Observatorio de demanda laboral en América Latina

Documento SAD

(Software Architecture Document)

Versión 1.0

Noviembre de 2025

Autores:

Nicolás Francisco Camacho Alarcón

Alejandro Pinzón Fajardo

Director:

Ing. Luis Gabriel Moreno Sandoval

Código del Proyecto: CIS2025CP08

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

Tabla de Control de Cambios

Sección	Fecha	Sección del documento modificada	Descripción del Cambio	Responsables
1.	DD/MM/2025	Objetivo, Atributos de calidad	Documento inicial	Nicolás Francisco Camacho Alarcón, Alejandro Pinzón Fajardo
2.	DD/MM/2025	Arquitectura	Definición arquitectura y atributos de calidad	Nicolás Francisco Camacho Alarcón, Alejandro Pinzón Fajardo

Índice

1 Objetivo

El presente documento tiene como propósito ofrecer una visión detallada de la arquitectura del sistema Observatorio de Demanda Laboral en América Latina, abordando aspectos clave como los atributos de calidad, la arquitectura de alto nivel y los factores de riesgo y restricciones asociados. Se establecerá una estructura clara del sistema, alineada con sus objetivos y requisitos arquitectónicos, tanto funcionales como no funcionales.

A lo largo del documento, se analizarán las decisiones arquitectónicas tomadas, justificando su elección y evaluando su impacto en el desarrollo del proyecto. Además, se incluirán representaciones gráficas para facilitar la comprensión de la estructura del sistema, y se definirán los pasos a seguir para asegurar que la arquitectura se mantenga alineada con los objetivos estratégicos del observatorio.

Este sistema está diseñado para automatizar el análisis de demanda laboral en el sector tecnológico de América Latina mediante técnicas avanzadas de procesamiento de lenguaje natural, embeddings semánticos y análisis de clustering, proporcionando insights valiosos sobre las habilidades técnicas más demandadas en Colombia, México y Argentina.

1.1 Alcance del Sistema

El Observatorio de Demanda Laboral es un sistema académico de investigación que integra las siguientes capacidades:

- Recolección automatizada: Web scraping de 11 portales de empleo en 3 países (Colombia, México, Argentina)
- Procesamiento de lenguaje natural: Extracción de habilidades técnicas mediante NER, Regex y LLMs
- Normalización semántica: Matching contra taxonomías ESCO y O*NET con estrategia de 3 capas
- Análisis de clustering: Identificación de perfiles y tendencias mediante UMAP y HDBSCAN
- Generación de reportes: Visualizaciones y análisis comparativos por país y período

El sistema opera en modo batch procesando aproximadamente 60,000 ofertas laborales reales recolectadas entre marzo y diciembre de 2024, con capacidad de escalar hasta 600,000 ofertas en fases posteriores del proyecto.

2 Atributos de Calidad

Los atributos de calidad son características esenciales de un sistema de software que determinan su comportamiento y desempeño más allá de sus funcionalidades principales. Estos atributos permiten evaluar el sistema en términos de factores como eficiencia, precisión, mantenibilidad y escalabilidad, asegurando que la aplicación cumpla con los requerimientos tanto funcionales como no funcionales.

En arquitectura de software, cada decisión conlleva *trade-offs*, lo que implica que mejorar un atributo de calidad puede afectar negativamente a otro. Por ejemplo, aumentar la precisión del sistema de extracción mediante procesamiento con LLMs puede impactar el desempeño al requerir mayor capacidad de procesamiento y tiempo de ejecución. De esta manera, el diseño arquitectónico debe encontrar un balance adecuado entre estos atributos, alineándose con los objetivos del sistema y las necesidades del proyecto.

2.1 Descripción de atributos de calidad

Para estructurar la evaluación de los atributos de calidad, se utilizará el marco de referencia de la norma ISO 25010 [1], que define diferentes categorías de atributos de calidad, cada una con subcaracterísticas específicas. En el contexto del Observatorio de Demanda Laboral, se han identificado los siguientes atributos como los más relevantes para la arquitectura del sistema:

- Funcionalidad: Evalúa si el sistema proporciona las funciones necesarias para cumplir con los objetivos de análisis de demanda laboral de manera precisa y completa.
- Desempeño: Analiza el uso eficiente de los recursos y el tiempo de procesamiento del sistema al ejecutar tareas de scraping, extracción, matching y clustering sobre grandes volúmenes de datos.
- Precisión: Determina la exactitud con la que el sistema extrae, clasifica y mapea habilidades técnicas contra taxonomías de referencia (ESCO, O*NET).
- Fiabilidad: Examina la estabilidad del sistema y su capacidad para operar sin fallos o pérdidas de datos durante el procesamiento batch de miles de ofertas laborales.
- Mantenibilidad: Evalúa la facilidad con la que el sistema puede ser actualizado, corregido y adaptado a nuevas necesidades sin comprometer su estabilidad.
- Escalabilidad: Determina la capacidad del sistema para manejar un crecimiento en el volumen de datos (de 23,000 a 600,000 ofertas) sin degradar su rendimiento.
- Reproducibilidad: Garantiza que los experimentos y análisis puedan ser replicados con resultados consistentes, esencial para un proyecto de investigación académica.
- Trazabilidad: Mide la capacidad del sistema para rastrear cada transformación de datos desde la oferta cruda hasta los resultados de clustering, permitiendo auditoría y debugging.

