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 las siguientes etapas: (1) scraping de portales de empleo (Computrabajo, elempleo.com, Bumeran), (2) extracción de habilidades mediante NER y regex, (3) enriquecimiento semántico con LLMs, (4) representación vectorial con embeddings multilingües, (5) clustering con UMAP y HDBSCAN, y (6) generación de visualizaciones macro estáticas.

Quedan fuera del alcance: el desarrollo de dashboards interactivos, la construcción de portales web, la automatización continua del pipeline y la integración con bases de datos externas o APIs privadas.

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

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

1.3 Supuestos y Restricciones

1.3.1 Supuestos

- Se mantendrá acceso continuo a portales de empleo y/o APIs para extracción de vacantes.
- Existirá un corpus suficiente y variado de vacantes en español.
- Se contará con infraestructura local adecuada para ejecutar las tareas técnicas.
- El equipo mantendrá una coordinación fluida durante las semanas de ejecución.
- Se contará con modelos a utilizar que tengan el funcionamiento adecuado en español.

1.3.2 Restricciones

- Existencia de carga académica paralela por parte del equipo.
- Capacidad limitada de procesamiento local (sin uso de GPUs o servidores externos).
- Tiempo acotado para ajuste fino de prompts, clustering y visualizaciones.

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 las siguientes fases principales:

1. **Comprensión del dominio y diseño del sistema:** incluye la revisión crítica del estado del arte, la definición del pipeline modular, y la planificación por etapas.
2. **Extracción de datos (scraping):** implementación de spiders personalizados por portal y país, almacenamiento estructurado en base de datos relacional.
3. **Procesamiento semántico y enriquecimiento:** aplicación de NER, regex y validación con LLMs para extracción explícita e implícita de habilidades.
4. **Vectorización y clustering:** obtención de embeddings semánticos, reducción de dimensionalidad y agrupación de perfiles con HDBSCAN.
5. **Visualización y validación:** generación de gráficos estáticos para evaluación cualitativa y cuantitativa por expertos.
6. **Documentación y empaquetado final:** consolidación de la guía metodológica, código versionado, resultados y recomendaciones futuras.

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

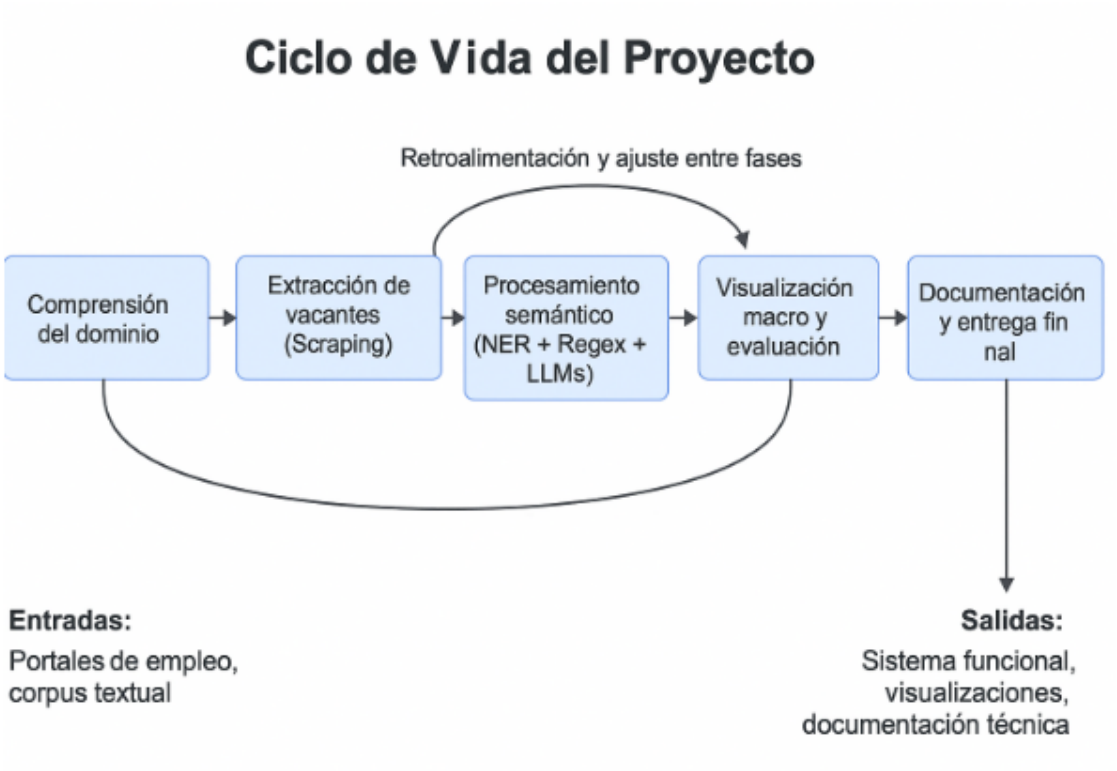

Figura 3.1: Ciclo de Vida del Proyecto basado en CRISP-DM con prácticas ágiles

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

Scrapy

Descripción: Framework asíncrono de scraping en Python, diseñado para proyectos de extracción a gran escala con arquitectura modular basada en spiders.

Ventajas:

- Alto rendimiento mediante ejecución asíncrona (Twisted).
- Gestión automática de concurrencia, reintentos y delays.
- Exportación nativa a JSON, CSV, SQL.
- Comunidad activa y documentación robusta.

Limitaciones:

- No ejecuta JavaScript de forma nativa.
- Curva de aprendizaje moderada para scrapers complejos.

Selenium

Descripción: Herramienta de automatización de navegadores que permite interactuar con contenido dinámico cargado mediante JavaScript.

Ventajas:

- Renderiza JavaScript completamente, ideal para sitios dinámicos.
- Permite simulación de interacciones humanas (clicks, scroll).
- Compatible con múltiples navegadores (Chrome, Firefox).

Limitaciones:

- Significativamente más lento que Scrapy.
- Consume más recursos (CPU, memoria).
- Requiere gestión explícita de drivers (ChromeDriver).

Playwright (evaluado como respaldo)

Descripción: Alternativa moderna a Selenium, desarrollada por Microsoft, con API más simple y ejecución más rápida.

Ventajas:

- Más rápido y estable que Selenium.
- API más limpia y moderna.
- Soporte nativo para capturas de pantalla y manejo de red.

