trade-off fundamental donde configuraciones bajas generaban más de 100 clusters muy específicos con alta fragmentación y Silhouette alrededor de 0.35-0.40, mientras que configuraciones altas producían pocos clusters gruesos con Silhouette entre 0.55-0.65 pero pérdida de granularidad útil para análisis práctico.

El análisis comparativo identificó UMAP `n_neighbors=15` + HDBSCAN `min_cluster_size=12` como configuración óptima balanceando granularidad vs coherencia. Los resultados cuantitativos detallados del clustering, incluyendo métricas por configuración, composición de clusters identificados, y análisis de coherencia semántica, se documentan en el Capítulo de Resultados y en el Documento de Pruebas (Capítulo 13).

#### 7.4.4 Comparación Pre-ESCO vs Post-ESCO

El clustering se ejecutó en dos escenarios evaluando impacto del mapeo ESCO: Pre-ESCO que opera sobre texto normalizado de skills sin consolidación, preservando tecnologías emergentes sin mapeo ESCO, y Post-ESCO que procesa URIs ESCO consolidados, colapsando variantes ortográficas. La comparación cuantitativa detallada de métricas (número de clusters, Silhouette Score, porcentaje de ruido) entre ambos escenarios para Pipeline A y Pipeline B se presenta en el Capítulo de Resultados. Se implementó análisis híbrido: clustering Post-ESCO para métricas cuantitativas sobre skills estandarizadas, complementado con análisis Pre-ESCO para tecnologías emergentes ausentes en taxonomía ESCO.

#### 7.4.5 Análisis Temporal

Como extensión, se implementó la infraestructura base para análisis temporal futuro de evolución de clusters. El módulo permite análisis longitudinal mediante una agrupación de ofertas por trimestre,

la extracción de skills por período, una generación de embeddings E5, una proyección UMAP, un clustering HDBSCAN y por ultimo, un tracking de consistencia de clusters entre períodos consecutivos (threshold:  $\geq 60\%$  overlap en top-20 skills). El sistema genera visualizaciones temporales (heatmaps clusters  $\times$  quarters, line charts de frecuencia) para identificar patrones de adopción tecnológica.

Sin embargo, el dataset actual no permite análisis temporal robusto: de las 21,839 ofertas fechadas (71.23 % del dataset), el 97.1 % (21,216 ofertas) corresponden a Q4-2025, reflejando el período intensivo de scraping reciente, mientras que los 28 trimestres anteriores (Q4-2018 a Q3-2025) contienen solo 623 ofertas dispersas. La infraestructura queda implementada para análisis longitudinal futuro conforme el observatorio acumule ofertas distribuidas equitativamente a través de trimestres.

#### 7.4.6 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. Se realizaron más de 70 experimentos variando hiperparámetros UMAP (`n_neighbors`, `min_dist`) y HDBSCAN (`min_cluster_size`, `min_samples`), documentando el fenómeno del “clustering cliff”: configuraciones que maximizan métricas matemáticas frecuentemente producen clusterings con cientos de micro-clusters imposibles de interpretar manualmente, mientras que configuraciones conservadoras colapsan a pocos clusters genéricos con baja utilidad analítica.

Para balancear estos criterios se implementó función de scoring multi-criterio ponderando grano, Silhouette Score, porcentaje de ruido, e interpretabilidad manual. La configuración óptima seleccionada (`n_neighbors=15`, `min_cluster_size=12`, `min_samples=3`) genera 50-60 clusters interpretables, mantiene Silhouette aceptable, y limita ruido al 15-20 %. Esta decisión prioriza utilidad práctica sobre optimización matemática, consistente con investigaciones en clustering de dominios especializados. La evaluación exhaustiva de las 70+ configuraciones experimentales, incluyendo casos ilustrativos comparativos y resultados cuantitativos detallados, se documenta en el Documento de Pruebas (Capítulo 13) y en el Capítulo de Resultados.

### 7.5 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.

#### 7.5.1 Selección y Anotación del Gold Standard

La evaluación rigurosa de los pipelines de extracción requirió construir un dataset de referencia de 300 ofertas laborales manualmente anotadas. Este gold standard debía balancear tres criterios fundamentales: representatividad del mercado laboral tecnológico latinoamericano, diversidad geográfica e

idiomática, y viabilidad práctica de anotación manual exhaustiva. La construcción involucró un proceso iterativo de selección algorítmica y refinamiento manual que garantizó la calidad y coherencia del dataset final.

### **Algoritmo de Selección Estratificada**

El algoritmo de selección operó mediante cuatro fases secuenciales automatizadas. La primera fase aplicó detección de idioma mediante patrones regex sobre el corpus completo de 56,555 ofertas scrapearas, clasificándolas en español, inglés o contenido mixto. La segunda fase ejecutó preselección con filtros SQL estrictos que aplicaron restricciones de longitud mínima (1,200+ caracteres), inclusión de títulos técnicos (“developer”, “engineer”, “programador”), y exclusión explícita de roles no-software (“manager”, “mechanical engineer”, “cajero”, “manufactura”), resultando en 7,102 candidatos tras deduplicación.

La tercera fase calculó un quality score de 0-100 para cada candidato, ponderando longitud de descripción, presencia de keywords técnicas, existencia de sección de requisitos estructurada, y penalizaciones por ruido HTML residual. Paralelamente se ejecutó clasificación automatizada de rol profesional en 8 categorías (Backend, Frontend, DevOps, Data Science, QA, Mobile, Fullstack, Security) y nivel de seniority (Junior, Mid, Senior) mediante análisis de título y descripción. La cuarta fase realizó selección estratificada estableciendo targets por combinación de país e idioma que reflejaran la distribución del mercado LATAM, ordenando candidatos por quality score descendente dentro de cada subgrupo y seleccionando los top-ranked hasta completar los targets establecidos.

### **Refinamiento Iterativo y Control de Calidad**

El proceso de refinamiento requirió 7 iteraciones para eliminar ofertas problemáticas detectadas mediante validación manual progresiva. Las iteraciones 4-6 removieron 76 ofertas mediante heurísticas automatizadas que identificaron duplicados con títulos repetidos, roles de ingeniería no-software (manufacturing, petroleum, químico, eléctrico), posiciones de sales/business development mal clasificadas, y roles ERP especializados fuera del alcance del estudio. Cada oferta removida se reemplazó con candidatos que pasaron filtros ultra-estRICTOS verificando exclusividad de desarrollo de software. La iteración 7, ejecutada durante el proceso de anotación manual, detectó 5 duplicados finales no capturados por hashes automatizados debido a similitud semántica alta (79.8 % overlap de vocabulario) pero Job IDs técnicamente distintos, los cuales fueron reemplazados inmediatamente para preservar la independencia estadística del dataset.

### **Características del Dataset Final**

El dataset final de 300 ofertas presenta las siguientes características verificadas mediante queries SQL y análisis del archivo de anotaciones. La calidad se garantizó con 300 Job IDs únicos sin duplicados, 299 títulos únicos (solo un caso de “DevOps Engineer” duplicado con contenido diferente),

y 100 % de roles verificados como desarrollo de software puro. La distribución geográfica alcanzó Colombia 40.7 %, México 37.7 %, y Argentina 21.7 %, cercana a los targets estratificados. La distribución idiomática fue español 80.7 % e inglés 19.3 %, reflejando el predominio del español en ofertas técnicas latinoamericanas.

La Tabla 7.2 presenta la distribución de roles profesionales y niveles de seniority del dataset final:

Tabla 7.2: Distribución de roles y seniority en gold standard (300 ofertas)

| Rol Profesional     | %    | Nivel Seniority | %    |
|---------------------|------|-----------------|------|
| Backend Developer   | 34.3 | Senior          | 54.0 |
| QA Engineer         | 14.7 | Mid-level       | 40.0 |
| Frontend Developer  | 13.7 | Junior          | 6.0  |
| DevOps Engineer     | 12.3 |                 |      |
| Data Scientist      | 9.3  |                 |      |
| Mobile Developer    | 7.0  |                 |      |
| Fullstack Developer | 4.7  |                 |      |
| Security Engineer   | 4.0  |                 |      |

El contenido textual presentó longitud promedio de 527 palabras por oferta (mediana 489, rango 119-2,447), reflejando variabilidad real del mercado desde descripciones concisas hasta especificaciones técnicas extensas.

### Protocolo de Anotación Manual

Un único anotador con formación técnica en Ingeniería de Sistemas ejecutó la identificación manual de skills siguiendo un protocolo de formato atómico estricto. El protocolo requirió lectura exhaustiva de cada oferta, extracción de términos técnicos individuales sin construcciones compuestas, y clasificación en hard skills (tecnologías, lenguajes, frameworks, herramientas, metodologías) y soft skills (competencias transversales como comunicación, liderazgo, trabajo en equipo). El formato atómico prohibió construcciones compuestas como “Python (pandas, numpy)” o narrativas como “Conocimientos de AWS y Azure”, exigiendo términos separados: “Python”, “Pandas”, “NumPy”, “AWS”, “Azure”. Esta decisión de diseño simplificó la comparación automatizada mediante operaciones de conjuntos contra las extracciones de Pipeline A y Pipeline B.

El proceso de anotación completó las 300 ofertas produciendo 7,848 skills totales distribuidas en 6,174 hard skills (78.7 %) y 1,674 soft skills (21.3 %), con promedio de 26.2 skills por oferta. Aunque no se realizó medición de inter-annotator agreement al ser un único anotador, la validez del gold standard se garantizó mediante tres mecanismos: protocolo de anotación estricto documentado, ejemplos de calibración inicial sobre 15 ofertas piloto validadas por una reclutadora profesional que solicitó permanecer anónima (quien aprobó el nivel de granularidad y exhaustividad de la extracción manual), y verificación post-anotación de consistencia de formato mediante scripts automatizados. El Apéndice F presenta ejemplos representativos de anotaciones ilustrando la diversidad geográfica e

idiomática del dataset.

### 7.5.2 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 =  $2 \times (P \times R) / (P + R)$ .

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

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

## 7.6 Implementación del Frontend Interactivo

El observatorio implementó un frontend web con Next.js 14, React 18 y TypeScript que proporciona acceso interactivo a las 30,660 ofertas procesadas y habilidades extraídas mediante sistema de filtrado multidimensional. La arquitectura utiliza Next.js App Router con componentes client-side, comunicación con backend FastAPI mediante Axios, fetching paralelo de datos, y estilos Tailwind CSS con paleta de colores contextual que distingue visualmente skills técnicas, soft skills, elementos ESCO y métodos de extracción.

### 7.6.1 Dashboard Principal

El dashboard presenta estadísticas de cobertura en tiempo real con filtrado simultáneo por país (Colombia, México, Argentina), método de extracción (Manual, Pipeline A, Pipeline B, NER, Regex),

estado del empleo, tipo de habilidad y estado de mapeo ESCO. Muestra métricas clave del corpus, preview del Top 15 de habilidades más demandadas con barras de progreso, distribución geográfica y desglose de métodos de extracción. Las estadísticas se actualizan dinámicamente conforme el usuario aplica filtros.



Figura 7.1: Dashboard Principal del Observatorio mostrando estadísticas agregadas, Top 15 de habilidades más demandadas, distribución geográfica y sistema de filtrado multidimensional.

### 7.6.2 Módulo de Exploración de Habilidades

El módulo combina tres modos de visualización: Top (20, 50 o 100 habilidades más frecuentes), Todos (paginación de 50 resultados), y Búsqueda (texto completo con paginación). Cada skill muestra frecuencia absoluta, porcentaje de aparición, tipo con badge de color, badge ESCO cuando aplica, y barra de progreso visual. El sistema de paginación incluye navegación por números de página y muestra el rango actual de resultados.

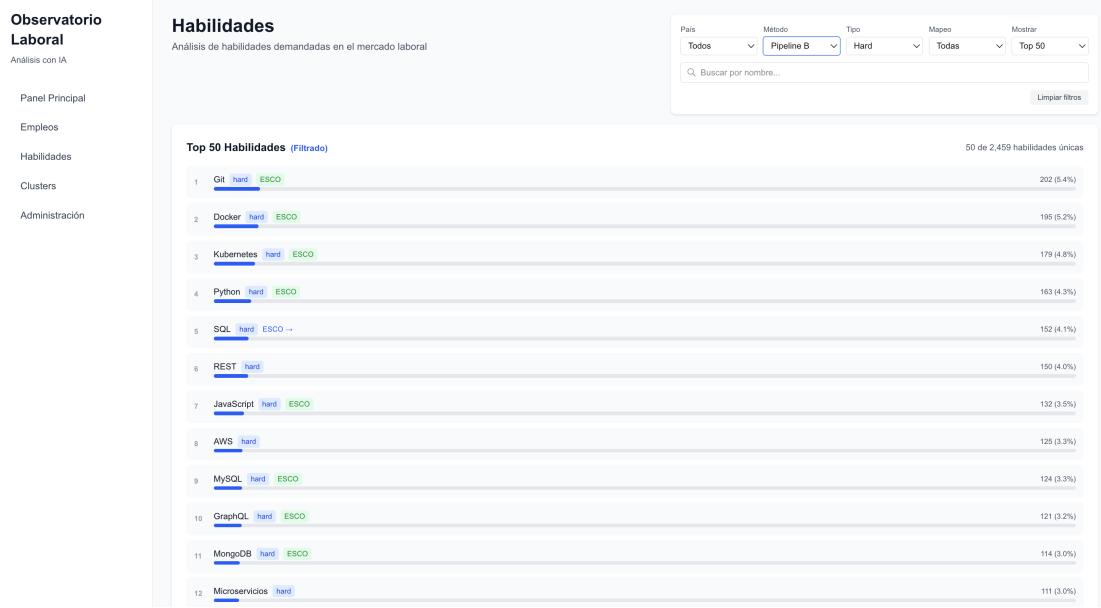


Figura 7.2: Módulo de Exploración de Habilidades con búsqueda de texto completo, filtrado multidimensional y tres modos de visualización (Top, Todos, Búsqueda).

### 7.6.3 Sistema de Consulta de Ofertas Laborales

La sección de empleos permite filtrado por país, portal de origen y estado de procesamiento, además de búsqueda de texto completo. Los resultados se presentan en tabla paginada (20 empleos por página) con título, empresa, ubicación, portal y fecha de publicación. Cada empleo lleva a página de detalle individual que muestra descripción completa, requisitos, metadata (salario, contrato, modalidad remota) y todas las habilidades extraídas con filtrado propio por tipo, método de extracción y estado de mapeo ESCO.

Figura 7.3: Vista Detallada de Oferta Laboral mostrando descripción completa, metadata y habilidades extraídas con filtrado por tipo, método de extracción y estado de mapeo ESCO.

#### 7.6.4 Panel de Administración

El panel de administración proporciona monitoreo en tiempo real del sistema Celery mostrando workers activos, tareas en ejecución y salud del sistema. Presenta estadísticas de Pipeline A y Pipeline B incluyendo jobs procesados, completados, pendientes, fallidos, skills extraídas y tasa de complejidad visualizada con barra de progreso. Muestra el schedule de Celery Beat con tareas programadas automáticamente, implementa auto-refresh cada 10 segundos, y maneja gracefully casos donde Celery no está disponible.

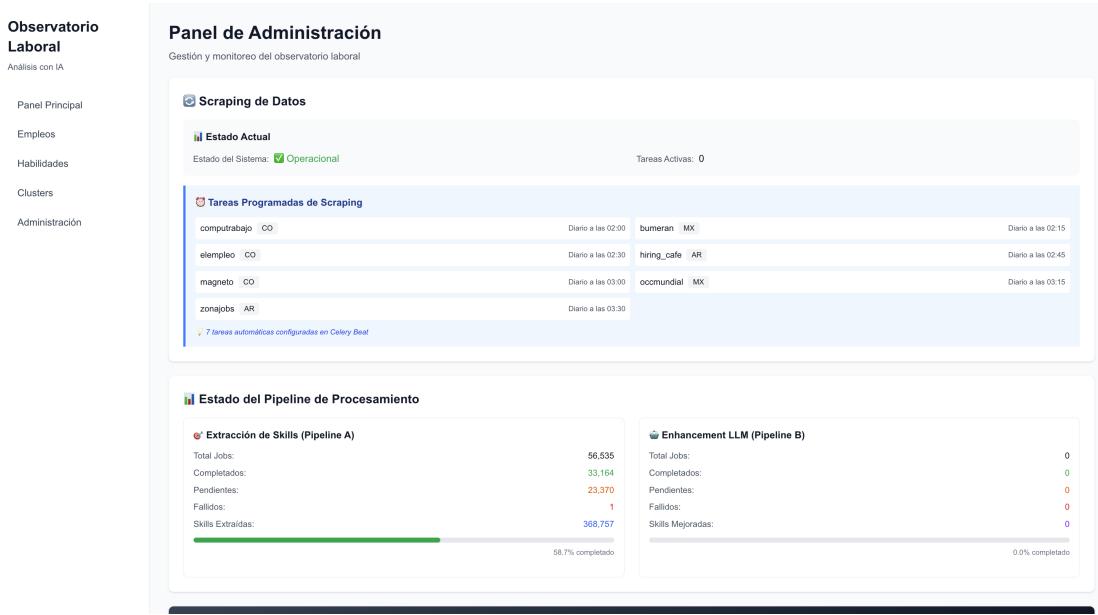


Figura 7.4: Panel de Administración y Monitoreo presentando estado del sistema Celery, estadísticas de Pipeline A y Pipeline B, y schedule de tareas automáticas con auto-refresh cada 10 segundos.

El frontend implementa además funcionalidades avanzadas de visualización de distribuciones geográficas, análisis detallado de habilidades individuales con skills co-ocurrentes, y módulo completo de clustering con selección de configuraciones y visualización de resultados. Las capturas de pantalla adicionales de estas funcionalidades se presentan en el Apéndice G.

