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 2 capas (exact a fuzzy con threshold 0.92, semantic deshabilitada)
Artefacto	Módulo esco_matcher_3layers.py (2 capas activas) 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 procesa ofertas laborales recolectadas automáticamente desde 8 portales de empleo principales (Computrabajo, Bumeran, ElEmpleo, HiringCafe, OCC Mundial, ZonaJobs, Indeed, Magneto). La arquitectura híbrida integra procesamiento síncrono de baja latencia para consultas de usuarios con procesamiento asíncrono distribuido de tareas computacionalmente intensivas, diseñada para maximizar precisión y reproducibilidad científica.

La plataforma implementa dos pipelines paralelos de extracción de habilidades mediante Pipeline A (NER con spaCy más Regex patterns más Matching ESCO de 3 capas) y Pipeline B (extracción basada en LLM con Gemma 3 4B para comparación científica). Los resultados se almacenan en PostgreSQL con trazabilidad completa, generan embeddings mediante E5 Multilingual de 768 dimensiones, aplican reducción dimensional con UMAP, ejecutan clustering con HDBSCAN, y exportan visualizaciones y reportes analíticos.

3.2 Decisiones arquitectónicas

El diseño arquitectónico del Observatorio de Demanda Laboral responde a un equilibrio 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.

3.2.1 Arquitectura Híbrida: Justificación y Patrones

La decisión arquitectónica fundamental fue adoptar 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 de usuarios, y procesamiento asíncrono distribuido de tareas computacionalmente intensivas que pueden requerir minutos u horas.

El primer patrón corresponde al API Gateway implementado mediante Nginx, que actúa como punto único de entrada para todas las peticiones HTTP/HTTPS externas proporcionando routing inteligente, terminación SSL/TLS, rate limiting para protección contra abusos, y logging centralizado para auditoría. El segundo patrón implementa Microservicios en Capas para comunicación Request/Response mediante arquitectura de tres capas: Presentación (Frontend con Next.js), Lógica de Negocio (API con FastAPI), y Persistencia (PostgreSQL). El tercer patrón utiliza Event-Driven Architecture mediante comunicación Pub/Sub donde la API y Celery Beat publican tareas a Redis, mientras que Celery Workers consumen estas tareas, ejecutan el procesamiento, y persisten resultados en PostgreSQL.

La selección de arquitectura híbrida se fundamentó en cinco razones principales: dualidad de requisitos de latencia (consultas inmediatas versus procesamiento de horas), escalabilidad horizontal selectiva (workers escalables dinámicamente sin modificar código), simplicidad operativa con potencia de procesamiento (mantiene trazabilidad mientras obtiene paralelismo), optimización de recursos (Request/Response evita overhead para operaciones simples mientras Event-Driven maximiza CPU para procesamiento intensivo), y madurez del ecosistema tecnológico (Celery y Redis como combinación probada industrialmente).

La Tabla 5 presenta la evaluación de estilos arquitectónicos considerados durante el diseño.

Tabla 5: Comparación de Estilos Arquitectónicos Evaluados

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

3.2.2 Pipeline Secuencial de 7 Etapas CRISP-DM

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 7 etapas especializadas siguiendo la metodología CRISP-DM adaptada para minería de textos, donde cada componente mantiene una responsabilidad única y bien definida según el principio de separación de responsabilidades.

La Tabla 6 describe las etapas del pipeline con sus tecnologías, responsabilidades y tiempos característicos de procesamiento.

Tabla 6: Etapas del Pipeline CRISP-DM y Especificaciones Técnicas

Etapa	Tecnología	Responsabilidad	Tiempo
1. Scraping	Scrapy + Selenium	Recolección automatizada desde 8 portales (Colombia, México, Argentina)	2-4 horas
2. Cleaning	BeautifulSoup	Limpieza HTML, normalización Unicode, extracción texto plano	5-10 minutos
3. Extraction	spaCy NER + Regex + LLM	Identificación habilidades explícitas (Pipeline A) e implícitas (Pipeline B)	20-30 minutos
4. Enhancement	Gemma 3 4B	Enriquecimiento semántico, matching ESCO con 3 capas (exact/fuzzy/semantic)	1-2 horas
5. Embedding	E5 Multilingual	Generación vectores densos 768D normalizados L2, construcción índice FAISS	15-20 minutos
6. Clustering	UMAP + HDBSCAN	Reducción dimensional 768D a 2-3D, clustering no supervisado con detección ruido	10-15 minutos
7. Visualization	Matplotlib + Seaborn	Generación gráficos interactivos, reportes analíticos, exportación multi-formato	5-10 minutos

El diseño modular del pipeline garantiza que cada etapa opera de forma autónoma mediante el siguiente flujo: 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 al permitir inspeccionar resultados intermedios en cualquier punto del flujo, habilita la reejecución de etapas individuales sin reprocesar el dataset completo (reduciendo tiempo de desarrollo), y asegura trazabilidad end-to-end mediante foreign keys que conectan cada resultado analítico con la oferta laboral original que lo generó.

