
**Ingeniería de Sistemas**

# PLAN DE ADMINISTRACIÓN DE PROYECTO

# TÍTULO

Observatorio de Demanda Laboral en Tecnología en Latinoamérica

# ESTUDIANTE(S)

**Nicolas Camacho Alarcón**
**Documento:** cc. 1000942178
**Celular:** 3175714599
**Correo Javeriano:** camachoa.nicolas@javeriana.edu.co

**Alejandro Pinzón Fajardo**
**Documento:** cc. 1052411260
**Celular:** 3115369454
**Correo Javeriano:** alejandro_pinzon@javeriana.edu.co

# DIRECTOR

**Ing. Luis Gabriel Moreno Sandoval**

**Documento:** cc.
**Celular:** 3016278993
**Correo Javeriano:** morenoluis@javeriana.edu.co
**Trabajo:** Pontificia Universidad Javeriana; Profesor Temporal Departamento de Sistemas

Noviembre de 2025

# Prefacio

Este documento tiene como objetivo presentar el plan de gestión del proyecto titulado “Observatorio Automatizado de Demanda Laboral en Tecnología para Latinoamérica”, con el fin de que el lector determine si le interesa profundizar en su contenido. En él se describen los objetivos, alcance, entregables, cronograma, riesgos y herramientas del proyecto, con un enfoque académico y técnico. Está dirigido principalmente a los estudiantes integrantes del equipo, al director del trabajo, a los docentes evaluadores y a cualquier interesado en comprender la planificación de un proyecto de software de mediano alcance con componentes de inteligencia artificial, procesamiento de lenguaje natural y scraping de datos en contexto latinoamericano.

## Historial de Cambios

| Versión | Fecha | Autor(es) | Descripción del cambio | Sección afectada |
|---|---|---|---|---|
| 0.1 | 2025-06-05 | Nicolás Camacho, Alejandro Pinzón | Creación inicial del documento con estructura base | Todas |
| 0.2 | 2025-07-01 | Alejandro Pinzón | Actualización del cronograma y ajuste de estimaciones de esfuerzo | Cap. 4 - Administración |
| 0.3 | 2025-07-20 | Nicolás Camacho | Expansión del análisis de alternativas tecnológicas para scraping y NLP | Cap. 3 - Contexto |
| 0.4 | 2025-08-05 | Alejandro Pinzón | Incorporación de glosario técnico completo con 29 términos y referencias bibliográficas | Glosario y Bibliografía |
| 0.5 | 2025-08-25 | Nicolás Camacho, Alejandro Pinzón | Detalle de criterios de aceptación cuantitativos por fase | Cap. 3 - Plan de Aceptación |
| 0.6 | 2025-09-15 | Alejandro Pinzón | Inclusión de diagramas BPMN para procesos de administración, calidad y riesgos | Cap. 5, 6 |
| 1.0 | 2025-11-15 | Nicolás Camacho, Alejandro Pinzón | Versión final aprobada para entrega formal | Todas |

Tabla 1: Historial de versiones del documento SPMP

**Nota:** Este historial refleja las versiones principales del documento durante su desarrollo. Cambios menores de formato, corrección de errores tipográficos y ajustes de redacción no se registran individualmente. El control de versiones detallado del código y documentación técnica se gestiona mediante Git y GitHub según lo descrito en el Capítulo 6.

# CONTENIDO

# LISTA DE TABLAS

# LISTA DE FIGURAS

# Vista General del Proyecto

## 1.1 Visión del Producto

El presente proyecto busca desarrollar un sistema semiautomatizado, ejecutable de forma periódica y local, que permita procesar y segmentar la demanda de habilidades tecnológicas en Colombia, México y Argentina mediante un pipeline de scraping, procesamiento semántico y visualización macro. El sistema permitirá generar insumos estructurados para el posterior análisis de tendencias laborales en el sector tecnológico latinoamericano, fortaleciendo la toma de decisiones por parte de instituciones educativas, investigadores y organismos interesados en cerrar brechas entre la formación y el mercado laboral.

Se espera que, una vez completado, el sistema siente las bases para futuras ampliaciones en cobertura territorial, frecuencia de actualización, refinamiento técnico y despliegue institucional, permitiendo construir un observatorio laboral replicable, contextualizado al español latinoamericano y alineado a estándares internacionales como ESCO o CIUO.

## 1.2 Propósito, Alcance y Objetivos

### 1.2.1 Propósito

Este proyecto se realiza con el fin de sentar las bases técnicas y metodológicas para un observatorio de demanda laboral tecnológica en Latinoamérica, integrando en un solo sistema fases de extracción, procesamiento y visualización de habilidades demandadas en vacantes reales en línea, con especial enfoque en el idioma español y el contexto regional.

### 1.2.2 Alcance

El proyecto contempla la implementación de un pipeline local compuesto por siete etapas secuenciales. Inicia con scraping de ocho portales de empleo en Colombia, México y Argentina, seguido de limpieza y normalización de datos. Posteriormente ejecuta extracción de habilidades mediante dos pipelines paralelos: Pipeline A basado en NER y expresiones regulares, y Pipeline B utilizando modelos de lenguaje grandes para enriquecimiento semántico. Las etapas finales incluyen representación vectorial con embeddings multilingües, clustering con UMAP y HDBSCAN, y generación de visualizaciones macro estáticas junto con análisis temporal.

Quedan fuera del alcance el desarrollo de dashboards interactivos en tiempo real, la construcción de portales web públicos, la automatización continua mediante orquestadores empresariales, y la integración con bases de datos externas o APIs privadas de portales de empleo.

### 1.2.3 Objetivo general

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

### 1.2.4 Objetivos Específicos

El proyecto contempla cuatro objetivos específicos que estructuran su desarrollo. El primero consiste en construir un estado del arte exhaustivo que permita comparar trabajos existentes en observatorios laborales automatizados y técnicas de procesamiento de lenguaje natural aplicadas al español, estableciendo así el marco conceptual y metodológico del sistema. El segundo objetivo busca diseñar una arquitectura modular, escalable y reutilizable para el observatorio, fundamentada en las mejores prácticas identificadas durante la revisión del estado del arte y adaptada a las particularidades del contexto latinoamericano.

El tercer objetivo se enfoca en implementar e integrar técnicas de inteligencia artificial para la identificación, normalización y agrupación semántica de habilidades tecnológicas en ofertas laborales en español, combinando métodos tradicionales basados en reglas con modelos de lenguaje modernos. Finalmente, el cuarto objetivo establece la necesidad de validar el desempeño y la robustez tanto de la arquitectura propuesta como de los modelos implementados, mediante métricas cuantitativas de precisión y recall, así como estudios empíricos sobre conjuntos de datos anotados manualmente.

## 1.3 Supuestos y Restricciones

### 1.3.1 Supuestos

El proyecto opera bajo cinco supuestos fundamentales que condicionan su viabilidad técnica y operativa. Se asume que será posible mantener acceso continuo a los portales de empleo seleccionados o sus APIs públicas para la extracción sistemática de vacantes durante todo el período de ejecución. Adicionalmente, se presupone la existencia de un corpus suficientemente amplio y variado de ofertas laborales publicadas en español que permita entrenar y validar los modelos de extracción con representatividad estadística adecuada.

Desde la perspectiva de infraestructura, se asume que el equipo contará con recursos computacionales locales adecuados para ejecutar las tareas técnicas del pipeline, incluyendo procesamiento de lenguaje natural, generación de embeddings y clustering de habilidades. En términos organizacionales, se supone que el equipo mantendrá coordinación fluida y comunicación efectiva durante las semanas de ejecución del proyecto. Finalmente, se presume que los modelos de lenguaje y bibliotecas

de NLP seleccionados tendrán funcionamiento adecuado para el procesamiento de texto en español, particularmente considerando las variantes dialectales de Colombia, México y Argentina.

### 1.3.2 Restricciones

El proyecto enfrenta tres restricciones principales que delimitan su alcance y enfoque metodológico. La primera corresponde a la existencia de carga académica paralela por parte de los integrantes del equipo, quienes deben balancear el desarrollo del observatorio con otras responsabilidades curriculares del programa de Ingeniería de Sistemas, limitando así las horas semanales disponibles para dedicación exclusiva al proyecto.

La segunda restricción corresponde a la capacidad limitada de procesamiento computacional local, operando sin acceso a GPUs de alto rendimiento ni infraestructura de servidores externos, lo cual condiciona la selección de modelos de lenguaje hacia alternativas de tamaño intermedio ejecutables en hardware consumer con cuantización. La tercera restricción refleja el tiempo acotado disponible para experimentación iterativa en componentes que requieren ajuste manual, como el refinamiento de prompts para LLMs, la optimización de parámetros de clustering, y el diseño de visualizaciones interpretables, priorizando implementaciones funcionales sobre exploraciones exhaustivas de hiperparámetros.

## 1.4 Entregables

| Entregable | Descripción | Destinatario | Fecha estimada |
|---|---|---|---|
| Repositorio funcional del sistema | Código completo de scraping, NER, LLMs, clustering y visualización | Docentes evaluadores | Semana 14 |
| Dataset limpio de vacantes y habilidades | Base de datos estructurada con datos procesados | Equipo y docentes | Semana 13 |
| Diccionario de habilidades y embeddings | Archivo con habilidades extraídas, enriquecidas y vectorizadas | Equipo técnico | Semana 10 |
| Visualizaciones macro | Gráficos estáticos de resultados para validación cualitativa | Asesor y evaluadores | Semana 12 |
| Documento explicativo del sistema | Guía metodológica, arquitectura y funcionamiento | Universidad / Archivo final | Semana 15 |
| Pruebas funcionales y logs | Evidencia de validación de módulos y resultados intermedios | Asesor y docentes | Semana 14 |

Tabla 1.1: Entregables del Proyecto

## 1.5 Resumen de Calendarización y Presupuesto

| Fase | Actividad | Semanas |
|---|---|---|
| F1 | Diseño y arquitectura del sistema | 1–2 |
| F2 | Scraping y carga a base de datos | 3–5 |
| F3 | Extracción de habilidades (NER + regex) | 6–7 |
| F4 | Enriquecimiento con LLMs | 8–9 |
| F5 | Embeddings + clustering | 10–11 |
| F6 | Visualización macro y evaluación | 12 |
| F7 | Pruebas, documentación y entrega final | 13–16 |

Tabla 1.2: Fases del Proyecto

| **Recurso** | **Descripción** | **Costo** |
|---|---|---|
| Infraestructura local | Uso de computadoras personales | Sin costo adicional |
| Modelos preentrenados | HuggingFace, spaCy, BETO, etc. | Gratuito (open source) |
| API de LLMs | GPT-3.5 para pruebas (en caso de acceso) | Versión gratuita / académica |
| Licencias y software | VSCode, GitHub, Dash, PostgreSQL | Gratuito |
| Herramientas de documentación | Google Docs, Overleaf, GitHub Wiki | Gratuito |

Tabla 1.3: Ítems Consolidados

## 1.6 Evolución del Plan

El plan seguirá una lógica iterativa inspirada en CRISP-DM y Scrum no estricto. Cualquier cambio al plan será discutido en reuniones semanales y validado por consenso. Las actualizaciones serán registradas en Google Docs y GitHub, y se comunicarán al equipo con antelación. Las decisiones sobre cambios mayores (como rediseño de etapas o reemplazo de técnicas) deberán estar documentadas en el acta de revisión de fase correspondiente.

# Glosario

**1. Portales de empleo**

Son plataformas web donde empresas publican vacantes laborales y profesionales buscan oportunidades. En este proyecto se consideran fuentes como LinkedIn, Computrabajo, Bumeran, ZonaJobs e Indeed, que constituyen insumos primarios para los procesos de scraping y análisis [1, 2].

**2. Web Scraping**

Técnica de recolección automatizada de datos desde páginas web, utilizando librerías como BeautifulSoup, Selenium o Playwright. Permite extraer de forma estructurada información relevante de las ofertas publicadas [3].

**3. Oferta laboral**

Se refiere al anuncio publicado por una organización donde se describe el perfil buscado, incluyendo título del cargo, funciones, requisitos y habilidades deseadas [4].

**4. Base de datos relacional (PostgreSQL)**

Sistema que organiza los datos recolectados en tablas interconectadas, facilitando su consulta, limpieza y posterior análisis mediante estructuras SQL [5].

**5. Normalización de datos**

Proceso de limpieza, estandarización y unificación de formatos para reducir ambigüedad, errores y duplicados, y mejorar la coherencia del análisis posterior [6].

## Procesamiento de texto y extracción de habilidades

**6. Expresiones regulares (Regex)**

Lenguaje sintáctico utilizado para identificar y extraer patrones textuales específicos (como frases que contengan habilidades o requisitos) en grandes volúmenes de texto [7].

**7. Named Entity Recognition (NER)**

Técnica de procesamiento de lenguaje natural (NLP) que identifica y clasifica entidades en un texto, como nombres de habilidades, empresas o tecnologías [8].

**8. Tokenización**

Consiste en dividir un texto en unidades mínimas llamadas “tokens” (palabras, signos u oraciones), facilitando el análisis lingüístico automatizado [8].

**9. Lematización**

Proceso que transforma las palabras a su forma canónica o raíz gramatical, permitiendo uniformar variaciones morfológicas del lenguaje [9].

**10. Stopwords**

Términos frecuentes sin valor informativo (como “de”, “por”, “la”), comúnmente eliminados en tareas de procesamiento textual [8].

**11. Co-ocurrencia**

Medida estadística que indica la frecuencia con que dos o más términos aparecen juntos en un texto, útil para detectar relaciones semánticas [6].

**12. Bigramas y trigramas**

Secuencias de dos o tres palabras consecutivas utilizadas para capturar patrones de lenguaje más complejos que las palabras individuales [1].

## Modelado con LLMs y enriquecimiento semántico

**13. LLM (Large Language Models)**

Modelos de lenguaje de gran escala (como GPT o T5) entrenados sobre corpus masivos, capaces de generar texto, extraer conocimiento implícito y realizar razonamiento contextualizado [8, 10].

**14. Prompt Engineering**

Diseño estratégico de instrucciones o ejemplos para guiar la salida de un LLM, crucial en tareas de extracción de habilidades o clasificación de ocupaciones [10].

**15. Few-shot learning**

Habilidad de los LLMs para realizar tareas complejas con pocos ejemplos, lo cual resulta clave cuando se carece de datasets etiquetados masivamente en español [8].

**16. Chain-of-Thought Reasoning (CoT)**

Técnica que induce a los modelos a razonar paso a paso, mejorando precisión en tareas como clasificación y desambiguación semántica [10].

**17. Infer-Retrieve-Rank (IRR)**

Enfoque que primero infiere una entidad, luego recupera candidatos posibles, y finalmente los rankea con base en relevancia, utilizado para seleccionar habilidades o clasificar ocupaciones [11].

**18. Habilidades explícitas vs implícitas**

Las primeras están textualmente expresadas (“manejo de Python”), mientras que las segundas deben inferirse por contexto (“implementación de modelos supervisados”) [8].

## Representación vectorial y análisis semántico

**19. Embeddings semánticos**

Representaciones numéricas de textos que capturan similitudes semánticas, permitiendo análisis cuantitativos y clustering. Ejemplos incluyen word2vec, BERT y E5 [12, 13].

**20. Embeddings multilingües**

Vectores entrenados para representar texto en múltiples idiomas en un mismo espacio semántico. Son esenciales para manejar contenido mixto español-inglés en ofertas laborales [9, 10].

**21. Modelos de lenguaje en español**

Incluyen variantes como BETO, MarIA, T5-español, que han sido entrenadas en corpus hispanos y se adaptan mejor a tareas de extracción en este idioma [8].

**22. Espacio vectorial**

Marco matemático donde entidades como palabras, frases o documentos son representadas como vectores en un espacio multidimensional [12].

**23. Reducción de dimensionalidad (UMAP)**

Técnica que transforma espacios de alta dimensionalidad en representaciones más simples, conservando la estructura semántica subyacente para facilitar análisis y visualización [7].

## Segmentación y visualización

**24. Clustering (HDBSCAN)**

Algoritmo no supervisado que detecta grupos naturales de observaciones (como habilidades o perfiles laborales) según su similitud semántica, sin requerir número de clusters predefinido [7].

**25. Evaluación por coherencia semántica**

Métrica que mide qué tan bien están agrupadas las instancias similares dentro de un modelo, clave para validar la efectividad del clustering [13].

**26. Silhouette Score**

Indicador que evalúa la calidad de los clusters considerando qué tan cohesionados y separados están entre sí [7].

**27. Visualización de datos**

Proceso de representar información compleja en formatos gráficos o interactivos que permiten interpretar resultados, comunicar hallazgos y apoyar decisiones [4].

**28. Python**

Lenguaje de programación ampliamente utilizado en ciencia de datos y NLP, por su sintaxis sencilla y librerías especializadas como scikit-learn, spaCy, transformers y pandas [8].

**29. Taxonomía de habilidades (ESCO, CIUO-08, O*NET)**

Sistemas jerárquicos y normalizados de clasificación de habilidades y ocupaciones, fundamentales para anclar el análisis a estándares internacionales y mejorar interoperabilidad de los resultados [2, 9].

# Contexto del proyecto

## 3.1 Modelo de Ciclo de Vida

El proyecto "Observatorio Automatizado de Demanda Laboral en Tecnología para Latinoamérica" adopta un enfoque metodológico mixto que combina elementos del modelo CRISP-DM con prácticas ágiles inspiradas en Scrum, adaptadas a un entorno académico. Esta combinación busca asegurar tanto una estructura clara y validada como una flexibilidad en la ejecución iterativa del desarrollo.