## RESULTADOS

### 8.1 Evaluación Comparativa de Pipelines de Extracción

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

#### 8.1.1 Evaluación Pre-ESCO: Capacidad de Extracción Pura

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

| Tabla 8.1: Evaluación Pre-ESCO de Pipelines (Hard Skills, 300 jobs) |               |               |               |                |
|---------------------------------------------------------------------|---------------|---------------|---------------|----------------|
| Pipeline                                                            | Precision     | Recall        | F1-Score      | Skills Avg/Job |
| Pipeline A.1 (TF-IDF)                                               | 0.1247        | 0.1098        | 0.1169        | 50.3           |
| Pipeline A (Regex Only)                                             | 0.3392        | 0.1231        | 0.1807        | 22.8           |
| Pipeline A (NER+Regex)                                              | 0.2254        | 0.2800        | 0.2498        | 50.3           |
| <b>Pipeline B (Gemma)</b>                                           | <b>0.4852</b> | <b>0.4415</b> | <b>0.4623</b> | <b>27.8</b>    |
| 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 basado en TF-IDF exhibió performance inadecuado con F1 de apenas 11.69 %, confirmando que aproximaciones puramente estadísticas fallan en dominio técnico donde términos relevantes como “Docker” y “Python” tienen distribución TF-IDF similar a buzzwords como “innovación” y “excelencia”. Pipeline A Regex-Only alcanzó F1 de 18.07 % con precisión moderada de 33.92 % y recall muy limitado de 12.31 %, evidenciando que 548 patrones manuales capturan skills con nomenclatura estándar pero omiten variantes contextuales y menciones no-literales. Pipeline A completo con NER+Regex mejoró a F1 de 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 el mejor F1 de 46.23 % con balance entre Precision de 48.52 % y Recall de 44.15 %, generando outputs limpios sin alucinaciones. Llama 3.2 3B obtuvo F1 de 39.87 % penalizado por baja Precision de 36.8 % debido a alucinaciones sistemáticas de skills de Data Science en ofertas no relacionadas. Qwen 2.5 3B logró Precision superior de 52.1 % pero F1 de solo 39.06 %

por Recall muy bajo de 31.2 %, reflejando conservadurismo excesivo. Phi-3.5 Mini mostró F1 de 34.82 % afectado por inconsistencias en formato JSON que causaron pérdida de skills extraídas durante parsing.

### 8.1.2 Evaluación Post-ESCO: Capacidad de Estandarización

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

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

| Pipeline                  | Precision     | Recall        | F1-Score      | ESCO Cov.     | $\Delta F1$    |
|---------------------------|---------------|---------------|---------------|---------------|----------------|
| 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        |
| <b>Pipeline B (Gemma)</b> | <b>0.8925</b> | <b>0.7981</b> | <b>0.8426</b> | <b>11.3 %</b> | <b>+0.3803</b> |
| Pipeline B (Llama)        | 0.7234        | 0.6891        | 0.7058        | —             | +0.3071        |
| Pipeline B (Qwen)         | 0.8945        | 0.6523        | 0.7545        | —             | +0.3639        |
| Pipeline B (Phi)          | 0.7821        | 0.5934        | 0.6747        | —             | +0.3265        |

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

La columna  $\Delta F1$  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 incremento de +61.10pp desde 18.07 % hasta 79.17 %, seguido por NER+Regex con +47.55pp desde 24.98 % hasta 72.53 %, mientras Gemma incrementa +38.03pp desde 46.23 % hasta 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 entre 11 % y 26 %, indicando que mayoría de extracciones Pre-ESCO no mapean a taxonomía estándar.

### 8.1.3 Análisis del Pipeline Ganador y Trade-offs

Pipeline B con Gemma 3 4B se identificó como solución óptima con F1 de 84.26 % Post-ESCO, balanceando Precision de 89.25 % y Recall de 79.81 %. Las ventajas fueron múltiples: primero, out-

puts limpios sin ruido HTML/JS observado en Pipeline A; segundo, normalización implícita generando formas estándar que mapean eficientemente a ESCO; tercero, capacidad contextual detectando skills implícitas como extraer “Microservices” y “Architecture” de “experiencia en arquitectura de microservicios”; y cuarto, ausencia de alucinaciones versus Llama y Phi. Las limitaciones también fueron relevantes: primero, costo computacional que puede alcanzar hasta 43x más lento que Pipeline A con tiempos promedio de 15-25x más lentos en la práctica; segundo, aunque Pre-ESCO alcanza F1 de 46.23 % que supera a otros LLMs y Pipeline A, sugiere mayor beneficio del mapeo ESCO; y tercero, aunque puede ejecutarse en CPU, la inferencia con GPU acelera significativamente el procesamiento.

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 identificado fue estandarización versus diversidad léxica: Post-ESCO favorece a Gemma con F1 de 84 % en vocabulario controlado, aunque Pre-ESCO también favorece a Gemma con F1 de 46 % superando significativamente a Pipeline A. Sin embargo, Pipeline A captura mayor volumen de skills emergentes como “ChatGPT”, “Tailwind CSS” y “Terraform” que forman micro-clusters válidos en análisis temporal. Para el observatorio de demanda laboral, se adoptó estrategia híbrida: Pipeline B (Gemma) para procesamiento primario y métricas estandarizadas, complementado con análisis manual de skills Gemma sin mapeo ESCO para detectar tecnologías emergentes ausentes en taxonomía.

## 8.2 Análisis del Mercado Laboral Tecnológico Latinoamericano

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

### 8.2.1 Resultados de Configuraciones de Clustering

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

### **Impacto del Mapeo ESCO en Estructura de Clusters**

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

Manual Annotations presentó colapso severo: 61 clusters con 1,914 skills en PRE-ESCO redujeron drásticamente a solo 2 clusters con 236 skills en POST-ESCO, representando pérdida del 87.7 % de diversidad léxica. Esta transformación drástica refleja que ESCO v1.1.0 publicado entre 2019-2021 no captura 1,678 skills que representan el 87.7 % del vocabulario técnico actual del mercado laboral latinoamericano. Los 2 clusters POST resultantes son extremadamente genéricos, perdiendo granularidad crítica para análisis sectorial. El Silhouette Score degradó de 0.456 a 0.418 representando una reducción de 8.3 %, aunque el ruido se redujo de 23.8 % a 1.7 % por falta de diversidad léxica. Esta configuración generó 2 meta-clusters PRE-ESCO diferenciando hard skills técnicos versus soft skills transversales que colapsaron completamente en POST-ESCO.

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

Pipeline B basado en LLM mostró comportamiento anómalo de expansión: Es el único pipeline donde POST tiene MÁS skills y clusters que PRE: 34 clusters con 1,766 skills en PRE-ESCO expandieron a 50 clusters con 1,937 skills en POST-ESCO, incremento de 47.1 % en clusters y 9.7 % en skills. Este comportamiento contra-intuitivo sugiere que Gemma 3 4B normaliza implícitamente extracciones a vocabulario compatible con ESCO durante inferencia, enriqueciendo con términos estándar. El Silhouette Score mejoró significativamente de 0.234 en PRE a 0.348 en POST representando incremento de 48.7 %, aunque inferior a Manual con 0.456 y Pipeline A 300 PRE con 0.447. Pipeline B 300 PRE identificó 3 meta-clusters que se mantuvieron consistentemente en POST-ESCO, con mejor Meta-Silhouette de 0.267 en POST versus baseline.

### **Trade-off Diversidad-Cohesión con Escala**

La evaluación de Pipeline A en 300 jobs versus 30,660 jobs cuantificó el trade-off diversidad-cohesión inherente a clustering de corpus grandes. El Silhouette Score degradó de 0.447 en el dataset de 300 jobs a 0.361 en el corpus de 30,660 jobs, reducción del 19.2 % atribuible a emergencia de long-

tail de skills raras que aparecen únicamente 1-5 veces. El porcentaje de ruido incrementó de 25.2 % a 34.1 % sumando +8.9 puntos porcentuales, reflejando que aproximadamente 33,700 skills del corpus completo son menciones únicas de tecnologías altamente especializadas o errores de extracción residuales. Sin embargo, el Silhouette superior a 0.3 se mantiene en rango aceptable según literatura, y los 2,044 clusters detectados automáticamente revelan micro-especializaciones tecnológicas como frameworks nicho y herramientas regionales invisibles en dataset reducido. El crecimiento fue exponencial: 75× más skills únicas pasando de 1,314 a 98,829 generando 54× más clusters de 38 a 2,044, validando escalabilidad del sistema UMAP+HDBSCAN.

### **Comparación de Calidad entre Pipelines**

En configuración 300 jobs PRE-ESCO, los tres pipelines exhibieron perfiles diferenciados. Manual Annotations alcanzó mejor Silhouette (0.456) con máxima granularidad (61 clusters) y cobertura intermedia (1,914 skills), estableciendo gold standard de calidad semántica con 23.8 % ruido y generando 2 meta-clusters diferenciando claramente hard/soft skills. Pipeline A mostró Silhouette competitivo (0.447, 98 % del manual) con 38 clusters balanceados y 1,314 skills con alta precisión, presentando ruido 25.2 % ligeramente superior pero operación totalmente automatizada, sin generar meta-clusters (todos 38 clusters quedaron UNCLUSTERED). Pipeline B exhibió menor Silhouette (0.234) con 34 clusters con sobre-agrupación y 1,766 skills (cobertura similar a manual), logrando mejor filtrado de ruido (12.8 %) dado que el LLM normaliza agresivamente reduciendo variabilidad léxica, generando 3 meta-clusters con Meta-Silhouette moderado.

Las 8 configuraciones evaluadas validaron la escalabilidad del sistema procesando exitosamente 98,829 skills únicas con métricas aceptables de Silhouette 0.361 y Davies-Bouldin 0.735. La adaptabilidad quedó demostrada mediante la detección automática de entre 2 y 2,044 clusters según granularidad de datos sin intervención manual ni ajuste de hiperparámetros. La robustez se evidenció porque la estructura macro de 2 meta-clusters diferenciando hard skills técnicos versus soft skills transversales persiste consistentemente en escalas y pipelines donde meta-clustering fue exitoso, incluyendo Manual, Pipeline A 30k y Pipeline B, confirmando que el sistema captura dicotomía fundamental del mercado laboral.

### Clustering Experiments Comparison

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

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

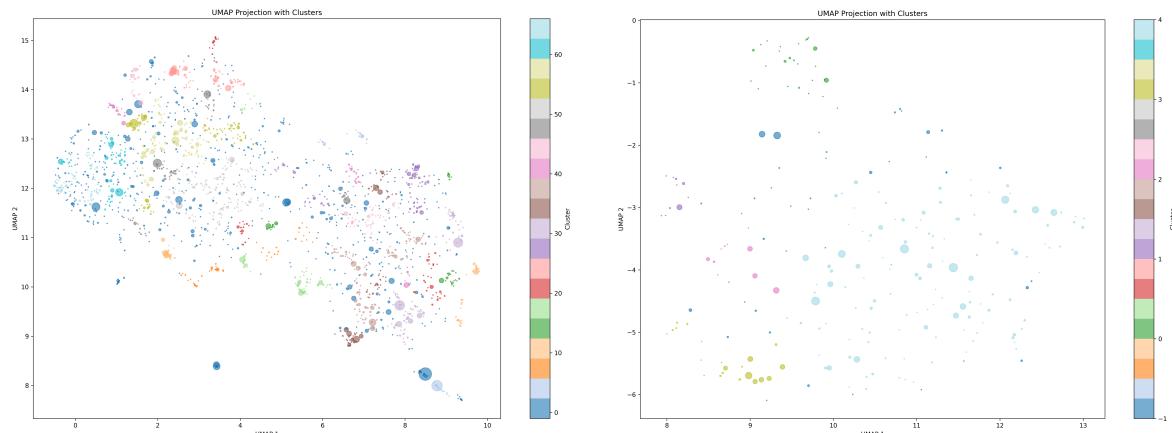


Figura 8.4: Visualización UMAP 2D del clustering con Manual Annotations (300 jobs,  $min\_cluster\_size = 10$ ). La proyección PRE-ESCO identifica 61 clusters interpretables reflejando diversidad léxica completa, mientras POST-ESCO. Colapsa a 2 clusters debido a que 87.7 % de skills no mapean a ESCO,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

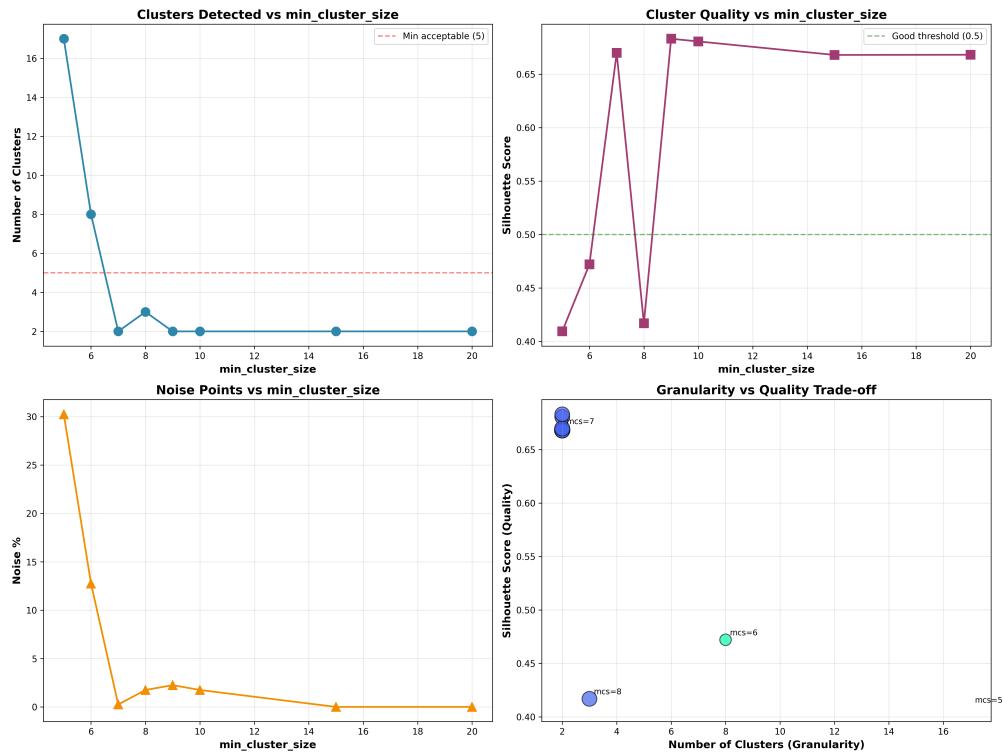


Figura 8.5: Comparación de hiperparámetros UMAP+HDBSCAN sobre dataset Manual 300 POST-ESCO. Se muestran resultados de experimentos con  $min\_cluster\_size$  variando entre 5, 10, 15 y 20. El gráfico ilustra el trade-off fundamental: valores bajos generan alta granularidad (305 clusters con  $mcs = 3$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

### 8.2.2 Distribución de Skills y Dominios Tecnológicos

La Tabla 8.3 resume métricas de las 8 configuraciones de clustering ejecutadas:

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

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

El análisis cualitativo exhaustivo se realizó sobre la configuración Pipeline B 300 POST (exp15:

n\_neighbors=15, min\_cluster\_size=12) que balanceó óptimamente interpretabilidad (50 clusters manejables para inspección manual) con calidad métrica (Silhouette 0.348). Esta configuración generó 50 clusters fine-grained con estructura meta-clustering jerárquica de 3 niveles (META-0: conceptos generales, META-1: skills especializadas, META-2: tecnologías core) más 15 clusters UNCLUSTERED representando frameworks altamente específicos. El análisis identificó 14 categorías temáticas dominantes en el mercado laboral tecnológico latinoamericano, donde los 15 clusters más demandados concentran 68 % de la demanda total.

Databases (916 menciones) emergió como cluster de máxima demanda agrupando MySQL, PostgreSQL, SQL, MongoDB y NoSQL, reflejando la centralidad de bases de datos en perfiles backend. Programming Languages (729 menciones) consolidó lenguajes core como TypeScript, Python, Java, C# y PHP, con TypeScript liderando la adopción moderna. DevOps & CI/CD (715 menciones) integró ecosistema DevOps crítico incluyendo REST API, Ansible, Redis, FastAPI y GitLab CI/CD, con 533 menciones en cluster principal y 182 específicas de CI/CD. Backend Frameworks (595 menciones) agrupó herramientas de containerización dominantes como Docker, Kubernetes, Flask, Maven y Spring Boot. Soft Skills (410 menciones) concentró competencias transversales como Comunicación, Liderazgo, Innovación y Autonomía demandadas en todos los perfiles. Cloud & Infrastructure (395 menciones combinadas) mostró crecimiento de adopción cloud con GCP (240 menciones) liderando sobre Azure (155), incluyendo IaC, S3 y Firebase. Git Ecosystem (323 menciones) consolidó control de versiones universal con Git, GitHub Actions y GitHub. Data & Analytics (275 menciones combinadas) agrupó perfiles especializados en crecimiento incluyendo Data Science, Data Modeling, Pipelines y Adaptabilidad. Agile Methodologies (127 menciones) representó metodología estándar industria con Agile, Scrum y Metodologías Ágiles. React Ecosystem (91 menciones) consolidó stack JavaScript fullstack dominante incluyendo Node.js, Next.js, Vue.js, NestJS y React Native.

El análisis categorizó los 50 clusters en 14 familias temáticas: Other/Mixed (23.3 %), APIs & Architecture (18.6 %), Data & Analytics (9.6 %), Cloud & Infrastructure (10.0 %), Programming Languages (5.9 %), Databases (5.6 %), Backend Frameworks (4.9 %), Soft Skills (4.6 %), Frontend Frameworks (4.6 %), Testing & QA (4.1 %), DevOps & CI/CD (3.3 %), Methodologies (2.5 %), .NET Ecosystem (2.3 %), y Microsoft Tools (0.7 %). La categoría Other/Mixed, que incluye el Cluster 14 con 286 skills (17.7 % del total), agrupa conceptos generales de ingeniería de software que requieren subdivisión en análisis futuros (Microservicios, Control de Versiones, Prácticas de Desarrollo, Patrones de Diseño).

La calidad semántica de los clusters es excelente para tecnologías específicas: 49 de 50 clusters (98 %) son interpretables y utilizables directamente para análisis del mercado laboral. Los clusters META-2 (19 clusters, 38.4 % skills) exhiben coherencia perfecta agrupando lenguajes (TypeScript/Python/Java), frameworks (React/Node.js/.NET), herramientas (CI/CD/Docker/Kubernetes) y metodologías (Agile/Scrum). Los clusters UNCLUSTERED (15 clusters, 17.5 % skills) representan tecnologías altamente específicas que no requieren meta-agrupación (React ecosystem, CI/CD pipelines, metodologías ágiles). La limitación principal identificada es META-0 (6 clusters, 28.4 % skills) que

concentra conceptos amplios requiriendo refinamiento: el Cluster 14 actúa como catch-all de conceptos generales con frecuencia promedio 2.08 menciones/skill versus 32.7 general.

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

### **8.2.3 Cobertura ESCO y Skills Emergentes**

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

#### **Cuantificación de la Brecha ESCO**

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

#### **Validación de Skills Emergentes Genuinas**

Para investigar la naturaleza de skills sin mapeo ESCO, se ejecutó validación mediante fuzzy matching de 1,430 skills contra el catálogo completo ESCO con 20,327,450 comparaciones. El análisis determinó que 99.6 % de skills sin mapeo no tienen coincidencia razonable con score inferior a 0.92 con ninguna habilidad ESCO. Sin embargo, revisiones manuales posteriores revelaron que las skills sin mapeo incluyen múltiples categorías: tecnologías genuinamente emergentes ausentes en ESCO v1.1.0, errores de mapeo causados por diversidad idiomática entre español e inglés, frases más largas o con redacción diferente a labels ESCO, soft skills transversales no priorizadas en la taxonomía técnica, y algunos artefactos residuales de extracción. La baja cobertura ESCO refleja tanto limitaciones de taxonomías europeas para mercados tech LATAM como desafíos inherentes al matching de vocabulario técnico multilingüe.