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 tecnologías fueron seleccionadas tras evaluación comparativa rigurosa, priorizando madurez del ecosistema, calidad de documentación, comunidad activa de soporte, y alineación estratégica con los objetivos del proyecto.

PostgreSQL 15+ fue seleccionado como sistema de persistencia central por su combinación de robustez transaccional ACID, soporte nativo para JSONB permitiendo metadatos flexibles sin esquema fijo, extensión pgvector para almacenamiento de vectores densos de 768 dimensiones, y capacidades de particionamiento

para escalabilidad futura. Para búsqueda vectorial de alta performance se implementó FAISS desarrollado por Facebook AI Research, que alcanza 30,147 queries por segundo mediante exact search con IndexFlatIP garantizando 100 por ciento de recall, superando por factor de 25x el rendimiento de pgvector en consultas de similitud.

El procesamiento de lenguaje natural utiliza spaCy con el modelo entrenado es core news lg de 97 millones de parámetros optimizado para español, complementado con EntityRuler poblado con las 14,174 skills de la taxonomía ESCO para reconocimiento de entidades de alta precisión con latencia inferior a 100 milisegundos por documento. La taxonomía de referencia corresponde a ESCO v1.1.0 publicada bajo licencia Creative Commons BY 4.0, que provee 13,939 habilidades con etiquetas en español e inglés organizadas mediante estructura ontológica con URIs persistentes, expandida estratégicamente con 152 habilidades de O*NET y 124 skills identificadas manualmente durante el gold standard totalizando 14,174 competencias normalizadas.

La Tabla 7 resume las decisiones tecnológicas fundamentales con sus justificaciones técnicas.

Tabla 7: 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 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. La Tabla 8 presenta las características

diferenciales de ambos pipelines.

Tabla 8: Comparación de Pipeline A (Control) vs Pipeline B (Tratamiento)

Criterio	**Pipeline A (Control)**	**Pipeline B (Tratamiento)**
Técnica	NER con spaCy + Regex patterns	LLM local (Gemma 3 4B o Llama 3 3B)
Cobertura	100 % ofertas laborales (30,660 jobs)	Subconjunto estratégico 300-1,000 ofertas
Precisión	78-95 % según gold standard	80-90 % esperada
Latencia	Menor a 2 segundos por oferta	5-10 segundos por oferta
Habilidades	Explícitas con menciones literales	Explícitas + implícitas inferidas
Spanglish	Limitado a diccionario ESCO	Manejo contextual nativo
Rol	Baseline de alta precisión	Evaluación capacidad LLM
Restricción	N/A	Computacional (GPU/CPU local)

La clave metodológica es que ambos pipelines convergen en el mismo módulo de matching ESCO con estrategia de 3 capas (exact matching, fuzzy matching con umbral 85 por ciento, semantic matching con embeddings), garantizando que las diferencias observadas en los resultados provengan exclusivamente de la técnica de extracción utilizada en cada pipeline, no de variaciones en la normalización posterior. Esta arquitectura dual con punto de convergencia controlado permite responder la pregunta de investigación con validez científica.

El conjunto de decisiones documentadas en esta sección configura un sistema que equilibra pragmáticamente las restricciones de un proyecto académico desarrollado por dos personas con recursos computacionales limitados, con los estándares de rigor científico requeridos para investigación reproducible. La arquitectura híbrida con 7 etapas CRISP-DM especializadas, el stack tecnológico basado en herramientas maduras de código abierto con comunidad activa, y la estrategia dual de pipelines para comparación experimental controlada, conforman una arquitectura coherente y justificada que habilita la investigación propuesta dentro de los recursos disponibles manteniendo trazabilidad completa y reproducibilidad metodológica.

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 seis estrategias de diseño complementarias que maximizan robustez, performance y calidad de datos. La Tabla 9 resume las estrategias implementadas con sus beneficios cuantificados.