El modelo CRISP-DM (Cross-Industry Standard Process for Data Mining) proporciona una base sólida para proyectos de análisis de datos, al organizar el trabajo en fases encadenadas y recurrentes. Su estructura es especialmente adecuada para proyectos que incluyen scraping, procesamiento textual, análisis semántico y generación de visualizaciones, como es el caso de este sistema.

El ciclo de vida del proyecto se compone de seis fases principales encadenadas y recurrentes. La primera fase corresponde a la comprensión del dominio y diseño del sistema, incluyendo revisión crítica del estado del arte, definición del pipeline modular, y planificación por etapas metodológicas. La segunda fase implementa la extracción de datos mediante scraping con spiders personalizados por portal y país, almacenando resultados de forma estructurada en base de datos relacional.

La tercera fase ejecuta procesamiento semántico y enriquecimiento, aplicando NER, expresiones regulares y validación con LLMs para extracción explícita e implícita de habilidades técnicas. La cuarta fase realiza vectorización y clustering, obteniendo embeddings semánticos, reducción de dimensionalidad con UMAP y agrupación de perfiles mediante HDBSCAN. La quinta fase genera visualización y validación, produciendo gráficos estáticos para evaluación cualitativa y cuantitativa por expertos del dominio. Finalmente, la sexta fase consolida documentación y empaquetado final, integrando guía metodológica, código versionado, resultados obtenidos y recomendaciones para futuras iteraciones.

Cada una de estas fases se articula mediante entregables intermedios verificables, como bases de datos limpias, corpus anotados, scripts funcionales y reportes interpretables.

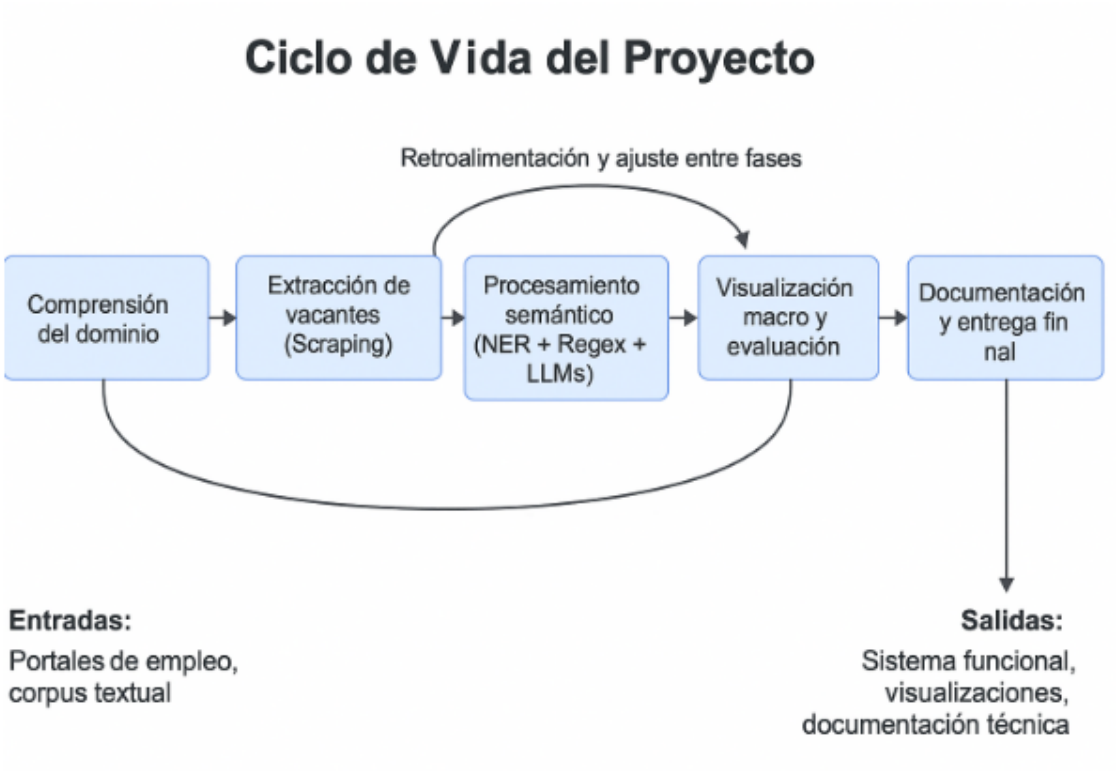


Figura 3.1: Ciclo de Vida del Proyecto basado en CRISP-DM con prácticas ágiles

### 3.1.1 Análisis de Alternativas y Justificación

En el contexto del desarrollo del sistema "Observatorio Automatizado de Demanda Laboral en Tecnología para Latinoamérica", se evaluaron distintas alternativas de modelos de ciclo de vida y enfoques metodológicos. A continuación, se analizan tres enfoques representativos: cascada tradicional, metodología ágil (Scrum completo), y CRISP-DM adaptado con prácticas ágiles.

#### Modelo de Cascada Tradicional

El modelo de cascada propone una secuencia rígida de etapas: análisis de requerimientos, diseño, implementación, pruebas y despliegue. Su lógica lineal facilita la planificación y la documentación desde el inicio, siendo útil en proyectos con requerimientos estables y completamente definidos.

**Ventajas:**

- Claridad en los entregables por fase.
- Control estricto del avance y dependencias.
- Buena trazabilidad documental.

**Limitaciones:**

- Supone requerimientos estables, lo cual no aplica a este proyecto, donde los resultados intermedios pueden modificar fases posteriores.
- No permite incorporar descubrimientos progresivos ni resultados exploratorios.
- La validación se da muy tarde, sin retroalimentación temprana.

**Conclusión:** Debido a la naturaleza exploratoria, técnica y adaptable del proyecto, el modelo de cascada resulta inadecuado.

#### Scrum completo (ágil puro)

Scrum es un marco ágil iterativo que organiza el trabajo en sprints, con roles definidos y eventos regulares. Favorece la adaptación continua y la entrega incremental de valor.

**Ventajas:**

- Flexibilidad ante cambios y descubrimientos técnicos.
- Retroalimentación constante.
- Priorización de entregables más relevantes en cada ciclo.

**Limitaciones:**

- Supone la existencia de un cliente activo y disponible para validar cada sprint, lo cual no aplica a un proyecto académico sin cliente externo.
- En su forma pura, Scrum puede fragmentar demasiado procesos que requieren una visión de pipeline.
- Requiere una madurez organizacional y disciplina de roles que excede el alcance del equipo.

**Conclusión:** Si bien Scrum ofrece beneficios para la iteración y mejora continua, su estructura estricta no se ajusta del todo al carácter investigativo y académico del presente proyecto.

#### CRISP-DM adaptado + prácticas ágiles (modelo elegido)

El modelo CRISP-DM fue diseñado para proyectos de minería de datos y análisis avanzado. Sus fases se ajustan de forma natural a un proyecto basado en scraping, NLP, embeddings y clustering. La combinación con prácticas ligeras de Scrum permite mantener orden sin renunciar a la flexibilidad.

**Ventajas:**

- Alineación directa con las etapas técnicas del proyecto.

- Permite iteración interna por módulo o fase.
- No requiere cliente externo constante.
- Favorece la documentación, trazabilidad y replicabilidad.
- Facilita la planeación modular, con entregables verificables en cada fase.

**Limitaciones:**

- No cubre explícitamente aspectos de comunicación o roles.
- Requiere una adaptación cuidadosa al contexto académico.

**Justificación final:**
Se adopta un modelo híbrido fundamentado en CRISP-DM como columna vertebral del flujo de trabajo, complementado con prácticas ágiles inspiradas en Scrum para planificación, validación y control de avances.

## 3.2 Análisis de Alternativas Tecnológicas

En complemento al análisis metodológico, se presenta a continuación un estudio comparativo de las principales alternativas tecnológicas evaluadas para cada componente crítico del sistema, con justificación de las decisiones finales adoptadas.

### 3.2.1 Herramientas de Web Scraping

Se evaluaron tres alternativas de herramientas para scraping de portales de empleo, cada una con características diferenciadas según el tipo de contenido web a procesar. La Tabla 3.1 presenta la comparación sistemática de estas herramientas considerando descripción, ventajas principales y limitaciones técnicas.

Tabla 3.1: Comparación de Herramientas de Web Scraping

| Herramienta | Descripción | Ventajas | Limitaciones |
|---|---|---|---|
| Scrapy | Framework asíncrono en Python para extracción a gran escala con arquitectura modular basada en spiders | Alto rendimiento vía ejecución asíncrona (Twisted), gestión automática de concurrencia/reintentos, exportación nativa JSON/CSV/SQL, comunidad activa | No ejecuta JavaScript nativamente, curva aprendizaje moderada |
| Selenium | Herramienta de automatización de navegadores para contenido dinámico JavaScript | Renderiza JavaScript completamente, simulación de interacciones humanas (clicks/scroll), compatible multi-navegador | Significativamente más lento que Scrapy, alto consumo CPU/memoria, gestión explícita de drivers |
| Playwright | Alternativa moderna a Selenium (Microsoft) con API simplificada | Más rápido y estable que Selenium, API limpia y moderna, soporte nativo capturas/-red | Comunidad más pequeña, menos ejemplos específicos scraping laboral |

La decisión final adoptó Scrapy como herramienta principal por su rendimiento superior y robustez para scraping masivo de páginas estáticas o semi-dinámicas, características esenciales dado el volumen esperado de ofertas a procesar. Selenium se empleará como respaldo estratégico para portales que requieran ejecución de JavaScript para renderizar contenido dinámico, aceptando el trade-off de mayor latencia a cambio de cobertura completa. Playwright queda documentado como alternativa secundaria viable en caso de problemas técnicos con Selenium, ofreciendo mejor performance que este último pero menor madurez ecosistémica.

### 3.2.2 Sistemas de Gestión de Bases de Datos

Se evaluaron tres alternativas de sistemas de gestión de bases de datos con paradigmas diferentes: relacional tradicional, NoSQL orientado a documentos, y relacional embebido. La Tabla 3.2 sintetiza las características, ventajas y limitaciones de cada opción.

Tabla 3.2: Comparación de Sistemas de Gestión de Bases de Datos

| Sistema | Descripción | Ventajas | Limitaciones |
|---|---|---|---|
| PostgreSQL | SGBD relacional open-source con soporte avanzado para consultas complejas e integridad referencial | Soporte JSON nativo, ACID completo, extensiones búsqueda texto (pg_trgm), escalabilidad probada, integración pandas/SQLAlchemy | Requiere instalación/-configuración local, mayor complejidad administrativa |
| MongoDB | Base NoSQL orientada a documentos con formato JSON/BSON flexible | Esquema flexible para datos semi-estructurados, almacenamiento anidaciones complejas, alto rendimiento escritura masiva | Sin integridad referencial estricta, consultas relacionales difíciles, menos adecuado para análisis estructurado |
| SQLite | BD relacional ligera embebida sin necesidad de servidor | Configuración mínima (archivo único), ideal prototipado rápido/datasets pequeños, compatible SQL estándar | Rendimiento limitado con grandes volúmenes, sin concurrencia escritura eficiente, carece funciones avanzadas PostgreSQL |

La decisión final seleccionó PostgreSQL como base de datos principal por cuatro razones técnicas fundamentales. Primero, su capacidad para manejar esquemas estructurados con integridad referencial garantiza consistencia de datos crítica para análisis posteriores. Segundo, su soporte avanzado para consultas analíticas complejas con índices GIN y B-tree permite agregaciones eficientes sobre el corpus completo. Tercero, su extensibilidad para búsqueda textual mediante pg_trgm facilita matching fuzzy de habilidades. Cuarto, su integración sólida con el ecosistema Python de ciencia de datos mediante SQLAlchemy y pandas reduce fricción en etapas de procesamiento y análisis.

### 3.2.3 Bibliotecas de Procesamiento de Lenguaje Natural (NLP)

Se evaluaron tres alternativas de bibliotecas para procesamiento de lenguaje natural en español, cada una con diferentes trade-offs entre velocidad, precisión y facilidad de uso. La Tabla 3.3 presenta la comparación sistemática.

Tabla 3.3: Comparación de Bibliotecas de Procesamiento de Lenguaje Natural

| **Biblioteca** | **Descripción** | **Ventajas** | **Limitaciones** |
|---|---|---|---|
| spaCy | Biblioteca industrial NLP en Python optimizada para eficiencia con soporte español | Rápida y eficiente producción, modelos preentrenados calidad (es_core_news), soporte tokenización/lematización/POS/NER, integración HuggingFace | Modelos NER genéricos no especializados dominio laboral, requiere ajuste fino para habilidades técnicas |
| Stanford NLP / Stanza | Suite herramientas NLP Stanford con implementación Python | Modelos lingüísticos fundamento académico sólido, análisis sintáctico profundo (dependencias), modelos multilingües corpus universales | Más lento que spaCy, configuración compleja, menor integración embeddings modernos |
| Transformers HuggingFace | Modelos Transformer (BERT, RoBERTa) preentrenados español (BETO, RoBERTuito) | Representaciones contextuales alta calidad, BETO específico español, adaptable fine-tuning dominios | Requiere recursos computacionales significativos, latencia alta tiempo real, fine-tuning necesita datasets grandes |

La decisión final adoptó spaCy como biblioteca principal de NLP por tres razones fundamentales. Primero, su balance entre velocidad y calidad en tareas básicas de NER y tokenización permite procesar el corpus completo en tiempo razonable. Segundo, su facilidad de uso y documentación robusta reduce la curva de aprendizaje del equipo. Tercero, su integración nativa con modelos Transformer de HuggingFace permite combinar eficiencia en tareas básicas con capacidades avanzadas cuando sea necesario. La estrategia complementará spaCy con expresiones regulares personalizadas para habilidades técnicas específicas no cubiertas por modelos genéricos, y evaluará el uso de modelos Transformer como BETO para tareas de enriquecimiento semántico en fases posteriores si los recursos computacionales lo permiten.

### 3.2.4 Modelos de Embeddings Semánticos

Se evaluaron cuatro alternativas de modelos para generación de representaciones vectoriales semánticas de habilidades, considerando tanto modelos especializados en similitud como embeddings generales. La Tabla 3.4 sintetiza la comparación.

Tabla 3.4: Comparación de Modelos de Embeddings Semánticos

| Modelo | Descripción | Ventajas | Limitaciones |
|---|---|---|---|
| BETO | BERT preentrenado corpus masivo español para comprensión contextual | Entrenado específicamente español, representaciones contextuales alta calidad, compatible biblioteca Transformers | No optimizado similitud semántica densos, requiere fine-tuning sentence similarity, alto costo computacional |
| Multilingual-E5 / LaBSE | Modelos multilingües diseñados embeddings densos oraciones con similitud cross-lingual | Optimizados similitud semántica (cosine), soporte múltiples idiomas mismo espacio, LaBSE 100+ idiomas, E5 rendimiento superior benchmarks | Modelos grandes (memoria), menor especialización dominio laboral |
| SBERT | BERT extendido para embeddings oraciones vía siamese networks | Rápido generación vs BERT estándar, modelos multilingües disponibles (paraphrase-multilingual), adopción amplia similitud | Rendimiento variable idioma/dominio, algunos modelos sin español entrenamiento principal |
| fastText | Embeddings basados subpalabras livianos entrenables localmente | Rápido y ligero, maneja OOV vía subpalabras, entrenable corpus específico proyecto | No captura contexto semántico profundo, embeddings estáticos no contextuales, menor rendimiento similitud compleja |

La decisión final adoptó Multilingual-E5 como modelo principal de embeddings por tres características críticas. Primero, su optimización específica para similitud semántica mediante entrenamiento contrastivo lo hace ideal para tareas de matching y clustering de habilidades. Segundo, su soporte multilingüe nativo permite procesar seamlessly el código mixto español-inglés característico del dominio tecnológico latinoamericano. Tercero, su rendimiento superior en benchmarks recientes de sentence similarity garantiza robustez empírica validada. Como alternativa secundaria se documentó LaBSE para escenarios que requieran mayor cobertura multilingüe, y fastText como respaldo computacionalmente liviano para experimentos rápidos o entornos con recursos limitados.

## 3.3 Lenguajes y Herramientas

El desarrollo del Observatorio de Demanda Laboral en Tecnología para Latinoamérica requiere una combinación de herramientas de software, lenguajes de programación, marcos de trabajo y bibliotecas especializadas que conforman el stack tecnológico integral del sistema.

### 3.3.1 Stack Tecnológico Principal

Python constituye el lenguaje de programación principal del proyecto debido a su simplicidad sintáctica, versatilidad para prototipado rápido y robusto ecosistema de bibliotecas especializadas en

ciencia de datos, scraping web y procesamiento de lenguaje natural. La madurez de su comunidad open-source garantiza soporte continuo y abundante documentación técnica para los componentes críticos del pipeline.

La arquitectura de extracción de datos emplea Scrapy como framework principal para scraping asíncrono de alta eficiencia, complementado estratégicamente con Selenium para portales que requieren ejecución de JavaScript y renderizado dinámico de contenido, tal como se detalló en la comparación de herramientas de web scraping presentada anteriormente. El almacenamiento persistente se fundamenta en PostgreSQL como sistema de gestión de bases de datos relacionales, seleccionado por su robustez transaccional, soporte avanzado para consultas complejas mediante índices GIN y JSONB, y capacidades de integridad referencial críticas para garantizar consistencia de datos históricos.

El procesamiento de lenguaje natural integra spaCy con el modelo es_core_news_lg como biblioteca principal, aprovechando su eficiencia computacional y modularidad para tokenización, lematización y reconocimiento de entidades nombradas en español. Esta capa se complementa con expresiones regulares personalizadas diseñadas específicamente para detectar patrones técnicos del dominio laboral latinoamericano que no están cubiertos por modelos genéricos. Como respaldo metodológico se consideran NLTK y Stanza para casos que requieran análisis sintáctico de dependencias o mayor granularidad lingüística, aunque su uso queda condicionado a evaluaciones específicas de desempeño versus complejidad.