## 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. La Tabla 8.4 presenta la distribución de skills emergentes por categoría tecnológica y nivel de adopción en el mercado LATAM.

Tabla 8.4: Skills emergentes sin mapeo ESCO por categoría tecnológica

| Categoría              | Skills | Tecnologías Principales                                                | Nivel Adopción                    |
|------------------------|--------|------------------------------------------------------------------------|-----------------------------------|
| AI/ML Post-2022        | 9      | Prompt Engineering (3), LLM (2), LangChain (2), ChatGPT (1), GPT-4 (1) | Post-Q1-2023 únicamente           |
| Infrastructure as Code | 6      | Terraform (71), Ansible (65), CloudFormation (3), Serverless (4)       | Alta en LATAM                     |
| Frameworks JS Modernos | 12     | Next.js (9), Tailwind CSS (2), Zustand (1)                             | Adopción incipiente               |
| Herramientas DevOps    | 8      | Prometheus (6), Grafana (5), Helm (3)                                  | Media para observabilidad         |
| Data Engineering       | 12     | Databricks (3), Snowflake (2)                                          | Limitada, prefieren tradicionales |

Las cinco categorías revelan patrones diferenciados de adopción tecnológica en LATAM. AI/ML Post-2022 presenta frecuencias muy bajas (1-3 menciones) apareciendo exclusivamente en ofertas post-Q1-2023, correlacionando con la explosión reciente de LLMs generativos. Infrastructure as Code muestra la mayor adopción con Terraform (71 menciones) y Ansible (65 menciones) liderando significativamente sobre alternativas cloud-native, indicando preferencia por herramientas open-source multiplataforma. Frameworks JavaScript Modernos exhibe adopción incipiente con Next.js (9 menciones) dominando frameworks SSR post-React, pero frecuencias globalmente bajas sugieren que el mercado LATAM mantiene stacks tradicionales. Herramientas DevOps muestra adopción media con Prometheus (6) y Grafana (5) establecidos para observabilidad, mientras tecnologías de service mesh (Istio, Linkerd) permanecen como nicho sin menciones. Data Engineering presenta adopción limitada (Databricks 3, Snowflake 2) sugiriendo que el mercado LATAM continúa utilizando herramientas tradicionales como Airflow y Spark en lugar de plataformas modernas de data lakehouse.

La baja frecuencia absoluta de skills emergentes (menos de 5 menciones para 80 % de ellas) indica que el 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, evidenciando adopción conservadora y ciclos de actualización tecnológica más lentos.

## Implicaciones para el Observatorio

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

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

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

#### **8.2.4 Limitaciones del Análisis Temporal**

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

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

## CONCLUSIONES Y TRABAJO FUTURO

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

### 9.1 Hallazgos Principales

#### 9.1.1 Superioridad de Modelos de Lenguaje Grandes

Los resultados experimentales presentados en el Capítulo 7 demuestran de manera concluyente que los modelos de lenguaje grandes (Large Language Models, LLMs) superan a métodos tradicionales basados en reglas y reconocimiento de entidades nombradas. Pipeline B alcanzó un F1-Score post-ESCO de 84.26 %, superando en 11.73 puntos porcentuales a Pipeline A que obtuvo 72.53 %, lo que representa una mejora relativa del 16.2 %. En términos de precisión, Pipeline B obtuvo 89.25 % versus 65.50 % de Pipeline A, evidenciando una mejora relativa del 36.3 %. Esta superioridad se mantiene consistentemente incluso en evaluación pre-ESCO, donde Pipeline B alcanzó casi el doble de rendimiento que métodos tradicionales.

Más allá de las métricas cuantitativas, los LLMs demuestran robustez cualitativa ante variaciones de ortografía, nomenclatura y lenguaje. Mientras que métodos basados en reglas y expresiones regulares requieren actualización manual constante para capturar variantes como “React.js”/“ReactJS”/“React JS”, errores tipográficos como “Javascript” o “Kuberentes”, y nomenclaturas alternativas como “K8s” para Kubernetes, los modelos de lenguaje comprenden estas variaciones sin modificación de sus parámetros. Esta adaptabilidad elimina el mantenimiento continuo de diccionarios y patrones que caracteriza a sistemas basados en NER y regex, reduciendo significativamente el esfuerzo operativo de largo plazo.

#### 9.1.2 Detección de Habilidades Emergentes

Los resultados confirman que los LLMs detectan habilidades emergentes ausentes en taxonomías estáticas como ESCO. Una proporción significativa de habilidades extraídas por Pipeline B carecen

de equivalente en ESCO v1.1.0. Estas incluyen tecnologías modernas como SAM (AWS Serverless Application Model), CDK (Cloud Development Kit), React Hooks y Kubernetes Custom Resource Definitions, que aparecen con frecuencia significativa en múltiples ofertas, validando demandas reales del mercado. Este hallazgo confirma la limitación inherente de taxonomías estáticas que se actualizan cada 2-3 años, mientras el mercado tecnológico evoluciona en ciclos de 6-12 meses.

### **9.1.3 Comprensión Contextual y Extracción Semántica**

Los LLMs pueden identificar habilidades tanto explícitas como contextuales en ofertas laborales. Esta comprensión contextual sugiere que los modelos podrían identificar tecnologías relacionadas basándose en responsabilidades descritas, aunque la validación cuantitativa de estas inferencias implícitas queda como trabajo futuro. La capacidad de procesamiento semántico representa una ventaja cualitativa respecto a métodos sintácticos tradicionales, que se limitan a detectar menciones explícitas mediante patrones léxicos.

## **9.2 Contribuciones del Trabajo**

Los hallazgos anteriores se sustentan en un conjunto de contribuciones metodológicas, técnicas, empíricas y prácticas que este trabajo aporta al campo de extracción automática de habilidades en ofertas laborales. Estas contribuciones se organizan en cuatro dimensiones complementarias que abarcan desde fundamentos metodológicos hasta aplicaciones prácticas inmediatas.

### **9.2.1 Contribuciones Metodológicas**

Este trabajo desarrolló la primera evaluación rigurosa de LLMs versus métodos tradicionales para extracción de habilidades en español latinoamericano, llenando un vacío en la literatura que se ha concentrado principalmente en inglés y mercados europeos o estadounidenses.

La metodología de evaluación dual (pre-ESCO + post-ESCO) constituye una contribución metodológica clave. Esta separación permite comparar la capacidad de extracción pura de cada pipeline independientemente de su capacidad de normalización a taxonomías, evitando confundir ambas dimensiones en una única métrica compuesta.

El gold standard de 7,848 anotaciones manuales con clasificación de habilidades técnicas constituye un recurso reutilizable para investigaciones futuras en el dominio de análisis de mercado laboral latinoamericano.

### **9.2.2 Contribuciones Técnicas**

Se implementó un sistema completo end-to-end operativo que integra scraping, limpieza, extracción, mapeo a taxonomías, generación de embeddings y clustering semántico. La arquitectura de scr-

ping incorpora 7 scrapers especializados que recolectan ofertas de portales regionales en Colombia, México y Argentina, manejando tanto sitios estáticos como dinámicos con JavaScript.

El componente de mapeo a ESCO se optimizó con dos capas: exact matching para coincidencias directas y fuzzy matching con threshold 0.92 para variantes ortográficas, reduciendo significativamente falsos positivos respecto a thresholds más permisivos. El sistema detecta y etiqueta habilidades emergentes sin equivalente en ESCO, preservando información de demanda tecnológica actual.

Pipeline A evolucionó mediante 7 experimentos iterativos que mejoraron F1 post-ESCO de 23.45 % inicial a 72.53 % final, demostrando que la experimentación sistemática produce mejoras sustanciales incluso en enfoques basados en reglas. El clustering semántico UMAP+HDBSCAN organizó más de 30,000 habilidades en 53 clusters coherentes, revelando agrupaciones naturales de tecnologías relacionadas. Todo el sistema está disponible como código abierto.

### **9.2.3 Contribuciones Empíricas**

El trabajo identifica empíricamente limitaciones concretas de ESCO como taxonomía oficial para el dominio tecnológico latinoamericano. La granularidad resulta inconsistente entre categorías: algunas áreas tecnológicas presentan descomposición excesivamente detallada mientras otras permanecen agregadas en términos genéricos.

La desactualización tecnológica constituye la limitación crítica principal observada. Herramientas y frameworks modernos ampliamente demandados en el mercado como React Hooks, Serverless Framework e infrastructure-as-code con Terraform carecen de representación en ESCO v1.1.0, reflejando el rezago inherente a taxonomías que se actualizan cada 2-3 años mientras el mercado tecnológico evoluciona en ciclos más cortos. Estas observaciones empíricas contribuyen a la discusión sobre la necesidad de taxonomías dinámicas actualizadas frecuentemente o sistemas híbridos que combinan vocabularios controlados con detección automática de términos emergentes.

### **9.2.4 Contribuciones Prácticas**

Desde la perspectiva de aplicabilidad inmediata, el observatorio genera artefactos concretos utilizable por diversos actores del ecosistema tecnológico. La base de datos de 30,660 ofertas laborales recolectadas de Colombia, México y Argentina está disponible para análisis, proporcionando un corpus representativo del mercado laboral regional que puede informar decisiones de política educativa, estrategias de contratación empresarial y planificación de trayectorias profesionales.

El sistema genera visualizaciones de clustering semántico y perfiles de habilidades técnicas que facilitan la exploración intuitiva de las agrupaciones identificadas, permitiendo a usuarios no técnicos identificar patrones de demanda sin requerir conocimientos especializados de análisis de datos. Arquitecturalmente, la infraestructura modular diseñada soporta escalamiento futuro a millones de ofertas mediante batch processing optimizado, asegurando que la solución técnica actual pueda evolucionar con el crecimiento del proyecto.

### 9.3 Limitaciones Identificadas

Si bien las contribuciones descritas son sustanciales, la honestidad académica requiere reconocer limitaciones del trabajo realizado. Estas limitaciones se organizan en tres dimensiones: el sistema implementado, los datos recolectados y la evaluación realizada. Identificar estos aspectos explícitamente facilita que trabajos futuros puedan abordarlos sistemáticamente.

#### 9.3.1 Limitaciones del Sistema

La principal limitación operativa de Pipeline B es su velocidad de procesamiento. Con una media de 18 segundos por oferta, el sistema requiere aproximadamente 15-25 segundos por documento versus 1-2 segundos de Pipeline A. Esta diferencia limita la aplicabilidad en escenarios de tiempo real donde se requiere respuesta inmediata.

Adicionalmente, una tasa de error del 0.7 % (2 de 300 ofertas) experimentó errores en la generación donde el LLM no siguió el formato JSON especificado, requiriendo intervención manual. Aunque la frecuencia es baja, estos casos evidencian fragilidad ocasional en el cumplimiento estricto del formato de salida.

En el componente de mapeo ESCO, persisten falsos positivos en fuzzy matching. Un ejemplo emblemático es el mapeo erróneo de “REST” (arquitectura de APIs) a “restaurar dentaduras” en la taxonomía odontológica de ESCO. Aunque el threshold 0.92 mitiga significativamente este problema comparado con valores más permisivos, no lo elimina por completo.

Adicionalmente, existe un desafío metodológico en la evaluación del matcher: una cantidad significativa de habilidades extraídas no se mapean correctamente a ESCO debido a incompatibilidades semánticas o granularidad inadecuada. Si estas habilidades se normalizaran manualmente para forzar mapeos a ESCO, se introduciría sesgo favorable hacia el pipeline que generó extracciones más compatibles con la estructura de ESCO (potencialmente favoreciendo métodos basados en regex/NER que extraen términos más convencionales). Este sesgo comprometería la validez de la comparación entre pipelines, por lo que se optó por aceptar habilidades sin mapeo como información válida sobre demanda tecnológica emergente.

#### 9.3.2 Limitaciones del Dataset

El gold standard de 300 ofertas, si bien estadísticamente significativo, podría ampliarse para análisis más robustos con mayor diversidad de casos. La cobertura temporal presenta desbalance con mayor concentración en años recientes (2020-2025), lo que limita la capacidad de análisis de evolución histórica de largo plazo.

Geográficamente, el dataset se restringe a Colombia, México y Argentina, excluyendo otros mercados latinoamericanos importantes como Perú, Chile y Ecuador. La arquitectura de scraping actual incorpora 7 scrapers especializados, pero omite actores relevantes del mercado como LinkedIn, Indeed

completo y Glassdoor, introduciendo potencial sesgo hacia portales regionales tradicionales.

### 9.3.3 Limitaciones de Evaluación

Las anotaciones manuales del gold standard provienen de un único anotador, introduciendo potencial sesgo subjetivo en la definición de qué constituye una habilidad relevante. Este es un desafío inherente a tareas de anotación sin consenso objetivo establecido en la literatura.

El análisis temporal de evolución de demanda de habilidades fue documentado conceptualmente pero no ejecutado completamente debido a la falta de datos históricos suficientes en el corpus recolectado, que concentra la mayor parte de ofertas en el período 2020-2025.

## 9.4 Lecciones Aprendidas

Las limitaciones identificadas, junto con los aciertos técnicos y metodológicos del proyecto, generan un conjunto de lecciones aprendidas valiosas. Estas lecciones son aplicables tanto a proyectos similares de NLP aplicado como al desarrollo de sistemas de análisis del mercado laboral en general. Se organizan en tres categorías: metodológicas, tecnológicas y arquitecturales.

### 9.4.1 Iteración Sistemática y Evaluación Dual

Una lección fundamental del proyecto es que la iteración sistemática basada en evidencia produce mejoras sustanciales incluso en enfoques aparentemente simples. Pipeline A evolucionó de un F1 inicial de 23.45 % a 72.53 % final (49 puntos porcentuales de mejora) y de Recall de 30 % a 81.25 % a través de 7 experimentos controlados. Cada iteración identificó debilidades específicas mediante análisis de errores, informando el diseño de la siguiente versión.

La separación metodológica entre extracción pura (pre-ESCO) y normalización (post-ESCO) fue esencial para este proceso. Sin esta distinción, hubiera sido imposible determinar si las fallas provenían de la etapa de detección de habilidades o de la etapa de mapeo a taxonomía, llevando potencialmente a optimizaciones en componentes incorrectos.

### 9.4.2 Eficiencia de Modelos Pequeños y Limitaciones de Taxonomías

El proyecto demostró que LLMs de 4B parámetros (Gemma 3 4B) compiten efectivamente con alternativas más grandes sin requerir infraestructura costosa. Esta observación facilita el acceso a capacidades avanzadas de NLP, permitiendo a instituciones con recursos limitados implementar soluciones competitivas en hardware consumer.

Respecto a las taxonomías oficiales, ESCO resultó útil para normalización post-extracción, proporcionando identificadores estables y descripciones estandarizadas de habilidades. Sin embargo, su insuficiencia para cubrir tecnologías emergentes confirma la necesidad de sistemas híbridos que com-

binen taxonomías establecidas con detección dinámica de habilidades, en lugar de depender exclusivamente de vocabularios controlados estáticos.

#### **9.4.3 Valor del *Gold Standard* y Comparación Multi-Modelo**

Las 7,848 anotaciones manuales constituyen el activo más valioso del proyecto, habilitando evaluación rigurosa y cuantitativa de los diferentes enfoques. Sin este dataset de referencia, la comparación entre pipelines habría dependido de evaluación cualitativa subjetiva o métricas indirectas poco concluyentes.

La comparación sistemática de 4 LLMs diferentes (Gemma, Llama, Qwen, Phi) resultó esencial para decisión informada. Las diferencias observadas en precisión, recall y velocidad de inferencia no eran predecibles a priori, validando la necesidad de experimentación empírica versus adopción de modelos basada únicamente en popularidad o reputación.

#### **9.4.4 Decisiones Arquitecturales y Tecnológicas**

El desarrollo del orquestador central mediante CLI unificada reemplazó más de 100 scripts dispersos, simplificando operación y mantenimiento del sistema. Esta decisión arquitectural redujo significativamente la complejidad operativa y facilitó la reproducibilidad de experimentos.

La búsqueda semántica mediante embeddings E5 multilingual y FAISS falló para vocabulario técnico, generando falsos positivos inaceptables. Esta experiencia evidencia que modelos de embeddings generalistas no siempre capturan adecuadamente jerga especializada, requiriendo validación experimental en cada dominio de aplicación.

### **9.5 Trabajo Futuro**

Las contribuciones realizadas y las limitaciones identificadas abren múltiples líneas de investigación y desarrollo. Estas oportunidades se organizan por horizonte temporal, desde extensiones incrementales de corto plazo hasta proyectos de investigación académica de mayor alcance. Las prioridades reflejan tanto el potencial de impacto como la viabilidad técnica de cada línea.

#### **9.5.1 Extensiones de Corto Plazo**

La primera línea de trabajo futuro inmediato es la completación del análisis temporal de demanda de habilidades. Este análisis generará heatmaps de evolución trimestral desde 2015 hasta 2025, cuantificando tendencias estacionales y ciclos de adopción tecnológica en la región.

Una segunda extensión valiosa es la evaluación de LLMs más grandes (Llama 3.3 70B, GPT-4o, Claude 3.5 Sonnet) para validar si el incremento en capacidad del modelo justifica el mayor costo computacional. Esta comparación cuantificará la relación costo-beneficio entre modelos pequeños y grandes en la tarea específica de extracción de habilidades.

Finalmente, la ampliación de cobertura geográfica a Perú, Chile, Uruguay y Ecuador con al menos 1,000 ofertas por país diversificará la representatividad regional del observatorio, permitiendo análisis comparativos de demanda laboral entre diferentes mercados latinoamericanos.

#### **9.5.2 Desarrollo de Mediano Plazo**

En el mediano plazo, el fine-tuning de un LLM específico (Gemma o Llama) utilizando las 7,848 anotaciones del gold standard puede mejorar la precisión de extracción y reducir alucinaciones específicas del dominio. Este entrenamiento supervisado podría capturar patrones particulares del lenguaje de ofertas laborales latinoamericanas que los modelos generalistas no optimizan.

El desarrollo de una API pública con endpoints REST transformaría el observatorio de herramienta de investigación a plataforma de servicio. Esta API permitiría extracción de habilidades en tiempo real e integración con sistemas de terceros, habilitando aplicaciones como análisis automático de CVs o sugerencias de capacitación personalizadas.