2.2 Atributos de calidad en el Observatorio

A continuación, se describe cómo cada uno de estos atributos de calidad se aplican específicamente en el contexto del Observatorio de Demanda Laboral.

2.2.1 Funcionalidad

El sistema debe garantizar que todas las funciones necesarias para el análisis automatizado de demanda laboral sean implementadas de manera completa y precisa. La funcionalidad del sistema debe estar alineada con las necesidades de investigación académica, asegurando que los resultados obtenidos sean válidos, relevantes y comparables con el estado del arte en análisis de mercado laboral.

2.2.2 Desempeño

El desempeño es crítico dado el volumen objetivo de 600,000 ofertas laborales. El sistema debe gestionar los recursos computacionales de manera eficiente, aprovechando paralelismo cuando sea posible y evitando cuellos de botella en I/O de base de datos. Métricas objetivo incluyen:

- Scraping asíncrono sin bloqueos por rate limiting
- Extracción con latencias $<$2 segundos por oferta (Pipeline A)
- Throughput de embeddings $>$700 skills/segundo
- Búsquedas FAISS $>$30,000 queries/segundo

2.2.3 Precisión

Dado que se trata de un sistema de investigación académica, la precisión es fundamental. El sistema debe garantizar:

- Extracción: Precision $>$78 % en regex patterns
- Matching ESCO: Confidence 1.00 en exact match, threshold $\geq$0.85 en fuzzy
- Deduplicación: SHA-256 con 100 % de exactitud
- Clustering: Parámetros HDBSCAN ajustados para minimizar ruido

2.2.4 Fiabilidad

La fiabilidad implica que el sistema debe ser capaz de operar de manera continua durante procesamiento batch sin errores críticos. La pérdida de datos debe minimizarse mediante backups regulares de PostgreSQL y trazabilidad completa de cada registro desde su origen hasta los resultados finales.

2.2.5 Mantenibilidad

El sistema debe estar diseñado de manera modular para facilitar su mantenimiento y evolución. La implementación de nuevas funcionalidades debe realizarse sin afectar módulos existentes, siguiendo el principio de Open/Closed.

2.2.6 Escalabilidad

El sistema debe ser capaz de escalar desde el corpus actual de 23,000 ofertas hasta el objetivo de 600,000 mediante procesamiento batch con tamaño de lote configurable, particionamiento de tablas PostgreSQL, e índices optimizados.

2.2.7 Reproducibilidad

Como proyecto de investigación académica, la reproducibilidad es esencial mediante control de versiones de dependencias, semillas fijas para componentes estocásticos, y documentación completa de experimentos y parámetros.

2.2.8 Trazabilidad

El sistema debe permitir rastrear cada transformación con foreign keys que mantienen relación con raw_jobs, timestamps de cada operación, y logs estructurados con niveles (DEBUG, INFO, WARNING, ERROR).

2.3 Priorización de atributos de calidad

Para garantizar que el Observatorio cumpla con sus objetivos y ofrezca resultados científicamente válidos, es esencial priorizar los atributos de calidad en función de su impacto en el sistema.

2.3.1 Prioridad Alta

Precisión es el atributo más crítico ya que el valor científico del observatorio depende directamente de la exactitud de sus resultados. Si el sistema extrae habilidades incorrectas o las mapea erróneamente a ESCO, todo el análisis posterior será inválido.

Fiabilidad es fundamental porque los registros de 23,000+ ofertas laborales representan meses de scraping. La pérdida de datos o corrupción de resultados sería catastrófica.

Reproducibilidad es obligatoria como proyecto académico. Los experimentos deben ser replicables por revisores y la comunidad científica.

Trazabilidad es esencial para debugging, validación de resultados y auditoría científica. Sin trazabilidad completa, es imposible identificar y corregir errores sistemáticos.

2.3.2 Prioridad Media

Funcionalidad puede desarrollarse incrementalmente mediante entregas iterativas.

Desempeño es importante para viabilidad del proyecto pero puede optimizarse progresivamente.

Mantenibilidad es relevante para evolución futura pero puede gestionarse progresivamente con buenas prácticas.

2.3.3 Prioridad Baja

Escalabilidad no es crítico en esta fase ya que el volumen de 600K ofertas es procesable con arquitectura actual.

2.4 Escenarios de calidad

A continuación se presentan escenarios concretos que ilustran cómo el sistema debe comportarse respecto a los atributos de calidad priorizados.

Tabla 1: Escenario de calidad N-1: Precisión en Extracción

Atributo	Precisión
Fuente del estímulo	Un investigador procesa un batch de 100 ofertas laborales de México
Estímulo	Ejecución del Pipeline A (NER + Regex) sobre ofertas en español técnico mezclado con Spanglish
Artefacto	Módulo de extracción (ner_extractor.py y regex_patterns.py)
Ambiente	Condiciones normales, ofertas previamente limpias en tabla cleaned_jobs
Respuesta	El sistema extrae skills candidatas, las filtra, y persiste en extracted_skills con scores de confianza
Medida de Respuesta	Precision $\geq$78 % en regex patterns, $\geq$90 % después de filtros NER