Tabla 9: Estrategias de Optimización y Mitigación de Limitaciones

Estrategia	**Implementación**	**Beneficio Cuantificado**
Persistencia Intermedia y Checkpointing	Cada etapa persiste en PostgreSQL antes de continuar. Campo extraction_status permite reanudar desde última etapa completada.	Sistema reinicia desde cualquier etapa sin reprocesar dataset completo
Deduplicación Multi-Nivel	Nivel 1: SHA-256 hash (title+company+desc). Nivel 2: UNIQUE(job_id, skill, method). Nivel 3: UNIQUE(skill_text)	Elimina duplicados con 0 % falsos positivos. Reduce de 30K a 23,188 ofertas únicas
Batch Processing Optimizado	Embeddings batch=32. Inserts BD batch=100. Cursor server-side PostgreSQL	Throughput 721 skills/s (vs 50 skills/s sin batching)
Índices BD Optimizados	Índices GIN para full-text en ESCO labels. Índices en foreign keys para joins	Matching 2,756 skills en menor a 5 segundos
Logging Estructurado y Monitoreo	Niveles DEBUG/INFO/WARNING/ERROR. Timestamps. Métricas por etapa. Progress bars tqdm	Identificación rápida de cuellos de botella y errores sistemáticos
Validación y Tests Automatizados	37 tests en test_embeddings.py. Validación normalización L2. Tests similitud semántica	Detección temprana de degradación de calidad

La estrategia de persistencia intermedia garantiza que fallos en etapas avanzadas del pipeline (como clustering después de 4 horas de procesamiento) no requieran reejecutar desde scraping, reduciendo significativamente el tiempo de debugging durante desarrollo. La deduplicación multi-nivel implementa tres capas de verificación progresivamente más estrictas: el hash SHA-256 en scraping previene almacenamiento de ofertas completamente idénticas, la restricción UNIQUE compuesta en extracción elimina redundancia de habilidades detectadas múltiples veces en la misma oferta por diferentes métodos, y la unicidad de skill text en embeddings asegura que cada competencia se vectorice exactamente una vez independientemente de su frecuencia en el corpus.

El batch processing optimizado reconoce que los modelos de embeddings (E5 Multilingual) y las operaciones de base de datos tienen overhead fijo por invocación, por lo que procesar 32 skills simultáneamente amortiza el costo de carga del modelo en GPU y reduce el número de roundtrips a PostgreSQL de 10,000 transacciones individuales a 100 transacciones batch, multiplicando throughput por factor de 14x. Los índices GIN (Generalized Inverted Index) de PostgreSQL permiten búsqueda full-text eficiente en las 14,174 etiquetas ESCO sin realizar table scans lineales, mientras que índices en foreign keys aceleran joins entre raw jobs y extracted skills de tiempo cuadrático a logarítmico.

El logging estructurado con niveles jerárquicos permite debugging granular durante desarrollo (DEBUG) mientras mantiene logs limpios en producción (INFO), con timestamps precisos que facilitan análisis de per-

formance y detección de regresiones entre ejecuciones. La suite de 37 tests automatizados verifica invariantes críticos como normalización L2 de embeddings (norma euclidiana igual a 1.0 con tolerancia de 1e-6), ausencia de valores NaN o Infinito que corromperían clustering, y coherencia semántica donde habilidades relacionadas como Python y Django deben tener similitud coseno mayor a 0.7.

3.4 Vistas arquitectónicas del sistema

Para documentar completamente la arquitectura híbrida del observatorio, se presentan tres vistas complementarias siguiendo el Modelo 4+1 de Vistas Arquitectónicas. La Vista de Contexto establece las fronteras del sistema y actores externos, la Vista Lógica describe la organización funcional de componentes y sus relaciones, la Vista Física documenta la topología de despliegue en el servidor de producción con especificaciones de infraestructura, y los Diagramas de Secuencia ilustran las interacciones dinámicas entre componentes para los flujos críticos del sistema.

3.4.1 Vista de Contexto

La Vista de Contexto (Figura 1) establece las fronteras del sistema del Observatorio de Demanda Laboral y define los actores externos que interactúan con él. El diagrama muestra cómo el sistema se relaciona con tres categorías de actores: usuarios finales (investigadores académicos, analistas de mercado laboral, estudiantes universitarios), fuentes de datos (8 portales de empleo en Colombia, México y Argentina: Computrabajo, Bumeran, ElEmpleo, HiringCafe, OCC Mundial, ZonaJobs, Indeed, Magneto), y recursos de conocimiento (taxonomía ESCO v1.1.0 con 14,174 habilidades normalizadas, modelos de lenguaje Gemma 3 4B y Llama 3 3B, modelo de embeddings E5 Multilingual).