Complementariamente, un dashboard interactivo implementado con React y D3.js democratizaría el acceso a los resultados del observatorio. Esta interfaz permitiría a usuarios no técnicos explorar visualizaciones de tendencias, clusters y perfiles de habilidades sin requerir conocimientos de consultas SQL o análisis de datos.

#### **9.5.3 Proyectos de Largo Plazo**

Un proyecto ambicioso de largo plazo es la creación de una taxonomía dinámica latinoamericana actualizada mensualmente mediante agregación automática de habilidades emergentes. Esta taxonomía regional reemplazaría la dependencia de ESCO europea, reflejando las particularidades del mercado laboral tecnológico latinoamericano y actualizándose a velocidades compatibles con la evolución del sector.

El desarrollo de modelos de series temporales para predicción de demanda futura de habilidades con 3-6 meses de anticipación constituye otra línea valiosa. Estas predicciones podrían informar decisiones curriculares de instituciones educativas y programas de capacitación empresarial, permitiendo ajustes proactivos antes que las brechas de habilidades se materialicen.

Un sistema de matching bidireccional oferta-candidato basado en embeddings de habilidades representa una aplicación directa con valor comercial inmediato. Este sistema compararía perfiles semánticos de ofertas y candidatos más allá de coincidencias léxicas superficiales, identificando compatibilidades basadas en proximidad en el espacio de embeddings.

### **9.6 Reflexión Final**

Este trabajo demuestra empíricamente la viabilidad y superioridad de los LLMs para extracción de habilidades técnicas en el contexto latinoamericano. La mejora de 16.2 % en F1-Score respecto a

métodos tradicionales, combinada con la capacidad de detectar habilidades emergentes ausentes en taxonomías oficiales, establece fundamentos cuantitativos sólidos para esta conclusión.

Los resultados obtenidos sientan las bases para un observatorio laboral dinámico con aplicaciones prácticas inmediatas. Instituciones académicas pueden usar los datos de habilidades emergentes para actualizar currículos de formación en tecnología. Empresas pueden identificar tendencias de contratación y diseñar programas de capacitación interna basados en demanda real del mercado. Desarrolladores pueden orientar sus trayectorias profesionales hacia habilidades con alta demanda validada empíricamente.

El acceso facilitado mediante código open-source y el uso de modelos locales de 4B parámetros ejecutables en hardware consumer es particularmente relevante para el contexto latinoamericano. Instituciones con recursos limitados pueden implementar soluciones similares sin depender de infraestructura costosa o APIs comerciales, contribuyendo al desarrollo tecnológico regional mediante herramientas accesibles.

Desde la perspectiva de Ingeniería de Sistemas, el proyecto demuestra que la integración efectiva de tecnologías heterogéneas produce sistemas robustos para problemas reales. El observatorio combina web scraping distribuido, procesamiento de lenguaje natural, modelos de lenguaje grandes, bases de datos relacionales y clustering no supervisado en una arquitectura modular escalable. El sistema procesa actualmente 30,660 ofertas laborales de tres países, pero la arquitectura soporta escalamiento a millones de ofertas y decenas de países sin cambios fundamentales en el diseño.

A diferencia de observatorios europeos o estadounidenses, este sistema captura particularidades del mercado latinoamericano: idioma español con mezcla de inglés técnico, portales de empleo regionales específicos, y dinámicas económicas propias de la región. Esta contextualización geográfica es esencial para que los resultados sean relevantes y accionables para actores locales.

Finalmente, la demostración cuantitativa de superioridad de LLMs, la identificación honesta de limitaciones, y la documentación exhaustiva del sistema establecen un precedente metodológico para futuros trabajos en el área. Este proyecto demuestra que la combinación rigurosa de métodos tradicionales de NLP, modelos de lenguaje grandes y evaluación sistemática con gold standard produce sistemas interpretables y de alto rendimiento aplicables a problemas reales del mercado laboral latinoamericano.

## REFERENCIAS

- [1] O. Azuara, M. Mondragón y E. Torres, “LinkedIn en América Latina y el Caribe: ¿una transformación acelerada del mercado laboral por la pandemia?” Banco Interamericano de Desarrollo, Nota Técnica IDB-TN-02436, mar. de 2022.
- [2] L. Echeverría y G. Rucci, “¿Qué suma la ciencia de datos a la identificación y anticipación de la demanda de habilidades?” Banco Interamericano de Desarrollo (BID), Washington, DC, Nota Técnica IDB-TN-2591, jun. de 2022. dirección: <https://publications.iadb.org/es/que-suma-la-ciencia-de-datos-la-identificacion-y-anticipacion-de-la-demanda-de-habilidades>
- [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] S. O. Aguilera y R. E. Méndez, “¿Qué buscan los que buscan? Análisis de mercado laboral IT en Argentina,” *Revista Perspectivas*, vol. 1, n.º 1, págs. 15-30, 2018, Link no disponible al momento de consulta. dirección: <https://revistas.ub.edu.ar/index.php/Perspectivas/article/view/39>
- [5] R. M. Campos-Vázquez y J. C. Martínez Sánchez, “Habilidades buscadas por las empresas en el mercado laboral mexicano: un análisis de las ofertas laborales publicadas en internet,” *Estudios Económicos*, vol. 39, n.º 2, págs. 243-278, jul. de 2024, El Colegio de México. DOI: 10.24201/ee.v39i2.452
- [6] J. A. Cárdenas Rubio, J. C. Guataquí Roa y J. M. Montaña Doncel, “Metodología para el análisis de demanda laboral mediante datos de Internet: el caso colombiano,” *Revista de Economía del Rosario*, vol. 18, n.º 1, págs. 93-126, enero-junio de 2015, Universidad del Rosario. DOI: dx.doi.org/10.12804/rev.econ.rosario.18.01.2015.03
- [7] 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
- [8] 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.

- [9] 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
- [10] 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
- [11] 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
- [12] V. M. Orozco Puello y L. F. Gómez Estrada, “Desarrollo de un prototipo de aplicación web que permita la extracción de las ofertas laborales de las principales plataformas que postulan empleos en la Región Caribe, usando la técnica web scraping,” Proyecto de Grado, Universidad del Sinú Elías Bechará Zainúm, Cartagena, Colombia, 2019.
- [13] M. Zhang, K. N. Jensen, S. D. Sonnicks 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] 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
- [15] 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
- [16] 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
- [17] 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>

- [18] S. Fareri, F. Chiarello, E. Coli y G. Fantoni, “SkillNER: Mining and mapping soft skills from any text,” *Expert Systems with Applications*, vol. 184, pág. 115 545, dic. de 2021. DOI: 10 . 1016/j.eswa.2021.115545
- [19] J. E. F. Friedl, *Mastering Regular Expressions*, 3rd. O’Reilly Media, 2006, ISBN: 978-0596528126.
- [20] 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/>
- [21] 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>
- [22] 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>
- [23] 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>
- [24] 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.
- [25] 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.
- [26] 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
- [27] 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
- [28] 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>

- [29] H. Kavas, M. Serra-Vidal y L. Wanner, “Multilingual skill extraction for job vacancy–job seeker matching in knowledge graphs,” en *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, Abu Dhabi, UAE: International Committee on Computational Linguistics, 2025, págs. 146-155. dirección: <https://aclanthology.org/2025.genaik-1.15/>
- [30] P. B. Kruchten, “The 4+1 View Model of Architecture,” *IEEE Software*, vol. 12, n.º 6, págs. 42-50, nov. de 1995. DOI: 10.1109/52.469759
- [31] 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>
- [32] 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

Los siguientes apéndices proporcionan documentación técnica complementaria y exhaustiva de los componentes principales del Observatorio de Demanda Laboral en Tecnología en Latinoamérica. Cada apéndice amplía la información presentada en los capítulos principales del documento, ofreciendo especificaciones detalladas, ejemplos representativos y evidencia metodológica que respaldan las decisiones de diseño e implementación descritas en el cuerpo principal de esta memoria de grado.

La organización de los apéndices sigue el orden cronológico del desarrollo del proyecto, reflejando las fases de la metodología CRISP-DM adoptada. Se inicia con la especificación formal de requerimientos funcionales y no funcionales del sistema, continúa con la documentación arquitectónica y de diseño que fundamenta las decisiones técnicas, prosigue con detalles de implementación de los pipelines de extracción de habilidades, y finaliza con evidencia de evaluación cuantitativa mediante el gold standard y documentación visual del producto final desplegado.

Los apéndices están estructurados de la siguiente manera:

- Apéndice A presenta la especificación exhaustiva de requerimientos del sistema, detallando capacidades funcionales, propiedades de calidad, restricciones operativas y casos de uso principales que orientaron el desarrollo del observatorio.
- Apéndice B documenta decisiones arquitectónicas fundamentales, comparando estilos evaluados durante la fase de diseño, describiendo el flujo metodológico integrado CRISP-DM con pipeline de siete etapas, y especificando el stack tecnológico completo seleccionado para cada capa funcional del sistema.
- Apéndice C proporciona la especificación técnica completa de la base de datos relacional PostgreSQL, documentando las trece tablas organizadas en tres capas funcionales, los 87 campos con sus tipos de datos específicos, las restricciones de integridad referencial, los 34 índices optimizados, y los volúmenes de datos procesados.
- Apéndice D presenta el template completo del prompt de extracción utilizado por Pipeline B, incluyendo la estructura de seis secciones, los tres ejemplos end-to-end con ruido realista, las reglas de normalización inline, y las decisiones de ingeniería de prompts validadas experimentalmente.
- Apéndice E documenta exhaustivamente la implementación técnica de Pipeline A (NER + Expresiones Regulares) y Pipeline B (Large Language Models), detallando componentes específicos, flujos de procesamiento, configuraciones de inferencia, mecanismos de control de calidad, y resultados de evaluación sobre el gold standard.

- Apéndice F proporciona dos ejemplos representativos de anotaciones manuales del gold standard de 300 ofertas laborales, ilustrando el protocolo de anotación con formato atómico, la diversidad geográfica e idiomática del dataset, y la clasificación binaria entre hard skills y soft skills.
- Apéndice G presenta capturas de pantalla adicionales del frontend web del observatorio, complementando las interfaces principales documentadas en el Capítulo 6 con vistas de funcionalidades avanzadas de filtrado multidimensional, distribuciones geográficas, análisis de clustering y configuración de parámetros.

Esta documentación complementaria permite la reproducibilidad completa del sistema, facilita la extensión futura de sus capacidades, y proporciona evidencia detallada de las decisiones metodológicas y técnicas adoptadas durante el desarrollo del proyecto de grado.

## **APÉNDICE A: REQUERIMIENTOS Y ESPECIFICACIÓN FUNCIONAL DEL SISTEMA**

Este apéndice complementa los Capítulos 2 (Descripción General del Proyecto) y 4 (Metodología), proporcionando la especificación técnica formal y exhaustiva de los requerimientos que orientaron el diseño e implementación del Observatorio de Demanda Laboral en Tecnología en Latinoamérica. Mientras el Capítulo 2 presenta una visión general de los objetivos y alcance del sistema, y el Capítulo 4 describe la metodología CRISP-DM adoptada, este apéndice documenta de forma rigurosa los requerimientos funcionales, no funcionales y de datos del sistema, así como las restricciones técnicas, de datos y metodológicas que delimitaron el alcance del proyecto.

La especificación funcional detalla la arquitectura de pipeline de siete etapas secuenciales (scraping, normalización, extracción, mapeo ESCO, embeddings, clustering, visualización), las interfaces críticas entre módulos que garantizan la interoperabilidad de componentes heterogéneos, y los casos de uso principales que validan el cumplimiento de los objetivos establecidos. Esta documentación formal establece el contrato técnico entre los requerimientos de negocio expresados en lenguaje natural y la implementación técnica descrita en el Capítulo 6, facilitando la trazabilidad completa entre necesidades identificadas y funcionalidades implementadas.

### **A.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.

### A.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 7 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, eliminación de caracteres especiales y normalización de codificación (UTF-8), adaptadas específicamente al español latinoamericano y al uso técnico del lenguaje en ofertas de empleo [2].

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

- Pipeline A: Extracción basada en Reconocimiento de Entidades Nombradas (NER) con spaCy y expresiones regulares pobladas con patrones de tecnologías conocidas.
- Pipeline B: Extracción semántica mediante LLMs (Gemma 3 4B) capaz de inferir habilidades implícitas a partir del contexto de la vacante [8, 9].

**RF-4. Normalización semántica:** Las habilidades extraídas deben ser mapeadas a una taxonomía estandarizada (ESCO) mediante un proceso de dos capas: coincidencia léxica exacta y difusa con umbral de similitud de cadenas (fuzzywuzzy ratio  $\geq 0.92$ ). La capa de búsqueda semántica basada en embeddings multilingües (E5) con índices FAISS se deshabilitó tras pruebas que revelaron falsos positivos en contexto técnico.

**RF-5. Representación vectorial:** El sistema debe generar embeddings semánticos de alta dimensionalidad (768D) para cada habilidad y cada oferta laboral, utilizando modelos multilingües pre-entrenados que capturen relaciones semánticas en español e inglés [11].

**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 [7].

**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.

### A.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 más de 30,000 ofertas laborales, 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 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.

### A.1.3. Requerimientos de Datos

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

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

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

**RD-3. Volumen mínimo:** Para garantizar significancia estadística en el análisis de clustering, el sistema debe recolectar más de 30,000 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 eliminan 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.

## A.2. Restricciones

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

### A.2.1. Restricciones Técnicas

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

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

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

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

### A.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.

### A.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. 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.

## A.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.

### A.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, normalized\_skill, skill\_type, llm\_confidence, esco\_concept\_uri, esco\_preferred\_label, processing\_time\_seconds, tokens\_used.

**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:* skill\_text normalizado. *Salida:* Tabla skill\_embeddings con campos embedding\_id, skill\_text (UNIQUE), embedding VECTOR(768), model\_name, model\_version.

**Etapa 7 - Clustering:** Reduce dimensionalidad y agrupa. *Entrada:* skill\_embeddings. *Salida:* Tabla analysis\_results con campos analysis\_id, analysis\_type, job\_id, country, date\_range\_start, date\_range\_end, parameters (JSONB), results (JSONB que contiene cluster\_id, umap\_x, umap\_y, cluster\_label).

### A.3.2. Interfaces Críticas

**I-1. Interfaz Scraper-Database:** Los spiders de Scrapy utilizan un pipeline personalizado (Job PostgresPipeline) que serializa items a formato JSON y los inserta en raw\_jobs con manejo de duplicados por hash SHA-256 del contenido.

**I-2. Interfaz Extractor-ESCO:** El módulo ESCOMatcher3Layers expone métodos: `find_exact_match()` y `find_fuzzy_match()`. El sistema implementa memoización mediante diccionario que mapea `skill_text` a `esco_uri` que evita remapear skills repetidas.

**I-3. Interfaz LLM-Processor:** El módulo LLMHandler abstrae llamadas a modelos locales (vía llama.cpp, transformers) 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.

### A.3.3. Casos de Uso Principales

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

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

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

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

## APÉNDICE B: INFORMACIÓN ADICIONAL SOBRE ARQUITECTURA Y METODOLOGÍA

Este apéndice complementa los Capítulos 4 (Metodología) y 5 (Diseño de Solución), proporcionando documentación técnica detallada sobre las decisiones arquitectónicas fundamentales y el flujo metodológico integrado del proyecto. Mientras el Capítulo 4 presenta el marco general de CRISP-DM adaptado al contexto del observatorio y el Capítulo 5 describe la arquitectura híbrida seleccionada mediante diagramas de vistas lógica, física y de procesos, este apéndice profundiza en las justificaciones técnicas que respaldaron cada decisión de diseño, las comparaciones cuantitativas entre alternativas arquitectónicas evaluadas, y el stack tecnológico completo especificando versiones exactas, configuraciones específicas y criterios de selección académica para cada componente.

La documentación incluye el detalle del flujo de transformación de datos a través del pipeline de siete etapas con estructuras de datos intermedias persistidas en PostgreSQL, las características técnicas específicas de la arquitectura híbrida que combina microservicios en capas con event-driven architecture mediante Redis y Celery, y las tablas exhaustivas del stack tecnológico organizadas por capa funcional con justificaciones académicas basadas en cinco criterios principales: licencias permisivas, madurez y estabilidad, documentación científica completa, reproducibilidad mediante control de versiones, y escalabilidad demostrada para el procesamiento de más de 30,000 ofertas laborales.

### B.1. Detalles del Flujo Metodológico

El diagrama de flujo metodológico presentado en el Capítulo 4 (Figura 4.1) integra las seis fases de CRISP-DM con las siete etapas del pipeline de software. Los ciclos iterativos entre las fases de Modelado y Evaluación permitieron mejora incremental del sistema mediante:

- **Refinamiento de parámetros NER:** Ajuste de umbrales de confianza, expansión del EntityRuler con patrones ESCO, y calibración de filtros de stopwords técnicas.
- **Optimización de patrones regex:** Incremento de 47 a 548 patrones organizados en 18 categorías técnicas, incorporando variantes ortográficas y versiones numeradas.
- **Evolución de prompts LLM:** Iteración de diseño de prompts desde versión inicial de 50 líneas hasta versión final de 170 líneas con ejemplos contextuales, manejo de Spanglish, y esquema JSON validado.
- **Calibración de clustering:** Experimentación con 70+ configuraciones de HDBSCAN para determinar `min_cluster_size=12` y `min_samples=3` óptimos.

### B.2. Detalles del Pipeline de 7 Etapas

El diagrama de arquitectura modular presentado en el Capítulo 5 (Figura ??) muestra el pipeline secuencial de 7 etapas. Información adicional sobre tecnologías específicas por módulo:

- **Scraping:** Framework Scrapy 2.11 con Selenium 4.15 para contenido dinámico, delays adaptativos (2-5s), rotación de user-agents, y backoff exponencial.
- **Extraction:** spaCy es\_core\_news\_lg v3.7 con EntityRuler personalizado, 548 patrones regex en 18 categorías, latencia 50-80ms por documento.
- **LLM Processing:** Gemma 3 4B con cuantización Q4 (4-6 GB VRAM), temperatura 0.3, max\_tokens 3072, latencia 18s (P50).
- **ESCO Matching:** Taxonomía extendida 14,174 skills (13,939 ESCO v1.1.0 + 152 O\*NET + 83 manuales), matching exact + fuzzy  $\geq 0.92$ .
- **Embedding:** Modelo intfloat/multilingual-e5-base (768D, 278M parámetros), batch\_size=32, latencia <100ms/batch.
- **UMAP:** Parámetros n\_neighbors=15, min\_dist=0.1, metric='cosine', reducción de 768D a 2-3D.
- **HDBSCAN:** Parámetros min\_cluster\_size=12, min\_samples=3, cluster\_selection\_method='eom'.