El enriquecimiento semántico se implementa mediante Hugging Face Transformers con Gemma 3 4B Instruct como modelo de lenguaje principal, seleccionado tras evaluación comparativa de cuatro alternativas open-source considerando precisión en español, capacidad de razonamiento semántico y viabilidad computacional en hardware limitado. Este modelo permite inferir habilidades implícitas mediante prompting estructurado y normalizar variantes dialectales técnicas. El diseño se complementa con técnicas de Prompt Engineering consistentes en el diseño iterativo de instrucciones especializadas para tareas como normalización de habilidades a taxonomías estándar y extracción contextual de competencias técnicas no explícitas.

La representación vectorial semántica emplea la biblioteca SentenceTransformers para generar embeddings densos mediante modelos multilingües como Multilingual-E5, LaBSE y SBERT, tal como se presentó en la comparación de modelos de embeddings. La selección de Multilingual-E5 como modelo principal se fundamenta en su optimización específica para similitud semántica cross-lingual y superior desempeño en benchmarks de recuperación de información en español. Como alternativa de respaldo se mantiene fastText para casos que requieran embeddings ligeros entrenables localmente sobre vocabulario específico del dominio.

### 3.3.2 Herramientas de Análisis y Visualización

El agrupamiento y reducción de dimensionalidad combina UMAP para proyección no lineal de espacios de embeddings de alta dimensión a representaciones bidimensionales o tridimensionales in-

terpretables, junto con HDBSCAN como algoritmo robusto de clustering basado en densidad que no requiere predefinir el número de grupos y maneja efectivamente ruido y outliers. Como métodos tradicionales de comparación se mantienen disponibles k-means para clustering particional y DBSCAN para validación cruzada de resultados, permitiendo evaluar la estabilidad de las agrupaciones obtenidas mediante múltiples técnicas.

La visualización de resultados emplea principalmente Plotly y Dash como bibliotecas especializadas para generación de gráficos interactivos y dashboards analíticos, aprovechando su capacidad de exportación a formatos estáticos de alta resolución para inclusión en documentos académicos. Como herramientas complementarias se utilizan Matplotlib y Seaborn para generar visualizaciones exploratorias rápidas durante el desarrollo y análisis preliminar de datos, facilitando iteraciones ágiles en el diseño de representaciones gráficas finales.

### 3.3.3 Infraestructura de Desarrollo y Documentación

El control de versiones y colaboración del código se gestiona mediante Git como sistema de control de versiones distribuido, integrado con GitHub como plataforma para hospedaje de repositorios, revisión de código mediante pull requests, seguimiento de issues y automatización de pruebas mediante GitHub Actions. Esta infraestructura garantiza trazabilidad completa de cambios, facilita trabajo paralelo en ramas independientes y documenta decisiones técnicas mediante commits descriptivos y discusiones en pull requests.

La documentación del proyecto se estructura en tres niveles complementarios. Google Docs se emplea para documentación colaborativa en tiempo real durante fases de diseño y planificación, permitiendo comentarios síncronos y versionado automático de decisiones arquitectónicas. Overleaf constituye el entorno principal para redacción final de documentos académicos formales en LaTeX, garantizando calidad tipográfica profesional y gestión eficiente de referencias bibliográficas mediante BibTeX. Finalmente, Markdown se estandariza como formato para documentación técnica embebida en el repositorio GitHub, incluyendo archivos README, guías de instalación, documentación de APIs internas y comentarios explicativos de notebooks Jupyter, facilitando navegación contextual de la documentación junto al código fuente.

## 3.4 Plan de Aceptación del Producto

El Plan de Aceptación del Producto define los criterios mediante los cuales cada entregable del Observatorio de Demanda Laboral en Tecnología para Latinoamérica será evaluado por el equipo académico para considerar su aceptación formal. Cada componente del sistema debe cumplir estándares específicos de calidad técnica, documentación y validación empírica antes de ser integrado al pipeline completo.

### 3.4.1 Diseño técnico y arquitectura

El entregable de diseño técnico y arquitectura será aceptado cuando se cumplan tres criterios fundamentales de completitud y rigor metodológico. Primero, debe existir un diagrama modular del pipeline completo documentado en notación BPMN o equivalente que represente claramente las siete etapas del sistema, los flujos de datos entre componentes y las tecnologías asignadas a cada módulo. Segundo, se requiere una justificación detallada y fundamentada de la elección de tecnologías específicas para cada componente, incluyendo análisis comparativo de alternativas evaluadas, trade-offs considerados y alineación con restricciones de hardware y tiempo del proyecto. Tercero, el documento metodológico inicial que formaliza el ciclo de vida CRISP-DM adaptado y la estrategia de evaluación dual debe estar validado formalmente por el asesor académico mediante firma o aprobación escrita.

La validación de este entregable se ejecutará mediante tres técnicas complementarias que garantizan rigor académico y coherencia técnica. La revisión por pares entre los integrantes del equipo asegurará consistencia interna de la documentación y detección temprana de inconsistencias arquitectónicas. La validación en reunión formal de asesoría permitirá al director del proyecto evaluar la viabilidad técnica y la alineación con objetivos académicos del trabajo de grado. Finalmente, el documento será compartido y versionado tanto en Google Docs para colaboración síncrona como en GitHub para trazabilidad histórica de decisiones arquitectónicas mediante commits descriptivos.

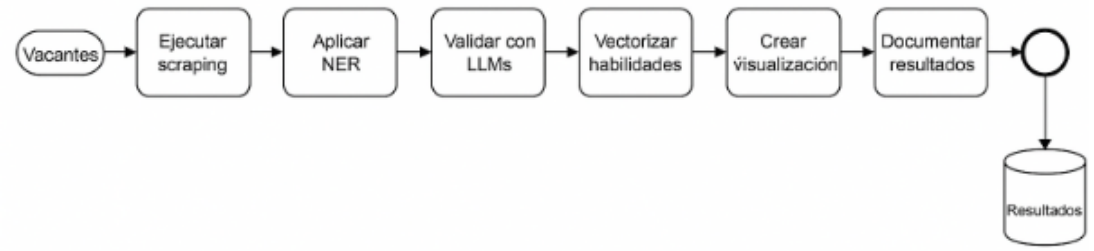


Figura 3.2: Diagrama BPMN del flujo general del proceso del observatorio

### 3.4.2 Sistema de extracción (scraping) y carga a base de datos

El módulo de extracción y carga de datos será considerado aceptable cuando demuestre robustez funcional en tres dimensiones operativas. Primero, deben existir spiders de Scrapy completamente funcionales para al menos dos portales de empleo por país objetivo, totalizando seis implementaciones independientes capaces de extraer título, descripción, empresa, ubicación y fecha de publicación de ofertas laborales. Segundo, el sistema debe demostrar almacenamiento exitoso y consistente de vacantes en PostgreSQL con un corpus mínimo de 500 registros reales extraídos de portales en producción, validando así la viabilidad de la estrategia de scraping a escala. Tercero, el código fuente debe estar documentado con docstrings descriptivos, estructurado modularmente mediante clases reutilizables y

equipado con manejo robusto de excepciones y mecanismos de respaldo ante fallos de red o cambios en estructura HTML de portales.

La validación técnica de este componente empleará tres estrategias de verificación empírica. La validación de funcionamiento en tiempo real será ejecutada por el asesor académico mediante demostración en vivo del proceso de scraping, observando logs de ejecución y confirmando inserción de registros en base de datos durante sesión de asesoría. La revisión del script y estructura de base de datos analizará la calidad del código Python, la normalización del esquema relacional PostgreSQL, y la adecuación de índices para consultas posteriores. Finalmente, se ejecutará comparación sistemática de outputs contra muestras manuales de portales y verificación automática de duplicados mediante consultas SQL de agregación, garantizando integridad y deduplicación efectiva del corpus.

### 3.4.3 Extracción de habilidades explícitas (NER y regex)

El componente de extracción de habilidades explícitas mediante Pipeline A alcanzará aceptación cuando satisfaga tres requisitos de funcionalidad y evaluabilidad. Primero, debe estar completamente integrado al menos un modelo funcional de reconocimiento de entidades nombradas en español, específicamente spaCy con es_core_news_lg extendido mediante EntityRuler con patrones ESCO, demostrando capacidad de identificar tecnologías, lenguajes de programación y frameworks en texto libre. Segundo, las expresiones regulares deben estar adaptadas específicamente al dominio laboral tecnológico latinoamericano, capturando variantes dialectales y patrones sintácticos característicos del español técnico, con resultados de extracción almacenados en formato estructurado evaluable mediante métricas cuantitativas. Tercero, el sistema debe generar anotaciones automatizadas de habilidades extraídas disponibles en formato JSON o CSV para validación posterior, incluyendo contexto de aparición y confianza de detección.

La evaluación de calidad se fundamentará en tres procedimientos de validación complementarios. La validación manual de muestras aleatorias será ejecutada sobre un subconjunto estadísticamente representativo de 50 ofertas laborales, verificando manualmente la completitud y precisión de habilidades detectadas versus lectura humana del texto completo. La revisión de logs y outputs del módulo de extracción analizará trazas de ejecución para identificar patrones de falsos positivos, falsos negativos y degradación de rendimiento en tipos específicos de ofertas. Finalmente, se ejecutará comparación sistemática con glosarios laborales estándar como ESCO y O*NET, calculando tasas de cobertura de la taxonomía y documentando habilidades emergentes no presentes en catálogos oficiales.

### 3.4.4 Enriquecimiento semántico con LLMs

El módulo de enriquecimiento semántico mediante Pipeline B con Gemma 3 4B Instruct será aceptado cuando cumpla tres estándares de diseño, desempeño y trazabilidad. Primero, deben existir prompts completamente definidos y documentados para al menos dos tareas críticas del sistema, específicamente normalización de habilidades a taxonomía ESCO e inferencia de competencias técnicas

implícitas, incluyendo instrucciones del sistema, ejemplos few-shot y formato de salida estructurado JSON. Segundo, los resultados del modelo deben demostrar tasa de precisión superior al 70 por ciento sobre una muestra controlada de 100 ofertas anotadas manualmente, medida mediante métricas de F1-score en extracción y exactitud en normalización taxonómica. Tercero, el sistema debe garantizar trazabilidad completa del razonamiento del modelo mediante logging de prompts enviados, respuestas generadas y justificaciones textuales producidas por el LLM, permitiendo auditoría posterior de decisiones automatizadas.

La validación de este componente experimental seguirá tres estrategias de evaluación rigurosa. La evaluación cualitativa será ejecutada en reunión conjunta con el asesor académico, presentando casos representativos de inferencia semántica exitosa, limitaciones detectadas y análisis de errores sistemáticos del modelo. La comparación contra listas de habilidades base establecerá línea de referencia mediante contraste con outputs de Pipeline A tradicional y anotaciones manuales del gold standard, cuantificando mejora incremental aportada por razonamiento LLM. Finalmente, se generará reporte técnico exhaustivo documentando arquitectura de prompts, ejemplos representativos de inputs y outputs, explicaciones de casos extremos y recomendaciones de refinamiento iterativo del diseño de instrucciones.

### 3.4.5 Embeddings y clustering

El pipeline de representación vectorial y agrupamiento semántico alcanzará criterios de aceptación cuando demuestre tres capacidades técnicas fundamentales. Primero, todas las habilidades extraídas deben estar representadas mediante embeddings densos de 768 dimensiones en formato vectorial numpy o tensors PyTorch almacenados persistentemente, generados mediante Multilingual-E5 con normalización L2 para garantizar comparabilidad mediante similitud coseno. Segundo, la aplicación de clustering debe ejecutarse exitosamente mediante HDBSCAN sobre proyección UMAP bidimensional o tridimensional, produciendo al menos tres clústeres semánticamente significativos interpretables por expertos del dominio, con análisis cualitativo detallado de coherencia temática intra-cluster y separabilidad inter-cluster. Tercero, debe existir visualización preliminar interactiva o estática de alta resolución generada mediante UMAP que represente espacialmente las agrupaciones de habilidades, incluyendo etiquetado de clústeres principales y destacado de habilidades representativas.

La validación de calidad del clustering combinará análisis cuantitativo y cualitativo mediante tres técnicas complementarias. La validación de cohesión semántica entre elementos de un mismo clúster será ejecutada manualmente por los investigadores, verificando que tecnologías agrupadas compartan dominio de aplicación, nivel de abstracción o contexto de uso típico, documentando casos de agrupación exitosa y outliers mal clasificados. La revisión técnica incluirá generación de gráficos de proyección UMAP coloreados por cluster y cálculo de métricas intrínsecas como Silhouette Score, DBCV y estadísticos de distribución de tamaños de clústeres, estableciendo umbrales mínimos de calidad de agrupamiento. Finalmente, se producirá informe de análisis de clústeres interpretado colaborativa-

mente por el equipo, identificando taxonomías emergentes de habilidades, tendencias de co-ocurrencia tecnológica y patrones geográficos o temporales en la demanda laboral.

### 3.4.6 Visualización macro

El módulo de visualización macro cumplirá estándares de aceptación cuando genere productos gráficos interpretables, exportables y validados técnicamente mediante tres criterios de completitud analítica. Primero, debe producir al menos tres tipos de visualizaciones complementarias que representen dimensiones críticas del observatorio: distribución de frecuencia de habilidades más demandadas mediante gráficos de barras o wordclouds ponderados, distribución geográfica de perfiles tecnológicos mediante mapas de calor por país o visualizaciones comparativas multi-región, y proyección espacial de clústeres semánticos mediante scatter plots UMAP coloreados por agrupación con leyendas descriptivas. Segundo, todas las visualizaciones deben ser exportables a formatos de alta resolución PNG o PDF con dimensiones adecuadas para inclusión en documentos académicos, acompañadas de reportes textuales interpretando tendencias identificadas, outliers relevantes y limitaciones de los análisis. Tercero, las interfaces de generación de gráficos deben exhibir usabilidad mínima para revisión técnica, permitiendo ajuste de parámetros básicos como rangos de visualización, filtros temporales o geográficos, y personalización de esquemas de color.

La validación de calidad gráfica y analítica seguirá tres procedimientos de evaluación iterativa. La presentación formal ante el asesor académico incluirá demostración en vivo de generación de visualizaciones con feedback inmediato sobre claridad interpretativa, adecuación de escalas y elección de representaciones gráficas, documentando sugerencias de refinamiento en actas de reunión. La validación del código en entorno local verificará reproducibilidad de gráficos mediante ejecución independiente por ambos integrantes del equipo, confirmando ausencia de dependencias de rutas absolutas y documentación adecuada de bibliotecas requeridas. Finalmente, se ejecutará revisión sistemática de consistencia visual y semántica de las gráficas, verificando que colores, leyendas y títulos sean autoexplicativos, que escalas numéricas sean apropiadas sin distorsiones perceptuales, y que interpretaciones textuales estén fundamentadas empíricamente en datos presentados.

### 3.4.7 Documentación técnica y guía metodológica

El entregable de documentación técnica y guía metodológica constituye componente crítico para asegurar reproducibilidad científica y transferencia de conocimiento, siendo aceptado cuando satisfaga tres estándares de completitud, claridad y trazabilidad. Primero, debe existir redacción clara, completa y estructurada de todas las etapas metodológicas del pipeline CRISP-DM adaptado, incluyendo justificación de decisiones arquitectónicas, descripción detallada de algoritmos implementados, especificación de hiperparámetros seleccionados y documentación de experimentos fallidos con lecciones aprendidas que informen futuras iteraciones. Segundo, el documento debe contener instrucciones de replicación paso a paso suficientemente detalladas para que un investigador externo con conocimientos

equivalentes pueda reproducir el sistema completo, especificando versiones exactas de dependencias Python, comandos de instalación, parámetros de configuración de base de datos y procedimientos de descarga de modelos preentrenados. Tercero, la documentación debe incluir artefactos técnicos completos como logs representativos de ejecuciones exitosas y fallidas, textos completos de prompts LLM utilizados con anotaciones explicativas, scripts auxiliares de preprocesamiento y análisis, y resultados clave tabulados incluyendo métricas de desempeño, ejemplos de outputs y visualizaciones finales.

La validación de calidad documental seguirá tres procedimientos rigurosos de evaluación académica. La revisión integral por parte del asesor académico examinará coherencia argumentativa, profundidad técnica, adecuación de referencias bibliográficas a trabajos relacionados y cumplimiento de estándares de formato institucional LaTeX de la universidad. La comparación con estándares académicos internacionales de replicabilidad verificará que el documento contenga todos los elementos mínimos especificados en guías como CRISP-DM, incluyendo diccionario de datos, diagramas de arquitectura, pseudocódigo de algoritmos críticos y análisis de limitaciones y trabajo futuro. Finalmente, se ejecutará validación cruzada exhaustiva entre documento metodológico y repositorio GitHub, confirmando que todos los módulos mencionados existan en el código, que versiones de software documentadas coincidan con requirements.txt, y que ejemplos de outputs presentados sean auténticamente generados por el sistema y no artificialmente construidos.