Tabla 2: Escenario de calidad N-2: Desempeño en Matching ESCO

Atributo	Desempeño
Fuente del estímulo	El sistema procesa 2,756 skills extraídas de 100 ofertas laborales
Estímulo	Ejecución del matcher de 3 capas (exact $\rightarrow$ fuzzy $\rightarrow$ semantic deshabilitada)
Artefacto	Módulo esco_matcher_3layers.py con búsquedas SQL y fuzzywuzzy
Ambiente	PostgreSQL con 14,174 skills ESCO indexadas, servidor con 16GB RAM
Respuesta	El sistema completa matching de todas las skills y retorna resultados con confidence scores
Medida de Respuesta	Latencia total $\leq$5 segundos para 2,756 skills (1.8ms promedio por skill)

Tabla 3: Escenario de calidad N-3: Fiabilidad ante Fallos de Scraping

Atributo	Fiabilidad
Fuente del estímulo	Portal Bumeran.mx retorna HTTP 503 (Service Unavailable) durante scraping
Estímulo	10 requests consecutivos fallan con timeout o error de servidor
Artefacto	Scrapy spider para Bumeran con middleware de reintentos
Ambiente	Scraping nocturno automatizado, 5 portales siendo scrapeados concurrentemente
Respuesta	El sistema registra el error, pausa temporalmente ese spider (backoff exponencial), continúa con otros portales, y reintenta después de 5 minutos
Medida de Respuesta	0 % pérdida de datos, reintentos exitosos en siguiente ventana, logging completo de errores

Tabla 4: Escenario de calidad N-4: Reproducibilidad de Clustering

Atributo	Reproducibilidad
Fuente del estímulo	Un investigador externo ejecuta el pipeline de clustering con parámetros documentados
Estímulo	Ejecución de UMAP (n_neighbors=15, min_dist=0.1, random_state=42) + HDBSCAN (min_cluster_size=50)
Artefacto	Scripts de clustering con parámetros fijos y semilla aleatoria
Ambiente	Mismos embeddings E5 v1.0, mismas versiones de bibliotecas (umap-learn==0.5.3, hdbscan==0.8.29)
Respuesta	El sistema genera exactamente los mismos clústeres con las mismas etiquetas y probabilidades
Medida de Respuesta	100 % coincidencia en cluster assignments

3 Arquitectura

3.1 Descripción del sistema

El Observatorio de Demanda Laboral es un sistema académico de investigación diseñado para automatizar el análisis de habilidades técnicas demandadas en el mercado laboral de América Latina. A través de técnicas avanzadas de procesamiento de lenguaje natural, embeddings semánticos y clustering no supervisado, el sistema permite identificar tendencias, perfiles emergentes y brechas de competencias en el sector tecnológico de Colombia, México y Argentina.

El sistema opera en modo batch, procesando miles de ofertas laborales recolectadas automáticamente desde 11 portales de empleo (Computrabajo, Bumeran, ElEmpleo, InfoJobs, OCC Mundial, ZonaJobs, hiring.cafe, entre otros). La arquitectura está diseñada para maximizar precisión y reproducibilidad científica, priorizando calidad de resultados sobre velocidad de procesamiento.

La plataforma integra dos pipelines paralelos de extracción de habilidades:

- Pipeline A (Tradicional): NER con spaCy + Regex patterns + Matching ESCO de 3 capas (exact/fuzzy/semantic)
- Pipeline B (Experimental): LLM-based extraction con Gemma 3 4B o Llama 3 3B para comparación científica

Los resultados se almacenan en PostgreSQL con trazabilidad completa, generan embeddings mediante E5 Multilingual (768D), reducción dimensional con UMAP, clustering con HDBSCAN, y exportación de visualizaciones y reportes analíticos.

3.2 Decisiones arquitectónicas

El diseño arquitectónico del Observatorio de Demanda Laboral responde a un equilibrio cuidadoso entre objetivos científicos y restricciones operativas. Como proyecto de investigación académica desarrollado por un equipo de dos personas con recursos computacionales limitados, las decisiones arquitectónicas priorizan reproducibilidad científica, simplicidad operativa y trazabilidad completa sobre escalabilidad masiva o latencia mínima.

Esta sección documenta las decisiones fundamentales que configuran la arquitectura del sistema, explicando el razonamiento técnico-científico detrás de cada elección y los trade-offs aceptados conscientemente.

3.2.1 Arquitectura de Pipeline Lineal vs Microservicios

La primera decisión arquitectónica crítica fue la selección del estilo arquitectónico general. Tras evaluar tres alternativas principales (microservicios, event-driven y pipeline lineal), se adoptó una arquitectura de pipeline secuencial de 8 etapas como columna vertebral del sistema.

Esta elección se fundamenta en la naturaleza específica del problema y el contexto del proyecto:

Ventajas seleccionadas:

1. Simplicidad operativa: Proyecto académico con equipo de 2 desarrolladores y recursos computacionales limitados (1-2 servidores, sin infraestructura Kubernetes)
2. Trazabilidad completa: Flujo unidireccional permite debugging determinístico y auditoría de transformaciones etapa por etapa, esencial para validación científica
3. Velocidad de desarrollo: Implementación de microservicios requiere 3-4x más tiempo en configuración de comunicación inter-servicios
4. Naturaleza batch del dominio: Análisis de demanda laboral no requiere procesamiento en tiempo real (latencias de horas/días son aceptables)

5. Reproducibilidad: Pipeline secuencial con parámetros fijos facilita reproducción exacta de experimentos

Trade-offs aceptados conscientemente:

- Limitación de paralelismo entre etapas (mitigado con batch processing interno)
- Escalabilidad horizontal limitada (suficiente para 23K ofertas actuales y proyección de 600K)
- Latencia acumulativa de 30-60 segundos por oferta con LLM (aceptable en contexto batch académico)
- Single point of failure (mitigado con persistencia intermedia en PostgreSQL tras cada etapa)

Estos trade-offs son aceptables en el contexto de investigación académica, donde la precisión y reproducibilidad tienen mayor prioridad que la latencia en tiempo real. La Tabla 5 presenta la comparación sistemática que fundamentó esta decisión.