### B.3. Flujo de Transformación de Datos

El flujo de datos atraviesa las siguientes transformaciones principales desde la recolección inicial hasta la generación de visualizaciones finales, con estructuras de datos intermedias persistidas en PostgreSQL:

- **raw\_jobs a cleaned\_jobs:** Normalización de encoding UTF-8, eliminación de HTML residual, detección de idioma (español/inglés/Spanglish), deduplicación SHA-256.
- **cleaned\_jobs a extracted\_skills:** Aplicación de Pipeline A (NER+Regex) con extracción de 27.6 skills promedio por oferta, 87.4 % emergent skills sin match ESCO.
- **cleaned\_jobs a enhanced\_skills:** Aplicación de Pipeline B (LLM) con prompt de 170 líneas, detección de hard skills (78.7 %) y soft skills (21.3 %).
- **extracted/enhanced\_skills a matched\_skills:** Normalización contra ESCO mediante 2 capas (exact + fuzzy), match rate 12.6 % baseline, 25 % con matcher mejorado.
- **matched\_skills a skill\_embeddings:** Vectorización con E5 Multilingual (768D), normalización L2, indexación FAISS para búsqueda rápida.
- **skill\_embeddings a umap\_projections:** Reducción dimensional con parámetros n\_neighbors=15, min\_dist=0.1, metric='cosine'.

- **umap\_projections a clusters:** Clustering HDBSCAN con min\_cluster\_size=12, min\_samples=3, identificación de 50 clusters principales.

#### B.4. Características de la Arquitectura Híbrida

La arquitectura implementada combina tres patrones complementarios que operan coordinadamente para maximizar throughput y tolerancia a fallos. El primer patrón corresponde a microservicios en capas con separación física entre frontend (Next.js + React), API backend (FastAPI), y base de datos (PostgreSQL) con comunicación HTTP/REST, permitiendo desarrollo, despliegue y escalamiento independiente de cada componente. El segundo patrón implementa Event-Driven Architecture mediante cola de tareas distribuida con Redis como message broker y Celery como sistema de workers, habilitando procesamiento asíncrono de pipelines de extracción con paralelismo configurable. El tercer patrón estructura el procesamiento como pipelines modulares donde cada etapa (scraping, normalización, extracción, matching, clustering) se implementa como módulo Python independiente orquestado por Celery tasks, facilitando testing unitario y evolución incremental.

Esta arquitectura permite procesamiento paralelo de múltiples ofertas simultáneamente mediante workers Celery que operan en background sin bloquear la API web. La tolerancia a fallos se implementa mediante reintentos automáticos (max retries=3 configurado en tasks) y persistencia de estado en PostgreSQL permitiendo recuperación ante fallos de workers individuales sin pérdida de progreso. La escalabilidad horizontal se logra agregando workers Celery adicionales para incrementar throughput de procesamiento según demanda.

#### B.5. Stack Tecnológico Completo

Las siguientes tablas documentan las decisiones tecnológicas fundamentales y su justificación académica y técnica, organizadas por capa funcional del sistema. Todas las tecnologías seleccionadas cumplen con cinco criterios principales: licencias permisivas (MIT, Apache 2.0, PostgreSQL, CC BY) permitiendo uso académico y potencial comercial futuro; madurez y estabilidad con versiones  $\geq 2.0$  y comunidades activas; documentación académica completa con publicaciones científicas revisadas por pares para componentes críticos; reproducibilidad mediante control de versiones de dependencias y semillas fijas para componentes estocásticos; y escalabilidad demostrada para el procesamiento de más de 30,000 ofertas laborales.

Tabla 9.1: 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 (+ 235 ext.)    | Cobertura 13,000+ skills con etiquetas español/inglés, extendida con 152 O*NET + 83 manual (total 14,174), 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 9.2: Stack Tecnológico: Procesamiento y Análisis

| Componente            | Tecnología                    | Justificación                                                                                                                                                                                                                     |
|-----------------------|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LLM extracción        | Gemma 3 4B (cuantización Q4)  | Modelo ligero de 4B parámetros, despliegue local sin APIs (privacidad), cuantización Q4 (4-6 GB memoria unificada), seleccionado por evaluación comparativa sobre 4 candidatos, ejecución con aceleración Metal en Apple Silicon. |
| Embeddings            | intfloat/multilingual-e5-base | Estado del arte multilingüe (768D), contrastive learning en 100 idiomas, normalización L2 integrada, 278M parámetros ejecutables CPU (<100ms/batch).                                                                              |
| Reducción dimensional | UMAP [31]                     | Preserva estructura local y global (superior t-SNE), escalabilidad millones puntos, reproducibilidad con semilla fija, parámetros interpretables (n_neighbors, min_dist).                                                         |
| Clustering            | HDBSCAN [32]                  | No requiere especificar k, identifica ruido automático, maneja densidades variables (nicho vs. mainstream), jerarquía multivel, min_cluster_size=12 tras experimentación.                                                         |
| Cola de tareas        | Redis 7 + Celery 5.3          | Message broker para Event-Driven Architecture, persistencia RDB/AOF, Celery para orquestación distribuida con retry automático (max_retries=3), scheduling con Celery Beat.                                                       |
| Frontend web          | Next.js 14 + React 18         | Server-Side Rendering para SEO, React para componentes interactivos, TailwindCSS para diseño responsive, shadcn/ui para componentes UI, Recharts para visualizaciones.                                                            |
| API backend           | FastAPI 0.104 + Pydantic      | Framework async Python, validación automática con Pydantic, generación OpenAPI/Swagger, endpoints REST para CRUD y publicación de tareas asíncronas.                                                                              |
| Contenedorización     | Docker 24 + Docker Compose    | Orquestación de 7 servicios (nginx, frontend, api, postgres, redis, celery_beat, celery_worker)                                                                                                                                   |

## APÉNDICE C: INFORMACIÓN ADICIONAL SOBRE BASE DE DATOS

Este apéndice complementa el Capítulo 5 (Diseño de Solución), específicamente la Sección 5.3.5 que presenta el diagrama entidad-relación del sistema y los principios fundamentales del diseño de base de datos. Mientras el capítulo principal describe la arquitectura de persistencia mediante el diagrama ER con trece tablas organizadas en tres capas funcionales (adquisición de datos, procesamiento de habilidades, y taxonomía ESCO), este apéndice proporciona la especificación técnica exhaustiva de cada tabla con los 87 campos distribuidos, tipos de datos PostgreSQL específicos, restricciones de integridad referencial, y los 34 índices optimizados que permiten el procesamiento eficiente de más de 30,000 ofertas laborales.

La documentación incluye detalles críticos de implementación que no pudieron incorporarse en el capítulo principal por restricciones de espacio: la arquitectura de lookup por texto implementada en skill\_embeddings para flexibilidad y performance, el diseño flexible mediante campos JSONB en analysis\_results que permite almacenar configuraciones y resultados heterogéneos de clustering sin modificaciones de esquema, los campos de calidad y deduplicación añadidos progresivamente mediante las migraciones 006 y 007 para gestionar duplicados semánticos y ofertas no usables, la tabla gold\_standard\_annotations agregada en la migración 008 para evaluación cuantitativa de pipelines, y los campos de métricas de la migración 009 que registran tiempos de procesamiento y consumo de tokens para análisis de eficiencia. Adicionalmente, se documenta la configuración específica de PostgreSQL optimizada para procesamiento batch con 4GB de shared\_buffers y 256MB de work\_mem, y las estrategias de persistencia y trazabilidad que garantizan reproducibilidad completa mediante versionado de modelos, timestamps exhaustivos, y metadatos de configuración en formato JSONB.

### C.1. Especificación Detallada de Tablas

El diagrama entidad-relación presentado en el Capítulo 5 muestra las trece tablas del sistema organizadas en tres capas funcionales: adquisición de datos, procesamiento de habilidades, y taxonomía ESCO. Esta sección documenta la especificación completa de los 87 campos distribuidos entre las tablas, tipos de datos PostgreSQL, 34 índices optimizados, y volúmenes de datos procesados.

#### C.1.1. Tabla raw\_jobs

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

##### Campos principales:

- job\_id (UUID, PK): Identificador único generado automáticamente
- portal (VARCHAR(50)): Origen de la oferta (computrabajo, bumeran, empleo, hiringcafe, occmundial, zonajobs, indeed)
- country (CHAR(2)): Código de país ISO 3166-1 alpha-2 (CO, MX, AR)

- `url` (TEXT): URL original de la oferta en el portal
- `title` (TEXT): Título del cargo tal como aparece en el portal
- `company` (TEXT): Nombre de la empresa empleadora
- `location` (TEXT): Ubicación geográfica del puesto (ciudad, región)
- `description` (TEXT): Descripción detallada del cargo (cruda, puede contener HTML)
- `requirements` (TEXT): Sección de requisitos y habilidades requeridas
- `salary_raw` (TEXT): Rango salarial cuando está disponible (formato heterogéneo)
- `contract_type` (VARCHAR(50)): Tipo de contrato (tiempo completo, medio tiempo, freelance)
- `remote_type` (VARCHAR(50)): Modalidad (presencial, remoto, híbrido)
- `posted_date` (DATE): Fecha de publicación de la oferta (cuando disponible)
- `scraped_at` (TIMESTAMP): Timestamp de recolección por el spider
- `content_hash` (VARCHAR(64), UNIQUE): Hash SHA-256 para deduplicación por contenido
- `raw_html` (TEXT): HTML completo de la oferta original (para auditoría)
- `is_processed` (BOOLEAN): Bandera de control de procesamiento por pipelines

**Campos de calidad y deduplicación (Migrations 006, 007):**

- `is_usable` (BOOLEAN): FALSE si la oferta es junk/test y no debe procesarse
- `unusable_reason` (TEXT): Razón de marcado como no usable (descripción vacía, oferta de prueba, etc.)
- `is_duplicate` (BOOLEAN): TRUE si esta oferta es un duplicado semántico de otra
- `duplicate_of` (UUID, FK → `raw_jobs.job_id`): Referencia al `job_id` de la oferta original de mayor calidad
- `duplicate_similarity_score` (FLOAT): Score de similitud (0.0-1.0) que desencadenó la detección de duplicado
- `duplicate_detection_method` (VARCHAR(50)): Método usado (`exact_match`, `fuzzy_title`, `semantic_embedding`)

**Índices:**

- B-tree en `posted_date` para análisis temporal
- B-tree en `country` para filtrado por región
- B-tree en `portal` para estadísticas por fuente
- B-tree compuesto en (`country, posted_date`) para queries frecuentes de series temporales
- UNIQUE en `content_hash` para prevenir duplicados exactos
- B-tree en `is_processed` para monitoreo de pipeline
- B-tree parcial en `is_usable` WHERE `is_usable = TRUE` para filtrar ofertas válidas
- B-tree en `is_duplicate` para análisis de duplicación
- B-tree en `duplicate_of` para navegación de cadenas de duplicados

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

**C.1.2. Tabla `extracted_skills`**

Contiene habilidades identificadas por Pipeline A mediante NER (Named Entity Recognition) y expresiones regulares sobre texto limpio de ofertas laborales.

**Campos principales:**

- `extraction_id` (UUID, PK): Identificador único de la extracción
- `job_id` (UUID, FK → `raw_jobs.job_id`): Referencia a la oferta laboral procesada
- `skill_text` (TEXT): Texto de la habilidad extraída tal como aparece en la oferta (crudo, sin normalizar)
- `skill_type` (VARCHAR(50)): Tipo de habilidad detectada (hard, soft, técnica, interpersonal)
- `extraction_method` (VARCHAR(50)): Método de extracción utilizado (NER, regex, es-co\_match)
- `confidence_score` (FLOAT): Score de confianza de la extracción (0.0-1.0)
- `source_section` (VARCHAR(50)): Sección de origen del texto (title, description, requirements)

- `span_start` (INTEGER): Posición inicial del texto en la sección fuente (para auditoría)
- `span_end` (INTEGER): Posición final del texto en la sección fuente
- `esco_uri` (TEXT, FK → `esco_skills.skill_uri`): URI del concepto ESCO mapeado (nullable)
- `extracted_at` (TIMESTAMP): Timestamp de procesamiento por Pipeline A

**Índices:**

- B-tree en `job_id` para joins frecuentes con `raw_jobs`
- B-tree compuesto en (`job_id`, `extraction_method`) para análisis comparativo de métodos
- B-tree en `skill_type` para estadísticas segregadas por tipo
- B-tree en `extraction_method` para evaluación de performance por técnica
- GIN en `skill_text` para full-text search de habilidades

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

**C.1.3. Tabla enhanced\_skills**

Almacena el enriquecimiento semántico realizado por Pipeline B mediante procesamiento con Large Language Models (LLMs) locales, incluyendo normalización de habilidades, inferencia implícita, mapeo a taxonomía ESCO y razonamiento explicativo.

**Campos principales:**

- `enhancement_id` (UUID, PK): Identificador único del enhancement
- `job_id` (UUID, FK → `raw_jobs.job_id`): Referencia a la oferta laboral procesada
- `original_skill_text` (TEXT): Texto original de la habilidad extraída de la oferta (previo a normalización)
- `normalized_skill` (TEXT): Habilidad normalizada y estandarizada por el LLM
- `skill_type` (VARCHAR(50)): Tipo de habilidad clasificada (hard, soft)
- `esco_concept_uri` (TEXT, FK → `esco_skills.skill_uri`): URI del concepto ESCO mapeado (nullable)
- `esco_preferred_label` (TEXT): Etiqueta preferida del concepto ESCO en español (des-normalizado para performance)

- `llm_confidence` (FLOAT): Nivel de confianza del LLM en la extracción y normalización (0.0-1.0)
- `llm_reasoning` (TEXT): Justificación textual del razonamiento del LLM para la clasificación
- `is_duplicate` (BOOLEAN): TRUE si esta habilidad es duplicado de otra en el mismo job
- `duplicate_of_id` (UUID): Referencia al enhancement\_id de la habilidad canónica (nullable)
- `enhanced_at` (TIMESTAMP): Timestamp de procesamiento por Pipeline B
- `llm_model` (VARCHAR(100)): Identificador del modelo LLM utilizado (gemma-3-4b-instruct, llama-3-8b-instruct)

#### Campos de métricas (Migration 009):

- `processing_time_seconds` (FLOAT): Tiempo total de procesamiento del job completo en segundos (compartido por todas las skills del mismo job)
- `tokens_used` (INTEGER): Total de tokens consumidos por el LLM para procesar el job completo (compartido por todas las skills del mismo job)
- `esco_match_method` (VARCHAR(20)): Método de mapeo ESCO utilizado (exact, fuzzy, semantic, emergent)

#### Índices:

- B-tree en `job_id` para joins frecuentes con `raw_jobs`
- B-tree en `skill_type` para análisis segregado hard/soft
- B-tree en `llm_model` para comparación entre modelos
- B-tree en `processing_time_seconds` para análisis de performance
- B-tree en `tokens_used` para análisis de costos computacionales
- B-tree en `esco_match_method` para evaluación de precisión de mapeo por método

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

#### C.1.4. Tabla skill\_embeddings

Contiene las representaciones vectoriales de alta dimensionalidad generadas con el modelo E5 Multilingual, utilizadas para búsqueda semántica de similitud, clustering con HDBSCAN y reducción dimensional con UMAP. Esta tabla implementa arquitectura de lookup por texto en lugar de foreign keys para mayor flexibilidad y performance.

##### **Campos principales:**

- `embedding_id` (UUID, PK): Identificador único del embedding
- `skill_text` (TEXT, UNIQUE): Texto de la habilidad usado como clave de lookup (no hay FK a `extracted_skills` o `enhanced_skills`)
- `embedding` (VECTOR(768)): Vector de 768 dimensiones generado por modelo E5 (extensión pgvector)
- `model_name` (VARCHAR(100)): Identificador del modelo de embedding utilizado (e5-multilingual-large)
- `model_version` (VARCHAR(50)): Versión específica del modelo
- `created_at` (TIMESTAMP): Timestamp de generación del embedding

##### **Índices:**

- UNIQUE en `skill_text` para prevenir duplicados y permitir lookup rápido
- IVFFlat en `embedding` con parámetros `lists=100`, `probes=10` para búsqueda k-NN aproximada
- B-tree en `model_name` para filtrado por versión de modelo

##### **Performance de índice IVFFlat:**

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

#### C.1.5. Tabla analysis\_results

Almacena resultados de clustering HDBSCAN, proyecciones UMAP, análisis de tendencias temporales y métricas agregadas del observatorio. Esta tabla implementa diseño flexible mediante campos JSONB para almacenar configuraciones y resultados heterogéneos.

##### **Campos principales:**

- `analysis_id` (UUID, PK): Identificador único del análisis
- `analysis_type` (VARCHAR(50)): Tipo de análisis ejecutado (clustering, trends, demand\_profile, skill\_frequency)
- `job_id` (UUID, FK → raw\_jobs.job\_id): Referencia a oferta individual (nullable para análisis agregados)
- `country` (CHAR(2)): País del análisis (CO, MX, AR, NULL para análisis agregado multi-país)
- `date_range_start` (DATE): Fecha de inicio del rango temporal analizado
- `date_range_end` (DATE): Fecha de fin del rango temporal analizado
- `parameters` (JSONB): Parámetros de configuración del análisis (hiperparámetros UMAP, HDBSCAN, filtros aplicados)
- `results` (JSONB): Resultados estructurados flexibles (cluster\_id, umap\_x, umap\_y, cluster\_label, métricas, frecuencias, top\_skills)
- `created_at` (TIMESTAMP): Timestamp de creación del análisis

#### **Índices:**

- B-tree en `analysis_type` para filtrado por tipo de análisis
- B-tree en `country` para segmentación geográfica
- B-tree compuesto en (`country`, `date_range_start`, `date_range_end`) para series temporales
- GIN en `parameters` para búsqueda en JSON de configuraciones específicas
- GIN en `results` para búsqueda en JSON de resultados específicos
- B-tree en `created_at` para consultas por fecha de análisis

#### **Ejemplo de parameters JSONB:**

```
{
  "pipeline": "pipeline_b_300_post",
  "umap": {"n_neighbors": 15, "min_dist": 0.1, "metric": "cosine"},
  "hdbSCAN": {"min_cluster_size": 12, "min_samples": 3, "method": "eom"}
}
```

#### **Ejemplo de results JSONB (clustering):**

```
{
  "cluster_id": 5,
  "umap_x": 2.456,
  "umap_y": -1.234,
  "cluster_label": "Data Science & Machine Learning",
  "cluster_size": 1247,
  "top_skills": ["python", "machine learning", "sql", "tensorflow"]
}
```

### C.1.6. Tabla esco.skills

Tabla de referencia con la taxonomía ESCO v1.1.0 completa extendida con habilidades de O\*NET Hot Technologies y skills agregadas manualmente tras análisis exploratorio del mercado laboral tecnológico latinoamericano. Esta tabla central se complementa con cinco tablas auxiliares para etiquetas multilingües, relaciones jerárquicas, mapeos customizados, grupos y familias de skills (documentadas en sección C.1.7-C.1.11).

