

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA



ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Optimización de la Gestión de Recursos Humanos: Asignación, Viabilidad y Análisis de Rendimiento en Proyecto

Autor:

Brais Míguez Varela

Tutores:

Eduardo Manuel Sánchez Vila

Óscar Ramos Macías

Máster Universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Julio 2025

Trabajo de Fin de Máster presentado en la Escola Técnica Superior de
Enxeñaría de la Universidade de Santiago de Compostela para la obtención del
Máster Universitario en Tecnologías de Análisis de Datos Masivos: Big Data

Agradecimientos

En primer lugar, quiero agradecer a Óscar Ramos, tutor en la empresa, por confiar en este proyecto desde el primer momento y por su acompañamiento a lo largo de todo el proceso. Su apoyo ha sido fundamental para que esta idea tomara forma y se convirtiera en una realidad.

También quiero dar las gracias a Arnau Garriga y a Nieves Casteñeda, por su paciencia infinita, por ayudarme tanto en la empresa como con la revisión del código de este proyecto, y por estar siempre dispuestos a echar una mano cuando más lo necesitaba.

A Tamara, Lucía, Sandra y Lety, gracias por estar a mi lado durante estos meses de caos, por aguantar mis momentos de agobio y por recordarme que sí era posible, incluso cuando yo lo dudaba. Vuestra amistad ha sido un apoyo constante.

A mi familia, por todo. Por su apoyo incondicional, por su comprensión, y por enseñarme a perseverar incluso en los momentos más complicados. Este logro también es vuestro.

En definitiva, gracias a todas las personas que han formado parte de este último año. Sin vosotros, este proyecto no habría sido lo mismo.

Resumen

El presente Trabajo de Fin de Máster tiene como objetivo principal diseñar e implementar un sistema para la asignación eficiente de recursos humanos a proyectos empresariales, basándose en técnicas avanzadas de análisis de datos y aprendizaje automático.

El sistema aborda un problema real de planificación en entornos dinámicos, integrando múltiples variables como habilidades del personal, carga de trabajo, tiempos de ejecución, y experiencia previa, para realizar recomendaciones personalizadas de asignación de tareas. Se parte de datos extraídos desde Jira, plataforma de gestión de proyectos, aplicando técnicas de limpieza, anonimización y enriquecimiento semántico.

El modelo de predicción se estructura en varios módulos funcionales: estimación de tiempos, clasificación de habilidades, generación de perfiles técnicos de empleados y predicción de candidatos óptimos. La arquitectura del sistema se apoya en una base de datos relacional PostgreSQL y se orquesta mediante scripts automatizados desplegados con Docker, con posibilidad de ejecución manual o periódica.

Los resultados generados se presentan en un archivo Excel con hojas estructuradas para facilitar el análisis por parte de gestores, incluyendo filtros por tareas, empleados y proyectos. La evaluación del sistema ha sido realizada en colaboración con un gestor de proyectos, validando la utilidad de las recomendaciones en un entorno real.

En las conclusiones se reconocen las limitaciones del sistema, como la ausencia de variables subjetivas (motivaciones, preferencias) y la necesidad de mejorar la granularidad en la asignación de experiencia por habilidad. Finalmente, se proponen líneas de mejora futura, como la integración de un calendario laboral, el uso de dashboards interactivos (Power BI), la expansión del conjunto de datos pre-etiquetados, y la adopción de herramientas avanzadas de orquestación como Apache Airflow.

Este proyecto representa una solución modular, escalable y basada en datos para optimizar la gestión de talento humano en organizaciones, con un enfoque práctico y orientado a la toma de decisiones estratégicas.