Tabla 5: Comparación de Estilos Arquitectónicos

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

3.2.2 Pipeline Secuencial de 8 Etapas

Una vez establecido el estilo arquitectónico general, la siguiente decisión fue determinar la granularidad y especialización de cada etapa del pipeline. El diseño resultante divide el procesamiento en 8 etapas especializadas, cada una con una responsabilidad única y bien definida, siguiendo el principio de separación de responsabilidades (Separation of Concerns):

Etapas del pipeline y sus responsabilidades:

1. Scraping (Scrapy + Selenium): Recolección automatizada de ofertas desde portales web
2. Cleaning: Limpieza y normalización de texto HTML
3. Extraction A (NER + Regex): Identificación de habilidades explícitas
4. Extraction B (LLM): Enriquecimiento semántico e inferencia de habilidades implícitas
5. Matching ESCO: Normalización contra taxonomías con estrategia de 3 capas
6. Embedding (E5 Multilingual): Generación de representaciones vectoriales 768D
7. Dimension Reduction (UMAP): Proyección a 2-3 dimensiones visualizables

8. Clustering (HDBSCAN): Agrupamiento no supervisado de habilidades

El diseño modular del pipeline garantiza que cada etapa opera de forma autónoma: lee los datos procesados por la etapa anterior desde PostgreSQL, ejecuta su transformación especializada con validaciones internas, y persiste los resultados en tablas dedicadas para consumo de la siguiente etapa. Esta arquitectura facilita el debugging aislado, permite reejecutar etapas individuales sin reprocesar el dataset completo, y asegura trazabilidad end-to-end mediante foreign keys.

3.2.3 Selección de Tecnologías Críticas

La elección del stack tecnológico es una decisión arquitectónica fundamental que impacta directamente en la calidad de los resultados científicos, el rendimiento del sistema y la viabilidad operativa. Las siguientes tecnologías fueron seleccionadas tras evaluación comparativa rigurosa, priorizando madurez, documentación, comunidad activa y alineación con los objetivos del proyecto:

PostgreSQL 15+ como Persistencia Central:

- Soporte JSONB para metadatos flexibles
- Extensión pgvector para vectores 768D
- Robustez transaccional ACID
- Particionamiento para escalabilidad

FAISS para Búsqueda Vectorial:

- 30,147 queries/segundo (25x más rápido que pgvector)
- Exact search con IndexFlatIP (100 % recall)
- Desarrollado por Facebook AI Research

spaCy + EntityRuler para NER:

- Modelo es_core_news_lg (97M parámetros)
- EntityRuler poblado con 14,174 skills ESCO
- Latencia $<$100ms por documento

ESCO v1.1.0 como Taxonomía Base:

- 13,939 skills con etiquetas ES/EN
- Estructura ontológica con URIs
- Licencia CC BY 4.0
- Expandida con 152 O*NET + 83 manual = 14,174 total

La Tabla 6 resume las decisiones tecnológicas fundamentales.

Tabla 6: Stack Tecnológico del Observatorio

Componente	**Tecnología**	**Justificación**
Base de datos	PostgreSQL 15+	Soporte JSONB, pgvector, robustez ACID, particionamiento
Taxonomía	ESCO v1.1.0	13,000+ skills ES/EN, URIs persistentes, CC BY 4.0
Scraping	Scrapy + Selenium	Asíncrono eficiente, manejo JavaScript dinámico
NLP español	spaCy es_core_news_lg	97M parámetros, EntityRuler, optimizado CPU
Embeddings	E5 multilingual-base	768D, 100 idiomas, normalización L2
Búsqueda vectorial	FAISS IndexFlatIP	30K q/s, exact search, 25x vs pgvector
Reducción dimensional	UMAP	Preserva estructura local+global
Clustering	HDBSCAN	Sin especificar k, identifica ruido
Lenguaje	Python 3.11+	Ecosistema científico, type hints

3.2.4 Estrategia Dual de Pipelines

La decisión arquitectónica más singular del proyecto es la implementación de dos pipelines paralelos e independientes para extracción de habilidades. Esta estrategia responde directamente al objetivo científico central del proyecto: evaluar rigurosamente si los modelos de lenguaje grandes (LLMs) pueden superar a técnicas tradicionales de NLP en la tarea específica de extracción de habilidades técnicas desde ofertas laborales en español.