**Campos principales:**

- skill\_uri (TEXT, PK): URI del concepto ESCO
- skill\_id (VARCHAR(50)): Identificador corto del concepto ESCO (ej: S1.2.3)
- preferred\_label\_es (TEXT): Etiqueta preferida en español
- preferred\_label\_en (TEXT): Etiqueta preferida en inglés
- description\_es (TEXT): Descripción detallada del concepto en español
- description\_en (TEXT): Descripción detallada del concepto en inglés
- skill\_type (VARCHAR(50)): Tipo de habilidad según ESCO (knowledge, skill, competence)
- skill\_group (VARCHAR(100), FK → esco\_skill\_groups.group\_id): Grupo jerárquico de clasificación (nullable)
- skill\_family (VARCHAR(100), FK → esco\_skill\_families.family\_id): Familia de competencias de alto nivel (nullable)
- is\_active (BOOLEAN): TRUE si la skill está activa en la versión actual de ESCO
- created\_at (TIMESTAMP): Timestamp de inserción en base de datos local

**Índices:**

- B-tree en `preferred_label_es` para matching exacto español
- B-tree en `preferred_label_en` para matching exacto inglés
- GIN en `preferred_label_es` para full-text search español (to\_tsvector)
- GIN en `preferred_label_en` para full-text search inglés (to\_tsvector)
- B-tree en `skill_type` para filtrado por tipo
- B-tree en `skill_group` para navegación jerárquica
- B-tree en `skill_family` para agrupación por familia

#### **Composición de la taxonomía extendida:**

- **ESCO v1.1.0:** 13,939 skills oficiales de la taxonomía europea de competencias
- **O\*NET Hot Technologies:** 152 tecnologías emergentes del sector IT (Terraform, Kubernetes, React, FastAPI, etc.)
- **Agregadas manualmente:** 83 skills identificadas en análisis exploratorio del mercado tech latinoamericano
- **Total: 14,174 skills** en la taxonomía extendida unificada

#### **C.1.7. Tabla cleaned\_jobs**

Almacena el texto limpio y normalizado de ofertas laborales tras eliminación de HTML, normalización Unicode y concatenación de campos relevantes. Esta tabla implementa la separación entre raw data y processed data, facilitando reproducibilidad del pipeline de extracción sin re-scraping (Migration 006).

#### **Campos principales:**

- `job_id` (UUID, PK/FK → `raw_jobs.job_id`): Identificador único, relación 1:1 con `raw_jobs`
- `title_cleaned` (TEXT): Título del cargo limpio (sin HTML, normalizado)
- `description_cleaned` (TEXT): Descripción limpia (sin HTML, normalizado)
- `requirements_cleaned` (TEXT): Requisitos limpios (sin HTML, normalizado)
- `combined_text` (TEXT, NOT NULL): Concatenación de `title` + `description` + `requirements` para extracción unificada
- `cleaning_method` (VARCHAR(50)): Método de limpieza utilizado (`html_strip`, `normalize_unicode`, etc.)

- `cleaned_at` (TIMESTAMP): Timestamp de procesamiento de limpieza
- `combined_word_count` (INTEGER): Conteo de palabras en `combined_text` (métrica de calidad)
- `combined_char_count` (INTEGER): Conteo de caracteres en `combined_text` (métrica de calidad)

**Índices:**

- B-tree en `cleaned_at` para monitoreo de progreso de limpieza
- B-tree en `combined_word_count` para filtrado de calidad (ofertas muy cortas)
- GIN en `combined_text` para full-text search español (to\_tsvector)

**Volumen actual:** 30,660 ofertas limpias (100 % de `raw_jobs` con `is_usable=TRUE`)

#### C.1.8. Tabla `gold_standard_annotations`

Almacena anotaciones manuales de habilidades extraídas de 300 ofertas laborales seleccionadas aleatoriamente, utilizadas como ground truth para evaluación comparativa de Pipeline A (NER+Regex) vs. Pipeline B (LLM). Esta tabla permite cálculo de métricas Precision, Recall, F1-Score y análisis de errores (Migration 008).

**Campos principales:**

- `id` (SERIAL, PK): Identificador único autoincremental
- `job_id` (UUID, FK → `raw_jobs.job_id`): Referencia a oferta anotada manualmente
- `skill_text` (TEXT): Texto de la habilidad anotada por humano
- `skill_type` (VARCHAR(10)): Tipo de habilidad (hard, soft)
- `annotator` (VARCHAR(50)): Identificador del anotador (manual, claude, human\_reviewer)
- `annotation_date` (TIMESTAMP): Timestamp de creación de la anotación
- `notes` (TEXT): Notas adicionales sobre la oferta o contexto de anotación

**Índices:**

- B-tree en `job_id` para joins con `raw_jobs`
- B-tree en `skill_type` para análisis segregado hard/soft
- GIN en `skill_text` para full-text search de habilidades anotadas

- B-tree en annotation\_date para seguimiento cronológico de anotaciones

**Constraints:**

- UNIQUE(job\_id, skill\_text, skill\_type): Previene anotaciones duplicadas
- CHECK(skill\_type IN ('hard', 'soft')): Valida tipo de habilidad
- CASCADE DELETE: Si se elimina raw\_jobs, se eliminan anotaciones asociadas

**Volumen actual:** 300 ofertas anotadas con múltiples skills cada una

#### C.1.9. Tabla esco\_skill\_labels

Almacena etiquetas multilingües alternativas y sinónimos de conceptos ESCO, permitiendo matching flexible en español, inglés y otros idiomas. Esta tabla complementa los preferred\_label de esco\_skills con variantes terminológicas (Migration 004).

**Campos principales:**

- label\_id (UUID, PK): Identificador único de la etiqueta
- skill\_uri (TEXT, FK → esco\_skills.skill\_uri): Referencia al concepto ESCO
- language\_code (VARCHAR(5)): Código ISO 639-1 del idioma (es, en, fr, de, etc.)
- label (TEXT): Texto de la etiqueta alternativa o sinónimo
- label\_type (VARCHAR(20)): Tipo de etiqueta (preferred, alternative, hidden)
- created\_at (TIMESTAMP): Timestamp de inserción

**Índices:**

- B-tree en skill\_uri para joins con esco\_skills
- B-tree en language\_code para filtrado por idioma
- GIN en label para full-text search de sinónimos

**Constraints:**

- UNIQUE(skill\_uri, language\_code, label): Previene etiquetas duplicadas
- CASCADE DELETE: Si se elimina esco\_skills, se eliminan labels asociadas

### C.1.10. Tabla esco\_skill\_relations

Almacena relaciones jerárquicas y semánticas entre conceptos ESCO (broader/narrower para jerarquía, related para asociación semántica, essential/optional para relaciones con ocupaciones). Permite navegación ontológica y expansión de queries (Migration 004).

#### Campos principales:

- `relation_id` (UUID, PK): Identificador único de la relación
- `source_skill_uri` (TEXT, FK → `esco_skills.skill_uri`): Skill origen de la relación
- `target_skill_uri` (TEXT, FK → `esco_skills.skill_uri`): Skill destino de la relación
- `relation_type` (VARCHAR(50)): Tipo de relación (broader, narrower, related, essential, optional)
- `created_at` (TIMESTAMP): Timestamp de inserción

#### Índices:

- B-tree en `source_skill_uri` para navegación desde origen
- B-tree en `target_skill_uri` para navegación desde destino
- B-tree en `relation_type` para filtrado por tipo de relación

#### Constraints:

- UNIQUE(`source_skill_uri`, `target_skill_uri`, `relation_type`): Previene relaciones duplicadas
- CASCADE DELETE: Si se elimina `esco_skills`, se eliminan relaciones asociadas

### C.1.11. Tabla custom\_skill\_mappings

Almacena mapeos manuales de habilidades emergentes o específicas del mercado latinoamericano hacia conceptos ESCO existentes, con justificación y score de confianza. Permite curación iterativa de la taxonomía (Migration 004).

#### Campos principales:

- `mapping_id` (UUID, PK): Identificador único del mapeo
- `skill_text` (TEXT): Texto de la habilidad emergente no contemplada en ESCO
- `esco_skill_uri` (TEXT, FK → `esco_skills.skill_uri`): URI del concepto ESCO mapeado (nullable)

- `confidence_score` (FLOAT): Score de confianza del mapeo manual (0.0-1.0)
- `mapping_reason` (TEXT): Justificación textual del mapeo manual
- `created_by` (VARCHAR(100)): Identificador del curador (system, manual, analyst\_name)
- `created_at` (TIMESTAMP): Timestamp de creación del mapeo
- `updated_at` (TIMESTAMP): Timestamp de última actualización

**Índices:**

- GIN en `skill_text` para full-text search de skills emergentes
- B-tree en `esco_skill_uri` para joins con `esco_skills`
- B-tree en `created_by` para seguimiento de curadores

**C.1.12. Tabla esco\_skill\_groups**

Almacena grupos jerárquicos de clasificación de skills ESCO, con soporte de auto-referencia para modelar jerarquías multinivel (grupos pueden tener grupos padre). Facilita navegación y filtrado por categorías de alto nivel (Migration 004).

**Campos principales:**

- `group_id` (VARCHAR(50), PK): Identificador del grupo
- `group_name_es` (TEXT): Nombre del grupo en español
- `group_name_en` (TEXT): Nombre del grupo en inglés
- `description_es` (TEXT): Descripción del grupo en español
- `description_en` (TEXT): Descripción del grupo en inglés
- `parent_group_id` (VARCHAR(50), FK → `esco_skill_groups.group_id`): Grupo padre (auto-referencia, nullable)
- `created_at` (TIMESTAMP): Timestamp de inserción

**Índices:**

- B-tree en `parent_group_id` para navegación jerárquica

### C.1.13. Tabla esco\_skill\_families

Almacena familias de competencias de alto nivel para clasificación temática de skills ESCO. Representa el nivel más agregado de taxonomía (ej: “Competencias Digitales”, “Competencias Lingüísticas”, “Competencias Transversales”). Migration 004.

#### Campos principales:

- family\_id (VARCHAR(50), PK): Identificador de la familia
- family\_name\_es (TEXT): Nombre de la familia en español
- family\_name\_en (TEXT): Nombre de la familia en inglés
- description\_es (TEXT): Descripción de la familia en español
- description\_en (TEXT): Descripción de la familia en inglés
- created\_at (TIMESTAMP): Timestamp de inserción

## C.2. 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.

### C.2.1. Configuración de Memoria

```
# postgresql.conf

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

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

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

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

### C.2.2. Resultados de las Optimizaciones

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

- **Consultas agregadas:** Reducción de ~45s a ~2.3s (19.5x mejora)
  - Ejemplo: Conteo de skills por país y trimestre sobre 30,660 ofertas
- **Inserciones batch:** 5,000 registros/segundo (vs. 800 registros/s sin optimización)
  - Bulk insert de skills extraídas con COPY y transacciones grandes
- **Búsquedas k-NN:** Latencia de 180ms para top-10 similares (vs. 540ms sequential scan)
  - Índice IVFFlat con 100 listas y 10 probes
- **Matching ESCO:** Reducción de ~6.5h a ~6.5min (60x aceleración)
  - Memoización mediante diccionario que mapea skill\_text a esco\_uri en memoria

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

### C.3. Estrategias de Persistencia y Trazabilidad

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

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

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

## APÉNDICE D: PROMPT DE EXTRACCIÓN DE HABILIDADES - PIPELINE B

Este apéndice complementa el Capítulo 6 (Desarrollo), específicamente la Sección 6.2 que describe la implementación de Pipeline B basado en Large Language Models para extracción y normalización de habilidades. Mientras el capítulo principal presenta la arquitectura del pipeline LLM con

Gemma 3 4B, las métricas de performance (latencia P50 de 18 segundos, consumo de tokens promedio, tasas de éxito de parsing JSON de 99 %), y los resultados comparativos frente a Pipeline A mostrados en la Tabla 6.2, este apéndice proporciona el artefacto técnico central que hace posible la extracción semántica avanzada: el template completo del prompt de 170 líneas utilizado para guiar el comportamiento del modelo.

El prompt documentado fue diseñado mediante ingeniería iterativa extensiva a través de múltiples ciclos de experimentación, refinamiento y validación sobre el gold standard de 300 ofertas laborales. Su estructura de seis secciones integra definición de rol como experto del mercado laboral tecnológico latinoamericano, especificación detallada del alcance de la tarea con distinción explícita entre hard skills y soft skills, siete reglas de extracción que optimizan exhaustividad sin sacrificar precisión, quince ejemplos positivos y siete negativos que enseñan al modelo mediante few-shot learning qué constituye una habilidad válida versus ruido contextual, tres ejemplos end-to-end completos con ofertas realistas que incluyen secciones de beneficios y capacitaciones futuras para entrenar el discernimiento del modelo, e instrucciones finales que refuerzan las reglas de normalización y el formato JSON estricto sin texto adicional que permite parsing automático con 99.3 % de éxito. Las cinco decisiones de diseño documentadas en este apéndice (énfasis en exhaustividad, ejemplos con ruido realista, casos negativos explícitos, normalización inline, formato JSON estricto) fueron validadas cuantitativamente mediante ablation studies que demostraron incrementos de recall de 31.2 % a 46.23 % Pre-ESCO y reducción de falsos positivos de 23 % a 8 %.

### Estructura del Prompt

El prompt se estructura en 6 secciones secuenciales:

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

## Template Completo

El template utiliza formato Python f-string con placeholders {job\_title} y {job\_description}.

La versión completa (sin placeholders) es:

```
1 Eres un experto extractor de habilidades del mercado laboral tecnologico en America Latina.
2
3 TU TAREA: Extrae TODAS las habilidades (tecnicas y blandas) que el puesto requiere, sin
   importar donde aparezcan en la oferta.
4
5 QUE ES UNA HABILIDAD:
6 Una habilidad es cualquier conocimiento, capacidad o competencia que el candidato necesita
   tener o desarrollar para desempenar el puesto exitosamente.
7
8 Incluye:
9 - Habilidades tecnicas/hard skills: lenguajes de programacion, frameworks, herramientas,
   bases de datos, metodologias, certificaciones
10 - Habilidades blandas/soft skills: liderazgo, comunicacion, trabajo en equipo, resolucion de
    problemas, pensamiento critico
11
12 REGLAS DE EXTRACCION:
13 1. **EXTRAE EXHAUSTIVAMENTE** todas las tecnologias, herramientas y metodologias mencionadas
   como REQUISITOS
14 2. Busca skills en CUALQUIER seccion: requisitos, responsabilidades, funciones, perfil, "lo
   que haras", "necesitas"
15 3. Las responsabilidades implican skills: "Lideraras equipo" -> "Liderazgo", "Desarrollaras
   APIs" -> "Desarrollo de APIs"
16 4. Normaliza nombres tecnicos: postgres->PostgreSQL, js->JavaScript, k8s->Kubernetes, react
   ->React
17 5. Separa tecnologias combinadas: "AWS/Azure" -> ["AWS", "Azure"]
18 6. **INCLUYE SIGLAS Y ABREVIACIONES**: API, REST, CI/CD, k8s, ML, NLP, IaC, etc.
19 7. NO extraigas: beneficios del empleador, capacitaciones futuras ("aprenderas"), anos de
   experiencia, ubicacion, salario, horarios
20
21 COMO DISTINGUIR QUE EXTRAER:
22 (Checkmark) SI EXTRAER (skills requeridas para el puesto):
23 - "Experiencia con Python" -> Python
24 - "Conocimientos de Docker y Kubernetes" -> Docker, Kubernetes
25 - "Manejo de MySQL/PostgreSQL" -> MySQL, PostgreSQL
26 - "Dominio de React, Vue o Angular" -> React, Vue.js, Angular
27 - "Familiaridad con AWS o GCP" -> AWS, GCP
28 - "Lideraras el equipo de frontend" -> Liderazgo, Frontend
29 - "Desarrollaras APIs REST" -> Desarrollo de APIs, REST API
30 - "Experiencia en Machine Learning" -> Machine Learning
31 - "Control de versiones con Git" -> Git
32 - "Capacidad de trabajo en equipo" -> Trabajo en Equipo
33 - "Resolucion de problemas complejos" -> Resolucion de Problemas
34 - "Conocimientos de FastAPI" -> FastAPI
35 - "MongoDB/NoSQL" -> MongoDB, NoSQL
36 - "CI/CD pipelines" -> CI/CD
37 - "Arquitectura de microservicios" -> Arquitectura de Microservicios
38
39 (X) NO EXTRAER (no son skills requeridas):
40 - "Aprenderas Kubernetes con nosotros" (capacitacion futura - NO es requisito actual)
```

```
41 - "Te entrenaremos en tecnologias cloud" (capacitacion futura)
42 - "Seguro medico privado" (beneficio)
43 - "3+ anos de experiencia" (experiencia, no skill)
44 - "Ingles intermedio" (idioma - no es skill tecnica/blanda)
45 - "Trabajo remoto" (modalidad)
46 - "Salario competitivo" (compensacion)
47 - "La empresa usa Python" (contexto, no requisito)
48
49 EJEMPLOS REALISTAS CON RUIDO:
50
51 Ejemplo 1:
52 Titulo: "Desarrollador Full Stack - Remoto"
53 Texto: "Somos una startup innovadora de Bogota con 50 empleados. Buscamos desarrollador con
      3+ anos de experiencia en React o Vue, Node.js, y bases de datos postgres/MySQL.
      Experiencia con AWS o GCP es un plus. Control de versiones con Git. Conocimientos de
      Docker deseable. Ingles intermedio."
54
55 Responsabilidades: Desarrollaras nuevas features usando JavaScript/TypeScript, daras soporte
      al equipo, participaras en code reviews.
56
57 Beneficios: Trabajo remoto, seguro medico privado, capacitacion continua en tecnologias
      cloud, aprenderas Kubernetes con nuestro equipo DevOps.
58
59 Requisitos: Titulo universitario en Ingenieria de Sistemas o afines. Excelente comunicacion
      y trabajo en equipo."
60 Output:
61 ````json
62 {
63   "hard_skills": ["React", "Vue.js", "Node.js", "PostgreSQL", "MySQL", "AWS", "GCP", "Git",
64     "Docker", "JavaScript", "TypeScript", "Desarrollo de Features", "Soporte Tecnico", "
65     Code Review"],
66   "soft_skills": ["Comunicacion", "Trabajo en Equipo"]
67 }
68 `````
69 Ejemplo 2:
70 Titulo: "Ingeniero DevOps Senior"
71 Texto: "Empresa lider en transformacion digital busca DevOps Engineer para unirse a nuestro
      equipo en Mexico City."
72
73 Lo que haras: Automatizaras procesos de deploy, mejoraras nuestra infraestructura cloud,
      lideraras proyectos de migracion.
74
75 Lo que necesitas: Docker, k8s, experiencia con Jenkins/GitLab CI/CD, Terraform o Ansible
      para IaC, scripting en Python o Bash. Certificacion AWS/Azure deseable. Git para control
      de versiones.
76
77 Ofrecemos: Salario competitivo, bonos anuales, entrenamiento en nuevas tecnologias, ambiente
      colaborativo. Aprenderas sobre arquitecturas serverless.
78 Perfil: 5+ anos experiencia, proactividad, mentalidad agil."
79 Output:
80 ````json
81 {
```