El sistema expone tres interfaces principales de interacción: API REST para consultas programáticas y automatización, interfaz web Next.js para visualización interactiva de análisis y reportes, y orquestador Celery para procesamiento batch asíncrono. La arquitectura de contexto refleja la naturaleza académica del proyecto, optimizado para análisis científico reproducible y exploración de datos antes que para operaciones transaccionales de alta frecuencia. Los límites del sistema están claramente definidos: el observatorio no almacena datos personales de candidatos, no realiza procesamiento de currículums, ni implementa sistemas de recomendación de empleos; su responsabilidad única es analizar y visualizar tendencias agregadas de habilidades demandadas en el mercado laboral latinoamericano.

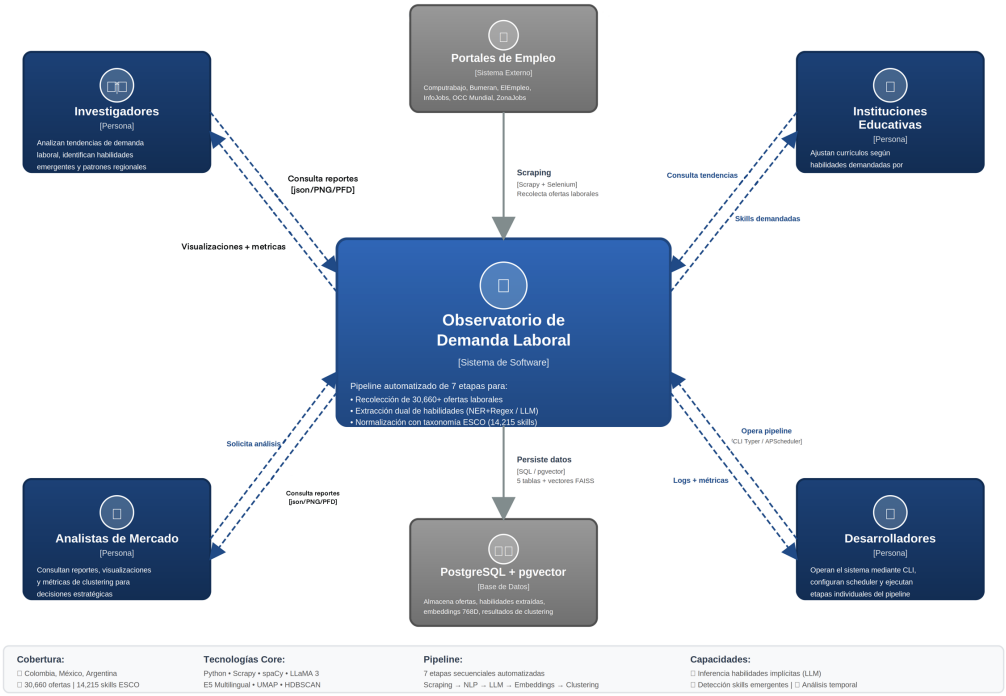

Figura 1: Vista de Contexto del Sistema. Fronteras del sistema, actores externos (usuarios finales, portales de empleo, recursos de conocimiento), interfaces de interacción (API REST, interfaz web, orquestador Celery), y límites de responsabilidad del observatorio.

3.4.2 Vista Lógica

La Vista Lógica (Figura 2) presenta la arquitectura híbrida con sus tres patrones integrados: API Gateway implementado con Nginx para routing y terminación SSL, Microservicios en Capas con comunicación Request/Response para operaciones síncronas (Frontend Next.js, API FastAPI, PostgreSQL), y Event-Driven Architecture con comunicación Pub/Sub mediante Redis para procesamiento asíncrono distribuido (Celery Beat publica tareas, Workers las ejecutan). El diagrama muestra cómo ambos flujos convergen en PostgreSQL como fuente única de verdad, garantizando consistencia de datos independientemente del mecanismo de acceso.