Índice general

1. Introducción	1
1.1. Objetivos del proyecto	2
1.2. Entregables del proyecto	2
1.3. Exclusiones del proyecto	3
1.4. Herramientas utilizadas	3
2. Especificación de Requisitos	4
2.1. Requisitos de información	4
2.2. Requisitos funcionales	7
2.3. Requisitos no funcionales	8
3. Diseño	9
3.1. Arquitectura del sistema	9
3.2. Base de datos	10
3.3. Contenedor principal	11
3.3.1. Planificador	11
3.3.2. Lectura de datos	12
3.3.3. Estimación de tiempo	13
3.3.4. Asignar habilidades a las tareas	14
3.3.5. Asignar habilidades a los empleados	16
3.3.6. Asignar tareas a los empleados	17
3.3.7. Escritura del excel	18
3.4. Tecnologías empleadas para el despliegue de la aplicación	19
3.5. Interfaz de la aplicación	19
3.6. Resultados	19
3.6.1. Presentación de resultados	19
3.6.2. Evaluación de los resultados de la asignación	21
4. Pruebas	22
4.1. Pruebas generales	22
4.2. Pruebas de la lectura de datos	23
4.3. Pruebas en la asignación de habilidades a las tareas	26

5. Conclusiones y posibles ampliaciones	28
5.1. Ampliaciones futuras	29

Índice de figuras

3.1. Estructura del sistema	9
3.2. Modelo Entidad-Relación	11
3.3. Excel de salida	21
4.1. Evidencia de superación de la P-01	23
4.2. Evidencia de superación de la P-02	24
4.3. Evidencia de superación de la P-03	25
4.4. Evidencia de superación de la P-04	25
4.5. Evidencia de superación de la P-05	26
4.6. Evidencia de superación de la P-06	27

Índice de tablas

2.1. Primer requisito de información	5
2.2. Segundo requisito de información	6
2.3. Tercer requisito de información	6
2.4. Primer requisito funcional	7
2.5. Segundo requisito funcional	7
2.6. Tercer requisito funcional	7
2.7. Cuarto requisito funcional	7
2.8. Quinto requisito funcional	8
2.9. Primer requisito no funcional	8
2.10. Segundo requisito no funcional	8
2.11. Tercer requisito no funcional	8
3.1. Resumen de Tareas, Proyectos y Empleados	20
3.2. Número de habilidades diferentes en la base de datos	20
3.3. Número de habilidades diferentes en la base de datos	20
4.1. Prueba de caída del servicio de base de datos	22
4.2. Prueba de conectividad de la base de datos	24
4.3. Prueba de error en el enlace	24
4.4. Prueba de conectividad de la base de datos	25
4.5. Prueba de no encontrar el fichero de tareas	26
4.6. Prueba de no encontrar las claves en la base de datos	27

Capítulo 1

Introducción

La gestión eficiente de los recursos humanos (RRHH) es un pilar fundamental para asegurar la productividad, la eficiencia operativa y la viabilidad de los proyectos dentro de una organización.

En un entorno empresarial dinámico y en constante transformación, las organizaciones deben enfrentarse al desafío de asignar a sus empleados de manera óptima. Esta tarea implica considerar diversos factores como las competencias individuales, la carga de trabajo existente y posibles incidencias que puedan afectar la planificación, tales como vacaciones, bajas laborales o huelgas.

El presente Trabajo de Fin de Máster (TFM) tiene como objetivo el desarrollo de un modelo que facilite una asignación eficiente del personal a los diferentes proyectos de la empresa, integrando múltiples variables que influyen en dicha gestión. Este modelo no solo permitirá optimizar el uso de los recursos humanos, sino que también contribuirá a:

- Evaluar la viabilidad de los proyectos en relación con sus fecha de inicio previstas.
- Detectar necesidades adicionales de personal.
- Identificar proyectos con bajo rendimiento o resultados no esperados.
- Analizar el desempeño de los proyectos en función de los costes asociados.

Además, el sistema posibilitará el análisis del rendimiento de los proyectos basándose en los costes asociados, proporcionando información clave para la toma de decisiones estratégicas dentro de la organización y generar los informes económicos y KPIs de rendimiento básicos de estas.

En etapas posteriores, el modelo podrá ampliarse para incluir estrategias de refuerzo en períodos críticos (como pueden ser los meses estivales), teniendo en cuenta la experiencia previa en proyectos similares y aspectos motivacionales del personal. De este modo, se busca lograr una planificación más eficaz y una mejor asignación de los recursos disponibles.