Limitaciones:

- Comunidad más pequeña que Selenium.
- Menos ejemplos específicos para scraping laboral.

Decisión final: Se adopta **Scrapy como herramienta principal** por su rendimiento y robustez para scraping masivo de páginas estáticas o semi-dinámicas. **Selenium se empleará como respaldo** para portales que requieran ejecución de JavaScript. Playwright queda como alternativa secundaria en caso de problemas con Selenium.

3.2.2 Sistemas de Gestión de Bases de Datos

PostgreSQL

Descripción: Sistema de gestión de bases de datos relacional de código abierto, con fuerte soporte para consultas complejas, integridad referencial y tipos de datos avanzados.

Ventajas:

- Soporte nativo para JSON y tipos de datos no estructurados.
- ACID completo, garantizando integridad transaccional.
- Extensiones para búsqueda de texto completo (pg_trgm).
- Escalabilidad probada en proyectos de mediano y gran tamaño.
- Integración directa con pandas y SQLAlchemy.

Limitaciones:

- Requiere instalación y configuración local o en servidor.
- Mayor complejidad administrativa que SQLite.

MongoDB

Descripción: Base de datos NoSQL orientada a documentos, almacena datos en formato JSON/BSON flexible.

Ventajas:

- Esquema flexible, ideal para datos semi-estructurados.
- Fácil almacenamiento de anidaciones complejas.
- Alto rendimiento en escritura masiva.

Limitaciones:

- No garantiza integridad referencial estricta.
- Consultas relacionales complejas son más difíciles.
- Menos adecuado para análisis estructurado y normalización posterior.

SQLite

Descripción: Base de datos relacional ligera, embebida, sin necesidad de servidor.
Ventajas:

- Configuración mínima (archivo único).
- Ideal para prototipos rápidos y datasets pequeños.
- Compatible con SQL estándar.

Limitaciones:

- Rendimiento limitado con grandes volúmenes de datos.
- Sin soporte para concurrencia de escritura eficiente.
- Carece de funciones avanzadas de PostgreSQL.

Decisión final: Se selecciona **PostgreSQL** como base de datos principal, por su capacidad para manejar esquemas estructurados con integridad referencial, su soporte avanzado para consultas analíticas, su extensibilidad para búsqueda textual, y su integración sólida con el ecosistema Python de ciencia de datos.

3.2.3 Bibliotecas de Procesamiento de Lenguaje Natural (NLP)

spaCy

Descripción: Biblioteca industrial de NLP en Python, optimizada para eficiencia y modularidad, con soporte para múltiples idiomas incluyendo español.
Ventajas:

- Rápida y eficiente en producción.
- Modelos preentrenados de calidad para español (es_core_news).
- Soporte nativo para tokenización, lematización, POS tagging y NER.
- Fácil integración con transformers de HuggingFace.

Limitaciones:

- Modelos de NER genéricos, no especializados en dominio laboral.
- Requiere ajuste fino o complementos para habilidades técnicas específicas.

Stanford NLP / Stanza

Descripción: Suite de herramientas NLP de Stanford, con Stanza como implementación en Python.
Ventajas:

- Modelos lingüísticos sólidos con fundamento académico.
- Soporte para análisis sintáctico profundo (dependencias).
- Modelos multilingües entrenados en corpus universales.

Limitaciones:

- Más lento que spaCy en ejecución.
- Configuración más compleja.
- Menor integración con ecosistema de embeddings modernos.

Transformers de HuggingFace (BETO, RoBERTuito)

Descripción: Modelos de lenguaje basados en arquitectura Transformer (BERT, RoBERTa) pre-entrenados en español.
Ventajas:

- Representaciones contextuales de alta calidad.
- BETO entrenado específicamente en español.
- Adaptable mediante fine-tuning para dominios específicos.

Limitaciones:

- Requiere recursos computacionales significativos.
- Latencia alta para procesamiento en tiempo real.
- Fine-tuning requiere datasets etiquetados grandes.

Decisión final: Se adopta **spaCy como biblioteca principal de NLP** por su balance entre velocidad, facilidad de uso y calidad en tareas básicas de NER y tokenización. Se complementará con **expresiones regulares personalizadas** para habilidades técnicas específicas, y se evaluará el uso de **modelos Transformer (BETO)** para tareas de enriquecimiento semántico en fases posteriores si los recursos computacionales lo permiten.

3.2.4 Modelos de Embeddings Semánticos

BETO (BERT en español)

Descripción: Modelo BERT preentrenado en corpus masivo de español, adecuado para tareas de comprensión de lenguaje contextual.

Ventajas:

- Entrenado específicamente en español.
- Representaciones contextuales de alta calidad.
- Compatible con biblioteca Transformers.

Limitaciones:

- No optimizado para similitud semántica (embeddings densos).
- Requiere fine-tuning para tareas de sentence similarity.
- Alto costo computacional.

Multilingual-E5 / LaBSE

Descripción: Modelos multilingües diseñados específicamente para generar embeddings densos de oraciones con alta similitud semántica cross-lingual.

Ventajas:

- Optimizados para similitud semántica (cosine similarity).
- Soporte para múltiples idiomas en un mismo espacio vectorial.
- LaBSE entrenado en más de 100 idiomas.
- E5 con rendimiento superior en benchmarks recientes.

Limitaciones:

- Modelos grandes (requisitos de memoria).
- Menor especialización en dominio laboral específico.

SBERT (Sentence-BERT)

Descripción: Extensión de BERT optimizada para embeddings de oraciones mediante siamese networks.

Ventajas:

- Rápido en generación de embeddings comparado con BERT estándar.
- Modelos multilingües disponibles (paraphrase-multilingual).
- Amplia adopción en tareas de similitud semántica.

Limitaciones:

- Rendimiento variable según idioma y dominio.
- Algunos modelos no incluyen español en su entrenamiento principal.

fastText

Descripción: Embeddings basados en subpalabras, livianos y entrenables localmente.

Ventajas:

- Rápido y ligero.
- Maneja palabras fuera de vocabulario (OOV) mediante subpalabras.
- Entrenable en corpus específico del proyecto.

Limitaciones:

- No captura contexto semántico profundo.
- Embeddings estáticos (no contextuales).
- Menor rendimiento en similitud semántica compleja.

Decisión final: Se adopta **Multilingual-E5 como modelo principal de embeddings** por su optimización específica para similitud semántica, su soporte multilingüe (español-inglés), y su rendimiento superior en benchmarks de sentence similarity. Como alternativa secundaria, se evaluará **LaBSE** si se requiere mayor cobertura multilingüe, y **fastText** como respaldo ligero 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.