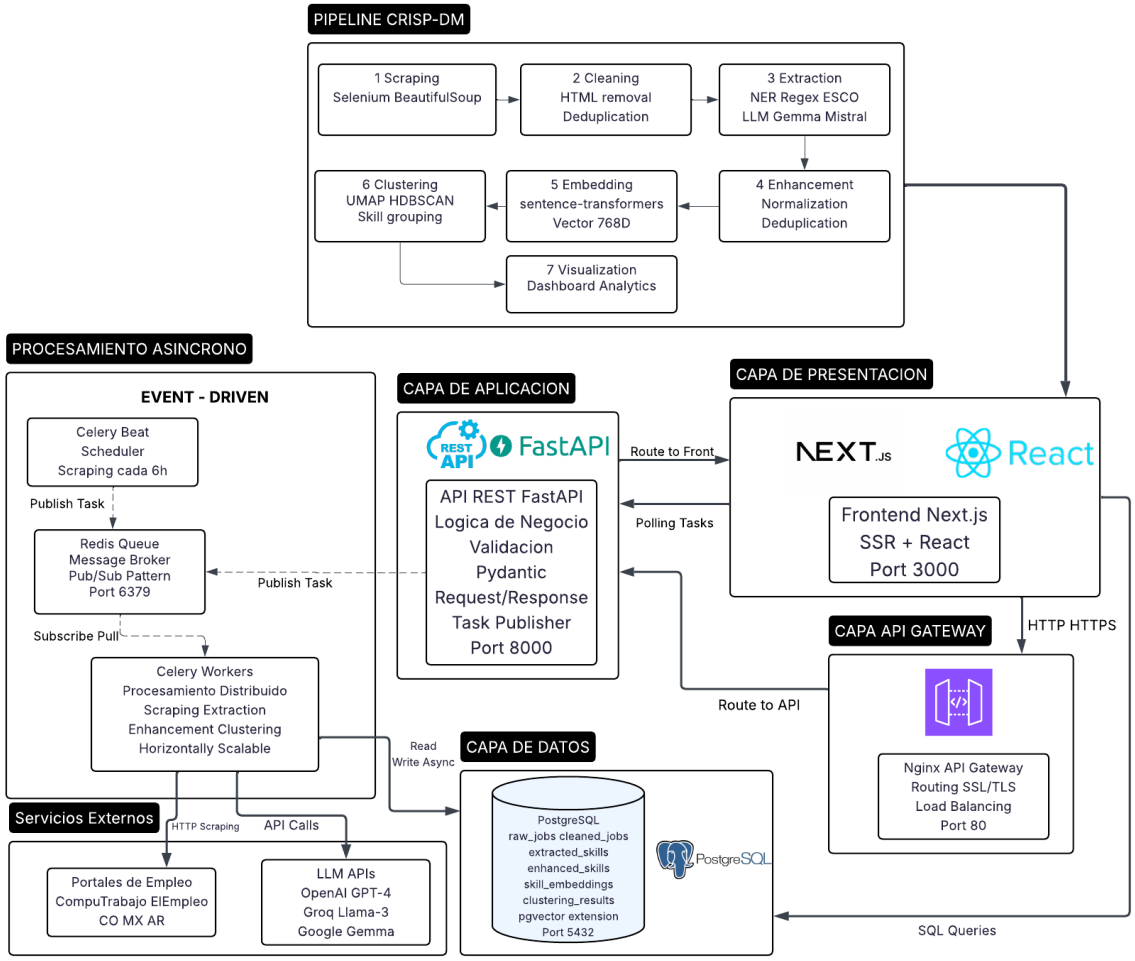

Figura 2: Vista Lógica del Sistema. Arquitectura híbrida integrando API Gateway (Nginx), Microservicios en Capas (comunicación síncrona Request/Response), y Event-Driven Architecture (comunicación asíncrona Pub/Sub mediante Redis y Celery).

3.4.3 Vista Física

La Vista Física (Figura 3) documenta el despliegue en servidor de producción con especificaciones de hardware: CPU Intel Xeon E5-2686 v4 de 8 núcleos a 2.3 GHz, 32 GB RAM DDR4, almacenamiento SSD NVMe de 500 GB, y sistema operativo Ubuntu 22.04 LTS. El diagrama muestra los siete contenedores Docker orquestados mediante Docker Compose (frontend, api, postgres, redis, celery-worker, celery-beat, nginx), red bridge interna con aislamiento de seguridad, mapeo de puertos (80/443 para Nginx, 3000 para Frontend, 8000 para API), y volúmenes persistentes para PostgreSQL y Redis que garantizan durabilidad de datos ante reinicios de contenedores.