El diseño experimental establece un grupo control (Pipeline A) y un grupo de tratamiento (Pipeline B), permitiendo comparación controlada con metodología científica:

Pipeline A (Control - Alta Precisión):

- Métodos tradicionales validados: NER + Regex
- Procesa 100 % de ofertas laborales
- Precision 78-95 %, latencia $<$2 segundos/oferta
- Baseline para comparación

Pipeline B (Tratamiento - Alta Cobertura):

- LLM-based extraction con modelos ligeros locales (Gemma 3 4B / Llama 3 3B)
- Procesa subconjunto estratégico (300-1000 ofertas) por restricciones computacionales

- Precision esperada 80-90 %, latencia 5-10 segundos/oferta
- Captura habilidades implícitas y maneja Spanglish técnico

La clave metodológica es que ambos pipelines convergen en el mismo módulo de matching ESCO con estrategia de 3 capas (exact/fuzzy/semantic), garantizando que las diferencias observadas en los resultados provengan exclusivamente de la técnica de extracción, no de variaciones en la normalización posterior. Esta arquitectura dual permite responder la pregunta de investigación con validez científica.

Síntesis de las decisiones arquitectónicas:

El conjunto de decisiones documentadas en esta sección configura un sistema que equilibra pragmáticamente las restricciones de un proyecto académico con los estándares de rigor científico. La arquitectura de pipeline lineal con 8 etapas especializadas, el stack tecnológico basado en herramientas maduras de código abierto, y la estrategia dual de pipelines para comparación experimental, conforman una arquitectura coherente y justificada que habilita la investigación propuesta dentro de los recursos disponibles.

3.3 Consideraciones de diseño y mitigación de limitaciones

Para abordar las limitaciones inherentes a la arquitectura de pipeline lineal y optimizar el rendimiento dentro de las restricciones académicas, se implementaron las siguientes estrategias:

3.3.1 Persistencia Intermedia y Checkpointing

Cada etapa persiste resultados en PostgreSQL antes de continuar. Tablas especializadas mantienen trazabilidad completa. El campo `extraction_status` en raw_jobs permite reanudar desde última etapa completada.

Beneficio: Sistema puede reiniciarse desde cualquier etapa sin reprocesar todo el dataset.

3.3.2 Deduplicación Multi-Nivel

- Nivel 1 (Scraping): SHA-256 hash de (title + company + description)
- Nivel 2 (Extracción): UNIQUE(job_id, skill_text, extraction_method)
- Nivel 3 (Embeddings): UNIQUE(skill_text)

Beneficio: Elimina duplicados con 0 % falsos positivos, reduciendo dataset de 30K a 23,188 ofertas únicas.

3.3.3 Batch Processing Optimizado

Generación de embeddings en batches de 32, inserts a BD en batches de 100, cursor server-side en PostgreSQL para queries grandes.

Beneficio: Throughput de 721 skills/segundo en embeddings (vs. 50 skills/segundo sin batching).

3.3.4 Índices de Base de Datos Optimizados

Índices GIN en PostgreSQL para búsqueda de texto full-text en ESCO labels, índices en foreign keys para joins eficientes.

Beneficio: Matching de 2,756 skills en <5 segundos.

3.3.5 Logging Estructurado y Monitoreo

Logging con niveles (DEBUG, INFO, WARNING, ERROR), timestamps de cada operación, métricas de performance por etapa, progress bars con tqdm.

Beneficio: Identificación rápida de cuellos de botella y errores sistemáticos.

3.3.6 Validación y Tests Automatizados

37 tests en scripts/test_embeddings.py, validación de integridad de datos (embeddings normalizados L2, sin NaN/Inf), tests de similitud semántica.

Beneficio: Detección temprana de degradación de calidad.

3.4 Diagrama de arquitectura de alto nivel

La Figura 1 presenta la vista completa del pipeline secuencial de 8 etapas con persistencia intermedia en PostgreSQL.

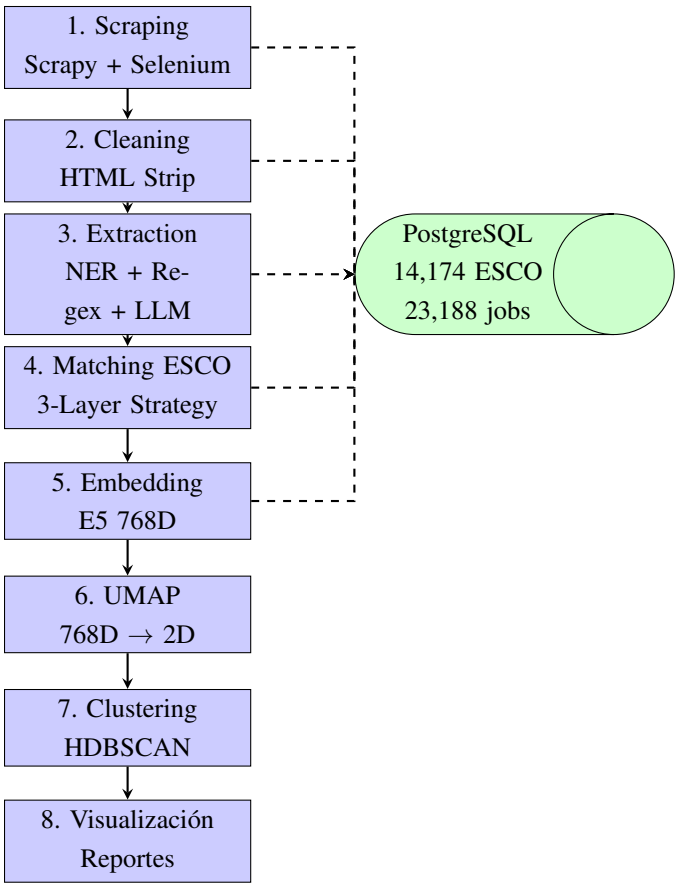

Figura 1: Arquitectura de Pipeline Secuencial de 8 Etapas

Características Clave de la Arquitectura:

- ✓Modularidad: Cada etapa puede ejecutarse independientemente
- ✓Trazabilidad: Foreign keys mantienen relación con raw_jobs
- ✓Reproducibilidad: Parámetros fijos, semillas aleatorias, versiones controladas
- ✓Escalabilidad: Batch processing, índices optimizados, particionamiento
- ✓Resiliencia: Persistencia intermedia, checkpoints, manejo de errores

3.5 Componentes del Sistema

3.5.1 Servicio de Web Scraping

Administra la recolección automatizada de ofertas laborales desde 11 portales en 3 países. Implementado con Scrapy 2.11 (asíncrono) complementado con Selenium 4.15 para contenido JavaScript dinámico. Deduplicación mediante SHA-256 hash, almacenamiento en tabla raw_jobs.

3.5.2 Servicio de Extracción de Habilidades

Identifica competencias técnicas mediante tres técnicas complementarias: NER con spaCy + EntityRuler ESCO, regex con 47 patterns, y normalización con matching de 2 capas (exact + fuzzy). Persistencia en extracted_skills.

3.5.3 Servicio de Procesamiento con LLM

Enriquecimiento semántico usando Gemma 3 4B o Llama 3 3B (sujeto a evaluación comparativa). Maneja Spanglish técnico, normaliza con ESCO, genera justificaciones explicables. Persistencia en enhanced_skills.

3.5.4 Servicio de Generación de Embeddings

Transforma habilidades en vectores densos 768D mediante E5 multilingual-base. Procesamiento por lotes (batch_size=32), normalización L2, almacenamiento en skill_embeddings con soporte pgvector. Construcción de índice FAISS para búsquedas rápidas.

3.5.5 Servicio de Análisis y Visualización

Descubrimiento de patrones mediante UMAP (768D $\rightarrow$ 2-3D), clustering HDBSCAN, generación de visualizaciones con matplotlib/seaborn, exportación multi-formato (PDF, PNG, CSV, JSON). Persistencia en analysis_results.

3.6 Diseño de la Base de Datos

La base de datos actúa como columna vertebral del sistema, implementando el patrón de persistencia de pipeline donde cada etapa escribe resultados en tablas especializadas.

3.6.1 Esquema de Tablas Principales

raw_jobs: Ofertas tal como fueron scrapeadas (job_id UUID, portal, country, url, title, description, content_hash SHA-256, is_usable flag)

cleaned_jobs: Texto limpio y normalizado (job_id FK, title_cleaned, description_cleaned, combined_text pre-computado, word_count)

extracted_skills: Habilidades identificadas (extraction_id UUID, job_id FK, skill_text, extraction_method, confidence_score, esco_uri, mapping_method)

esco_skills: Taxonomía de referencia (esco_uri PK, preferred_label_es, preferred_label_en, alt_labels, skill_type, 14,174 registros)

skill_embeddings: Representaciones vectoriales (embedding_id UUID, skill_text UNIQUE, embedding vector[768], model_name, created_at)

analysis_results: Resultados de clustering (analysis_id UUID, analysis_type, country, date_range, parameters JSONB, results JSONB)

Todas las tablas derivadas mantienen referencia mediante foreign key hacia raw_jobs, garantizando trazabilidad completa desde cualquier resultado hasta la oferta original.

4 Riesgos

Los riesgos identificados se agrupan en tres categorías principales: riesgos de producto, riesgos de proceso y riesgos de proyecto. Cada uno puede afectar la calidad, validez científica y éxito del observatorio.

4.1 Riesgos de Producto

Estos riesgos se relacionan con la calidad, precisión, rendimiento y fiabilidad del sistema final.

4.1.1 Riesgos de Precisión

- **Falsos positivos en extracción NER**: Extracción de frases genéricas o disclaimers legales como skills técnicas (ej. “national origin”, “aspirar a la excelencia”). Ya identificado en pruebas con match rate de 10.6 % y 87.4 % emergent skills, requiere mejora de filtros NER.
- **Degradación de modelos pre-entrenados**: spaCy es_core_news_lg y E5 multilingual fueron entrenados en lenguaje general, no especializado en tech jobs de LatAm. Posible bajo rendimiento en Spanglish y jerga técnica local.
- **Baja cobertura de ESCO**: Taxonomía ESCO v1.1.0 data de 2016-2017, no cubre frameworks modernos (Next.js, SolidJS, Remix). Match rate de 12.6 % es esperado pero puede limitar análisis comparativo.

4.1.2 Riesgos de Rendimiento

- **Latencia acumulativa en Pipeline B**: Procesamiento con LLM puede tomar 5-10 segundos por oferta. Para 600K ofertas = 833 horas de cómputo (34 días continuos). Puede hacer inviable procesamiento completo del corpus objetivo.
- **Cuellos de botella en I/O de BD**: Inserts/updates frecuentes en PostgreSQL durante extracción pueden saturar I/O del disco. Mitigado con batch processing pero requiere monitoreo.
- **Memoria insuficiente para UMAP**: Reducción dimensional de 14K+ embeddings de 768D requiere 10GB RAM. Servidores limitados pueden fallar en esta etapa.