3.3.1 Lenguaje de programación principal

Python: Python es el lenguaje principal del proyecto debido a su simplicidad, versatilidad y fuerte ecosistema para ciencia de datos, scraping y procesamiento de lenguaje natural.

3.3.2 Extracción de datos (Scraping)

- **Scrapy:** Framework robusto para scraping asíncrono, útil para manejar grandes volúmenes de páginas con estructuras HTML regulares.
- **Selenium:** Utilizado como respaldo para portales con contenido dinámico cargado por JavaScript.
- **Playwright (respaldo opcional):** Herramienta moderna para automatización de navegación.

3.3.3 Almacenamiento de datos

PostgreSQL: Sistema de gestión de bases de datos relacionales que garantiza integridad, consultas complejas y eficiencia.

3.3.4 Procesamiento de texto y NLP

- **spaCy (modelo spaCy-es):** Biblioteca moderna para NLP, enfocada en eficiencia y modularidad.
- **Expresiones Regulares (Regex):** Utilizadas para detectar patrones específicos dentro de campos.
- **NLTK y Stanza (respaldo opcional):** En caso de requerirse mayor granularidad lingüística.

3.3.5 Enriquecimiento semántico y LLMs

- **Hugging Face Transformers + LLaMA:** Para realizar razonamiento semántico e inferencia de habilidades implícitas mediante prompting en español.
- **Prompt Engineering:** Diseño iterativo de instrucciones para tareas como normalización de habilidades.

3.3.6 Embeddings y representación semántica

- **SentenceTransformers:** Biblioteca que permite el uso de modelos como BETO, LaBSE, E5 y SBERT para generar representaciones vectoriales semánticas.
- **fastText (respaldo):** Modelo entrenable localmente, útil en caso de requerir embeddings ligeros o adaptables.

3.3.7 Agrupamiento y reducción de dimensionalidad

- **UMAP:** Para proyectar espacios de embeddings de alta dimensión a representaciones más interpretables.
- **HDBSCAN:** Algoritmo robusto para agrupamiento sin necesidad de predefinir número de clusters.
- **k-means / DBSCAN (respaldo):** Métodos tradicionales considerados en caso de requerirse comparaciones.

3.3.8 Visualización de resultados

- **Plotly y Dash:** Librerías para la generación de visualizaciones interactivas o estáticas.
- **Matplotlib / Seaborn (respaldo):** Librerías complementarias para generar gráficos más simples.

3.3.9 Control de versiones y colaboración

Git + GitHub: Sistema de control de versiones distribuido y plataforma para gestión colaborativa del código.

3.3.10 Documentación y edición

- **Google Docs y Overleaf:** Google Docs se utilizará para documentación colaborativa. Overleaf será el entorno elegido para la redacción final en LaTeX.
- **Markdown (README):** Estandarizado para documentar módulos y scripts dentro del repositorio GitHub.

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.

3.4.1 Diseño técnico y arquitectura

Criterios de aceptación:

- Diagrama modular del pipeline documentado (en BPMN o equivalente).
- Justificación detallada de elección de tecnologías y estructura de datos.

- Documento metodológico inicial validado por el asesor.

Técnicas e instrumentos:

- Revisión por pares.
- Validación en reunión de asesoría.
- Documento compartido y versionado en Google Docs y GitHub.

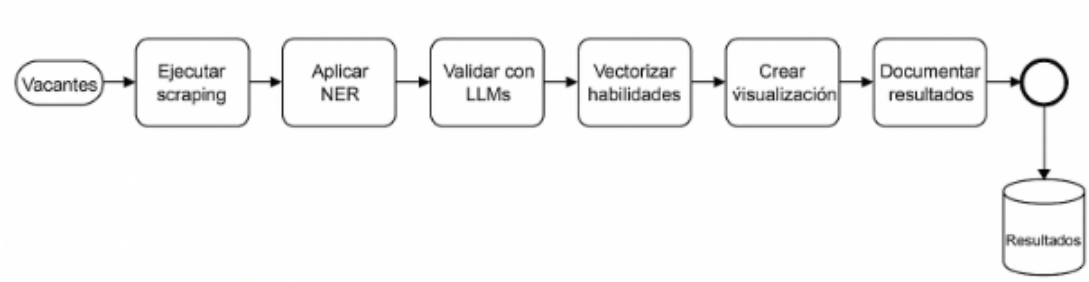

Figura 3.2: Diagrama BPMN del flujo general del proceso del observatorio

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

Criterios de aceptación:

- Spiders funcionales en al menos dos portales por país.
- Almacenamiento exitoso de vacantes en PostgreSQL con al menos 500 registros reales.
- Código funcional documentado, modular y con respaldo ante fallos.

Técnicas e instrumentos:

- Validación de funcionamiento en tiempo real por parte del asesor.
- Revisión del script y la estructura de la base de datos.
- Comparación de outputs y verificación de duplicados.

3.4.3 Extracción de habilidades explícitas (NER y regex)

Criterios de aceptación:

- Al menos un modelo funcional de NER en español integrado.
- Regex adaptadas al dominio y resultados de extracción evaluables.

- Anotaciones automatizadas disponibles para validación.

Técnicas e instrumentos:

- Validación manual de muestras aleatorias.
- Revisión de logs y outputs del módulo de extracción.
- Comparación con glosarios laborales como ESCO.

3.4.4 Enriquecimiento semántico con LLMs

Criterios de aceptación:

- Prompts definidos y documentados para al menos dos tareas.
- Resultados con tasa de precisión aceptable (¿70 %) sobre una muestra controlada.
- Trazabilidad de razonamiento y justificación de outputs.

Técnicas e instrumentos:

- Evaluación cualitativa en reunión conjunta con el asesor.
- Comparación contra listas de habilidades base.
- Reporte técnico con ejemplos y explicaciones.

3.4.5 Embeddings y clustering

Criterios de aceptación:

- Habilidades representadas mediante embeddings densos en formato vectorial.
- Aplicación exitosa de clustering con HDBSCAN y análisis cualitativo de al menos tres clústeres significativos.
- Visualización preliminar mediante UMAP u otra técnica.

Técnicas e instrumentos:

- Validación de cohesión semántica entre elementos de un mismo clúster.
- Revisión técnica con gráfico y métricas básicas (e.g., Silhouette Score).
- Informe de análisis de clústeres interpretado por el equipo.