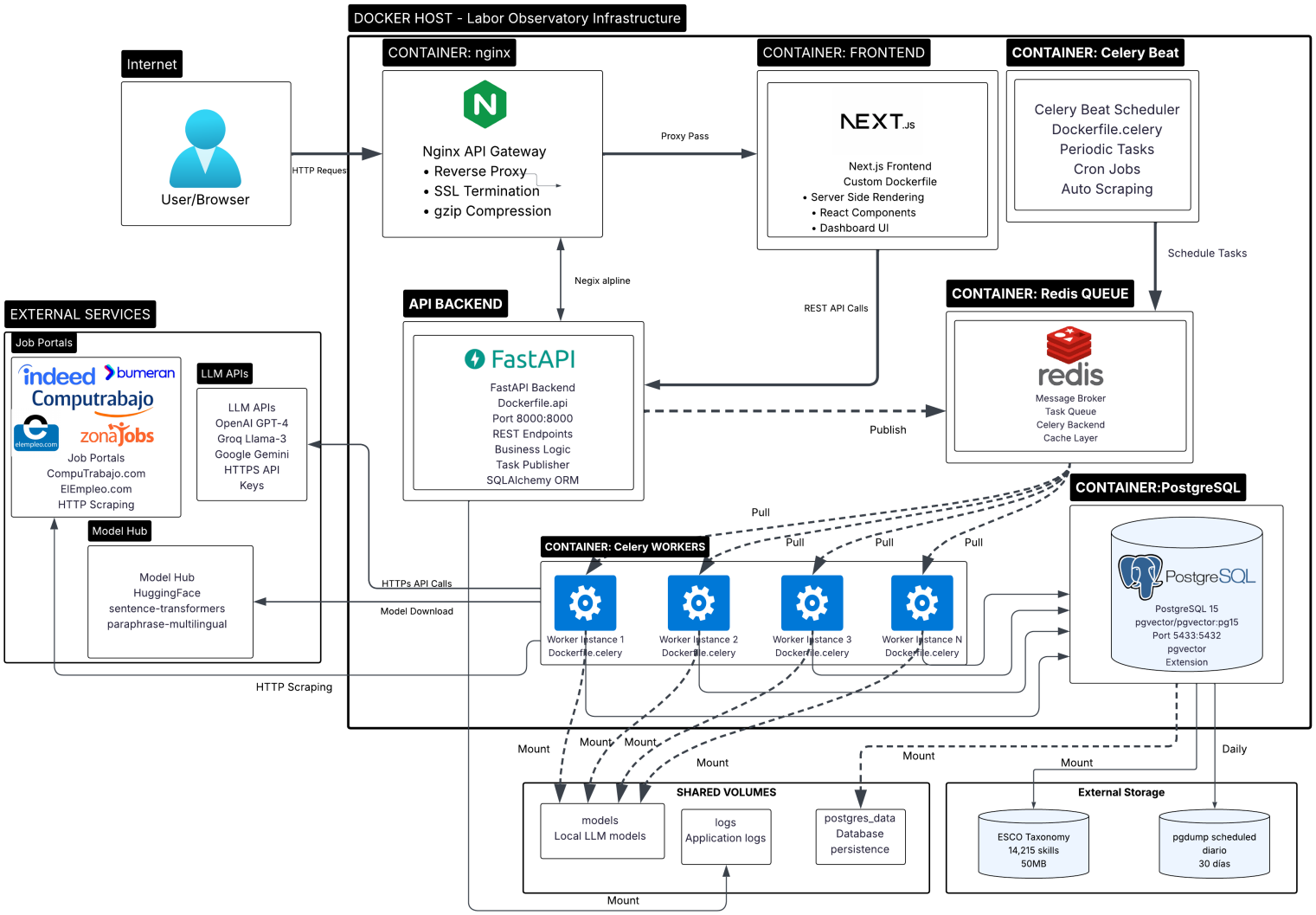

Figura 3: Vista Física del Sistema. Servidor de producción con especificaciones de hardware, contenedores Docker orquestados por Docker Compose, red bridge interna, mapeo de puertos, y volúmenes persistentes para PostgreSQL y Redis.

La arquitectura implementada cumple con cinco características esenciales para el proyecto: modularidad donde cada componente puede ejecutarse y desarrollarse independientemente facilitando debugging y mantenimiento, trazabilidad completa mediante foreign keys que conectan cualquier resultado analítico con la oferta laboral original que lo generó, reproducibilidad científica garantizada por parámetros fijos documentados en código con semillas aleatorias controladas y versiones específicas de dependencias, escalabilidad horizontal mediante batch processing optimizado con índices de base de datos especializados y capacidad de particionamiento futuro, y resiliencia operativa con persistencia intermedia en cada etapa del pipeline más checkpoints que permiten recuperación ante fallos y manejo robusto de errores.

3.4.4 Diagramas de Secuencia

Los Diagramas de Secuencia documentan las interacciones dinámicas entre componentes para los flujos críticos del sistema. La Figura 4 ilustra el flujo de procesamiento asíncrono mediante arquitectura event-driven, que constituye el mecanismo principal para ejecución de tareas computacionalmente intensivas en el observatorio.