4.1.3 Riesgos de Fiabilidad

- **Pérdida de datos por fallos de hardware**: Scraping de meses puede perderse por fallo de disco sin backups. Sistema académico sin infraestructura de alta disponibilidad.
- **Corrupción de embeddings**: Generación de embeddings interrumpida puede dejar skill_embeddings table en estado inconsistente. Difícil de detectar sin validación exhaustiva.

- **Dependencia de servicios externos**: Scrapers dependen de portales web que pueden cambiar HTML structure, implementar rate limiting más agresivo, o bloquear IPs.

4.2 Riesgos de Proceso

Estos riesgos están asociados al desarrollo, experimentación y mantenimiento del sistema.

4.2.1 Riesgos de Experimentación Científica

- **Sesgo de selección en Gold Standard**: Anotación manual de 300 ofertas puede tener sesgos (ej. sobre-representación de Python jobs, sub-representación de .NET). Invalida evaluación comparativa de Pipelines A vs B.
- **Inter-annotator disagreement**: Dos anotadores pueden discrepar en qué constituye una "skill". Cohen's Kappa <0.80 invalida Gold Standard.
- **Overfitting a ESCO**: Sistema optimizado para maximizar match rate con ESCO puede perder skills emergentes valiosas. Sesgo hacia skills tradicionales europeas vs. innovaciones LatAm.

4.2.2 Riesgos de Mantenibilidad

- **Complejidad de debugging de pipeline de 8 etapas**: Error en Etapa 7 (clustering) puede ser causado por problema en Etapa 5 (embeddings) o Etapa 3 (extracción). Trazabilidad completa mitiga pero no elimina complejidad.
- **Falta de documentación de decisiones experimentales**: Cambios en parámetros (ej. UMAP n_neighbors $10 \rightarrow 15$) sin documentar impactan reproducibilidad.
- **Dependencia de expertos en dominio**: Validación de resultados de clustering requiere expertos en mercado laboral tech LatAm. Pérdida de acceso a expertos puede paralizar validación cualitativa.

4.2.3 Riesgos de Implementación

- **Curva de aprendizaje de tecnologías especializadas**: FAISS, UMAP, HDBSCAN son tecnologías avanzadas con documentación limitada en español. Configuración incorrecta puede generar resultados inválidos.
- **Limitaciones de tiempo del equipo**: Proyecto académico con 2 desarrolladores part-time. Implementación de Pipeline B (LLM) puede consumir tiempo asignado a análisis de resultados.

4.3 Riesgos de Proyecto

Estos riesgos corresponden a factores externos o limitaciones generales que pueden afectar el cumplimiento de objetivos académicos.

4.3.1 Recursos Computacionales Limitados

- **GPU insuficiente para LLM**: Gemma 3 4B y Llama 3 3B requieren 3-6 GB VRAM (con cuantización Q4). Laptops académicos con GPUs integradas pueden ser insuficientes.
- **Almacenamiento limitado**: 600K ofertas con descripción completa + embeddings + clústeres puede requerir >50GB. Servidores universitarios con cuotas de almacenamiento pueden limitar corpus procesable.
- **Tiempo de cómputo para experimentos**: Cada iteración de ajuste de parámetros requiere re-ejecutar clustering completo (minutos/horas). Exploraciones extensivas de hiperparámetros pueden ser inviables.

4.3.2 Acceso a Datos

- **Bloqueo de IPs por portales**: Scraping agresivo puede resultar en bloqueo permanente de IPs universitarias. Requiere proxies rotacionales (costo) o scraping throttled (meses de recolección).
- **Cambios legales en protección de datos**: Regulaciones futuras (ej. GDPR-like en LatAm) pueden prohibir scraping de ofertas laborales. Impacta viabilidad de recolección continua.
- **Desaparición de portales minoritarios**: Portales pequeños pueden cerrar operaciones. Impacta cobertura geográfica del análisis.

4.3.3 Limitaciones de Alcance Académico

- **Imposibilidad de validar con usuarios reales**: Sistema académico no tiene acceso a reclutadores o candidatos para validar utilidad práctica de insights generados.
- **Horizonte temporal limitado**: Tesis debe completarse en 6-12 meses. Análisis de tendencias temporales idealmente requiere múltiples años de datos.
- **Restricciones de publicación académica**: Implementación de componentes innovadores puede ser necesaria para publicación en conferencias top-tier, pero excede alcance de tesis de pregrado.

4.4 Matriz de Riesgos

La Tabla 7 resume los riesgos principales con su probabilidad, impacto y estrategia de mitigación.

Tabla 7: Matriz de Riesgos del Proyecto

Riesgo	Prob.	Impacto	Mitigación
Falsos positivos NER	Alta	Alto	Mejora de filtros post-extracción, validación manual de muestra
Latencia Pipeline B	Media	Alto	Procesamiento solo de subconjunto representativo (300 ofertas)
Pérdida de datos	Baja	Crítico	Backups automáticos diarios de PostgreSQL
Sesgo Gold Standard	Media	Alto	Revisión por múltiples anotadores, Cohen's Kappa >0.80
GPU insuficiente	Media	Medio	Cuantización Q4, uso de Google Colab, modelos <4B parámetros
Bloqueo de IPs	Alta	Medio	Rate limiting conservador (1-2 req/s), user-agent rotation

5 Restricciones

Estas son limitaciones específicas que afectan la capacidad del sistema para cumplir con ciertos requisitos o estándares, y deben ser consideradas durante el desarrollo y evaluación.