3.4.6 Visualización macro

Criterios de aceptación:

- Generación de al menos tres visualizaciones que representen: frecuencia de habilidades, distribución geográfica, y clústeres semánticos.
- Reportes exportables con interpretaciones básicas.
- Usabilidad mínima para revisión técnica.

Técnicas e instrumentos:

- Presentación ante asesor con feedback inmediato.
- Validación del código en entorno local.
- Revisión de consistencia visual y semántica de las gráficas.

3.4.7 Documentación técnica y guía metodológica

Criterios de aceptación:

- Redacción clara, completa y estructurada de todas las etapas metodológicas.
- Instrucciones de replicación paso a paso.
- Inclusión de logs, prompts, scripts y resultados clave.

Técnicas e instrumentos:

- Revisión integral por parte del asesor.
- Comparación con estándares académicos de replicabilidad.
- Validación cruzada entre documento y repositorio.

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 3.1: Tabla de Interfaces Externas del Proyecto

Gestión de las interfaces:

La comunicación con el director del proyecto se realizará mediante reuniones quincenales programadas, comunicación por correo electrónico para consultas urgentes, y revisión de entregables mediante Google Drive compartido y GitHub.

El acceso a portales de empleo se gestionará respetando términos de servicio, robots.txt, y políticas anti-scraping, implementando delays, rotación de user agents y limitación de tasa de peticiones.

El uso de modelos y herramientas de código abierto se documentará adecuadamente, citando licencias y fuentes, y respetando términos de uso académico cuando aplique.

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, cada uno con responsabilidades claramente definidas según sus fortalezas técnicas y organizativas.

Nicolás Camacho - Líder Técnico y Arquitecto del Sistema:

Responsable del diseño de la arquitectura del sistema, coordinación técnica, toma de decisiones clave sobre modelos, herramientas y estructura del pipeline. Supervisa la integración de módulos y garantiza la coherencia técnica de todo el sistema.

Alejandro Pinzón - Desarrollo de Módulos y Documentación:

Encargado del desarrollo técnico de componentes específicos del sistema, incluyendo scraping, procesamiento, embeddings y visualización. También lidera la redacción de documentos formales y la planificación y ejecución de pruebas funcionales.

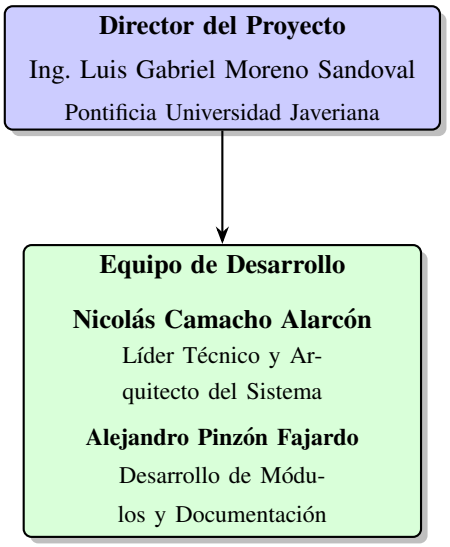

Figura 3.3: Organigrama del equipo de desarrollo del proyecto

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: el análisis de tareas específicas por fase metodológica, la disponibilidad horaria semanal del equipo, y la experiencia práctica en proyectos previos similares. Dado que se trata de un trabajo de grado con una duración prevista de entre 14 y 16 semanas y un equipo de tres personas, se estimó un rango total de **440 a 500 horas de trabajo distribuidas entre los tres integrantes**.

Método de estimación utilizado

Para esta estimación se empleó una **técnica de descomposición (WBS)**, donde cada fase del proyecto fue dividida en tareas concretas, a las que se asignaron horas aproximadas según su complejidad, herramientas requeridas y experiencia previa del equipo. Esta estimación fue luego contrastada con la disponibilidad semanal esperada de cada integrante, ajustada por compromisos académicos paralelos.

El enfoque adoptado es cualitativo, iterativo y conservador. En lugar de aplicar fórmulas matemáticas complejas (como COCOMO), se optó por la **estimación fundamentada en experticia del equipo**, validada contra planes previos y distribuida de forma proporcional.

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 45–55 horas** para estas actividades transversales, que completan las 440–500 horas globales.

Herramientas empleadas para la estimación

- **Google Sheets** para tabular las actividades y distribuir tiempos por fase e integrante.
- **Planificación semanal estimada** en base a 8–12 horas de dedicación por persona por semana.
- **Retroalimentación del director de proyecto**, validando la viabilidad y distribución del esfuerzo.

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, scraping dinámico, procesamiento semántico en español, embeddings multilingües y técnicas de clustering no supervisado, se ha establecido un plan de entrenamiento ligero pero enfocado, que permita nivelar al equipo en los aspectos técnicos esenciales sin comprometer el cronograma establecido.

El entrenamiento será liderado por Nicolás Camacho, quien cuenta con mayor experiencia técnica en las tecnologías clave del proyecto. Su rol incluirá el diseño de pequeñas cápsulas de formación y la transferencia directa de conocimiento al resto del equipo mediante los siguientes mecanismos:

- **Explicaciones prácticas durante las reuniones semanales:** Nicolás explicará en vivo el funcionamiento del código que esté desarrollando en cada fase, incluyendo justificación técnica y recomendaciones de buenas prácticas.
- **Ejemplos rápidos y contextualizados:** Se utilizarán notebooks o scripts breves con datos simples para ilustrar conceptos clave como scraping con Selenium, prompts con LLMs, embeddings, UMAP y clustering.
- **Lecturas y recursos recomendados:** Se compartirá documentación oficial, artículos clave y videos cortos en español o inglés, según el tema y nivel de dificultad.
- **Revisión cruzada de avances:** Los integrantes aplicarán lo aprendido directamente en sus tareas asignadas, recibiendo retroalimentación de Nicolás de forma continua.

A continuación, se detalla la planificación del entrenamiento interno:

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 las necesidades reales del proyecto. No se contempla una fase de formación previa extensa, sino que el aprendizaje estará integrado al flujo de trabajo, adaptado al tiempo disponible y priorizando las tareas más urgentes.