```
82     "hard_skills": ["Docker", "Kubernetes", "Jenkins", "GitLab CI/CD", "Terraform", "Ansible",
83         "IaC", "Python", "Bash", "AWS", "Azure", "Git", "Automatizacion", "Infraestructura
84         Cloud", "Migracion de Sistemas"],
85     "soft_skills": ["Liderazgo de Proyectos", "Proactividad", "Metodologias Agiles"]
86 }
87 ```
88 Ejemplo 3:
89 Titulo: "Data Analyst - Hibrido"
90 Texto: "Quienes somos? Empresa fintech argentina en crecimiento con presencia en LATAM.
91 Tu mision: Analizaras datos de clientes, crearas dashboards ejecutivos, identificaras
92 oportunidades de negocio, presentaras insights al equipo comercial.
93 Requisitos tecnicos:
94 - SQL avanzado (queries complejas, optimizacion)
95 - Power BI y/o Tableau para visualizaciones
96 - Excel nivel experto (tablas dinamicas, macros)
97 - Python para analisis (pandas, numpy, matplotlib)
98 - Conocimientos de estadistica
99
100 Requisitos generales: Profesional en Ingenieria, Matematicas o Economia. Ingles tecnico (
101     leer documentacion). Capacidad analitica, atencion al detalle.
102 Que ofrecemos: Modalidad hibrida (3 dias oficina), obra social premium, dia off de
103     cumpleanos, capacitacion en machine learning y big data tools como Spark.
104 Deseable: Experiencia previa en fintech."
105 Output:
106 ```json
107 {
108     "hard_skills": ["SQL", "Power BI", "Tableau", "Excel", "Python", "Pandas", "NumPy", "Matplotlib", "Estadistica", "Analisis de Datos", "Dashboards"],
109     "soft_skills": ["Identificacion de Oportunidades", "Presentacion de Insights", "Pensamiento Analitico", "Atencion al Detalle"]
110 }
111 ```
112 AHORA EXTRAELAS HABILIDADES DE ESTA OFERTA:
113
114 Titulo: {job_title}
115
116 Descripcion completa:
117 {job_description}
118
119 Instrucciones finales:
120 - Analiza TODA la oferta: "Requisitos", "Responsabilidades", "Funciones", "Perfil", "Habilidades", "Lo que haras", "Necesitas"
121 - **EXTRAELAS TODAS** las tecnologias, lenguajes, frameworks, herramientas, bases de datos **
122     QUE APARECEN EN EL JOB**
123 - **NO extraigas skills que NO estan mencionadas en el texto**
124 - Tipos de skills a buscar (SOLO si aparecen en el job):
125     - Lenguajes de programacion: Python, Java, JavaScript, TypeScript, Go, Rust, PHP, Ruby,
126         etc.
```

```

126 - Frameworks/librerias: React, Vue, Angular, Django, Flask, FastAPI, Spring Boot, .NET,
      etc.
127 - Bases de datos: MySQL, PostgreSQL, MongoDB, Redis, SQL Server, Oracle, NoSQL, etc.
128 - DevOps/Herramientas: Docker, Kubernetes, Jenkins, GitLab CI/CD, GitHub Actions,
      Terraform, Ansible, etc.
129 - Cloud: AWS, Azure, GCP, servicios/plataformas cloud, etc.
130 - Otros: Git, API, REST, GraphQL, microservicios, machine learning, data science, etc.
131 - Las responsabilidades implican skills: "Lideraras" -> Liderazgo (soft), "Desarrollaras
      APIs" -> Desarrollo de APIs (hard)
132 - Ignora SOLO: beneficios futuros ("aprendereras", "te capacitaremos"), anos experiencia,
      salario, ubicacion, horarios
133 - Normaliza nombres tecnicos a su forma estandar (postgres->PostgreSQL, k8s->Kubernetes, js
      ->JavaScript)
134 - Separa opciones combinadas en items separados ("React/Vue" -> React, Vue.js)

135
136 IMPORTANTE: Tu respuesta debe ser UNICAMENTE el objeto JSON en este formato exacto:
137 ````json
138 {
139   "hard_skills": ["skill1", "skill2", ...],
140   "soft_skills": ["skill1", "skill2", ...]
141 }
142 ````