Con este enfoque integral, el proyecto aspira a mejorar la gestión de los recursos humanos y los proyectos dentro de las organizaciones, ofreciendo una herramienta que promueva la optimización operativa y contribuya a la sostenibilidad empresarial.

Los datos de entrenamiento del modelo provendrán de un caso real en constante evolución, concretamente de los registros de tareas en Jira correspondientes a varios proyectos de la empresa *FDS, a DXC Company*. No obstante, debido a la sensibilidad de los datos, estos han sido alterados en sus definiciones.

1.1. Objetivos del proyecto

Tal como se ha expuesto anteriormente, el objetivo principal de este proyecto es desarrollar un sistema de recomendación que permita identificar habilidades a partir de textos y, en base a ello, realizar recomendaciones de empleados con el fin de maximizar la eficiencia en el uso de los recursos, tanto temporales como de personal.

Este objetivo general se desglosa en los siguientes subobjetivos específicos:

1. Extraer y procesar los datos procedentes de la plataforma de gestión de tareas utilizada por la organización (en este caso, Jira).
2. Realizar estimaciones de tiempo para la ejecución de tareas, considerando tanto al empleado como al proyecto específico.
3. Asignar habilidades a las tareas identificadas a partir de la información extraída en el primer paso.
4. Determinar las habilidades de los empleados basándose en las tareas previamente realizadas.
5. Recomendar la asignación de empleados a las tareas en función del grado de adecuación entre las habilidades requeridas y las habilidades disponibles.
6. Detectar posibles ineficiencias en la gestión, tales como sobrecarga de trabajo en determinados empleados o asignaciones subóptimas de recursos.

1.2. Entregables del proyecto

Los entregables del proyecto serán los siguientes:

- La documentación del proyecto: la memoria y el código fuente. La memoria corresponde al presente documento. El código fuente estará incluido tanto en la entrega oficial como en el repositorio de GitHub asociado, accesible

en el siguiente enlace: <https://github.com/Brais-MiguezV/Optimizacion-de-la-Gestion-de-Recursos-Humanos>, lo que permitirá su clonación y despliegue en un entorno local.

1.3. Exclusiones del proyecto

Aunque el modelo desarrollado busca optimizar la asignación de empleados a tareas según criterios técnicos y operativos, hay aspectos que quedan fuera del alcance del proyecto. No se garantiza que las asignaciones sean siempre óptimas desde una perspectiva humana, pudiendo repetirse tareas similares para un mismo empleado, lo que podría afectar a su motivación y bienestar.

Tampoco se consideran factores subjetivos como aspiraciones profesionales, preferencias individuales o dinámicas de equipo, ni elementos externos como vacaciones, bajas o huelgas. Estos aspectos, aunque relevantes, se plantean como posibles mejoras en desarrollos futuros.

1.4. Herramientas utilizadas

Durante el desarrollo del proyecto se utilizaron las siguientes herramientas:

- **Visual Studio Code - Insiders** [18]: Visual Studio Code es un editor de propósito general que se puede utilizar para el desarrollo en un gran número de lenguajes y frameworks. En su versión Insiders se añaden versiones beta de los plugins, que añaden nuevas funcionalidades al editor.
- **PyCharm** [15]: editor de código desarrollado por JetBrains enfocado en el desarrollo de código utilizando Python.
- **Overleaf** [19]: editor de texto en línea que permite la edición de documentos redactados utilizando código de LaTeX.
- **Docker** [3]: aplicación que automatiza el despliegue de aplicaciones o servicios gracias al uso de contenedores.
- **Jira** [2]: herramienta de gestión de proyectos y seguimiento de tareas desarrollada por Atlassian. Permite planificar, asignar, realizar seguimiento y supervisar el progreso de proyectos, especialmente en entornos de desarrollo ágil.