## 3.5 Organización del Proyecto y Comunicación

### 3.5.1 Interfaces Externas

En el desarrollo del proyecto se identifican varias entidades externas que, aunque no forman parte directa del equipo de desarrollo, cumplen funciones esenciales en el acompañamiento académico, la evaluación, la provisión de insumos y la validación conceptual del sistema.

| Entidad externa | Rol en el proyecto | Tipo de interacción | Frecuencia |
|---|---|---|---|
| Director del Proyecto (Ing. Luis Gabriel Moreno Sandoval) | Supervisión académica, validación técnica, orientación metodológica | Reuniones de seguimiento, revisión de entregables, aprobación de decisiones clave | Quincenal |
| Docentes evaluadores | Evaluación formal del trabajo de grado, validación de calidad académica | Presentaciones formales, defensa pública, retroalimentación escrita | 2-3 sesiones durante el proyecto |
| Portales de empleo (LinkedIn, Computrabajo, Bumeran, Indeed) | Fuentes de datos primarias (ofertas laborales) | Acceso web mediante scraping, consulta de APIs públicas (si disponibles) | Diaria durante fase de scraping |
| Comunidades técnicas (Stack Overflow, GitHub Issues, foros de spaCy/HuggingFace) | Soporte técnico, resolución de dudas, acceso a ejemplos y soluciones | Consulta de documentación, publicación de issues, revisión de ejemplos | Según necesidad |
| Proveedores de modelos preentrenados (HuggingFace, spaCy, OpenAI) | Acceso a modelos de NLP, embeddings y LLMs | Descarga de modelos, uso de APIs gratuitas o académicas | Según fase técnica |
| Pontificia Universidad Javeriana (Departamento de Sistemas) | Provisión de recursos institucionales, validación académica, aprobación formal del trabajo | Entrega de documentos formales, uso de recursos bibliotecarios, acceso institucional | Permanente |
| Colegas y compañeros de carrera | Revisión cruzada de código, retroalimentación informal, validación de usabilidad | Sesiones de código compartido, discusiones técnicas informales | Ocasional |

Tabla 3.5: Tabla de Interfaces Externas del Proyecto

La gestión de las interfaces externas sigue tres estrategias operativas diferenciadas según el tipo de entidad. La comunicación con el director del proyecto se realizará mediante reuniones quincenales programadas con agenda previa compartida, comunicación por correo electrónico para consultas urgentes que requieran respuesta en menos de 48 horas, y revisión colaborativa de entregables mediante Google Drive compartido para documentos en progreso y GitHub para código versionado con pull requests descriptivos. El acceso a portales de empleo se gestionará respetando estrictamente términos de servicio, archivos robots.txt y políticas anti-scraping de cada sitio, implementando delays entre peticiones de 2-5 segundos, rotación automática de user agents para simular navegación diversa y limitación de tasa de peticiones a máximo 1-2 requests por segundo por portal. El uso de modelos y herramientas de código abierto se documentará exhaustivamente, citando licencias específicas y

repositorios oficiales de cada biblioteca, respetando términos de uso académico cuando aplique y contribuyendo reportes de bugs o mejoras a comunidades open-source cuando sea técnicamente viable.

### 3.5.2 Organigrama y Descripción de Roles

El equipo de desarrollo del proyecto está conformado por dos estudiantes de la carrera de Ingeniería de Sistemas de la Pontificia Universidad Javeriana, cada uno con responsabilidades claramente definidas según sus fortalezas técnicas y organizativas, garantizando cobertura completa de las fases del pipeline mediante distribución balanceada de carga de trabajo.

Nicolás Camacho Alarcón asume el rol de Líder Técnico y Arquitecto del Sistema, siendo responsable del diseño de la arquitectura general del observatorio, coordinación técnica entre módulos interdependientes, toma de decisiones clave sobre selección de modelos de NLP, herramientas de scraping y estructura del pipeline de procesamiento. Adicionalmente supervisa la integración funcional de componentes desarrollados independientemente y garantiza coherencia técnica del sistema completo mediante revisiones de código y validación de estándares de calidad establecidos.

Alejandro Pinzón Fajardo cumple el rol de Desarrollador de Módulos y Responsable de Documentación, estando encargado del desarrollo técnico detallado de componentes específicos del sistema incluyendo implementación de spiders de scraping, módulos de procesamiento de texto, generación de embeddings y construcción de visualizaciones macro. También lidera la redacción de documentos formales académicos siguiendo estándares institucionales LaTeX, la planificación sistemática de pruebas funcionales por módulo y la ejecución de validaciones empíricas sobre muestras controladas del corpus.

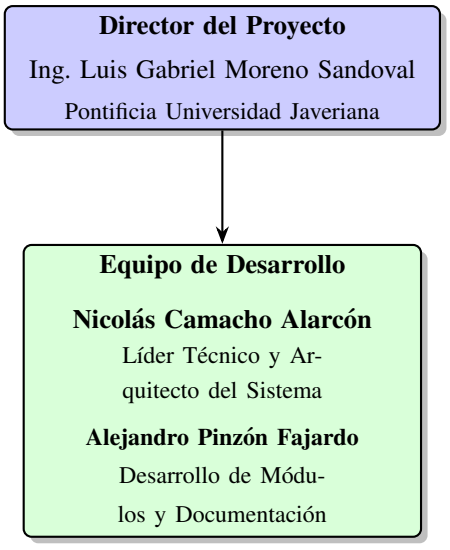


Figura 3.3: Organigrama del equipo de desarrollo del proyecto

# Administración del Proyecto

## 4.1 Métodos y Herramientas de Estimación

El presente proyecto ha sido estimado utilizando una combinación de métodos empíricos y de juicio experto, apoyados en la experiencia previa de los integrantes del equipo, la naturaleza del trabajo requerido, y la complejidad técnica de cada fase. Dado que no se cuenta con herramientas formales de estimación como PERT o COCOMO, se optó por un enfoque ágil y práctico, ajustado a las condiciones reales del equipo.

### 4.1.1 Método de Estimación del Proyecto

El presente proyecto se apoya en una estimación realista de esfuerzos basada en tres principios clave que garantizan viabilidad operativa y académica del cronograma propuesto. El primer principio corresponde al análisis detallado de tareas específicas por fase metodológica, descomponiendo cada etapa del pipeline CRISP-DM en actividades concretas con complejidad técnica estimable. El segundo principio considera la disponibilidad horaria semanal efectiva del equipo, balanceando la dedicación al proyecto con otras responsabilidades curriculares del programa de Ingeniería de Sistemas. El tercer principio integra la experiencia práctica acumulada por los integrantes en proyectos previos similares de scraping, procesamiento de texto y análisis de datos, permitiendo estimaciones fundamentadas empíricamente. Dado que se trata de un trabajo de grado con duración prevista de entre 14 y 16 semanas y un equipo conformado por dos estudiantes con roles complementarios, se estimó un rango total de 440 a 500 horas de trabajo distribuidas entre ambos integrantes según especialización técnica y responsabilidades asignadas.

#### Método de estimación utilizado

Para esta estimación se empleó técnica de descomposición jerárquica de trabajo conocida como Work Breakdown Structure o WBS, donde cada fase del proyecto fue sistemáticamente dividida en tareas concretas y medibles, a las cuales se asignaron horas aproximadas según complejidad algorítmica, herramientas requeridas, curva de aprendizaje esperada y experiencia previa documentada del equipo en componentes similares. Esta estimación bottom-up fue posteriormente contrastada con la disponibilidad semanal esperada de cada integrante mediante proyección de calendario académico, ajustada conservadoramente por compromisos académicos paralelos incluyendo otras asignaturas, evaluaciones y trabajos grupales que compiten por tiempo efectivo de dedicación.

El enfoque adoptado es cualitativo en naturaleza, iterativo en ejecución y conservador en supuestos de productividad. En lugar de aplicar fórmulas matemáticas complejas de estimación como COCOMO que requieren calibración histórica de proyectos previos con métricas detalladas de líneas de código y factores de complejidad, se optó por estimación fundamentada en experticia directa del equipo técnico, validada mediante comparación con planes de proyectos académicos previos similares y distribuida proporcionalmente según carga de trabajo balanceada entre ambos integrantes considerando fortalezas individuales.

**Estimación del esfuerzo por integrante**

| **Integrante** | **Rol principal** | **Horas estimadas** |
|---|---|---|
| Nicolás Camacho | Líder técnico, arquitectura | 170 |
| Alejandro Pinzón | Documentación y pruebas | 130 |
| Alejandro Pinzón | Desarrollo de módulos | 140 |
| **Total estimado** | | **440 horas** |

Tabla 4.1: Tiempo de Esfuerzo por Integrante

**Estimación por fase metodológica**

| **Fase** | **Horas estimadas** |
|---|---|
| Diseño inicial del sistema | 45 |
| Scraping y carga a base de datos | 70 |
| NER y normalización de habilidades | 60 |
| Enriquecimiento con LLMs | 60 |
| Embeddings y clustering | 50 |
| Visualización macro evaluativa | 40 |
| Documentación, validación y ajustes finales | 70 |
| **Total** | **395 horas** |

Tabla 4.2: Tiempo Estimado por Fase

*Nota:* El total por fases metodológicas no considera horas de reuniones, organización, imprevistos o revisión con el asesor, por lo que se reserva un margen adicional de 45 a 55 horas para estas actividades transversales de coordinación y gestión que completan las 440 a 500 horas globales estimadas del proyecto.

**Herramientas empleadas para la estimación**

El proceso de estimación se apoyó en tres herramientas complementarias que facilitaron planificación estructurada y validación colaborativa del cronograma propuesto. La herramienta principal consistió en Google Sheets como plataforma de tabulación y cálculo, donde se registraron todas las actividades descompuestas del WBS con distribución detallada de tiempos estimados por fase metodológica e integrante responsable, permitiendo visualización matricial de carga de trabajo y cálculo automático de totales agregados. Como segundo componente se estableció planificación semanal estimada fundamentada en disponibilidad real del equipo, proyectando dedicación efectiva de 8 a 12 horas por persona por semana considerando compromisos académicos paralelos, lo cual permitió traducir estimaciones totales en horas a cronograma calendario de 14 a 16 semanas de duración. Finalmente, se incorporó retroalimentación continua del director de proyecto Ing. Luis Gabriel Moreno Sandoval mediante revisión de estimaciones en reuniones de asesoría quincenal, validando viabilidad técnica de las cargas propuestas, identificando actividades subestimadas o sobreestimadas, y aprobando formalmente la distribución final del esfuerzo entre ambos integrantes.

## 4.2 Inicio del proyecto

### 4.2.1 Entrenamiento del Personal

Dado que el proyecto Observatorio de Demanda Laboral en Tecnología en Latinoamérica involucra herramientas avanzadas como modelos de lenguaje grandes, scraping web dinámico con JavaScript, procesamiento semántico especializado en español técnico, generación de embeddings multilingües y técnicas de clustering no supervisado basado en densidad, se ha establecido un plan de entrenamiento ligero pero técnicamente enfocado que permita nivelar conocimientos del equipo en aspectos técnicos esenciales del stack tecnológico sin comprometer significativamente el cronograma establecido de 14 a 16 semanas.

El entrenamiento será liderado por Nicolás Camacho, quien cuenta con mayor experiencia técnica previa en tecnologías clave del proyecto incluyendo Python para ciencia de datos, bibliotecas de NLP en español y frameworks de scraping web. Su rol de transferencia de conocimiento incluirá diseño de cápsulas formativas concisas y demostración práctica de implementaciones mediante cuatro mecanismos pedagógicos complementarios.

El primer mecanismo consiste en explicaciones prácticas durante reuniones técnicas semanales, donde Nicolás presentará en vivo el funcionamiento del código que esté desarrollando en cada fase, incluyendo justificación técnica de decisiones arquitectónicas, recomendaciones de buenas prácticas de ingeniería de software y anticipación de errores comunes detectados durante implementación. El segundo mecanismo emplea ejemplos rápidos y contextualizados mediante notebooks Jupyter o scripts Python breves ejecutados sobre datos sintéticos simples, ilustrando conceptos clave como configuración de spiders Scrapy, interacción con Selenium WebDriver, diseño de prompts estructurados para

LLMs, generación de embeddings con SentenceTransformers, proyección dimensional con UMAP y parametrización de clustering HDBSCAN.

El tercer mecanismo facilita lecturas y recursos técnicos recomendados mediante curación de documentación oficial de bibliotecas utilizadas, artículos académicos clave de estado del arte y videos tutoriales cortos disponibles en español o inglés según el tema y nivel de dificultad técnica, compartidos en repositorio GitHub del proyecto con anotaciones contextualizadas. El cuarto mecanismo establece revisión cruzada continua de avances, donde los integrantes aplicarán conocimientos adquiridos directamente en sus tareas asignadas del WBS, recibiendo retroalimentación técnica detallada de Nicolás mediante comentarios en pull requests de GitHub y validación funcional de implementaciones durante sesiones de pair programming remoto.

La planificación detallada del entrenamiento interno por componente técnico se presenta a continuación:

| **Habilidad o tema técnico** | **Integrantes a entrenar** | **Responsable del entrenamiento** | **Método principal** | **Tiempo estimado** |
|---|---|---|---|---|
| Scraping web con Scrapy, Selenium y Playwright | Alejandro | Nicolás | Ejemplos en vivo + código comentado | Semanas 2–3 |
| Procesamiento de lenguaje en español (spaCy, regex, taxonomías laborales) | Alejandro | Nicolás | Lecturas + demos | Semanas 3–5 |
| Uso de LLMs y diseño de prompts | Alejandro | Nicolás | Ejercicios guiados + discusión en grupo | Semanas 5–6 |
| Embeddings multilingües y reducción dimensional | Alejandro | Nicolás | Notebooks breves + aplicación directa | Semanas 6–7 |
| Clustering con HDBSCAN y UMAP | Alejandro | Nicolás | Ejemplo sintético + retroalimentación | Semana 8 |
| Visualización macro con Dash / Plotly | Alejandro | Nicolás | Revisión cruzada de visualizaciones | Semanas 9–10 |
| Documentación técnica y guía metodológica | Todo el equipo | Alejandro | Plantillas + validación por Nicolás | Permanente |

Tabla 4.3: Plan de Entrenamiento Interno

Este entrenamiento se desarrollará de manera continua, práctica y orientada a resolver necesidades técnicas reales del proyecto mediante aprendizaje situado en contexto de desarrollo. No se contempla fase de formación teórica previa extensa desacoplada de implementación, sino que el aprendizaje técnico estará completamente integrado al flujo de trabajo del WBS, adaptado al tiempo disponible según cronograma y priorizando componentes más urgentes del pipeline según secuencia de depen-

dencias entre fases.

Nicolás brindará adicionalmente apoyo técnico puntual cuando surjan dudas específicas de debugging, optimización de rendimiento o selección entre alternativas de implementación, fomentando cultura de colaboración técnica horizontal y mejora iterativa donde se valoran preguntas fundamentadas, exploración autónoma de documentación oficial y experimentación controlada con validación empírica de resultados. La documentación técnica del sistema, liderada por Alejandro Pinzón en su rol de responsable de documentación académica, incluirá además explicaciones simplificadas didácticas de procesos técnicos clave mediante diagramas de flujo y pseudocódigo, con el fin de reforzar aprendizaje colectivo del equipo, facilitar onboarding de futuros colaboradores y dejar registro permanente de decisiones técnicas críticas para futuras iteraciones o réplicas académicas del observatorio.

### 4.2.2 Infraestructura

Para el adecuado desarrollo del proyecto "Observatorio de Demanda Laboral en Tecnología en Latinoamérica", se requiere una infraestructura técnica que permita la implementación modular del sistema, desde la recolección de datos hasta el análisis semántico y la documentación de resultados. Esta infraestructura combina herramientas de software de código abierto, recursos computacionales locales y servicios colaborativos en la nube, con una planeación clara sobre su adquisición, configuración y mantenimiento.

Los detalles completos de herramientas de software, especificaciones de equipos, y actividades de obtención, despliegue y mantenimiento se encuentran documentados en las tablas 9, 10, 11, 12 y 13 del presente documento.

## 4.3 Planes de Trabajo del Proyecto

### 4.3.1 Descomposición de Actividades

La estructura de descomposición de actividades del proyecto (WBS - Work Breakdown Structure) se organiza por fases metodológicas, cada una con sus respectivas subactividades. La siguiente tabla representa la estructura general adoptada para la ejecución, según se detalla en la Tabla 14.

### 4.3.2 Calendarización

La calendarización define las fechas estimadas de inicio y finalización de cada fase principal, así como su secuencia de ejecución, distribuida en una duración total de 16 semanas, tal como se muestra en la Tabla 15 y la Figura 4 (Carta Gantt).

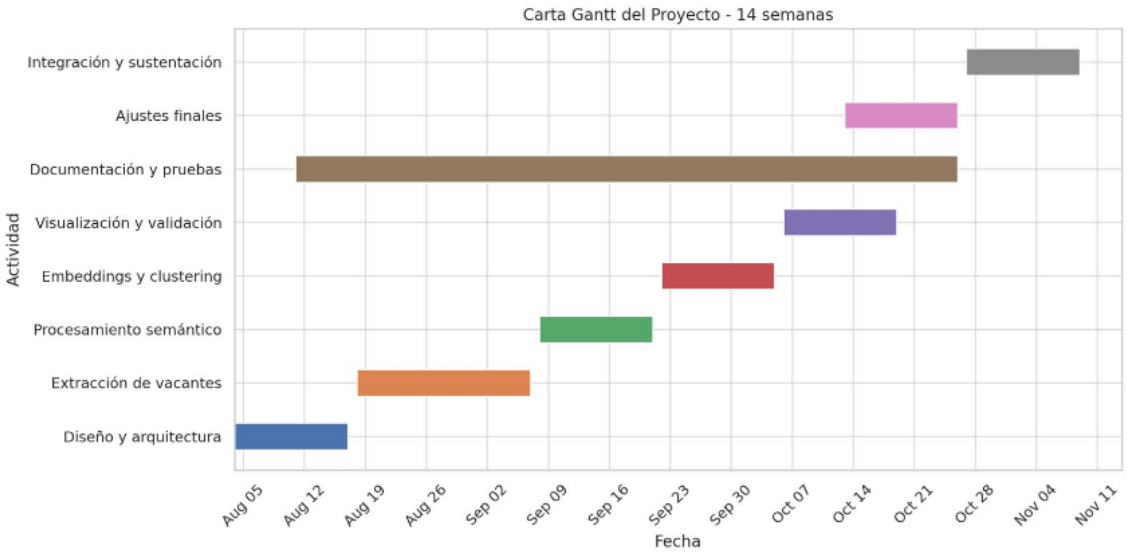


Figura 4.1: Carta Gantt del cronograma del proyecto (16 semanas)

### 4.3.3 Asignación de Recursos

Para cada fase principal del proyecto, se han identificado los recursos necesarios en términos de recursos humanos, software, hardware y documentación, con el fin de asegurar el cumplimiento eficiente de los objetivos. Los detalles completos se encuentran en la Tabla 16.