Nicolás también brindará apoyo puntual cuando surjan dudas específicas, y fomentará una cultura de colaboración y mejora técnica, donde se valoran las preguntas, la exploración autónoma y la mejora iterativa. La documentación técnica, liderada por Daniel, incluirá además las explicaciones simplificadas de los procesos clave, con el fin de reforzar el aprendizaje colectivo y dejar registro de las decisiones técnicas para futuras iteraciones o réplicas del proyecto.

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).

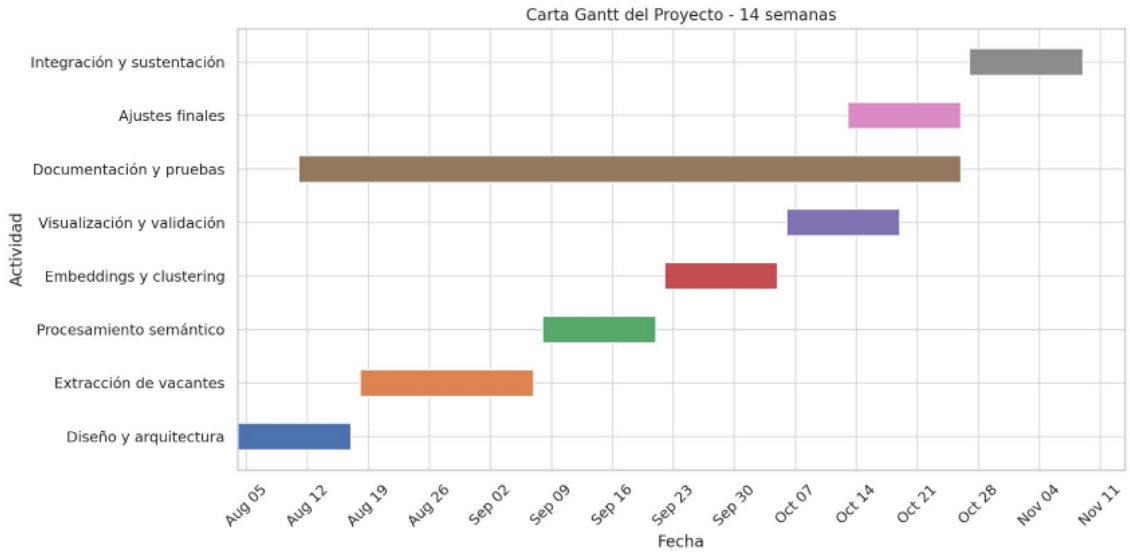

Figura 4.1: Carta Gantt del cronograma del proyecto (16 semanas)

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:

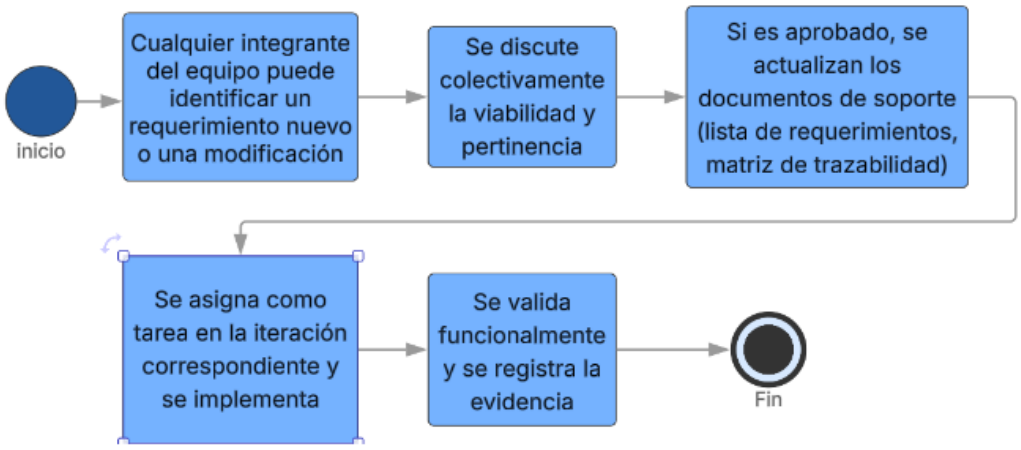

Figura 5.1: Proceso de Administración de Requerimientos (BPMN)

5.1.1 Proceso de Gestión de Cambios

Los cambios a los requerimientos podrán ser propuestos por cualquier miembro del equipo, el asesor del proyecto, o por descubrimiento técnico en ejecución. Todo cambio seguirá el siguiente flujo:

1. **Identificación del cambio:** El integrante que identifica la necesidad de cambio documenta el motivo, impacto esperado y alternativas posibles.
2. **Evaluación de impacto:** Se discute en la reunión semanal si el cambio afecta cronograma, carga de trabajo, dependencias técnicas o entregables.
3. **Decisión por consenso:** Se aprueba o rechaza por acuerdo del equipo. Si hay desacuerdo, el asesor tendrá voz decisoria.
4. **Actualización de documentación:** Si se aprueba, se registra el cambio en el acta semanal, en el documento técnico del proyecto, y se actualiza la planificación afectada.
5. **Comunicación y ajuste:** Los integrantes afectados ajustan su trabajo conforme al cambio aprobado. Se notifica formalmente en la siguiente sesión de seguimiento.

5.1.2 Trazabilidad de Requerimientos

Los requerimientos funcionales están vinculados directamente a las fases del proyecto y sus entregables asociados. La trazabilidad se mantiene mediante los siguientes mecanismos:

- **Tabla de requerimientos funcionales y fases:** Cada requerimiento está asociado a una o más fases metodológicas, lo que facilita identificar el momento de su validación y prueba.
- **Repositorio de código en GitHub:** Cada módulo tiene una carpeta identificada y commits descriptivos que referencian la fase técnica correspondiente.
- **Actas de seguimiento:** Registro semanal de avances, donde se indica qué requerimientos fueron implementados, probados o modificados.
- **Documento técnico final:** En la documentación se incluirá una matriz de trazabilidad que vincule cada requerimiento funcional con su fase de diseño, implementación, prueba y validación.

5.1.3 Aprobación y Validación de Cambios

Todos los cambios a requerimientos deben ser validados por:

- El equipo de desarrollo (validación técnica de viabilidad)
- El asesor del proyecto (validación académica y coherencia con los objetivos)

- El director del proyecto (aprobación formal si el cambio implica modificación del alcance o fechas)

La aprobación se formaliza mediante firma digital o confirmación escrita en el acta correspondiente.