Capítulo 2

Especificación de Requisitos

2.1. Requisitos de información

Un requisito de información se define como una especificación en detalle de las necesidades y las expectativas relacionadas con los datos y la información que el sistema maneja. La escala que se utiliza para clasificar la importancia de estos requisitos es:

- **Vital:** que un requisito tenga importancia vital implica que la aplicación no podrá funcionar sin él.
- **Importante:** que un requisito tenga este nivel de importancia implica que la aplicación podrá funcionar sin él, pero no de forma eficiente.

En el caso de esta aplicación solo se tienen tres requisitos de este tipo:

RQI - 01	Datos de las tareas
Descripción	El sistema debe almacenar la información relevante y necesaria asociada a cada una de las tareas que se generan en la plataforma de gestión del proyecto.

Datos específicos	<ul style="list-style-type: none"> ▪ <u>id</u>: campo que actuará a forma de identificador único. ▪ <u>clave</u>: representa el identificador único de la tarea dentro de cada proyecto. ▪ <u>fecha</u>: día y hora en el que se dió de alta la tarea. ▪ <u>timespent_real</u>: tiempo que se ha invertido en la tarea, medido en horas. ▪ <u>timespent_estimado</u>: tiempo estimado para la tarea en base a la tarea y el proyecto, medido en horas. ▪ <u>bien_estimado</u>: campo que indica si el tiempo estimado es igual al tiempo invertido con un margen del 5 %. ▪ <u>project_key</u>: campo que corresponde con el identificador del proyecto. ▪ <u>assignee</u>: campo que indica el empleado asignado para realizar la tarea. ▪ <u>status_text</u>: texto que indica el estado de la tarea. ▪ <u>issue_type</u>: campo que indica el tipo de tarea. ▪ <u>texto</u>: texto del cuerpo de la tarea que contiene la descripción. ▪ <u>candidatos</u>: campo que indica los 3 mejores empleados para realizar la tarea. ▪ <u>habilidades_extraidas</u>: campo que representa las habilidades extraidas de las tareas. ▪ <u>fecha_modificacion</u>: campo que representa la fecha y hora en la que se ha modificado algún campo de la tarea.
Importancia	Vital

Tabla 2.1: Primer requisito de información

RQI - 02	Datos de los empleados
Descripción	El sistema debe almacenar la información relevante y necesaria asociada a cada empleado para poder realizar las predicciones.

Datos específicos	<ul style="list-style-type: none"> ▪ <u>id</u>: campo que actuará a forma de identificador único. ▪ <u>codificación</u>: campo que indica la codificación del nombre del empleado. ▪ <u>empleado</u>: campo que indica el nombre del empleado. ▪ <u>habilidades</u>: campo que representa las habilidades del empleado. ▪ <u>is_active</u>: campo que representa si el empleado está activo o no. ▪ <u>fecha_modificacion</u>: campo que representa la fecha y hora en la que se ha modificado algún campo de la tarea.
Importancia	Vital

Tabla 2.2: Segundo requisito de información

RQI - 03	Datos de los proyectos
Descripción	El sistema debe almacenar la información relevante y necesaria asociada a cada proyecto para poder realizar las predicciones.
Datos específicos	<ul style="list-style-type: none"> ▪ <u>id</u>: campo que actuará a forma de identificador único. ▪ <u>codificación</u>: campo que indica la codificación del nombre del proyecto. ▪ <u>proyecto</u>: campo que indica el nombre del proyecto. ▪ <u>habilidades_necesarias</u>: campo que representa las habilidades asociadas al proyecto. ▪ <u>fecha_modificacion</u>: campo que representa la fecha y hora en la que se ha modificado la tarea.
Importancia	Importante

Tabla 2.3: Tercer requisito de información

2.2. Requisitos funcionales

Los requisitos funcionales se pueden definir como las declaraciones de los servicios que proveerá el sistema, cómo reaccionará a entradas particulares y cómo actúa en situaciones concretas. Las funcionalidades que va a tener el sistema en este proyecto son:

RF - 01	Almacenar tareas de los proyectos
Descripción	El sistema deberá guardar toda la información necesaria y procesada para sacar el mayor partido posible a partir de los datos en crudo.