### 4.3.4 Asignación de Presupuesto y Justificación

Este proyecto no contempla flujo de dinero real ni remuneración alguna para los integrantes. Sin embargo, se presenta una estimación simbólica del costo técnico potencial, que refleja lo que implicaría económicamente replicar el esfuerzo si se ejecutara en un entorno profesional o institucional. Esta estimación es útil para evaluar necesidades técnicas, justificar decisiones y proyectar posibles inversiones en caso de escalamiento.

El presupuesto estimado se presenta en la Tabla 17, con un total aproximado de hasta $8.600.000 COP en valor técnico para réplica profesional del proyecto.

*Esta asignación tiene fines estimativos y no representa una solicitud ni gestión presupuestal formal. Ningún recurso económico será solicitado ni entregado a los integrantes.*

# Monitoreo y Control del Proyecto

## 5.1 Administración de Requerimientos

La administración de requerimientos en este proyecto sigue un enfoque pragmático y orientado a la adaptabilidad técnica, en el que se priorizan los requerimientos funcionales directamente vinculados con las fases metodológicas definidas y se permite cierto grado de flexibilidad en la implementación cuando surgen limitantes técnicas, cambios en herramientas disponibles o imprevistos identificados durante las retrospectivas semanales.

Dada la naturaleza académica del proyecto y su metodología híbrida basada en CRISP-DM y Scrum no estricto, los requerimientos serán gestionados de forma iterativa, sin pretender alcanzar niveles de rigidez formal propios de proyectos empresariales bajo contratos estrictos. En cambio, se promoverá la documentación continua de decisiones técnicas, la trazabilidad modular de los cambios, y la validación interna de que cada fase cumple con los objetivos esperados antes de avanzar a la siguiente.

El proceso general de gestión de requerimientos se presenta en el siguiente diagrama BPMN:

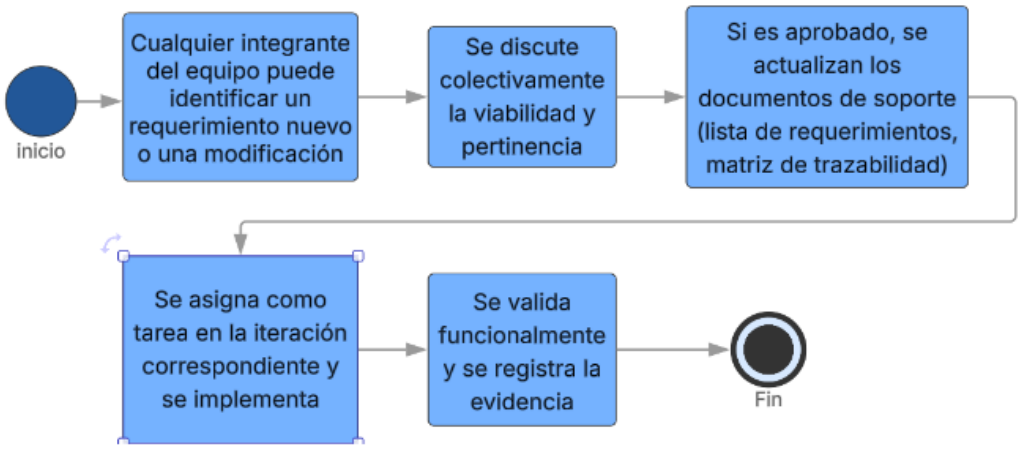


Figura 5.1: Proceso de Administración de Requerimientos (BPMN)

### 5.1.1 Proceso de Gestión de Cambios

Los cambios a los requerimientos podrán ser propuestos por cualquier miembro del equipo de desarrollo, el asesor académico del proyecto, o identificados mediante descubrimiento técnico emergente durante fases de ejecución cuando limitaciones no previstas o oportunidades de mejora se hacen evidentes. Todo cambio propuesto seguirá flujo estructurado de cinco etapas secuenciales que garantizan análisis riguroso de implicaciones y trazabilidad completa de decisiones.

La primera etapa corresponde a identificación formal del cambio, donde el integrante que detecta la necesidad documenta exhaustivamente el motivo técnico o metodológico que lo origina, el impacto esperado sobre funcionalidad del sistema o calidad de resultados, y las alternativas posibles de implementación con análisis comparativo preliminar de trade-offs. La segunda etapa ejecuta evaluación sistemática de impacto mediante discusión estructurada en reunión semanal del equipo, analizando si el cambio afecta cronograma global o de fases específicas, redistribución de carga de trabajo entre integrantes, introducción de nuevas dependencias técnicas entre módulos, o modificación de entregables comprometidos con el asesor.

La tercera etapa consiste en decisión formal por consenso del equipo técnico, donde el cambio se aprueba o rechaza mediante acuerdo explícito de ambos integrantes fundamentado en análisis de viabilidad técnica y alineación con objetivos del proyecto. En caso de desacuerdo persistente entre integrantes sobre viabilidad o prioridad del cambio, el asesor del proyecto Ing. Luis Gabriel Moreno Sandoval tendrá voz decisoria final para desbloquear la decisión. La cuarta etapa actualiza documentación del proyecto si el cambio es aprobado, registrando formalmente la modificación en acta semanal de reunión con justificación explícita, actualizando documento técnico del proyecto en sección de decisiones arquitectónicas, y ajustando planificación WBS afectada incluyendo redistribución de horas estimadas si aplica.

La quinta etapa final ejecuta comunicación formal y ajuste operativo del trabajo en curso, donde los integrantes afectados por el cambio aprobado ajustan implementación actual conforme a nueva especificación, actualizan código y documentación técnica correspondiente, y notifican formalmente el estado de implementación del cambio en la siguiente sesión de seguimiento quincenal con el asesor académico.

### 5.1.2 Trazabilidad de Requerimientos

Los requerimientos funcionales del observatorio están vinculados directamente a las siete fases del proyecto CRISP-DM adaptado y sus entregables técnicos asociados, garantizando que cada funcionalidad especificada pueda ser rastreada desde su origen conceptual hasta su validación empírica final. La trazabilidad se mantiene operativa mediante cuatro mecanismos complementarios de documentación y versionado que cubren diferentes niveles de granularidad del sistema.

El primer mecanismo consiste en tabla de requerimientos funcionales y fases incluida en el documento de especificación de requerimientos, donde cada requerimiento funcional está explícitamente

asociado a una o más fases metodológicas del WBS, facilitando identificación precisa del momento de su implementación técnica, validación funcional y prueba de aceptación. El segundo mecanismo emplea el repositorio de código versionado en GitHub, donde cada módulo del sistema tiene carpeta identificada con estructura jerárquica clara y commits descriptivos que referencian mediante tags o mensajes la fase técnica correspondiente, el requerimiento funcional implementado y el autor responsable de la implementación.

El tercer mecanismo mantiene actas de seguimiento semanal almacenadas en Google Drive compartido, constituyendo registro cronológico de avances donde se indica explícitamente qué requerimientos funcionales fueron implementados durante la semana, cuáles fueron probados exitosamente mediante validación funcional, y cuáles sufrieron modificaciones de especificación o priorización por decisiones del equipo o feedback del asesor. El cuarto mecanismo documenta en sección específica del documento técnico final una matriz de trazabilidad bidireccional que vincula cada requerimiento funcional con su fase de diseño arquitectónico inicial, etapa de implementación técnica, procedimientos de prueba ejecutados y resultados de validación empírica con métricas cuantitativas de aceptación.

### 5.1.3 Aprobación y Validación de Cambios

Todos los cambios propuestos a requerimientos funcionales o no funcionales del sistema deben ser validados formalmente por tres instancias complementarias de revisión que garantizan viabilidad técnica, coherencia académica y alineación con objetivos institucionales del trabajo de grado.

La primera instancia corresponde al equipo de desarrollo conformado por Nicolás Camacho y Alejandro Pinzón, quienes ejecutan validación técnica de viabilidad del cambio propuesto, analizando si es implementable con stack tecnológico actual, si requiere refactorización significativa de módulos ya desarrollados, si introduce deuda técnica aceptable o crítica, y si su implementación cabe dentro del tiempo restante del cronograma considerando carga de trabajo existente. La segunda instancia involucra al asesor del proyecto Ing. Luis Gabriel Moreno Sandoval, quien ejecuta validación académica del cambio propuesto, verificando coherencia con objetivos específicos del trabajo de grado, alineación con metodología CRISP-DM establecida, pertinencia para aportes esperados del proyecto y adecuación a estándares de calidad de trabajos de investigación en Ingeniería de Sistemas. La tercera instancia requiere aprobación formal del director del proyecto si el cambio implica modificación significativa del alcance definido inicialmente, ajuste de fechas de entregables comprometidos institucionalmente, o reasignación sustancial de recursos humanos o técnicos que afecte viabilidad global del cronograma.

La aprobación formal de cambios validados se materializa mediante firma digital del asesor en documento PDF del acta correspondiente o mediante confirmación escrita explícita en correo electrónico institucional archivado en carpeta compartida del proyecto, garantizando trazabilidad auditable de todas las decisiones de gestión de cambios ejecutadas durante el desarrollo.

## 5.2 Monitoreo y Control de Progreso

El monitoreo del proyecto se realizará de forma continua mediante reuniones semanales, indicadores de avance y mecanismos de reporte estructurado que permitan detectar desviaciones de forma temprana y aplicar correcciones con suficiente anticipación.

### 5.2.1 Métricas de Seguimiento

Se emplearán las siguientes métricas para evaluar el avance del proyecto:

| **Métrica** | **Descripción** | **Frecuencia de medición** | **Responsable** |
|---|---|---|---|
| Porcentaje de avance por fase | Proporción de actividades completadas en cada fase metodológica | Semanal | Líder técnico |
| Horas trabajadas acumuladas | Total de horas dedicadas por cada integrante | Semanal | Cada integrante |
| Número de requerimientos implementados | Cantidad de funcionalidades técnicas completadas y validadas | Quincenal | Líder técnico |
| Tasa de cumplimiento del cronograma | Relación entre fechas planificadas y fechas reales de finalización de fases | Al finalizar cada fase | Coordinador de proyecto |
| Cantidad de defectos encontrados en validación | Errores técnicos detectados en pruebas funcionales | Al finalizar cada fase | Encargado de pruebas |

Tabla 5.1: Métricas de Seguimiento del Proyecto

### 5.2.2 Actividades de Reporte

El reporte estructurado del avance del proyecto se realizará mediante cuatro mecanismos complementarios de comunicación que operan en diferentes frecuencias y niveles de detalle técnico, garantizando transparencia continua hacia el asesor académico y coordinación efectiva interna del equipo.

El primer mecanismo consiste en reuniones semanales de seguimiento ejecutadas cada lunes con duración de 1 hora, requiriendo presencia sincrónica de todo el equipo mediante videoconferencia o sesión presencial. La agenda estructurada incluye discusión del avance técnico de la semana anterior medido contra métricas del WBS, presentación de bloqueos técnicos encontrados con descripción de síntomas y hipótesis de causas, revisión cuantitativa de métricas de avance por fase y por integrante, y planificación detallada de tareas de la siguiente semana con asignación explícita de responsables y dependencias entre actividades.

El segundo mecanismo genera actas de reunión documentadas al finalizar cada sesión semanal,

incluyendo registro de asistencia con nombres y roles, temas técnicos y administrativos tratados, decisiones tomadas con justificación y votación si aplica, compromisos adquiridos por cada integrante con fechas límite específicas, y fecha confirmada de próxima reunión. Las actas se almacenan en carpeta compartida de Google Drive con nomenclatura estandarizada que incluye fecha ISO 8601 para facilitar búsqueda cronológica y auditoría posterior.

El tercer mecanismo establece reportes técnicos quincenales al asesor del proyecto, enviados cada dos semanas mediante correo electrónico institucional con documento PDF adjunto de 1 a 2 páginas estructurado en secciones estándar. El contenido incluye estado actual de cada fase metodológica con porcentaje de completitud estimado, problemas técnicos detectados durante el período con severidad clasificada, soluciones técnicas aplicadas o en evaluación con análisis de alternativas, y planes de trabajo para las próximas dos semanas con énfasis en hitos críticos del cronograma.

El cuarto mecanismo ejecuta reunión de revisión de fase al finalizar cada una de las siete fases metodológicas del CRISP-DM adaptado, consistiendo en sesión extendida con el asesor de hasta 2 horas de duración donde se presenta formalmente el entregable técnico correspondiente mediante demostración funcional en vivo, se valida calidad técnica y académica del resultado obtenido mediante inspección de código y análisis de métricas de evaluación, y se ajusta colaborativamente el plan de trabajo para fases subsiguientes si desviaciones detectadas o lecciones aprendidas así lo requieren.

### 5.2.3 Acciones Correctivas

Cuando se detecte desviación significativa en el cronograma planificado, degradación de recursos técnicos o humanos disponibles, o deterioro de calidad técnica de entregables contra criterios de aceptación establecidos, el equipo podrá aplicar cuatro tipos de acciones correctivas escaladas según severidad del problema identificado.

La primera acción correctiva consiste en reprogramación táctica de actividades del WBS, aplicable cuando una fase específica se retrasa más de una semana respecto a fechas comprometidas. El equipo evalúa reducir alcance técnico de otras fases menos críticas mediante priorización de requerimientos funcionales, redistribuir tareas entre integrantes según especialización y carga actual de trabajo, o ajustar cronograma global mediante extensión de fechas con aprobación formal del asesor académico y análisis de impacto en fecha de defensa final.

La segunda acción correctiva implementa refuerzo técnico colaborativo ante bloqueos técnicos que superan capacidad individual de resolución, donde Nicolás Camacho en su rol de líder técnico brindará acompañamiento adicional intensivo a los integrantes afectados, incluyendo sesiones de pair programming sincrónico de 2 a 4 horas para debugging conjunto de código problemático, revisión arquitectónica guiada de diseño de módulos con alto acoplamiento o baja cohesión, y transferencia acelerada de conocimiento especializado en componentes técnicos complejos del stack.

La tercera acción correctiva ejecuta simplificación técnica controlada cuando una técnica o algoritmo planificado resulta inviable por limitaciones computacionales de hardware disponible, insuficien-

cia de datos de entrenamiento o anotación, o complejidad de implementación excesiva que amenaza cronograma crítico. El equipo reemplazará la técnica original por alternativa más simple pero técnicamente válida y académicamente defendible, como migrar de fine-tuning de modelos transformer a prompting directo con LLMs preentrenados, sustituir clustering jerárquico por particionamiento k-means, o simplificar pipeline de limpieza de datos reduciendo etapas de normalización.

La cuarta acción correctiva aplica extensión controlada del cronograma como último recurso cuando calidad de entregables técnicos está severamente comprometida y amenaza viabilidad de defensa académica. El equipo podrá solicitar formalmente extensión de hasta 2 semanas adicionales, previa validación con el director del proyecto mediante reunión extraordinaria que presente evidencia de problemas críticos encontrados, ajuste formal del plan de trabajo con redistribución de actividades en tiempo extendido, y compromiso escrito de ambos integrantes de dedicación intensiva en período adicional.

Todas las acciones correctivas ejecutadas deberán quedar exhaustivamente documentadas en acta de reunión correspondiente, especificando problema técnico o administrativo detectado con métricas cuantitativas de desviación, decisión correctiva tomada con justificación de selección entre alternativas evaluadas, responsable designado de implementar la acción con accountability explícita, y plazo de ejecución con fecha límite verificable en siguiente sesión de seguimiento.

## 5.3 Cierre del Proyecto

El cierre del proyecto contempla un conjunto de actividades formales que garantizan la entrega completa de los resultados, la documentación adecuada de aprendizajes, la transferencia de conocimiento y la validación final por parte de los evaluadores designados.

### 5.3.1 Entrega del Producto

La entrega final del proyecto al asesor académico y a la universidad incluirá seis componentes técnicos y documentales complementarios que garantizan completitud funcional del sistema, reproducibilidad científica de resultados y transferencia efectiva de conocimiento generado durante el desarrollo.

El primer componente consiste en repositorio de código fuente funcional alojado en GitHub con estructura modular jerárquica que refleja arquitectura del sistema, código Python completamente comentado mediante docstrings en formato Google o NumPy, archivo README markdown con instrucciones detalladas de instalación y ejecución del sistema, archivo requirements.txt con dependencias especificando versiones exactas de bibliotecas utilizadas, y carpeta de ejemplos con scripts de demostración para ejecución del pipeline completo sobre datos sintéticos o muestra real reducida.