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 del avance del proyecto se realizará mediante los siguientes mecanismos:

- **Reuniones semanales de seguimiento:** Cada lunes, de 1 hora de duración, con presencia de todo el equipo. Se discute el avance de la semana anterior, se presentan los bloqueos encontrados, se revisan las métricas de avance y se planifican las tareas de la siguiente semana.

- **Actas de reunión:** Se redactará un acta resumida al final de cada reunión semanal, incluyendo asistencia, temas tratados, decisiones tomadas, compromisos adquiridos y fecha de próxima reunión. Las actas serán almacenadas en Google Drive compartido.
- **Reportes quincenales al asesor:** Cada dos semanas, el equipo enviará un reporte técnico breve (1–2 páginas) al asesor del proyecto, indicando el estado de las fases, problemas técnicos detectados, soluciones aplicadas y planes próximos.
- **Revisión de fase:** Al finalizar cada fase metodológica, se llevará a cabo una reunión de revisión extendida (de hasta 2 horas) con el asesor, donde se presenta el entregable correspondiente, se valida su calidad y se ajusta el plan para las siguientes fases si es necesario.

5.2.3 Acciones Correctivas

Cuando se detecte una desviación significativa en el cronograma, los recursos disponibles, o la calidad técnica de los entregables, el equipo podrá aplicar las siguientes acciones correctivas:

- **Reprogramación de actividades:** Si una fase se retrasa más de una semana, se evalúa reducir el alcance técnico de otras fases menos críticas, redistribuir tareas entre integrantes, o ajustar el cronograma global con aprobación del asesor.
- **Refuerzo técnico colaborativo:** Ante bloqueos técnicos, Nicolás Camacho brindará acompañamiento adicional a los integrantes afectados, incluyendo sesiones de pair programming o revisión de código guiada.
- **Simplificación técnica:** Si una técnica planificada resulta inviable por limitaciones computacionales, falta de datos o complejidad excesiva, se reemplazará por una alternativa más simple pero válida técnicamente (por ejemplo, pasar de fine-tuning a prompting directo con LLMs).
- **Extensión controlada del cronograma:** Como último recurso, si la calidad del entregable está comprometida, se podrá solicitar una extensión de hasta 2 semanas, previa validación con el director del proyecto y ajuste formal del plan.

Todas las acciones correctivas deberán quedar documentadas en el acta correspondiente, indicando el problema detectado, la decisión tomada, el responsable de implementarla y el plazo de ejecución.

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 incluirá los siguientes componentes:

- **Repositorio de código funcional:** Alojado en GitHub, con estructura modular clara, código comentado, archivo README con instrucciones de instalación y ejecución, requirements.txt con dependencias, y ejemplos de ejecución del pipeline completo.
- **Dataset procesado:** Base de datos PostgreSQL exportada (dump SQL) con las tablas de vacantes, habilidades extraídas, embeddings y clusters. Incluye un diccionario de datos explicativo.
- **Documento técnico final:** Informe completo del proyecto en formato PDF, con introducción, metodología, arquitectura del sistema, descripción de cada fase técnica, resultados obtenidos, análisis de validación, conclusiones y recomendaciones. Mínimo 60 páginas, formato IEEE o plantilla institucional.
- **Visualizaciones y notebooks:** Carpeta con gráficos generados (histogramas, clusters, mapas de calor, nubes de palabras) y notebooks de Jupyter con análisis exploratorios y explicaciones técnicas detalladas.
- **Manual de usuario técnico:** Guía breve para ejecutar el sistema en otro equipo, incluyendo requisitos de hardware, instalación de dependencias, configuración de variables de entorno, y pasos para replicar las fases del pipeline.
- **Evidencias de validación:** Logs de ejecución, métricas de evaluación (precisión de NER, coherencia de clusters, ejemplos de habilidades enriquecidas), y capturas de pantalla de visualizaciones funcionales.

5.3.2 Actividades de Cierre

Al finalizar el desarrollo del proyecto, se llevarán a cabo las siguientes actividades de cierre:

1. **Revisión final de entregables:** El equipo verifica que todos los componentes estén completos, funcionales y correctamente documentados.
2. **Sesión de retrospectiva final:** Reunión de 2 horas donde el equipo reflexiona sobre aprendizajes técnicos, dificultades enfrentadas, decisiones acertadas y mejoras aplicables a futuros proyectos similares. Se documenta en un acta de lecciones aprendidas.
3. **Presentación final al asesor:** Se expone el sistema completo al asesor del proyecto, demostrando su funcionamiento end-to-end, explicando decisiones técnicas clave y respondiendo preguntas técnicas y metodológicas.

4. **Preparación de defensa pública:** El equipo elabora una presentación de entre 20 y 30 minutos para la sustentación formal ante jurados, enfocada en los resultados técnicos, validación empírica y aportes del proyecto.

5. **Archivo del proyecto:** Todos los entregables finales se suben a un repositorio institucional (si aplica) y se entrega copia a la universidad en los formatos solicitados (PDF, código fuente, datasets).

6. **Liberación pública (opcional):** Si el equipo lo decide y el asesor lo autoriza, se puede liberar el código bajo licencia open source (MIT o Apache 2.0) para beneficio de la comunidad técnica y académica.

5.3.3 Criterios de Aceptación Final

El proyecto será considerado completo y aceptable cuando:

- Todos los requerimientos funcionales prioritarios hayan sido implementados y validados técnicamente.
- El sistema sea capaz de ejecutar el pipeline completo desde scraping hasta visualización sin errores críticos.
- La documentación técnica esté completa, coherente y permita replicar el sistema por terceros con conocimientos técnicos medios.
- El asesor del proyecto apruebe formalmente la calidad técnica y académica del trabajo.
- Se hayan cumplido las entregas programadas según el cronograma ajustado, o se hayan justificado formalmente las extensiones aplicadas.

Una vez cumplidos estos criterios, el proyecto podrá ser presentado a evaluación formal por parte de los jurados designados.

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 y de los artefactos técnicos se realizará mediante Git y GitHub, siguiendo las siguientes prácticas:

- **Rama principal (main):** Contiene la versión estable del sistema, probada y validada al cierre de cada fase metodológica. Los commits a esta rama requieren revisión previa.
- **Ramas de desarrollo por fase (dev-fase-X):** Cada fase metodológica tendrá una rama temporal donde se desarrollarán las funcionalidades correspondientes. Al validarse técnicamente, se fusionará a main mediante pull request.
- **Commits descriptivos:** Todo commit debe tener un mensaje claro que indique qué cambio se realizó, por qué y en qué fase del proyecto se hizo. Formato sugerido: "[FASE-X] Descripción breve del cambio".
- **Versionado semántico para entregas:** Cada entregable mayor será etiquetado con un tag de versión (v0.1, v0.2, v1.0, etc.) para facilitar la trazabilidad histórica.