Tabla 2.4: Primer requisito funcional

RF - 02	Asignar habilidades a las tareas
Descripción	El sistema deberá determinar las habilidades asociadas a cada tarea y guardarlas en la base de datos.

Tabla 2.5: Segundo requisito funcional

RF - 03	Crear perfiles de los empleados
Descripción	El sistema deberá agregar la información de las tareas realizadas por los empleados para crear así sus perfiles técnicos.

Tabla 2.6: Tercer requisito funcional

RF - 04	Asignar tareas a los empleados
Descripción	El sistema deberá asignar tareas a los empleados en función de las habilidades y de los perfiles obtenidos.

Tabla 2.7: Cuarto requisito funcional

RF - 05	Presentar resultados
Descripción	El sistema deberá guardar los datos obtenidos para que los gestores de proyectos puedan ver los resultados de una forma más clara que en base de datos.

Tabla 2.8: Quinto requisito funcional

2.3. Requisitos no funcionales

Los requisitos no funcionales se pueden definir como restricciones de los servicios o funciones que el sistema ofrece, incluyendo restricciones de tiempo, sobre los estándares, concurrencia, etc. En lo relativo a este proyecto se tienen los siguientes requisitos no funcionales:

RNF - 01	Coherencia de datos
Descripción	El sistema deberá garantizar la consistencia y congruencia de todos los datos relativos a la base de datos de las tareas, evitando estados de error.
Importancia	Vital

Tabla 2.9: Primer requisito no funcional

RNF - 02	Tamaño de la base de datos
Descripción	El sistema deberá garantizar una amplia capacidad para poder almacenar las tareas necesarias.
Importancia	Vital

Tabla 2.10: Segundo requisito no funcional

RNF - 03	Facilidad para ampliar el almacenamiento de la base de datos
Descripción	El sistema deberá garantizar que la ampliación de almacenamiento se pueda realizar de forma simple y que interrumpa lo mínimo posible al servicio.
Importancia	Alta

Tabla 2.11: Tercer requisito no funcional

Capítulo 3

Diseño

En este capítulo se detallará cómo ha sido el proceso de diseño y desarrollo, incluyendo la división de los componentes del sistema y el equipamiento software y hardware necesario para el correcto desarrollo e implementación.

3.1. Arquitectura del sistema

Los componentes del sistema se relacionan de la siguiente forma:

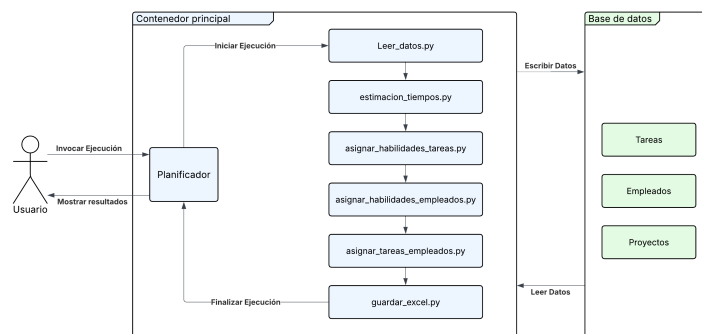


Figura 3.1: Estructura del sistema

A partir de esta figura se puede ver que el sistema se puede dividir en:

- **Base de datos:** es la parte del sistema encargada de almacenar toda la información necesaria para el correcto funcionamiento de la aplicación. Dentro de esta base de datos es necesario mencionar y explicar el proceso seguido para el diseño de las preguntas.
- **Contenedor principal:** esta parte corresponde al componente del sistema encargado de ejecutar las distintas fases del proceso de asignación, bajo la supervisión de un planificador de cargas. Una vez completada la ejecución,

el sistema genera un archivo Excel que contiene todos los datos resultantes del proceso.

A continuación, se detallarán todos los componentes para aportar más información sobre su diseño, implementación y tecnologías utilizadas en el proceso.