El segundo componente entrega dataset procesado exportado desde PostgreSQL mediante dump SQL comprimido conteniendo esquema completo de base de datos y datos de producción, incluyendo tablas de vacantes scrapeadas con metadata temporal y geográfica, habilidades extraídas por ambos

pipelines con scores de confianza, vectores de embeddings de 768 dimensiones en formato binario eficiente, asignaciones de clustering con parámetros HDBSCAN utilizados, y diccionario de datos explicativo en formato markdown documentando esquema relacional, tipos de datos, restricciones de integridad y semántica de cada columna.

El tercer componente produce documento técnico final del trabajo de grado en formato PDF profesional de mínimo 60 páginas siguiendo plantilla institucional de la Pontificia Universidad Javeriana o formato IEEE Computer Society, conteniendo introducción contextualizando problema de observación de demanda laboral, metodología CRISP-DM adaptada con justificación de decisiones arquitectónicas, descripción detallada de arquitectura del sistema con diagramas BPMN y de componentes, explicación técnica exhaustiva de cada una de las siete fases con pseudocódigo de algoritmos críticos, presentación de resultados obtenidos con métricas cuantitativas y análisis cualitativo de outputs, validación empírica mediante comparación con gold standard y benchmarks de estado del arte, conclusiones sintetizando aportes del proyecto y lecciones aprendidas, y recomendaciones fundamentadas para trabajo futuro y escalamiento del observatorio.

El cuarto componente compila visualizaciones generadas y notebooks analíticos en carpeta organizada jerárquicamente, incluyendo gráficos de alta resolución en formatos PNG y PDF vectorial representando histogramas de distribución de habilidades más frecuentes, proyecciones UMAP de espacios de embeddings coloreadas por clústeres identificados, mapas de calor geográficos comparando perfiles tecnológicos por país, nubes de palabras ponderadas por frecuencia de términos técnicos, y notebooks Jupyter completamente ejecutables con análisis exploratorios de datos, experimentos de parametrización de clustering, y explicaciones técnicas didácticas paso a paso de decisiones metodológicas.

El quinto componente provee manual de usuario técnico conciso de 5 a 10 páginas dirigido a desarrolladores o investigadores que deseen replicar el sistema en infraestructura diferente, especificando requisitos mínimos de hardware incluyendo CPU, RAM, almacenamiento y GPU opcional, instrucciones paso a paso de instalación de dependencias mediante pip o conda en entorno virtual Python aislado, configuración de variables de entorno para credenciales de base de datos y paths de modelos preentrenados descargados, y secuencia de comandos para ejecutar cada fase del pipeline con parámetros por defecto validados y troubleshooting de errores comunes encontrados durante desarrollo.

El sexto componente documenta evidencias empíricas de validación del sistema mediante carpeta de artefactos técnicos, conteniendo logs de ejecución representativos de scraping exitoso y manejo de errores de red, métricas cuantitativas de evaluación tabuladas incluyendo precisión y recall del NER, F1-score Post-ESCO de ambos pipelines, coherencia interna de clústeres mediante Silhouette Score y DBCV, ejemplos concretos de habilidades enriquecidas semánticamente por Pipeline B con justificaciones textuales del LLM, y capturas de pantalla de alta resolución de visualizaciones funcionales ejecutadas sobre corpus completo de 30,660 ofertas laborales.

### 5.3.2 Actividades de Cierre

Al finalizar el desarrollo técnico del proyecto e integración de todos los componentes del observatorio, se ejecutarán seis actividades formales de cierre que garantizan completitud de entregables, documentación de lecciones aprendidas y preparación adecuada para defensa académica ante jurados evaluadores.

La primera actividad consiste en revisión final exhaustiva de entregables mediante checklist estructurada, donde el equipo verifica sistemáticamente que todos los componentes técnicos y documentales estén completos según especificación del plan de proyecto, funcionales mediante ejecución end-to-end del pipeline sobre datos reales sin errores críticos, y correctamente documentados con comentarios de código, README actualizado y manual de usuario validado mediante prueba de instalación en máquina limpia.

La segunda actividad ejecuta sesión de retrospectiva final del equipo con duración de 2 horas, donde ambos integrantes reflexionan colaborativamente sobre aprendizajes técnicos adquiridos en cada fase del CRISP-DM, dificultades técnicas y organizacionales enfrentadas con análisis de causas raíz, decisiones arquitectónicas acertadas que facilitaron desarrollo ágil y modular, y mejoras metodológicas aplicables a futuros proyectos académicos o profesionales similares. Los insights se documentan formalmente en acta de lecciones aprendidas archivada en repositorio del proyecto para consulta posterior.

La tercera actividad presenta sistema completo al asesor académico mediante sesión formal de demostración técnica de 2 horas, donde se expone funcionamiento end-to-end del observatorio ejecutando pipeline completo en vivo sobre muestra representativa de datos, se explican decisiones técnicas clave como selección de Gemma 3 4B sobre alternativas evaluadas y diseño de arquitectura híbrida de pipelines, y se responden preguntas técnicas y metodológicas del asesor sobre implementación de componentes, validación de resultados y alineación con objetivos académicos del trabajo de grado.

La cuarta actividad prepara defensa pública del trabajo de grado mediante elaboración colaborativa de presentación profesional de 20 a 30 minutos en formato PowerPoint o Beamer LaTeX, enfocada estratégicamente en comunicación clara de resultados técnicos obtenidos con métricas cuantitativas destacadas, procedimientos de validación empírica ejecutados sobre gold standard anotado, aportes metodológicos del proyecto como adaptación de CRISP-DM a contexto latinoamericano, y demostración visual impactante del sistema funcionando mediante screenshots y videos cortos de scraping, clustering y visualizaciones generadas.

La quinta actividad archiva formalmente el proyecto completo mediante subida de todos los entregables finales a repositorio institucional de la Pontificia Universidad Javeriana si protocolo aplica, y entrega de copia completa a la universidad en formatos solicitados por programa académico incluyendo documento PDF del trabajo de grado firmado digitalmente, código fuente comprimido en archivo ZIP con estructura de carpetas preservada, datasets procesados exportados como dumps SQL, y licencia de uso académico firmada autorizando uso interno con fines educativos y de investigación.

La sexta actividad opcional ejecuta liberación pública del código fuente del observatorio bajo licencia open source permisiva como MIT o Apache 2.0, condicionada a decisión consensuada del equipo considerando beneficio para comunidad técnica y académica, y autorización formal del asesor verificando que no existan restricciones institucionales o de propiedad intelectual que impidan publicación abierta. La liberación incluye documentación clara de instalación, datasets sintéticos de ejemplo para testing, y sección de contribuciones invitando a comunidad a reportar issues y proponer mejoras mediante pull requests de GitHub.

### 5.3.3 Criterios de Aceptación Final

El proyecto del Observatorio de Demanda Laboral en Tecnología para Latinoamérica será formalmente considerado completo y aceptable para defensa académica ante jurados cuando satisfaga cinco criterios de completitud técnica, funcionalidad sistémica, documentación exhaustiva, validación académica y cumplimiento cronológico.

El primer criterio requiere que todos los requerimientos funcionales prioritarios del sistema identificados en documento de especificación hayan sido implementados técnicamente en código Python funcional y validados mediante pruebas de aceptación ejecutadas sobre datos reales, confirmando que módulos de scraping, extracción dual mediante Pipeline A y B, generación de embeddings, clustering HDBSCAN y visualizaciones macro operan correctamente según especificaciones técnicas establecidas.

El segundo criterio establece que el sistema debe ser capaz de ejecutar pipeline completo de siete fases desde scraping inicial de portales hasta generación de visualizaciones finales sin errores críticos que detengan ejecución, aceptando únicamente warnings no fatales o errores manejados gracefully mediante try-except con logging apropiado, y completando procesamiento de corpus completo de 30,660 ofertas laborales en tiempo razonable sin memory leaks o deadlocks.

El tercer criterio exige que documentación técnica del proyecto esté completa cubriendo todas las fases metodológicas con descripción detallada de implementación, coherente manteniendo consistencia terminológica y de notación entre secciones, y suficientemente clara para permitir replicación exitosa del sistema por terceros con conocimientos técnicos de nivel medio en Python, NLP y bases de datos relacionales, validado mediante prueba empírica de instalación por persona externa al equipo.

El cuarto criterio condiciona aceptación final a aprobación formal del asesor del proyecto Ing. Luis Gabriel Moreno Sandoval sobre calidad técnica del sistema implementado evaluada mediante inspección de código y demostración funcional, y calidad académica del documento de trabajo de grado evaluada mediante revisión de profundidad metodológica, rigurosidad de análisis de resultados y alineación con estándares de investigación en Ingeniería de Sistemas de la Pontificia Universidad Javeriana.

El quinto criterio verifica que se hayan cumplido entregas programadas de entregables según cronograma ajustado del proyecto considerando cambios aprobados formalmente durante desarrollo, o

alternativamente que extensiones de fechas aplicadas hayan sido justificadas formalmente mediante actas de reunión documentando causas de retrasos, aprobación escrita del asesor aceptando nueva calendarización, y evidencia de mitigación de riesgos identificados para evitar futuros incumplimientos.

Una vez satisfechos estos cinco criterios de aceptación mediante verificación documentada y aprobación formal del asesor académico, el proyecto podrá ser sometido a evaluación final por parte de los jurados designados por el Departamento de Sistemas en sesión de defensa pública programada según calendario académico institucional.

# Procesos de Soporte

## 6.1 Gestión de la Configuración

La gestión de la configuración del proyecto tiene como propósito mantener la integridad, trazabilidad y control de versiones de todos los artefactos generados durante el desarrollo, incluyendo código fuente, datasets, documentación técnica, modelos entrenados, scripts de procesamiento y archivos de configuración. Dado el carácter académico del proyecto y su naturaleza modular, se adoptará un enfoque pragmático basado en Git, GitHub y herramientas colaborativas de documentación, sin pretender alcanzar niveles de formalidad propios de entornos empresariales regulados.

### 6.1.1 Elementos de Configuración

Los elementos que estarán bajo control de configuración incluyen:

| Elemento de configuración | Descripción | Herramienta de gestión | Responsable |
|---|---|---|---|
| Código fuente del sistema | Scripts de scraping, NER, LLMs, clustering y visualización en Python | GitHub | Nicolás |
| Configuraciones de entorno | Archivos .env, config.json, requirements.txt, docker-compose.yml | GitHub | Nicolás |
| Bases de datos y esquemas | Dump SQL de PostgreSQL con estructura de tablas y datos procesados | GitHub + Google Drive | Nicolás |
| Documentación técnica del proyecto | Documento SPMP, SRS, memoria técnica, manuales | Google Docs + Overleaf | Alejandro |
| Notebooks de análisis | Jupyter notebooks con pruebas exploratorias, validaciones y visualizaciones | GitHub | Nicolás |
| Datasets intermedios | Archivos CSV, JSON o pickle con datos procesados por fase | Google Drive | Nicolás |
| Modelos y embeddings | Archivos de modelos descargados o ajustados, vectores precomputados | Google Drive | Nicolás |
| Actas de reunión y seguimiento | Registros de reuniones semanales y decisiones de equipo | Google Docs | Alejandro |

Tabla 6.1: Elementos de Configuración del Proyecto

### 6.1.2 Proceso de Control de Versiones

El control de versiones del código fuente y de los artefactos técnicos del proyecto se realizará mediante Git como sistema de control distribuido y GitHub como plataforma de hospedaje y colaboración remota, siguiendo cinco prácticas fundamentales de ingeniería de software que garantizan trazabilidad histórica, integración controlada y recuperabilidad ante errores.

La primera práctica establece rama principal denominada main como línea base estable del sistema, conteniendo únicamente código probado funcionalmente y validado técnicamente al cierre de cada fase metodológica del CRISP-DM. Los commits directos a esta rama están prohibidos, requiriendo obligatoriamente revisión previa mediante pull request aprobado por líder técnico Nicolás Camacho. La segunda práctica crea ramas de desarrollo dedicadas por fase metodológica siguiendo nomenclatura estandarizada dev-fase-X donde X identifica número de fase, permitiendo desarrollo paralelo aislado de funcionalidades correspondientes sin afectar estabilidad de main. Al completar validación técnica satisfactoria de fase, rama temporal se fusiona a main mediante pull request con revisión de código y resolución de conflictos de merge.

La tercera práctica exige commits descriptivos con mensajes claros estructurados que indiquen qué

cambio técnico se realizó describiendo alcance específico de modificación, por qué razón se ejecutó justificando decisión arquitectónica o corrección de bug, y en qué fase del proyecto se efectuó para facilitar trazabilidad cronológica. El formato estandarizado sugerido sigue convención de prefijo por fase: [FASE-X] seguido de descripción concisa del cambio en modo imperativo presente. La cuarta práctica implementa versionado semántico para entregas formales de entregables mayores, etiquetando commits significativos con tags de versión siguiendo convención v0.1 para prototipos iniciales, v0.2 para iteraciones intermedias, hasta v1.0 para release académico final, facilitando identificación y recuperación de estados históricos específicos del sistema.

La quinta práctica establece sincronización diaria obligatoria donde los integrantes deben ejecutar git push de avances locales al menos una vez al día laboral al finalizar sesión de trabajo, garantizando que repositorio remoto GitHub refleje estado actual de desarrollo y minimizando riesgo de conflictos masivos de integración acumulados. Esta práctica facilita colaboración efectiva permitiendo revisión continua cruzada de código entre integrantes y backup automático diario de trabajo en progreso.

### 6.1.3 Gestión de Cambios en la Configuración

Cualquier cambio propuesto en la configuración técnica del sistema incluyendo modificación de estructura de esquema relacional de base de datos PostgreSQL, reemplazo o actualización de versión de bibliotecas clave del stack Python, o ajuste significativo de arquitectura de scrapers web que afecte modularidad o extensibilidad, deberá seguir procedimiento formal estructurado de cinco etapas secuenciales que garantizan análisis riguroso de impacto y trazabilidad completa de decisiones técnicas.

La primera etapa requiere que el integrante que identifica necesidad del cambio documente exhaustivamente razón técnica fundamentada que motiva modificación propuesta con evidencia empírica de problema actual o limitación detectada, impacto esperado sobre funcionalidad del sistema cuantificando mejora en métricas de desempeño o calidad esperadas, y alternativas técnicas evaluadas mediante análisis comparativo de trade-offs entre opciones consideradas.

La segunda etapa ejecuta discusión estructurada en reunión semanal del equipo donde se debate colaborativamente si el cambio propuesto es técnicamente necesario para cumplir objetivos funcionales o no funcionales del proyecto, viable de implementar dentro de restricciones de tiempo y recursos computacionales disponibles, y académicamente justificado alineándose con metodología CRISP-DM establecida y aportes esperados del trabajo de grado.

La tercera etapa implementa el cambio aprobado mediante desarrollo en rama específica de Git denominada feature/nombre-cambio aislada de main, ejecuta pruebas funcionales exhaustivas localmente en entorno de desarrollo de integrante responsable validando que modificación no introduce regresiones críticas en módulos dependientes, y documenta técnicamente el cambio en archivo README principal del repositorio o mediante comentarios explicativos inline en código fuente modificado describiendo justificación y procedimiento de uso.

La cuarta etapa abre pull request formal en GitHub para revisión técnica obligatoria por parte

de Nicolás Camacho como líder técnico antes de autorizar fusión a rama principal main, incluyendo descripción detallada del cambio, screenshots o logs de pruebas ejecutadas exitosamente, y análisis de impacto en otros módulos del sistema para facilitar evaluación informada de revisor.

La quinta etapa final actualiza archivo de configuración correspondiente del sistema como config.json o variables de entorno en archivo .env con nuevos parámetros o versiones de dependencias, registra formalmente el cambio aprobado en acta semanal de reunión con justificación técnica resumida, y comunica modificación al equipo completo mediante mensaje en canal de comunicación del proyecto explicando qué cambió y cómo afecta workflow de desarrollo de otros integrantes.

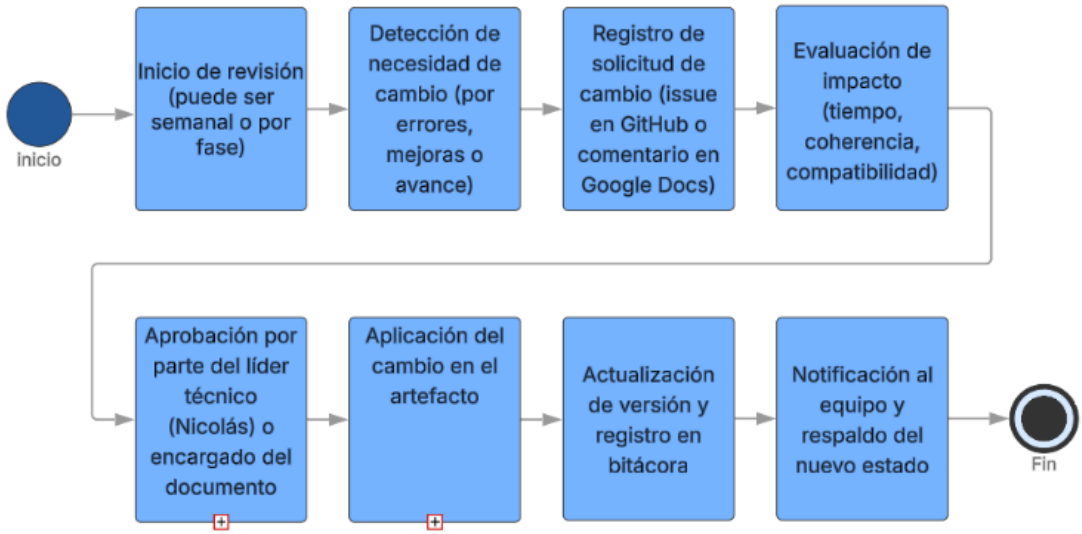


Figura 6.1: Proceso de Gestión de Cambios en la Configuración (BPMN)

### 6.1.4 Backup y Recuperación

Para garantizar disponibilidad continua de los artefactos del proyecto ante pérdidas accidentales por fallas de hardware, errores humanos de borrado o corrupción de archivos, se implementarán cuatro medidas complementarias de respaldo y recuperación que operan en diferentes niveles de granularidad temporal y tipo de contenido.