- **Sincronización diaria:** Los integrantes deberán hacer push de sus avances al menos una vez al día, para evitar conflictos de integración y facilitar la colaboración.

6.1.3 Gestión de Cambios en la Configuración

Cualquier cambio en la configuración del sistema (por ejemplo, modificación de estructura de base de datos, cambio de librerías clave, ajuste de arquitectura de scraping) deberá seguir este procedimiento:

1. El integrante que propone el cambio documenta la razón técnica, el impacto esperado y las alternativas evaluadas.
2. Se discute en reunión semanal si el cambio es necesario, viable y justificado.
3. Si se aprueba, se implementa en una rama específica, se prueba localmente, y se documenta en el README o en comentarios del código.
4. Se abre un pull request para revisión por parte de Nicolás antes de fusionar a la rama principal.
5. Se actualiza el archivo de configuración correspondiente, se registra el cambio en el acta semanal y se comunica formalmente al equipo.

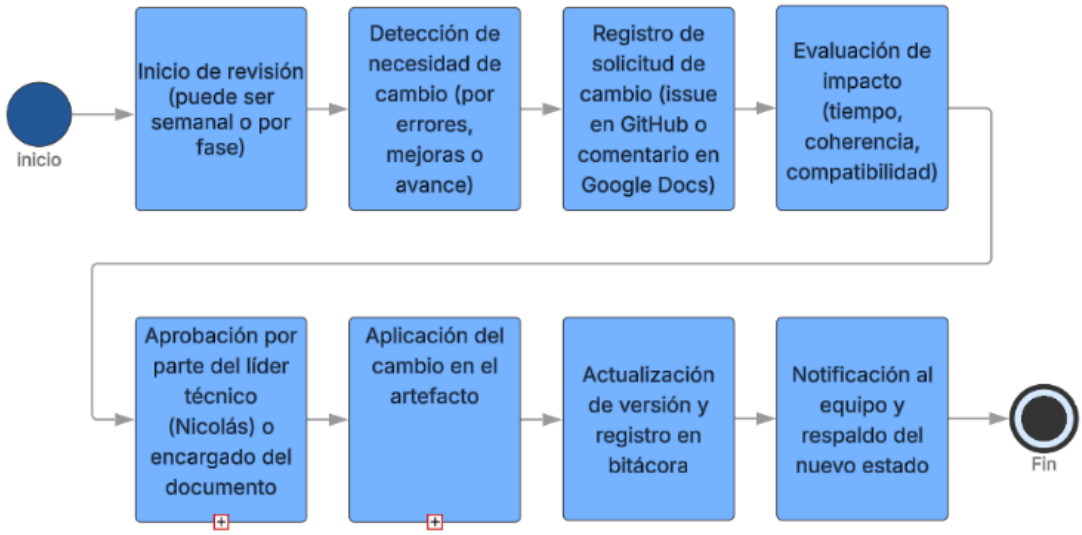

Figura 6.1: Proceso de Gestión de Cambios en la Configuración (BPMN)

6.1.4 Backup y Recuperación

Para garantizar la disponibilidad de los artefactos del proyecto ante pérdidas accidentales, se implementarán las siguientes medidas:

- **GitHub como repositorio central:** Todo el código y documentación técnica se alojará en GitHub, que ofrece redundancia automática y permite recuperar versiones previas en caso de errores.
- **Google Drive compartido:** Los datasets grandes, modelos preentrenados, documentos en edición activa y archivos binarios se almacenarán en una carpeta compartida de Google Drive con acceso restringido al equipo.
- **Backups semanales locales:** Cada integrante mantendrá una copia local actualizada del repositorio completo, sincronizada semanalmente mediante git pull.
- **Exportaciones SQL periódicas:** La base de datos PostgreSQL será exportada (dump) al finalizar cada fase técnica, con fecha y versión identificada, y almacenada en GitHub y Google Drive.

En caso de pérdida de datos, se recurrirá al historial de GitHub o a los backups de Google Drive, según corresponda. Si un integrante pierde su copia local, podrá clonar nuevamente el repositorio completo y descargar los archivos grandes desde Drive.

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

Se aplicarán los siguientes estándares de calidad técnica en el desarrollo del proyecto:

- **PEP 8 (Python):** Todo el código Python seguirá las convenciones de estilo de PEP 8, incluyendo nombres de variables descriptivos, separación clara de funciones, indentación de 4 espacios y límite de 100 caracteres por línea cuando sea posible.
- **Modularidad y reutilización:** Cada fase del pipeline será implementada como un módulo independiente con entradas y salidas claramente definidas, facilitando la depuración, el testing y la reutilización futura.

- **Documentación interna del código:** Funciones complejas deberán incluir docstrings explicativas en español, indicando propósito, parámetros de entrada, salida esperada y ejemplo de uso.
- **Manejo de errores:** Se implementarán validaciones básicas de entrada, manejo de excepciones con mensajes claros, y logging de eventos críticos para facilitar el diagnóstico de problemas.
- **Reproducibilidad técnica:** Todos los procesos estarán documentados con suficiente detalle como para que un tercero con conocimientos técnicos medios pueda replicar el pipeline completo.