El diagrama muestra la secuencia completa de eventos desde que un usuario solicita iniciar un proceso de scraping hasta que los resultados son almacenados y notificados. El flujo comienza cuando el usuario o Celery Beat envía una solicitud POST a la API FastAPI para iniciar scraping. La API valida la solicitud, publica un mensaje en la cola Redis mediante el broker Celery, retorna inmediatamente un task_id al cliente sin bloquear, y marca el estado inicial como PENDING en la tabla task_status de PostgreSQL.

Celery Worker, monitoreando continuamente la cola Redis, detecta el nuevo mensaje, extrae los parámetros de la tarea, actualiza el estado a STARTED, ejecuta el scraper correspondiente (Scrapy + Selenium), procesa las ofertas laborales recolectadas, persiste los resultados en la tabla raw_jobs de PostgreSQL, actualiza el estado final a SUCCESS con metadatos (ofertas procesadas, tiempo de ejecución), y opcionalmente publica un evento de notificación para tareas dependientes en el pipeline.

El cliente puede consultar el progreso de la tarea mediante polling periódico al endpoint GET /tasks/{task_id} de la API, que consulta PostgreSQL y retorna el estado actualizado, o mediante WebSocket para notificaciones en tiempo real (implementación futura). Este patrón de procesamiento asíncrono desacopla la API de las tareas de larga duración, permite escalabilidad horizontal agregando workers sin modificar código, garantiza tolerancia a fallos mediante reintentos configurables y dead letter queues, y mantiene trazabilidad completa registrando cada transición de estado con timestamps precisos.