La primera medida establece GitHub como repositorio central de versionado distribuido donde todo el código fuente Python, scripts de análisis, documentación técnica en formato Markdown y LaTeX, y archivos de configuración del sistema se alojarán con historial completo de commits. Esta plataforma ofrece redundancia automática geográfica mediante replicación en múltiples datacenters, control de versiones granular permitiendo recuperar cualquier estado histórico del código mediante git checkout o revert, y mecanismo de recuperación ante errores mediante creación de branches de rollback o cherry-pick de commits específicos.

La segunda medida utiliza Google Drive compartido institucional como almacenamiento de archivos de gran tamaño que exceden límites prácticos de Git, incluyendo datasets intermedios en formato CSV o pickle superiores a 100 MB, modelos preentrenados descargados de HuggingFace con pesos completos, documentos de trabajo en edición activa mediante Google Docs con sincronización en tiempo real, y archivos binarios de visualizaciones en alta resolución. La carpeta compartida tiene acceso restringido exclusivamente al equipo de desarrollo mediante autenticación institucional, versionado automático de Google Drive que retiene versiones de archivos modificados durante 30 días, y cuota de almacenamiento asignada de 15 GB por integrante.

La tercera medida exige backups semanales locales obligatorios donde cada integrante del equipo mantiene copia local completa del repositorio GitHub actualizada, sincronizada semanalmente mediante git pull origin main para incorporar cambios remotos, almacenada en disco duro local con respaldo adicional en disco externo o partición secundaria, y verificada mensualmente mediante ejecución de tests de integridad que confirmen que repositorio local puede ejecutar pipeline completo sin dependencias externas.

La cuarta medida implementa exportaciones SQL periódicas automatizadas de base de datos PostgreSQL ejecutadas al finalizar cada una de las siete fases técnicas del proyecto, generando archivo dump completo mediante comando $pg_dump con compresión gzip, nomenclatura estandarizada incluyendo fecha ISO$

En caso de materialización de pérdida de datos por cualquier causa identificada, el procedimiento de recuperación seguirá protocolo escalonado según severidad. Para pérdida de código fuente o documentación técnica versionada, se recurrirá al historial completo de GitHub mediante git log para identificar último commit válido y git reset para restaurar estado funcional previo al error. Para pérdida de datasets o modelos grandes, se descargarán copias de respaldo desde Google Drive compartido verificando integridad mediante checksums MD5 si disponibles. Si un integrante pierde completamente su copia local de trabajo por falla catastrófica de hardware, podrá ejecutar recuperación completa mediante git clone del repositorio completo desde GitHub para obtener código fuente, descarga de carpeta compartida de Google Drive para recuperar datasets y modelos grandes, y restauración de dump SQL más reciente de PostgreSQL para reconstruir base de datos operativa, completando proceso de recuperación en tiempo estimado de 2 a 4 horas dependiendo de velocidad de conexión a internet.

## 6.2 Aseguramiento de Calidad

El aseguramiento de calidad tiene como objetivo garantizar que los entregables del proyecto cumplan con los estándares técnicos esperados, funcionen correctamente en distintos escenarios de uso, y estén adecuadamente documentados para facilitar su comprensión, validación y réplica por terceros. Dado el contexto académico, se priorizará la validación funcional, la coherencia metodológica y la transparencia técnica por encima de métricas formales de calidad de software empresarial.

### 6.2.1 Estándares de Calidad Aplicados

El desarrollo del observatorio seguirá cinco estándares complementarios de calidad técnica que garantizan legibilidad del código, mantenibilidad del sistema, reproducibilidad científica y coherencia con mejores prácticas de ingeniería de software académico y profesional.

El primer estándar adopta convenciones de estilo PEP 8 como guía oficial para todo el código Python del proyecto, incluyendo uso obligatorio de nombres de variables y funciones descriptivos en snake$_c asequecomuniquenclaramentepropósitoycontenido, separaciónlógicaclaradefuncionesmediantelíneasen$

El segundo estándar exige modularidad y reutilización mediante diseño arquitectónico donde cada una de las siete fases del pipeline CRISP-DM será implementada como módulo Python independiente con responsabilidad única bien definida, entradas explícitamente tipadas mediante type hints de Python 3.10+ especificando tipos de datos esperados, salidas claramente documentadas retornando estructuras de datos consistentes como DataFrames de pandas o diccionarios con esquema predefinido, y acoplamiento mínimo entre módulos comunicándose exclusivamente mediante interfaces públicas sin dependencias circulares, facilitando depuración aislada de componentes individuales, testing unitario independiente por módulo y reutilización futura de componentes en proyectos relacionados.

El tercer estándar requiere documentación interna exhaustiva del código donde funciones con complejidad ciclomática superior a 5 o que implementen lógica de negocio no trivial deberán incluir docstrings explicativas detalladas en español siguiendo formato Google o NumPy, indicando propósito claro de la función describiendo qué problema resuelve y por qué existe, parámetros de entrada con tipos esperados y restricciones de valores válidos, salida esperada especificando tipo de dato retornado y significado semántico, excepciones que pueden lanzarse documentando condiciones que las activan, y ejemplo de uso mínimo mostrando llamada típica con valores reales y resultado esperado para facilitar comprensión rápida de API.

El cuarto estándar implementa manejo robusto de errores mediante estrategia defensiva que incluye validaciones básicas de entrada al inicio de funciones críticas verificando tipos de datos, rangos numéricos válidos y existencia de archivos requeridos antes de procesamiento, manejo explícito de excepciones mediante bloques try-except con captura específica de tipos de error esperados como FileNotFoundError, JSONDecodeError o ConnectionError en lugar de excepciones genéricas, mensajes de error claros y accionables que informen al usuario qué falló, por qué ocurrió el error y qué acción correctiva puede tomar, y logging estructurado de eventos críticos mediante biblioteca logging de Python con niveles apropiados DEBUG para trazas detalladas, INFO para hitos de progreso, WARNING para situaciones anómalas recuperables y ERROR para fallos críticos, facilitando diagnóstico post-mortem de problemas en producción.

El quinto estándar garantiza reproducibilidad técnica completa mediante documentación exhaustiva donde todos los procesos del pipeline estarán descritos con suficiente detalle granular en README técnico, notebooks Jupyter comentados y documento de metodología del trabajo de grado, permitiendo que investigador externo con conocimientos técnicos de nivel medio en Python, procesamiento de

lenguaje natural y bases de datos relacionales pueda replicar sistema completo desde cero, obteniendo resultados estadísticamente equivalentes a los reportados considerando variabilidad estocástica inherente a modelos de lenguaje y algoritmos de clustering no deterministas, validado mediante prueba empírica de instalación ejecutada por persona externa al equipo de desarrollo en máquina limpia sin configuración previa.

### 6.2.2 Actividades de Verificación y Validación

Las actividades de verificación (¿construimos el sistema correctamente?) y validación (¿construimos el sistema correcto?) se llevarán a cabo de forma iterativa en cada fase del proyecto:

| **Fase** | **Actividad de verificación** | **Actividad de validación** | **Responsable** |
|---|---|---|---|
| Scraping | Revisar que las vacantes extraídas tengan estructura completa y datos válidos | Comparar muestra manual con resultados del scraper para verificar precisión | Nicolás |
| NER y regex | Ejecutar el script sobre un conjunto de prueba y verificar que extrae habilidades sin errores de sintaxis | Revisar manualmente 50 vacantes y evaluar si las habilidades extraídas son correctas y relevantes | Alejandro |
| LLMs | Probar distintos prompts y verificar que el formato de salida es consistente | Evaluar cualitativamente si las habilidades enriquecidas tienen sentido técnico y semántico | Nicolás |
| Embeddings y clustering | Verificar que las dimensiones de los vectores son correctas y que HDBSCAN no arroja errores | Inspeccionar visualmente los clusters generados y verificar que agrupan habilidades coherentes | Nicolás |
| Visualización | Comprobar que los gráficos se generan sin errores y se exportan correctamente | Revisar con el asesor si las visualizaciones comunican información relevante de forma clara | Alejandro |

Tabla 6.2: Actividades de Verificación y Validación por Fase

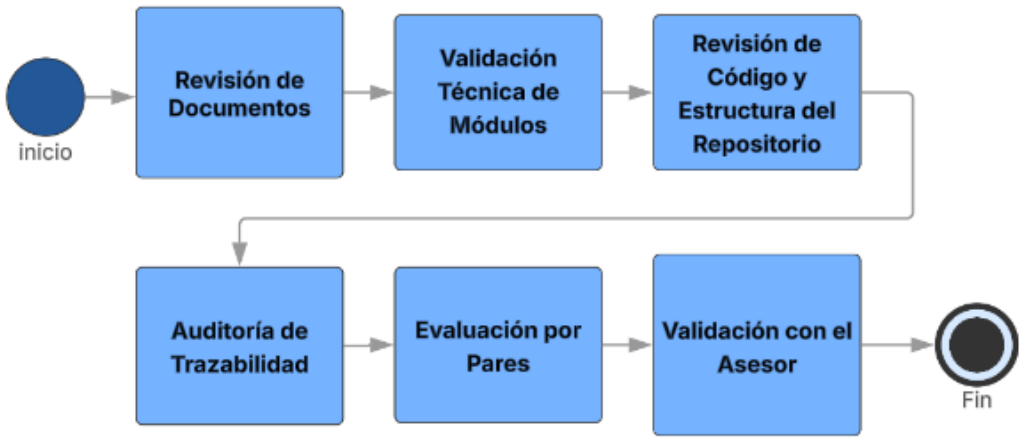


Figura 6.2: Proceso de Control de Calidad del Proyecto (BPMN)

### 6.2.3 Revisión Técnica de Entregables

Al finalizar cada una de las siete fases metodológicas del pipeline CRISP-DM, se ejecutará revisión técnica formal estructurada del entregable correspondiente siguiendo procedimiento de cinco etapas secuenciales que garantizan calidad funcional, coherencia arquitectónica y alineación con objetivos académicos del trabajo de grado.

La primera etapa consiste en presentación formal del entregable donde el integrante responsable de la fase expone módulo desarrollado al equipo completo durante reunión semanal de seguimiento con duración estimada de 30 a 45 minutos, explicando arquitectura e implementación técnica del código con énfasis en decisiones de diseño no triviales, decisiones clave tomadas durante desarrollo incluyendo alternativas evaluadas y criterios de selección aplicados, y resultados técnicos obtenidos mediante ejecución sobre datos reales con análisis de métricas de desempeño, calidad de outputs y tiempo de procesamiento observado.

La segunda etapa ejecuta revisión colaborativa de código donde los demás integrantes del equipo clonan rama de desarrollo del módulo en sus entornos locales, revisan exhaustivamente código fuente Python mediante lectura crítica de implementación buscando bugs potenciales, code smells o violaciones de estándares PEP 8, prueban módulo localmente ejecutando pipeline completo sobre muestra representativa de datos para verificar reproducibilidad de resultados reportados, y formulan preguntas técnicas fundamentadas sobre decisiones de implementación o identifican posibles mejoras de eficiencia, legibilidad o robustez ante casos extremos.

La tercera etapa redacta checklist de verificación formal con criterios mínimos de aceptación específicos al tipo de entregable, incluyendo preguntas técnicas binarias como si el código ejecuta sin errores críticos que detengan ejecución en condiciones normales de entrada, si los datos de salida

generados tienen estructura esperada conforme a especificación de interfaz del módulo validada mediante inspección de schemas, si está adecuadamente documentado el proceso mediante docstrings de funciones principales y comentarios explicativos inline en secciones complejas, si manejo de errores cubre casos extremos identificados mediante análisis de riesgos, y si desempeño de ejecución es razonable completando procesamiento de corpus completo en tiempo compatible con cronograma del proyecto.

La cuarta etapa toma decisión formal de aceptación donde si el entregable cumple todos los criterios obligatorios del checklist se aprueba formalmente mediante consenso del equipo, se fusiona a rama principal main mediante pull request con squash de commits para mantener historial limpio, y se etiqueta con tag de versión semántica correspondiente a fase completada; alternativamente si el entregable no cumple criterios críticos se registra lista detallada de correcciones pendientes priorizadas por severidad en documento de issues de GitHub, se establece plazo concreto de 3 a 5 días para aplicar correcciones identificadas según complejidad de cambios requeridos, y se programa sesión de re-revisión para validar implementación de mejoras solicitadas.

La quinta etapa final involucra validación académica donde el asesor del proyecto Ing. Luis Gabriel Moreno Sandoval revisa el entregable técnico aprobado internamente durante reunión quincenal de seguimiento con duración de 1 hora, valida que implementación esté técnicamente alineada con objetivos específicos del trabajo de grado y metodología CRISP-DM establecida, verifica que resultados obtenidos sean académicamente defendibles mediante comparación con benchmarks de literatura o análisis de coherencia semántica cualitativa, y provee retroalimentación formal escrita en acta de reunión con aprobación explícita o solicitudes de refinamiento adicional antes de proceder a siguiente fase del proyecto.

### 6.2.4 Métricas de Calidad

La evaluación cuantitativa y cualitativa de calidad del sistema desarrollado se fundamentará en cinco métricas complementarias que cubren dimensiones de completitud funcional, exactitud de outputs, interpretabilidad de resultados, complejidad del código y legibilidad para terceros.

La primera métrica mide cobertura de requisitos funcionales calculada como porcentaje de requisitos funcionales prioritarios del sistema especificados en documento SRS que han sido completamente implementados en código Python funcional y validados mediante pruebas de aceptación, respecto al total de requisitos funcionales definidos inicialmente, estableciendo umbral mínimo de aceptación de 90 por ciento para requisitos críticos del pipeline y 70 por ciento para requisitos opcionales de mejoras futuras.

La segunda métrica cuantifica tasa de errores en validación manual determinada como proporción de habilidades técnicas extraídas por Pipeline A mediante NER y regex o enriquecidas semánticamente por Pipeline B mediante LLM que son incorrectas por no corresponder a habilidades técnicas reales, irrelevantes por ser demasiado genéricas o fuera de dominio tecnológico, o mal normalizadas

a taxonomía ESCO sin correspondencia semántica adecuada, evaluada mediante inspección manual exhaustiva sobre muestra estadísticamente representativa de 100 registros seleccionados aleatoriamente del corpus completo, estableciendo umbral de calidad aceptable inferior a 15 por ciento de errores para Pipeline A y 10 por ciento para Pipeline B considerando su mayor sofisticación.

La tercera métrica evalúa coherencia semántica de clusters generados mediante análisis cualitativo experto de si los grupos de habilidades identificados por HDBSCAN tienen sentido técnico y semántico coherente agrupando tecnologías relacionadas por dominio de aplicación, stack tecnológico compartido o nivel de abstracción similar, mediante inspección visual de proyecciones UMAP bidimensionales con etiquetado de clústeres principales, revisión de listas de top 10 habilidades más frecuentes por cluster, y discusión estructurada con el asesor académico sobre interpretabilidad de taxonomía emergente identificada, documentando ejemplos de clusters exitosos y outliers problemáticos para refinamiento iterativo de hiperparámetros.

La cuarta métrica monitorea complejidad ciclomática moderada del código mediante análisis estático con herramientas como radon o flake8 verificando que funciones individuales del sistema mantengan complejidad ciclomática inferior a 10 para favorecer testabilidad y comprensión, evitando bloques de código excesivamente largos superiores a 100 líneas por función que indiquen falta de descomposición modular, o anidamiento excesivo de condicionales y loops superiores a 4 niveles de profundidad que dificulten seguimiento de flujo de control, estableciendo como objetivo que 80 por ciento de funciones tengan complejidad inferior a 5 indicando diseño simple y directo.

La quinta métrica evalúa legibilidad del código mediante análisis subjetivo de si el código fuente Python es comprensible y mantenible para desarrollador externo con experiencia intermedia en el lenguaje pero sin conocimiento previo del dominio específico del observatorio, verificada empíricamente mediante revisión cruzada entre integrantes del equipo donde cada desarrollador lee e interpreta código escrito por el otro sin consultar documentación externa, midiendo tiempo necesario para comprender propósito de módulo y localizar componente específico, y documentando sugerencias de mejora de nombres de variables, estructura de funciones o comentarios explicativos para incrementar claridad y reducir curva de aprendizaje de futuros mantenedores del sistema.

## 6.3 Gestión de Riesgos

La gestión de riesgos del proyecto tiene como objetivo identificar de forma anticipada las amenazas potenciales que puedan afectar el cronograma, la calidad técnica, los recursos disponibles o el cumplimiento de los objetivos, y definir estrategias de mitigación y contingencia para minimizar su impacto en caso de materializarse.