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

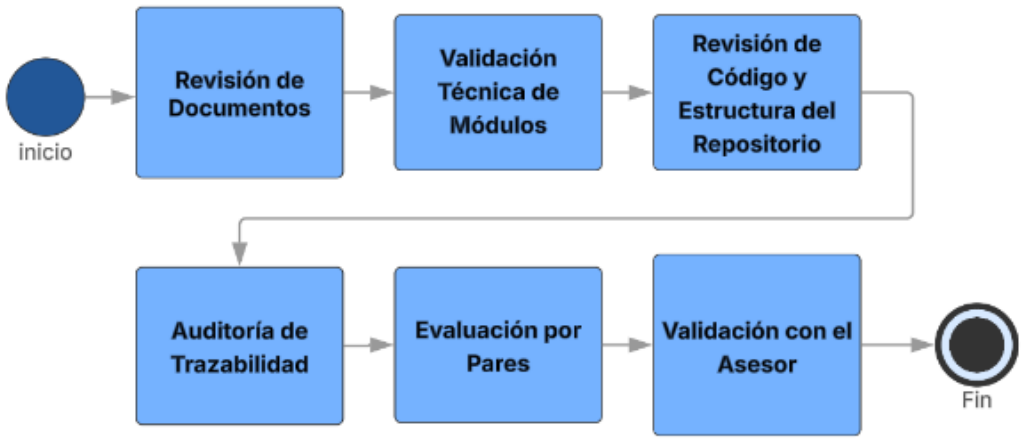

Figura 6.2: Proceso de Control de Calidad del Proyecto (BPMN)

6.2.3 Revisión Técnica de Entregables

Al finalizar cada fase metodológica, se realizará una revisión técnica formal del entregable correspondiente, siguiendo este procedimiento:

1. El responsable de la fase presenta el entregable al equipo completo en reunión semanal, explicando la implementación técnica, las decisiones tomadas y los resultados obtenidos.
2. Los demás integrantes revisan el código, prueban el módulo localmente, y formulan preguntas técnicas o identifican posibles mejoras.
3. Se redacta un checklist de verificación con criterios mínimos de aceptación (por ejemplo: "¿El código ejecuta sin errores?", "¿Los datos de salida tienen la estructura esperada?", "¿Está documentado el proceso?").
4. Si el entregable cumple todos los criterios, se aprueba y se fusiona a la rama principal. Si no, se registra una lista de correcciones pendientes y se establece un plazo para aplicarlas.
5. El asesor del proyecto revisa el entregable en la reunión quincenal de seguimiento y valida que esté alineado con los objetivos técnicos y académicos del proyecto.

6.2.4 Métricas de Calidad

Se emplearán las siguientes métricas cualitativas y cuantitativas para evaluar la calidad del sistema:

- **Cobertura de requisitos funcionales:** Porcentaje de requisitos implementados respecto al total definido en el documento SRS.

- **Tasa de errores en validación manual:** Proporción de habilidades extraídas o enriquecidas que son incorrectas o irrelevantes, evaluada sobre una muestra de 100 registros.
- **Coherencia de clusters:** Evaluación cualitativa de si los grupos de habilidades generados tienen sentido semántico y técnico, mediante inspección visual y discusión con el asesor.
- **Complejidad ciclomática moderada:** Funciones con complejidad razonable, evitando bloques de código excesivamente largos o anidados.
- **Legibilidad del código:** Evaluación subjetiva de si el código es comprensible para un desarrollador externo, verificada mediante revisión cruzada entre integrantes.

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

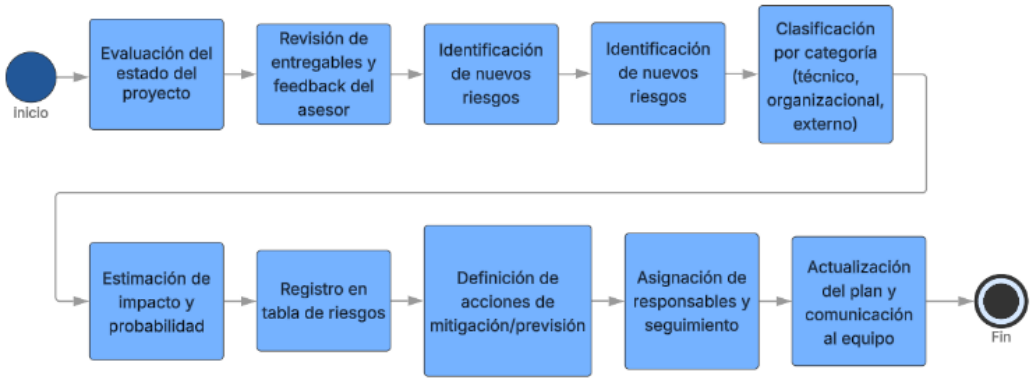

Figura 6.3: Proceso de Identificación y Gestión de Riesgos (BPMN)

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.

Los riesgos de mayor criticidad (probabilidad media-alta e impacto alto) son:

- **R01 y R02:** Problemas con el scraping, que afectarían directamente la disponibilidad de datos y podrían paralizar las fases posteriores.
- **R06 y R07:** Problemas organizacionales relacionados con disponibilidad de tiempo y cumplimiento del cronograma, que podrían generar retrasos en cascada.

Estos riesgos recibirán especial atención en las estrategias de mitigación y monitoreo continuo.

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 las siguientes actividades:

- **Revisión semanal de riesgos:** En cada reunión de seguimiento, se dedicará un espacio breve (10 minutos) para revisar si algún riesgo identificado se ha materializado, si han surgido nuevos riesgos, y si las estrategias de mitigación están funcionando.
- **Indicadores de alerta temprana:** Se monitorearán señales que indiquen la proximidad de riesgos, tales como:
  - Errores frecuentes en el scraper (indicador de R01)
  - Aumento en tiempo de respuesta de portales o CAPTCHAs (indicador de R02)
  - Bajo rendimiento en métricas de NER (indicador de R03)
  - Retrasos superiores a 3 días en tareas asignadas (indicador de R06 y R07)
- **Registro de incidentes:** Cualquier materialización de riesgo será documentada en el acta semanal, indicando el riesgo activado, la acción correctiva aplicada, el responsable y el resultado obtenido.
- **Actualización de la matriz de riesgos:** Si durante el proyecto se identifica un nuevo riesgo relevante, se agregará a la tabla de riesgos con su respectiva evaluación y estrategia de mitigación.

6.3.5 Responsabilidades en Gestión de Riesgos

- **Nicolás Camacho (Líder técnico):** Responsable de identificar y monitorear riesgos técnicos (R01–R05), proponer soluciones técnicas y coordinar la aplicación de estrategias de mitigación.
- **Alejandro Pinzón (Coordinador de proyecto):** Responsable de monitorear riesgos organizacionales (R06–R08), gestionar el cronograma, detectar desviaciones y proponer acciones correctivas.
- **Todo el equipo:** Responsable de reportar cualquier riesgo detectado, participar en la evaluación de impacto, y colaborar en la implementación de planes de contingencia cuando sea necesario.

La gestión de riesgos será un proceso continuo y colaborativo, integrado a las actividades semanales del proyecto, con el objetivo de minimizar incertidumbre y maximizar la probabilidad de éxito en la entrega final del observatorio laboral.

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