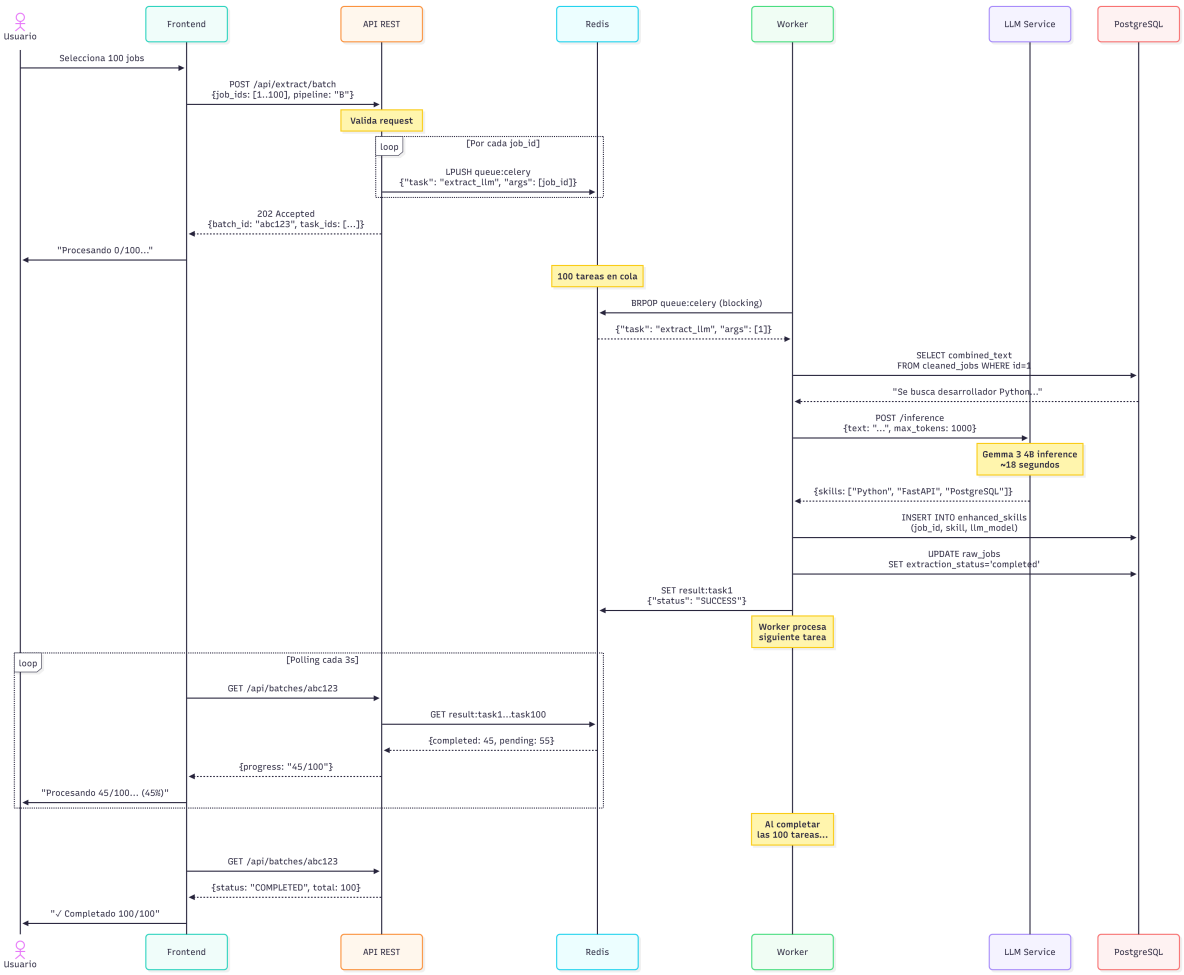

Figura 4: Diagrama de Secuencia Event-Driven. Flujo de procesamiento asíncrono mostrando interacciones entre Usuario/Celery Beat, API FastAPI, Redis (broker), Celery Worker, y PostgreSQL desde solicitud inicial hasta persistencia de resultados y notificación de estado.

La arquitectura event-driven implementada proporciona cuatro ventajas críticas para el observatorio: desacoplamiento temporal donde la API no bloquea esperando procesamiento de horas, escalabilidad elástica mediante workers que pueden agregarse dinámicamente según carga, resiliencia ante fallos con reintentos automáticos y persistencia de estado en cada etapa, y visibilidad operativa mediante logging estructurado y métricas de performance por tipo de tarea. Este diseño arquitectónico habilita procesamiento batch eficiente de decenas de miles de ofertas laborales manteniendo responsividad de la interfaz web y garantizando reproducibilidad científica mediante trazabilidad end-to-end.

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

La tabla raw_jobs almacena ofertas tal como fueron scrapeadas con campos job_id UUID, portal, country, url, title, description, content_hash SHA-256 e is_usable flag. La tabla cleaned_jobs contiene texto limpio y normalizado con job_id FK, title_cleaned, description_cleaned, combined_text pre-computado y word_count.

La tabla extracted_skills registra habilidades identificadas mediante extraction_id UUID, job_id FK, skill_text, extraction_method, confidence_score, esco_uri y mapping_method. La tabla esco_skills funciona

como taxonomía de referencia con esco_uri PK, preferred_label_es, preferred_label_en, alt_labels, skill_type totalizando 14,174 registros.

La tabla skill_embeddings almacena representaciones vectoriales con embedding_id UUID, skill_text UNIQUE, embedding vector[768], model_name y created_at. La tabla analysis_results persiste resultados de clustering con analysis_id UUID, analysis_type, country, date_range, parameters JSONB y 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 10 resume los riesgos principales con su probabilidad, impacto y estrategia de mitigación.

Tabla 10: 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 C-3. Hardware Disponible y Recursos Computacionales

- 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 con al menos 8GB VRAM requeridos para modelos de 7B parámetros, o Apple MPS
- Almacenamiento: 100-200 GB cuota en servidores universitarios

Implicaciones:

- LLMs ejecutables localmente requieren GPU con al menos 8GB VRAM para modelos de 7B 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) sin infraestructura adicional

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 Técnicas (continuación)

5.3.1 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.

5.4 Restricciones de Datos

5.4.1 C-1. Límites de Scraping y C-2. Dinamismo del DOM

- **C-1**: 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**: 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.
- Rate limiting: 1-2 requests/segundo por portal
- Bloqueo de IPs ante comportamiento sospechoso
- Contenido JavaScript requiere Selenium (más lento)

5.4.2 C-5. Heterogeneidad de Formatos y C-6. Incompletitud de Información

- **C-5**: 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**: Muchas ofertas omiten información relevante (salario, requisitos detallados, tecnologías específicas), limitando la profundidad del análisis para ciertos campos.

5.4.3 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.

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

5.4.4 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.

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

5.5 Cumplimiento Normativo

5.5.1 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)

5.6 Restricciones de Taxonomías

5.6.1 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

5.7 Restricciones Metodológicas

5.7.1 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.

- 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

5.7.2 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.

5.7.3 Validación de Clustering

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

5.8 Restricciones de Publicación Académica

5.8.1 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

5.9 Resumen de Restricciones

La Tabla 11 presenta un resumen consolidado de las principales restricciones del proyecto.

Tabla 11: Resumen de Restricciones del Proyecto

Categoría	**Restricción Principal**
Computacional	Hardware limitado: 16-32GB RAM, GPU mínimo 8GB VRAM para modelos 7B params
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.