### 6.3.1 Identificación de Riesgos

A continuación, se presentan los principales riesgos identificados para el proyecto, clasificados por categoría:

| ID | Descripción del riesgo | Prob. | Impacto | Categoría |
|---|---|---|---|---|
| R01 | Cambios en la estructura HTML de portales de empleo que rompan el scraper | Media | Alto | Técnico |
| R02 | Bloqueo o limitación de acceso por parte de los portales web (anti-scraping) | Media | Alto | Técnico |
| R03 | Rendimiento insuficiente del modelo NER en español para el dominio laboral | Media | Medio | Técnico |
| R04 | Falta de datos suficientes o de calidad en portales de algunos países | Baja | Alto | Técnico |
| R05 | Complejidad técnica excesiva en implementación de clustering semántico | Media | Medio | Técnico |
| R06 | Sobrecarga académica de los integrantes afectando disponibilidad semanal | Alta | Medio | Organizacional |
| R07 | Retrasos acumulados que comprometan el cronograma general del proyecto | Media | Alto | Organizacional |
| R08 | Falta de coordinación o conflictos internos en el equipo | Baja | Medio | Organizacional |
| R09 | Fallas técnicas en equipos personales (hardware, conectividad, software) | Baja | Medio | Infraestructura |
| R10 | Pérdida de datos por falta de backups adecuados | Baja | Alto | Infraestructura |

Tabla 6.3: Identificación y Clasificación de Riesgos

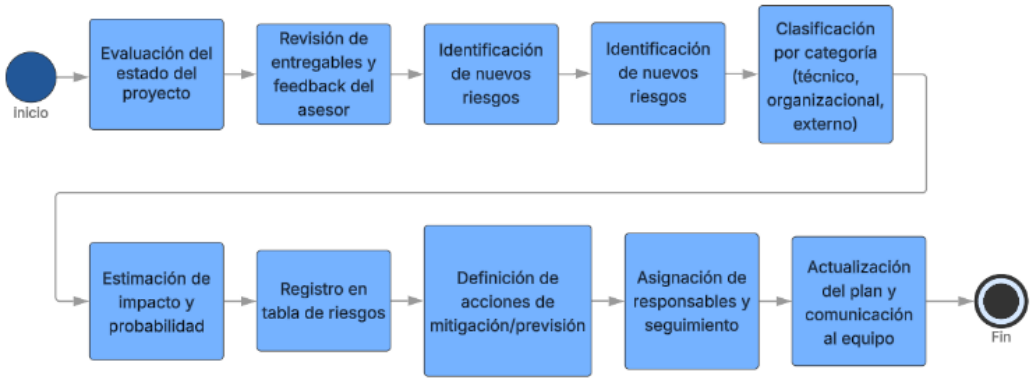


Figura 6.3: Proceso de Identificación y Gestión de Riesgos (BPMN)

### 6.3.2 Análisis de Riesgos

Cada riesgo ha sido evaluado en términos de probabilidad de ocurrencia (Baja, Media, Alta) e impacto en el proyecto (Bajo, Medio, Alto), permitiendo priorizarlos según su nivel de criticidad.

El análisis cuantitativo de riesgos identifica cuatro amenazas de mayor criticidad que combinan probabilidad media-alta con impacto alto sobre el proyecto, clasificadas en dos categorías principales. La primera categoría corresponde a riesgos técnicos de scraping, específicamente R01 relacionado con cambios estructurales en HTML de portales de empleo que rompan funcionamiento de scrapers implementados requiriendo mantenimiento correctivo urgente, y R02 asociado a bloqueo o limitación de acceso mediante técnicas anti-scraping como CAPTCHAs, rate limiting o blacklisting de IP que restrinjan capacidad de recolección masiva de datos. Estos riesgos técnicos afectarían directamente disponibilidad del corpus de vacantes laborales requerido como insumo fundamental del pipeline completo, pudiendo paralizar fases posteriores de procesamiento NLP y análisis semántico hasta que se resuelva problema de adquisición de datos.

La segunda categoría engloba riesgos organizacionales de gestión de proyecto, específicamente R06 relacionado con sobrecarga académica de integrantes por compromisos paralelos en otras asignaturas del programa curricular de Ingeniería de Sistemas que reduce disponibilidad horaria semanal efectiva por debajo de estimación inicial de 8 a 12 horas por persona, y R07 asociado a retrasos acumulados en entrega de fases metodológicas que comprometan cumplimiento del cronograma general de 16 semanas establecido para completar trabajo de grado. Estos riesgos organizacionales podrían generar efecto cascada de retrasos progresivos donde demoras tempranas en fases críticas de path crítico se amplifican en etapas posteriores, forzando reducción de alcance funcional o solicitud de extensión formal de plazo académico ante dirección de programa.

Estos cuatro riesgos de criticidad alta recibirán especial atención prioritaria en definición de estrategias de mitigación preventiva detalladas, monitoreo continuo semanal mediante indicadores de

alerta temprana específicos, y activación inmediata de planes de contingencia reactiva en caso de materialización confirmada durante ejecución del proyecto.

### 6.3.3 Estrategias de Mitigación

Para cada riesgo prioritario, se definen las siguientes estrategias de mitigación:

| ID | Estrategia de mitigación (preventiva) | Plan de contingencia (reactiva) |
|---|---|---|
| R01 | Diseñar scrapers modulares y flexibles. Revisar semanalmente la estructura de los portales durante la fase de scraping. | Si un portal cambia, ajustar el scraper en un plazo máximo de 3 días. Si no es viable, reemplazar por otro portal del mismo país. |
| R02 | Implementar delays aleatorios entre peticiones, rotar user agents, respetar robots.txt y usar Selenium/Playwright cuando sea necesario. | Si un portal bloquea el acceso, cambiar a scraping manual asistido o buscar datasets públicos alternativos (APIs, Kaggle). |
| R03 | Probar varios modelos NER en español (spaCy, BETO, modelos de HuggingFace) en etapa temprana. Validar con muestras pequeñas antes de procesar todo el corpus. | Si el rendimiento es bajo, complementar con expresiones regulares ad-hoc, diccionarios de habilidades predefinidos, o ajuste manual de resultados. |
| R04 | Validar acceso a portales y disponibilidad de vacantes antes de iniciar scraping masivo. Tener al menos 2 portales por país como respaldo. | Si un país tiene datos insuficientes, reducir su alcance o reemplazarlo por otro país latinoamericano con mayor disponibilidad. |
| R05 | Comenzar con técnicas simples (K-Means, DBSCAN) antes de pasar a HDBSCAN. Realizar pruebas con datasets sintéticos pequeños. | Si HDBSCAN no converge o genera clusters poco útiles, reducir dimensionalidad con PCA en lugar de UMAP, o aplicar clustering jerárquico tradicional. |
| R06 | Planificar cronograma considerando semanas de exámenes y entregas paralelas. Distribuir carga de trabajo de forma flexible. | Si un integrante tiene sobrecarga puntual, redistribuir tareas urgentes entre el resto del equipo temporalmente. |
| R07 | Hacer seguimiento semanal estricto del cronograma. Detectar retrasos tempranamente y aplicar correcciones inmediatas. | Si el retraso es mayor a 2 semanas, reducir el alcance de fases menos críticas o solicitar extensión formal del plazo final. |
| R09 | Mantener backups semanales en GitHub y Google Drive. Documentar configuraciones de entorno en README. | Si un equipo falla, el integrante afectado podrá clonar el repositorio completo en otro equipo y continuar trabajando con mínima pérdida de tiempo. |
| R10 | Implementar sistema de backups automáticos semanales en GitHub y Google Drive. Verificar integridad de backups mensualmente. | Si hay pérdida de datos, restaurar desde el último backup disponible. Si no hay backup, repetir el trabajo perdido priorizando lo más crítico. |

Tabla 6.4: Estrategias de Mitigación y Planes de Contingencia

### 6.3.4 Monitoreo de Riesgos

El seguimiento de riesgos se realizará de forma continua durante todo el proyecto mediante implementación de cuatro actividades complementarias de monitoreo sistemático que operan en diferentes escalas temporales y niveles de granularidad para detectar tempranamente señales de materialización de amenazas identificadas.

La primera actividad establece revisión semanal estructurada de riesgos donde en cada reunión de seguimiento del equipo se dedicará espacio breve de 10 minutos para revisar colaborativamente si algún riesgo identificado en matriz de análisis se ha materializado parcial o totalmente durante semana transcurrida, si han surgido nuevos riesgos emergentes no contemplados inicialmente que ameriten incorporación a tabla de seguimiento, si estrategias de mitigación preventiva implementadas están funcionando efectivamente según evidencia empírica observada, y si es necesario ajustar nivel de probabilidad o impacto de riesgos existentes en función de información actualizada obtenida durante ejecución del proyecto.

La segunda actividad implementa monitoreo continuo de indicadores de alerta temprana consistentes en señales observables que indiquen proximidad creciente de materialización de riesgos específicos antes de que se manifiesten completamente, permitiendo activación proactiva de medidas correctivas anticipadas. Estos indicadores cuantitativos y cualitativos incluyen como primer indicador errores frecuentes superiores a 3 fallos diarios en ejecución de scrapers por cambios detectados en estructura HTML de portales señalizando proximidad de R01, como segundo indicador aumento progresivo en tiempo promedio de respuesta HTTP de portales objetivo superiores a 5 segundos o aparición de CAPTCHAs en sesiones de scraping alertando sobre posible R02, como tercer indicador bajo rendimiento en métricas de evaluación de NER con F1-score inferior a 60 por ciento en validación de muestras indicando materialización de R03, y como cuarto indicador retrasos acumulados superiores a 3 días en entrega de tareas asignadas según cronograma WBS señalizando activación de R06 y potencial desencadenamiento de R07 por efecto cascada.

La tercera actividad exige registro formal detallado de incidentes donde cualquier materialización confirmada de riesgo será documentada exhaustivamente en acta semanal de reunión de seguimiento mediante plantilla estructurada que indique identificador del riesgo activado según código de tabla de análisis, descripción concreta de cómo se manifestó la amenaza con evidencia empírica específica observada, acción correctiva aplicada inmediatamente especificando plan de contingencia ejecutado y recursos movilizados, integrante responsable de implementar solución y realizar seguimiento de efectividad, y resultado final obtenido cuantificando tiempo de resolución y impacto residual sobre cronograma o calidad de entregables.

La cuarta actividad establece actualización dinámica de matriz de riesgos donde si durante desarrollo del proyecto se identifica nuevo riesgo relevante no contemplado en análisis inicial que tenga probabilidad media o alta y impacto medio o alto, será agregado formalmente a tabla de identificación y clasificación con asignación de código secuencial R11 en adelante, evaluación fundamentada de

probabilidad e impacto según criterios consistentes con análisis previo, categorización en tipo técnico, organizacional o infraestructura, definición de estrategia de mitigación preventiva y plan de contingencia reactivo siguiendo mismo formato que riesgos originales, y comunicación al equipo completo y asesor académico durante siguiente reunión de seguimiento para aprobación e incorporación al proceso de monitoreo continuo.

### 6.3.5 Responsabilidades en Gestión de Riesgos

La distribución de responsabilidades en gestión de riesgos del proyecto sigue modelo colaborativo diferenciado donde cada integrante asume liderazgo específico sobre categorías particulares de amenazas según especialización técnica y rol asignado en estructura organizacional del equipo, complementado con responsabilidades compartidas transversales que requieren participación activa de todos los miembros.

Nicolás Camacho, en su rol de líder técnico y arquitecto principal del sistema, asume responsabilidad primaria sobre identificación proactiva y monitoreo continuo de riesgos técnicos categorizados como R01 hasta R05 en matriz de análisis, incluyendo amenazas relacionadas con scraping web, rendimiento de modelos NLP, disponibilidad y calidad de datos, y complejidad de implementación de clustering semántico. Sus funciones específicas incluyen proponer soluciones técnicas fundamentadas ante materialización de riesgos de esta categoría mediante análisis de alternativas viables, coordinar aplicación operativa de estrategias de mitigación preventiva definidas ejecutando ajustes de código y configuración necesarios, y evaluar efectividad de medidas correctivas implementadas mediante análisis de métricas de desempeño y calidad de outputs generados.

Alejandro Pinzón, en su rol de coordinador de proyecto y responsable de documentación académica, asume responsabilidad primaria sobre monitoreo continuo de riesgos organizacionales categorizados como R06 hasta R08 en matriz de análisis, incluyendo amenazas relacionadas con sobrecarga académica de integrantes, retrasos acumulados en cronograma y conflictos internos de coordinación en equipo. Sus funciones específicas incluyen gestión activa del cronograma WBS verificando semanalmente cumplimiento de plazos establecidos mediante revisión de estado de tareas asignadas, detección temprana de desviaciones respecto a planificación original superior a 2 días mediante comparación entre fechas estimadas y reales de entrega, y propuesta de acciones correctivas oportunas incluyendo redistribución temporal de carga de trabajo entre integrantes o ajuste de alcance de fases no críticas para recuperar alineación con cronograma maestro.

Todo el equipo sin distinción de rol asume responsabilidades compartidas transversales consistentes en reportar inmediatamente cualquier riesgo detectado durante ejecución de tareas asignadas mediante comunicación en reunión semanal o canal urgente si criticidad lo amerita, participar activamente en evaluación colaborativa de impacto de riesgos materializados o emergentes mediante discusión estructurada fundamentada en evidencia empírica disponible, y colaborar constructivamente en implementación de planes de contingencia reactiva cuando sea necesario mediante ejecución de

tareas de emergencia asignadas o soporte técnico cruzado entre integrantes para acelerar resolución de incidentes críticos.

La gestión de riesgos será proceso continuo y colaborativo ejecutado de forma integrada a actividades semanales regulares del proyecto sin constituir overhead administrativo separado, con objetivo explícito de minimizar incertidumbre asociada a amenazas identificadas mediante preparación anticipada y respuesta ágil, y maximizar probabilidad de éxito en entrega final del observatorio laboral cumpliendo estándares de calidad técnica, cronograma académico establecido y objetivos funcionales definidos en alcance del trabajo de grado.

# BIBLIOGRAFÍA

[1] S. O. Aguilera y R. E. Méndez, "¿Qué buscan los que buscan? Análisis De mercado laboral IT en Argentina," *Revista Perspectivas*, vol. 1, n.º 1, págs. 15-30, 2018. dirección: https://revistas.ub.edu.ar/index.php/Perspectivas/article/view/39

[2] J. A. Cárdenas Rubio, J. C. Guataquí Roa y J. M. Montaña Doncel, "Metodología para el análisis de demanda laboral mediante datos de internet: caso Colombia," Organización Internacional del Trabajo / CINTERFOR, Informe técnico, 2015. dirección: https://www.oitcinterfor.org/sites/default/files/file_publicacion/Metodolog%C3%ADa%20An%C3%A1lisis%20Demanda%20Laboral%20Mediante%20Datos%20Internet%20caso%20Colombia.pdf

[3] O. Puello y L. F. Gómez Estrada, "Proyecto de Grado II – Web Scraping," Trabajo de grado, Universidad del Sinú Elías Bechara Zainúm, Seccional Cartagena, Facultad de Ciencias Exactas e Ingenierías, Escuela de Ingeniería de Sistemas, 2019. dirección: http://repositorio.unisinucartagena.edu.co:8080/jspui/bitstream/123456789/94/1/1.%20Proyecto%20de%20Grado%20II%20-%20WEB%20SCRAPING_FINAL.pdf

[4] J. A. Rubio Arrubla, "Demanda de habilidades tecnológicas: evidencia desde el mercado laboral colombiano," Tesis de maestría, Universidad de los Andes, 2024. dirección: https://repositorio.uniandes.edu.co/server/api/core/bitstreams/ea4ea129-d35e-498c-aa46-028e3d8ffb5e/content

[5] J. C. Martínez Sánchez, "Desajuste en el mercado laboral: análisis de los perfiles de candidatos y las ofertas de trabajo publicadas en internet," *Revista del INEGI*, vol. 44, 2024. dirección: https://rde.inegi.org.mx/wp-content/uploads/2024/pdf/RDE44/RDE44_art01.pdf

[6] R. M. Campos-Vázquez y J. C. Martínez Sánchez, "Skills sought by companies in the Mexican labor market: An analysis of online job vacancies," *Estudios Económicos De El Colegio De México*, vol. 39, n.º 2, págs. 243-278, 2024. dirección: https://estudioseconomicos.colmex.mx/index.php/economicos/article/view/452/619

[7] M. Lukauskas, V. Šarkauskaitė, V. Pilinkienė y A. Stundziene, "Enhancing Skills Demand Understanding through Job Ad Segmentation Using NLP and Clustering Techniques," *ResearchGate*, 2023. dirección: https://www.researchgate.net/publication/370816962

[8] K. Nguyen, M. Zhang, S. Montariol y A. Bosselut, "Rethinking Skill Extraction in the Job Market Domain using Large Language Models," en *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, St. Julian's, Malta: Association for Computational Linguistics, 2024, págs. 27-42. dirección: https://aclanthology.org/2024.nlp4hr-1.3/

[9] L. Echeverría y G. Rucci, "¿Qué suma la ciencia de datos a la identificación y anticipación de la demanda de habilidades?" Banco Interamericano de Desarrollo, inf. téc., 2022. dirección: https://publications.iadb.org/es/que-suma-la-ciencia-de-datos-la-identificacion-y-anticipacion-de-la-demanda-de-habilidades

[10] E. Razumovskaia et al., "Multilingual Skills Classification with LLMs," en *Proceedings of CLiC-it 2024*, Pisa, Italia, 2024.

[11] J. López et al., "Entity Linking and Skill Extraction with Chain-of-Thought Reasoning," en *Proceedings of GenAIK Workshop*, 2025.

[12] H. Kavas, M. Serra-Vidal y L. Wanner, "Multilingual Skill Extraction for Job Vacancy–Job Seeker Matching in Knowledge Graphs," en *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, Abu Dhabi, Emiratos Árabes Unidos: International Committee on Computational Linguistics, 2025. dirección: https://aclanthology.org/2025.genaik-1.15.pdf

[13] L. Vásquez-Rodríguez et al., "Hardware-effective Approaches for Skill Extraction in Job Offers and Resumes," en *RecSys in HR Workshop 2024*, Bari, Italia, 2024. dirección: https://publications.idiap.ch/attachments/papers/2024/Vasquez-Rodriguez_RECSYSINHR24_2024.pdf