3.2. Base de datos

La base de datos constituye el componente responsable de almacenar toda la información necesaria tanto para el entrenamiento como para la obtención de resultados de los distintos modelos implementados en el sistema. Se trata de un elemento fundamental, ya que sin esta estructura de almacenamiento, la aplicación carecería de contenido sobre el cual operar y no podría llevar a cabo las predicciones, perdiendo así su funcionalidad principal.

Para la implementación de esta aplicación, se ha optado por utilizar una base de datos relacional mediante **PostgreSQL**, un sistema de gestión de bases de datos de código abierto, robusto y ampliamente utilizado en entornos empresariales. PostgreSQL organiza la información en tablas estructuradas mediante filas y columnas, lo que permite definir esquemas de datos consistentes y relaciones entre distintas entidades, garantizando la integridad y coherencia de la información. Las principales ventajas de utilizar este gestor son:

- **Consistencia y control de integridad:** permite definir claves primarias, foráneas, restricciones y reglas que aseguran la validez de los datos, evitando problemas de inconsistencia.
- **Lenguaje estándar (SQL):** utiliza SQL para realizar consultas, lo que facilita la interoperabilidad con otras herramientas y el mantenimiento del sistema.
- **Soporte para transacciones:** ofrece un control fiable de las operaciones mediante transacciones ACID, lo que garantiza que los cambios se apliquen de forma segura y coherente.
- **Escalabilidad y rendimiento:** está optimizado para manejar grandes volúmenes de datos y consultas complejas, con opciones avanzadas como índices, particiones y paralelismo en la ejecución.
- **Extensibilidad:** permite definir funciones personalizadas, tipos de datos propios y usar extensiones.

En el caso de la aplicación y ante los requisitos de información mostrados previamente se ha decidido generar la base con la siguiente estructura:

- En el caso del requisito RQI - 01 (tabla 2.1), se ha creado una tabla denominada **tareas** en el que se guardará toda la información asociadas a ellas y serán identificadas por el atributo **clave**.
- En el caso del requisito RQI - 02 (tabla 2.2), se ha creado una tabla denominada **empleados** en el que se guardará la información asociada a cada una de las personas que trabajan en los diferentes proyectos y se identificarán por el campo **codificacion**.
- En el caso del requisito RQI - 03 (tabla 2.3), se ha creado una tabla denominada **proyectos** en el que se guardará la información asociada a cada uno de los proyectos de los que se extraen tareas y se identificarán por el campo **codificacion**.

A partir de lo descrito en los apartados anteriores, se deriva el siguiente esquema entidad-relación para la base de datos, el cual refleja la estructura lógica de las entidades involucradas y sus respectivas relaciones:

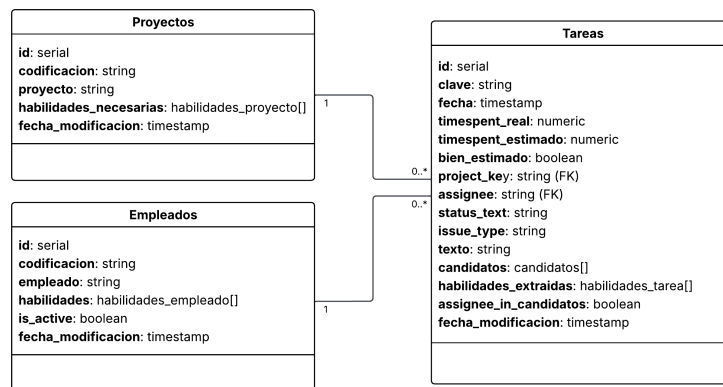


Figura 3.2: Modelo Entidad-Relación

3.3. Contenedor principal

En esta sección se describen los distintos componentes que intervienen en el proceso completo de asignación, detallando su funcionalidad y el papel que desempeñan dentro del sistema.

3.3.1. Planificador

El planificador es el componente responsable de coordinar y gestionar la ejecución de las distintas etapas del sistema de asignación. Su función principal

consiste en orquestar de forma ordenada los módulos que intervienen en el proceso, garantizando que cada uno se ejecute en el momento adecuado y con los datos correspondientes.