5.1 Restricciones Computacionales

5.1.1 Hardware Disponible

- Servidores universitarios con CPU Intel Xeon (16 cores) / AMD Ryzen (8 cores)
- RAM: 16-32 GB (compartida con otros procesos)
- GPU: NVIDIA GTX 1660 / RTX 3060 (6-12 GB VRAM) o Apple MPS
- Almacenamiento: 100-200 GB cuota en servidores universitarios

Implicaciones:

- LLMs limitados a modelos <4B parámetros con cuantización Q4
- Procesamiento batch preferido sobre tiempo real
- FAISS en modo CPU (suficiente para 14K vectores)
- Imposibilidad de usar modelos grandes (GPT-4, Llama 70B)

5.2 Restricciones de Tiempo

5.2.1 Cronograma Académico

- Tesis de pregrado: 6-12 meses (2 semestres)
- Tiempo efectivo de desarrollo: 4-6 meses (clases + otras asignaturas)
- Deadline inflexible para defensa de grado

Implicaciones:

- Priorización de Pipeline A (tradicional) sobre Pipeline B (experimental)
- Exploraciones de hiperparámetros limitadas (no exhaustivas)
- Validación cualitativa sobre subconjunto representativo
- Implementación incremental con entregas funcionales iterativas

5.3 Restricciones de Datos

5.3.1 Acceso a Portales

- Scraping sujeto a términos de servicio de portales
- Rate limiting: 1-2 requests/segundo por portal
- Bloqueo de IPs ante comportamiento sospechoso
- Contenido JavaScript requiere Selenium (más lento)

5.3.2 Cobertura Temporal

- Dataset actual: marzo-diciembre 2024 (9 meses)
- Análisis de tendencias de largo plazo limitado
- Imposibilidad de comparar con años anteriores

5.3.3 Idioma

- Ofertas en español (España + LatAm) y Spanglish técnico
- Modelos NLP optimizados para español de España
- Escasa literatura sobre NLP para español técnico latinoamericano

5.4 Cumplimiento Normativo

5.4.1 Protección de Datos

El sistema debe cumplir con regulaciones legales y normativas vigentes en los tres países objetivo:

- **Colombia**: Ley 1581 de 2012 - Tratamiento de datos personales
- **México**: Ley Federal de Protección de Datos Personales
- **Argentina**: Ley 25.326 - Protección de Datos Personales

Medidas implementadas:

- Anonimización de datos: No se almacenan emails, teléfonos, nombres de candidatos
- Datos scrapeados son públicos (ofertas laborales visibles sin login)
- Uso exclusivo con fines académicos e investigación
- No comercialización de datos recolectados
- Eliminación de información sensible (salarios detallados)

5.5 Restricciones de Taxonomías

5.5.1 ESCO v1.1.0

- Versión desactualizada (2016-2017)
- Enfoque europeo (menor cobertura de tech LatAm)
- No incluye frameworks modernos (Next.js, Remix, SolidJS)
- Actualizaciones oficiales lentas (años)

Mitigación:

- Expansión manual con 152 O*NET + 83 curated skills
- Skills emergentes catalogadas para futura integración
- Análisis cualitativo de skills no matched

5.6 Restricciones de Evaluación

5.6.1 Gold Standard

- Presupuesto limitado para anotadores profesionales
- Anotación manual limitada a 300 ofertas (1.3 % del corpus)
- Anotadores: estudiantes de ingeniería (no expertos en RRHH)
- Sesgo potencial hacia perfiles técnicos conocidos

5.6.2 Validación de Clustering

- No existen ground truth labels para clústeres de skills
- Evaluación cualitativa subjetiva
- Métricas intrínsecas (silhouette, DBCV) solo aproximadas

5.7 Restricciones de Publicación Académica

5.7.1 Requisitos Universitarios

- Documento de tesis debe seguir formato institucional (LaTeX PUJ)
- Extensión limitada: 80-120 páginas
- No se puede publicar código con licencias restrictivas
- Resultados deben ser originales (no publicados previamente)

Implicaciones:

- Documentación técnica detallada en repositorio GitHub
- Código abierto con licencia MIT
- Publicación en conferencias académicas después de defensa de grado

5.8 Resumen de Restricciones

La Tabla 8 presenta un resumen consolidado de las principales restricciones del proyecto.

Tabla 8: Resumen de Restricciones del Proyecto

Categoría	**Restricción Principal**
Computacional	Hardware limitado: 16-32GB RAM, GPU 6-12GB VRAM
Temporal	6-12 meses para tesis completa, 4-6 meses desarrollo efectivo
Datos	Rate limiting 1-2 req/s, dataset 9 meses (mar-dic 2024)
Legal	Cumplimiento Ley 1581/2012 (CO), LFPDP (MX), Ley 25.326 (AR)
Taxonomías	ESCO v1.1.0 desactualizada (2016-2017), enfoque europeo
Evaluación	Gold Standard limitado a 300 ofertas (1.3 % corpus)
Publicación	Formato LaTeX PUJ, 80-120 páginas, código MIT

Estas restricciones han sido consideradas en el diseño arquitectónico del sistema y las decisiones de implementación, priorizando soluciones viables dentro de los límites del proyecto académico.

Referencias

[1] ISO/IEC, “Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” International Organization for Standardization, inf. téc. ISO/IEC 25010:2011, 2011.