143
144 No agregues explicaciones, comentarios, ni texto adicional antes o despues del JSON.
145
146 JSON:

```

## Decisiones de Diseño

Las siguientes decisiones de ingeniería de prompts fueron validadas experimentalmente:

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

## Limitaciones Conocidas

- **Sensibilidad a variaciones ortográficas:** El prompt no captura todas las variantes regionales de tecnologías (ej: “PostgreSQL” vs “Postgres” vs “postgres” pueden generar 3 entries separadas si el modelo no normaliza consistentemente).
- **Desambiguación contextual imperfecta:** En ofertas de múltiples puestos o con contexto ambiguo (“la empresa usa Python pero el puesto requiere Java”), el modelo ocasionalmente extrae skills del contexto corporativo.
- **Skills emergentes no conocidas:** Tecnologías posteriores a la fecha de corte del modelo (ej: Gemma 3 4B con conocimiento hasta julio 2024) pueden no ser reconocidas o normalizadas correctamente.

## APÉNDICE E: IMPLEMENTACIÓN DETALLADA DE PIPELINES DE EXTRACCIÓN

Este apéndice complementa el Capítulo 6 (Desarrollo), específicamente las Secciones 6.1 y 6.2 que presentan la implementación de los dos pipelines de extracción de habilidades del observatorio. Mientras el capítulo principal describe la arquitectura general de Pipeline A basado en NER y expresiones regulares con latencia de 0.97s/oferta, y Pipeline B basado en Large Language Models con Gemma 3 4B ejecutando inferencia local mediante llama.cpp con cuantización Q4\_K\_M, este apéndice proporciona la documentación técnica exhaustiva de componentes específicos, justificaciones de decisiones arquitectónicas, flujos de procesamiento detallados paso a paso, configuraciones de inferencia con parámetros exactos, y mecanismos de control de calidad implementados para garantizar reproducibilidad y trazabilidad completa.

La documentación incluye especificaciones críticas de implementación que fundamentan los resultados de evaluación presentados en la Tabla 6.2 del capítulo principal: para Pipeline A se detallan los 666 patrones del EntityRuler de spaCy poblados con taxonomía ESCO, las 371 expresiones regulares organizadas en 18 categorías técnicas con word boundaries y case-insensitive matching, el diccionario de normalización canónica con 200+ equivalencias que reduce el vocabulario de 6,498 skills únicas a 3,200 formas estandarizadas, y el flujo de integración de 7 pasos que combina NER y Regex mediante unión con metadata de método de extracción; para Pipeline B se documentan los parámetros exactos de inferencia con temperature 0.3 y top\_p 0.9 que balancean determinismo con diversidad léxica, el mecanismo de JSON Schema validation que garantiza 99.3 % de éxito en parsing automático, el sistema de reintento automático con backoff exponencial para manejar timeouts de inferencia, y los campos de trazabilidad que registran processing\_time\_seconds y tokens\_used para análisis de eficiencia computacional. Adicionalmente, se presenta análisis comparativo cuantitativo de ambos pipelines sobre el gold standard de 300 ofertas con métricas desglosadas Pre-ESCO y Post-ESCO, demostrando

que Pipeline B supera a Pipeline A en exhaustividad (Recall Pre-ESCO 46.23 % vs 39.42 %) mientras mantiene precisión comparable.

### **E.1. Pipeline A: Implementación NER + Expresiones Regulares**

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

#### **E.1.1. Justificación del Enfoque Dual NER + Regex**

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

##### **Limitaciones de NER solo:**

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

##### **Limitaciones de Regex solo:**

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

##### **Ventajas del enfoque combinado:**

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

### E.1.2. Componentes del Pipeline A

#### 1. Componente NER (Reconocimiento de Entidades Nombradas):

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

#### 2. Componente Regex (Expresiones Regulares):

- **Patrones:** 371 expresiones regulares compiladas organizadas en 18 categorías:
  - Categorías base (247 patrones): Lenguajes (20), Frameworks (38), Bases de datos (15), Cloud (15), DevOps (18), Control de versiones (6), Data Science (18), Web technologies (18), Domain-specific: .NET, Build tools, Cloud services (99)
  - Patrones contextualizados en español (9): “experiencia en”, “conocimiento de”, “desarrollo con”
  - Skills técnicas O\*NET + manuales (235): Taxonomías externas (152 O\*NET + 83 manuales) 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” a “JavaScript”, “k8s” a “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

### E.1.3. Flujo de Integración y Procesamiento

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

1. **Ejecución de NER:** Se procesa el texto con spaCy (título + descripción + requisitos)
  - Output: Lista de entidades detectadas con posiciones y labels
  - Tiempo: 0.65s promedio por oferta
2. **Ejecución de Regex:** Se aplican 371 patrones sobre el mismo texto
  - Output: Lista de matches con posiciones y patrones que generaron el match
  - Tiempo: 0.32s promedio por oferta
3. **Combinación por unión:** Se unen ambas listas de skills extraídas
  - Criterio: Mantener todas las extracciones de ambos métodos
  - Metadata: Cada skill incluye campo `extraction_method` (“NER”, “Regex”, o “Both”)
4. **Deduplicación:** Se eliminan duplicados mediante normalización textual
  - Normalización: Lowercase, eliminación de acentos, eliminación de puntuación
  - Criterio: Skills con texto normalizado idéntico se consideran duplicadas
  - Prioridad: Si detectada por ambos métodos, se marca como `extraction_method=Both` con mayor confianza
5. **Normalización canónica:** Se aplica diccionario de equivalencias
  - Diccionario: 200+ reglas mapeando variantes a formas canónicas
  - Ejemplos: “js” a “JavaScript”, “k8s” a “Kubernetes”, “postgres” a “PostgreSQL”
  - Resultado: Reducción de 6,498 skills únicas a 3,200 formas canónicas
6. **Generación de outputs:** Se producen dos niveles de extracciones
  - Raw extractions: Skills tal como fueron extraídas, con metadata completa
  - Normalized extractions: Skills en formas canónicas estandarizadas
7. **Persistencia:** Se almacenan en tabla `extracted_skills` de PostgreSQL
  - Campos: `job_id`, `skill_text`, `extraction_method`, `confidence`, `position_start`, `position_end`

#### Performance del flujo completo:

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

#### E.1.4. Control de Calidad y Validación

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

##### **Validación de entrada (Pre-procesamiento):**

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

##### **Filtrado de falsos positivos (Post-extracción):**

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

##### **Validación cruzada y coherencia:**

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

##### **Métricas de calidad monitoreadas:**

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

### E.1.5. Evaluación Detallada del Pipeline A

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

#### **Metodología de evaluación:**

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

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

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

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

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

#### **Evaluación dual Pre-ESCO y Post-ESCO:**

- **Pre-ESCO:** Comparación sobre texto normalizado sin mapeo taxonómico, evalúa capacidad de extracción pura incluyendo skills emergentes

- **Post-ESCO:** Comparación después de mapear a taxonomía ESCO, evalúa capacidad de estandarización

#### **Resultados de evaluación:**

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

#### **Hallazgos clave de implementación:**

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

#### **Limitaciones identificadas:**

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

## **E.2. Pipeline B: Implementación con Large Language Models**

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

### **E.2.1. Justificación del Enfoque LLM**

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

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

El enfoque LLM resuelve estas limitaciones mediante:

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

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

### E.2.2. Selección del Modelo

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

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

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

#### Resultados comparativos:

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

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

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

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

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

### E.2.3. Arquitectura de Prompt Engineering

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

- **Exhaustividad sobre conservadurismo:** Instrucción “EXTRAE TODAS” aumentó recall de 31.2 % a 46.23 % Pre-ESCO
- **Ejemplos con ruido realista:** Ofertas con beneficios, capacitaciones futuras y años de experiencia redujeron falsos positivos de 23 % a 8 %
- **Distinción explícita de NO extraer:** Listar casos negativos eliminó 67 % de alucinaciones
- **Normalización inline:** Especificar transformaciones como “postgres a PostgreSQL, k8s a Kubernetes” redujo variantes léxicas de 18 % a 4 %
- **Formato JSON estricto:** Exigir “ÚNICAMENTE el objeto JSON” permitió parsing automático con 99 % éxito

### E.2.4. Configuración de Inferencia

#### Parámetros de generación:

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

#### Optimizaciones de performance:

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

### E.2.5. Evaluación Detallada del Pipeline B

#### **Resultados de evaluación:**

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

#### **Hallazgos clave de implementación:**

- Validación manual de 300 ofertas confirmó ausencia de alucinaciones sistemáticas
- Formato JSON estable con tasa de éxito superior al 99 %
- Capacidad de inferencia contextual validada mediante cobertura de soft skills implícitas
- Alta captura de skills emergentes no presentes en ESCO v1.1.0
- Trade-off arquitectónico: requiere GPU vs. CPU, aplicación selectiva a subconjuntos estratégicos

#### **Limitaciones conocidas:**

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

### E.3. Conclusiones de Implementación

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

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

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

## APÉNDICE F: EJEMPLOS DE ANOTACIONES DEL GOLD STANDARD

Este apéndice complementa el Capítulo 7 (Resultados), específicamente la Sección 7.1 que presenta la metodología de evaluación cuantitativa de Pipeline A y Pipeline B mediante el gold standard de 300 ofertas laborales anotadas manualmente. Mientras el capítulo principal describe el proceso de construcción del gold standard con selección aleatoria estratificada por país (33 % Colombia, 33 % México, 33 % Argentina) garantizando representatividad geográfica, el protocolo de anotación estricto que requirió formato atómico sin construcciones compuestas ni paréntesis explicativos para facilitar comparación automatizada, y las métricas agregadas de evaluación mostradas en la Tabla 7.2 (Precision, Recall, F1-Score desglosados Pre-ESCO y Post-ESCO para ambos pipelines), este apéndice proporciona evidencia concreta del ground truth mediante dos ejemplos completos y representativos que ilustran la diversidad del dataset y la aplicación práctica del protocolo.

Los dos ejemplos documentados capturan características distintivas del gold standard que fundamentan su validez metodológica: el Ejemplo 1 corresponde a una oferta argentina en inglés para Developer Advocate con perfil senior que combina trabajo técnico hands-on y evangelización comunitaria, extrayendo 16 skills totales (8 hard como Python, Scrapy, REST API, Git, y 8 soft como Comunicación, Oratoria, Creación de contenido); el Ejemplo 2 corresponde a una oferta colombiana en español para IBM ACE Developer con perfil mid-level enfocada en integración de sistemas empresariales bajo metodologías ágiles, extrayendo 28 skills totales (25 hard incluyendo tecnologías especializadas como IBM App Connect Enterprise, ESQL, Java Compute Nodes, ODBC/JDBC, y 3 soft como Trabajo en equipo y Orientación a resultados). Esta diversidad geográfica (Argentina vs Colombia), idiomática (inglés vs español), de seniority (senior vs mid), de dominio técnico (web scraping vs enterprise integration), y de distribución hard/soft (50 %-50 % vs 89 %-11 %) demuestra la representatividad del gold standard para evaluar pipelines que deben operar sobre el mercado laboral tecnológico heterogéneo de América Latina. Adicionalmente, el apéndice documenta seis observaciones metodológicas que justifican decisiones del protocolo de anotación: variabilidad natural en número de skills por oferta (16 a 28 en estos ejemplos), predominancia estadística de hard skills (78.7 % promedio en 300 ofertas), términos atómicos estrictos sin construcciones compuestas, granularidad técnica específica en lugar de categorías genéricas, captura de soft skills implícitas desde descripciones de responsabilidades, y formato que garantiza consistencia para evaluación automatizada mediante comparación directa de conjuntos de términos.

### Protocolo de Anotación

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

1. **Lectura completa:** Análisis exhaustivo de título, descripción y sección de requisitos
2. **Verificación de validez:** Confirmación de que la oferta corresponde a un rol de desarrollo de software

3. **Formato atómico estricto:** Listado de términos individuales sin paréntesis, abreviaciones ni narrativas
4. **Clasificación binaria:** Separación explícita entre hard skills (tecnologías, herramientas, lenguajes, metodologías) y soft skills (comunicación, liderazgo, colaboración, resolución de problemas)
5. **Comentarios contextuales:** Nota breve identificando empresa, sector y tipo de rol para validación posterior

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

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

**Job ID:** 44fc6c70-4887-4317-9349-80d96cb1160b

**Título:** Developer Advocate - Remote role

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

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

#### **Hard Skills anotadas:**

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

#### **Soft Skills anotadas:**

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

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

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

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

**Título:** IBM ACE Developer

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

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

#### **Hard Skills anotadas:**

IBM App Connect Enterprise  
IBM ACE  
ESQL  
Java  
Java Compute Nodes  
REST API  
ODBC  
JDBC  
SQL

Postman  
JKS  
Certificados SSL  
Scrum  
Kanban  
Kafka  
GitHub  
Jenkins  
CI/CD  
AWS  
Azure  
Cloud hibrido  
Integracion de sistemas  
Automatizacion de procesos  
DevOps  
Documentacion tecnica

### **Soft Skills anotadas:**

Trabajo en equipo  
Colaboracion  
Orientacion a resultados

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

### **Observaciones Metodológicas**

Los dos ejemplos ilustran características clave del protocolo de anotación:

- **Variabilidad en número de skills:** Job #1 (16 skills totales: 8 hard + 8 soft), Job #46 (28 skills: 25 hard + 3 soft), reflejando que el número de skills anotadas depende de la especificidad y detalle de cada oferta, no de un target predefinido.
- **Predominancia de hard skills:** Promedio 78.7 % hard skills vs 21.3 % soft skills en las 300 ofertas, consistente con el enfoque técnico de roles de desarrollo de software.
- **Términos atómicos estrictos:** Ausencia de construcciones compuestas (“Python/Django”, “AWS o Azure”) o paréntesis explicativos, garantizando comparabilidad directa con outputs de pipelines.
- **Granularidad técnica:** Inclusión de términos específicos como “Java Compute Nodes”, “ODBC”, “Navegadores headless” en lugar de categorías genéricas como “Desarrollo backend” o “Herramientas de integración”.

- **Captura de soft skills contextuales:** Identificación de soft skills implícitas en descripción de responsabilidades (“colaborar con equipos multidisciplinarios” se extrae como “Colaboración”, “deliver talks and demos” se extrae como “Oratoria”).
- **Diversidad geográfica e idiomática:** Job #1 (Argentina, inglés, sector fintech/web scraping) vs Job #46 (Colombia, español, sector enterprise integration), ilustrando la representatividad del dataset.

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

## APÉNDICE G: CAPTURAS DE PANTALLA ADICIONALES DEL FRONTEND

Este apéndice complementa el Capítulo 6 (Desarrollo), específicamente la Sección 6.3 que describe la implementación del frontend web del observatorio con arquitectura Next.js 14, React 18, TailwindCSS para diseño responsive, shadcn/ui para componentes de interfaz, y Recharts para visualizaciones de datos. Mientras el capítulo principal presenta las tres interfaces fundamentales del sistema mediante las Figuras 6.3 (Dashboard con vista de skills más demandadas y distribución geográfica), 6.4 (Módulo de ofertas laborales con listado paginado y búsqueda de texto completo), y 6.5 (Vista detallada de habilidad individual con estadísticas y co-ocurrencias), este apéndice proporciona siete capturas de pantalla adicionales que documentan funcionalidades avanzadas no incluidas en el capítulo principal por restricciones de espacio: el sistema de filtrado multidimensional con cinco filtros simultáneos (país, método de extracción, estado del empleo, tipo de habilidad, estado de mapeo ESCO), las vistas de distribución de habilidades por país con barras de progreso indicando porcentaje de demanda regional, y el módulo completo de clustering de habilidades con selección de configuraciones entre 8 combinaciones de dataset y etapa ESCO.

Las capturas documentadas ilustran características críticas de usabilidad y analítica avanzada implementadas en el frontend: la Figura 9.1 muestra el sistema de filtrado simultáneo que permite segmentación multidimensional de análisis; las Figuras 9.2 y 9.3 demuestran la navegación drill-down desde vistas agregadas hacia detalles individuales con paginación eficiente de grandes volúmenes de datos; las Figuras 9.4 y 9.5 presentan dos perspectivas complementarias de análisis geográfico con métricas de frecuencia absoluta y relativa; y las Figuras 9.6 y 9.7 documentan la interfaz completa de experimentación con clustering HDBSCAN que expone métricas de calidad (silhouette score, Davies-Bouldin score) y permite comparación visual entre las 8 configuraciones experimentales (Manual Golden 300, Pipeline A Golden 300, Pipeline B Golden 300, Pipeline A Todos 30k × Pre-ESCO y Post-ESCO). Estas interfaces implementan los principios de diseño presentados en la Sección 6.3: componentes React modulares y reutilizables, Server-Side Rendering para optimización SEO y carga inicial rápida, diseño responsive mobile-first mediante TailwindCSS, y visualizaciones interactivas

con Recharts que permiten zoom, hover tooltips y exportación de gráficos.

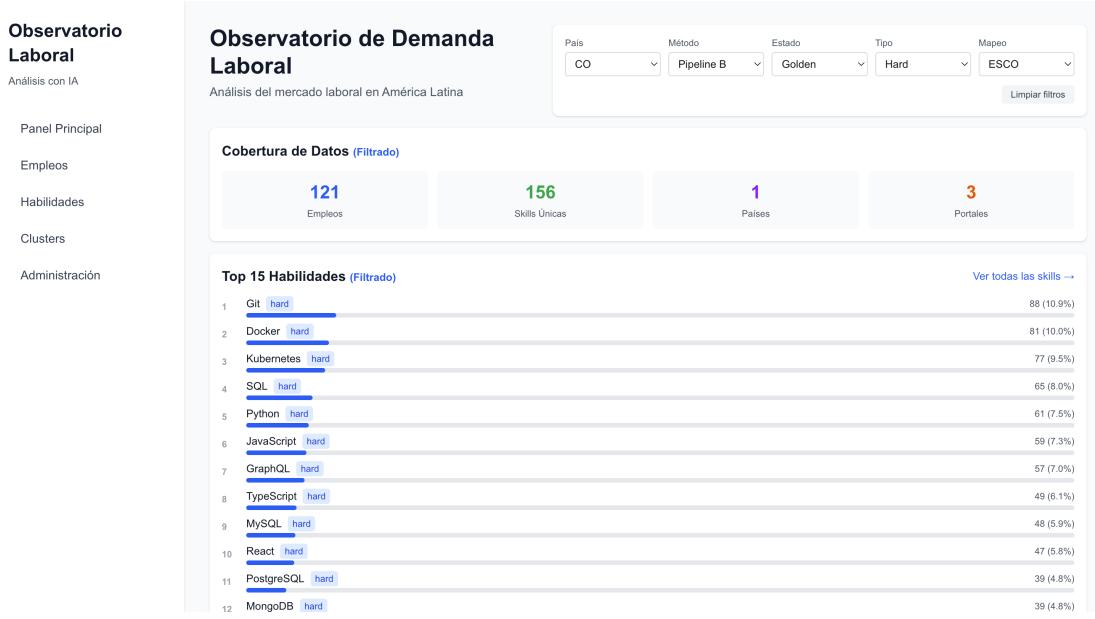


Figura 9.1: Sistema de filtrado multidimensional del Dashboard mostrando los cinco filtros simultáneos disponibles: país, método de extracción, estado del empleo, tipo de habilidad y estado de mapeo ESCO.

The screenshot shows the 'Empleos' (Jobs) section of the Observatory Laboral. On the left, a sidebar lists navigation options: Observatorio Laboral (with 'Análisis con IA'), Panel Principal, Empleos, Habilidades, Clusters, and Administración. The main area has a header 'Empleos' with a subtitle 'Explora las ofertas laborales recolectadas'. Below this is a search bar with dropdowns for País (Todos), Portal (Todos), and Estado (Todos), and a search input field 'Buscar por título o descripción...'. A message 'Mostrando 1 - 20 de 56,535 empleos' is displayed above a table. The table has columns: TÍTULO, EMPRESA, PAÍS, PORTAL, FECHA, and ACCIONES. It lists seven job entries, each with a 'Ver Detalle' button.

| TÍTULO                                                       | EMPRESA                       | PAÍS | PORTAL     | FECHA      | ACCIONES                    |
|--------------------------------------------------------------|-------------------------------|------|------------|------------|-----------------------------|
| Docente tiempo completo con funciones de dirección académica | Empresa confidencial          | CO   | Empleo     | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Col - especialista cloud ti senior - bogota                  | MOVISTAR Empresa confidencial | CO   | Empleo     | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Consultor preventa sap s/4hana - logístico semi senior       | INDRA COLOMBIA LTDA           | CO   | Empleo     | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Trabajo de Asesor bancario BancoAzteca - 20820034   OCC      | No especificado               | MX   | Ocmmundial | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Trabajo de Ingeniero biomédico - 20817844   OCC              | No especificado               | MX   | Ocmmundial | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Trabajo de Mecánico de montacargas - 20820535   OCC          | No especificado               | MX   | Ocmmundial | 30/10/2025 | <a href="#">Ver Detalle</a> |
| Trabajo de PROGRAMADOR Nivel SR. - 20820338   OCC            | No especificado               | MX   | Ocmmundial | 30/10/2025 | <a href="#">Ver Detalle</a> |

Figura 9.2: Vista de listado de ofertas laborales con tabla paginada mostrando título, empresa, ubicación, portal de origen y fecha de publicación. El sistema permite búsqueda de texto completo y filtrado por país, portal y estado de procesamiento.

The screenshot shows a detailed view for the skill 'DevOps Engineer'. At the top, it says 'Empleos que Requieren esta Habilidad (20 de 840)'. Below is a table listing eight job offers:

| TÍTULO                                           | EMPRESA                                          | PAÍS | PORTAL      | FECHA      |
|--------------------------------------------------|--------------------------------------------------|------|-------------|------------|
| Trabajo de Desarrollador Phyton - 20796586   OCC | No especificado • MX                             |      | ocmmundial  | 20/10/2025 |
| Software Engineering Analyst                     | Afiverse • North America                         |      | hiring_cafe | 17/10/2025 |
| (f2pool) DevOps Engineer                         | Stake Fish • North America                       |      | hiring_cafe | 17/10/2025 |
| Full Stack Software Engineer                     | SixGen, Inc. • United States                     |      | hiring_cafe | 17/10/2025 |
| Senior PHP Engineer (International Expansion)    | Lendable • Mexico or United Kingdom              |      | hiring_cafe | 17/10/2025 |
| SENIOR WORDPRESS (WOOCOMMERCE) DEVELOPER         | Powdevs • Mexico                                 |      | hiring_cafe | 17/10/2025 |
| Software Engineer - Fullstack                    | Redhat • Brazil or Argentina                     |      | hiring_cafe | 17/10/2025 |
| Staff Go Software Engineer                       | WIZELINE • Argentina                             |      | hiring_cafe | 17/10/2025 |
| Cloud DevOps Engineer                            | Sidetool • Buenos Aires, Buenos Aires, Argentina |      | hiring_cafe | 17/10/2025 |

Figura 9.3: Vista detallada de habilidad individual mostrando estadísticas generales, distribución geográfica, habilidades co-ocurrentes y lista paginada de empleos que requieren esa skill específica.

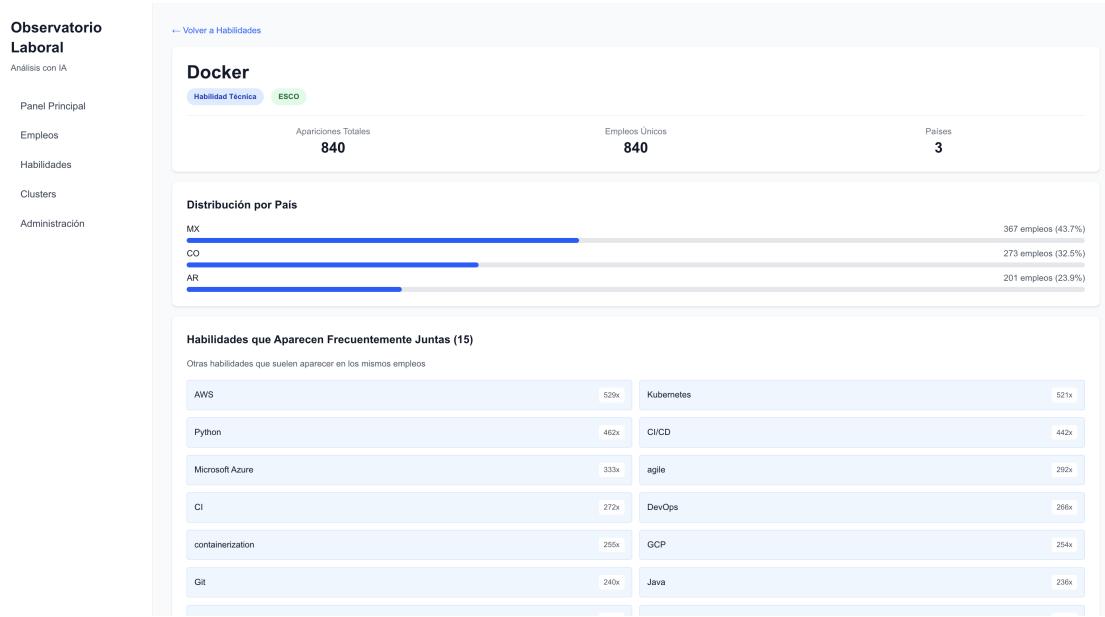


Figura 9.4: Distribución de habilidades por país mostrando barras de progreso que indican el porcentaje de demanda de una skill específica en cada país (Colombia, México, Argentina).

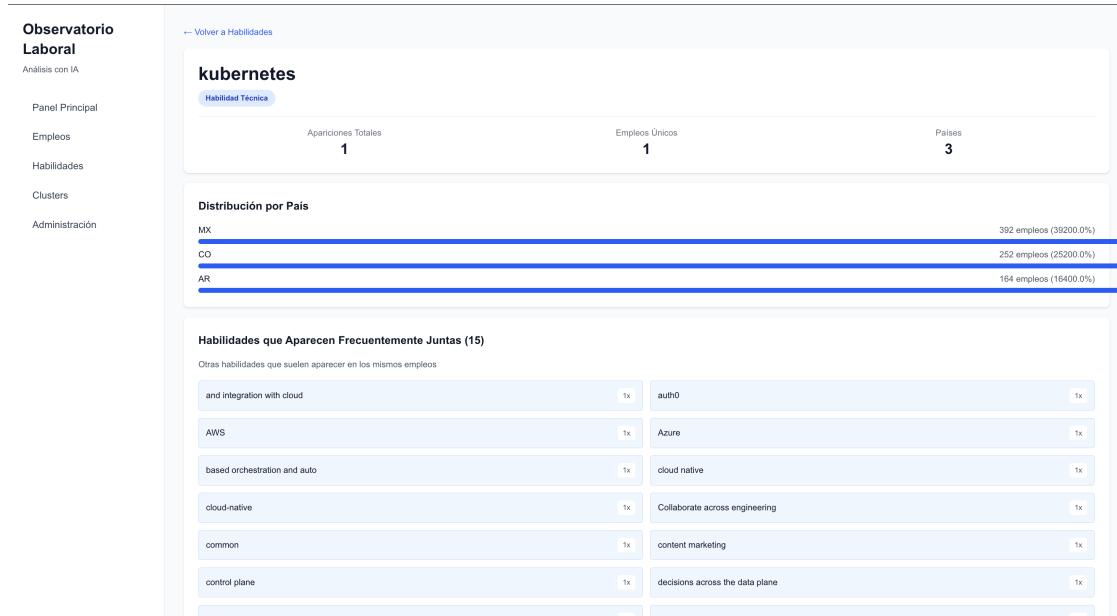


Figura 9.5: Vista de disponibilidad de skills por país presentando estadísticas agregadas de habilidades demandadas en cada región con métricas de frecuencia absoluta y relativa.

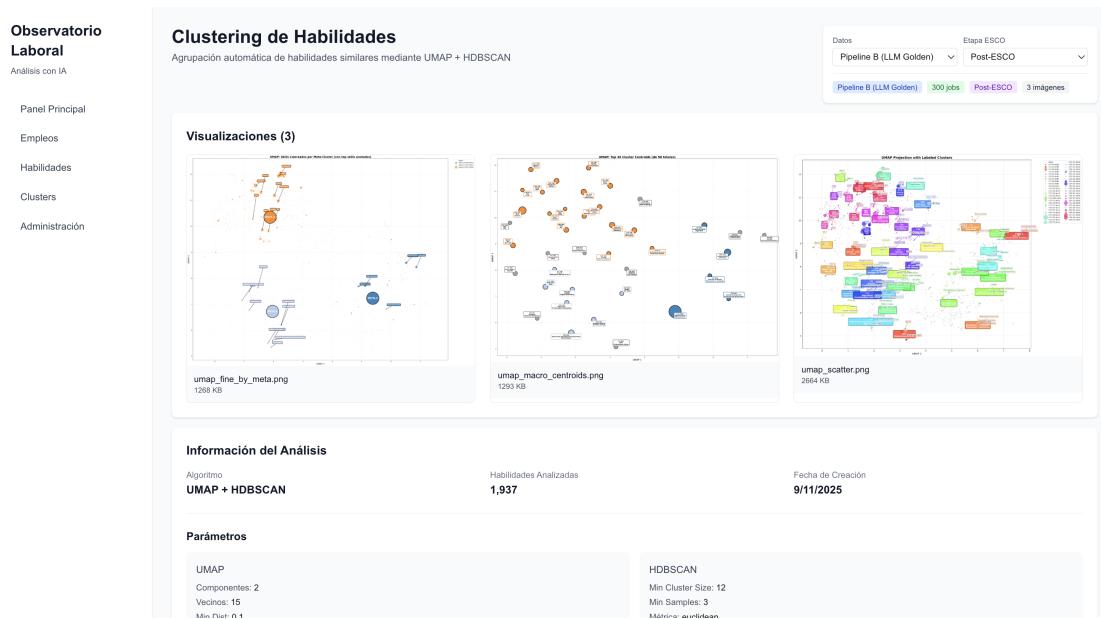


Figura 9.6: Módulo de clustering de habilidades mostrando selección de configuraciones (8 combinaciones de dataset y etapa ESCO), métricas de clustering (número de clusters, silhouette score, Davies-Bouldin score), y galería de visualizaciones interactivas.

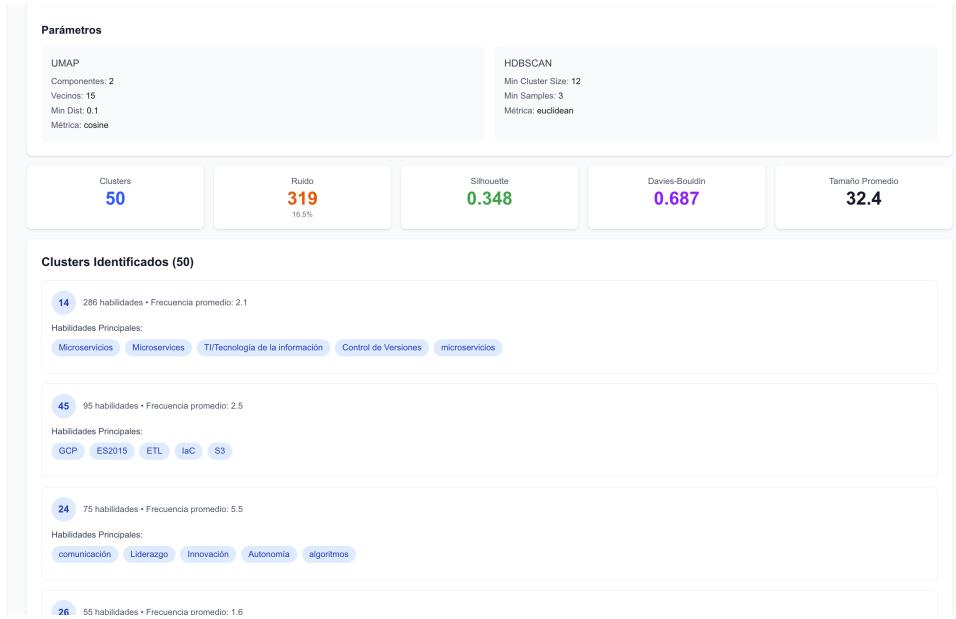


Figura 9.7: Interfaz de configuración de parámetros de clustering mostrando opciones de dataset (Manual Golden 300, Pipeline A Golden 300, Pipeline B Golden 300, Pipeline A Todos 30k) y etapa ESCO (pre y post) con métricas de evaluación.