Para ello, el planificador emplea una estrategia de ejecución secuencial, invocando de manera estructurada los distintos scripts que conforman el sistema. Esta secuencia está diseñada para respetar la dependencia lógica entre los módulos, asegurando que los resultados generados por una fase estén disponibles para la siguiente.

Además, este componente permite controlar el flujo completo del proceso, facilitando tanto la trazabilidad como la posibilidad de reiniciar la ejecución en caso de errores o modificaciones parciales. Su correcto funcionamiento es clave para garantizar la coherencia y eficiencia del sistema en su conjunto.

Una de las herramientas que se han investigado para realizar esto es **Tivoli Workload Scheduler (TWS)** [12]. TWS es una herramienta de planificación y automatización de cargas de trabajo desarrollada por IBM. Su objetivo principal es coordinar y optimizar la ejecución de procesos y tareas en entornos informáticos complejos, facilitando la gestión eficiente de recursos y la reducción de intervenciones manuales. TWS permite definir flujos de trabajo mediante calendarios, dependencias y reglas de ejecución, adaptándose tanto a entornos locales como distribuidos o en la nube. Gracias a su capacidad de integración con diversas plataformas, aplicaciones y sistemas operativos, se convierte en una solución robusta para organizaciones que requieren una alta fiabilidad en la programación de procesos críticos. Además, ofrece funcionalidades avanzadas de monitoreo, alertas y recuperación ante errores, lo que mejora la visibilidad y el control operativo del sistema.

Finalmente, se ha optado por una implementación más sencilla mediante un script en Bash. Esta decisión se debe a que el sistema está diseñado para ejecutar un único proceso de carga y asignación de manera secuencial, por lo que no resulta justificable, en términos de inversión temporal y complejidad, la incorporación de una solución más avanzada como Tivoli Workload Scheduler. Esta aproximación permite mantener la funcionalidad requerida con un menor coste de desarrollo y mantenimiento. Esta parte se corresponde con el fichero `ejecución_total.sh`, disponible en el repositorio mencionado al inicio de esta memoria.

3.3.2. Lectura de datos

Esta parte del sistema está dedicada a la obtención de datos directamente desde la API de Jira [1], capturando la información en su formato original. A continuación, se lleva a cabo un primer proceso de preprocesamiento cuyo objetivo es filtrar y seleccionar aquellos elementos que resultan relevantes para las etapas posteriores de análisis y predicción. Este filtrado inicial permite reducir la complejidad de los datos y centrar el modelo en las variables más significativas para la toma de decisiones.

Para la obtención de los datos, se emplea la librería `requests` de Python [17], la cual facilita el proceso de conexión, autenticación y envío de solicitudes HTTP a servicios web. En el contexto de esta aplicación, es necesario proporcionar tanto la URL del proyecto en Jira como un token de autenticación válido, requisito indispensable para que la API devuelva una respuesta autorizada.

Los datos recuperados desde la API se presentan en formato JSON, aunque sin una estructura fija predefinida, lo que ha supuesto un reto adicional durante la fase de preprocesamiento. Esta falta de uniformidad ha requerido un tratamiento específico para extraer de forma coherente la información relevante que será utilizada en las etapas posteriores del sistema.

Los datos recuperados incluyen numerosos campos que no son necesarios para los objetivos del modelo, tales como las versiones del software, las imágenes de perfil de los usuarios, la zona horaria, entre otros. Estos elementos han sido descartados durante el proceso de limpieza y selección, con el fin de reducir la complejidad del conjunto de datos y centrarse exclusivamente en la información útil para el sistema de predicción y asignación.

Asimismo, durante esta fase se lleva a cabo un proceso de anonimización de las tareas, con el objetivo de garantizar el cumplimiento de las normativas vigentes en materia de protección de datos y privacidad. Esta medida asegura que la información tratada no comprometa la identidad de los usuarios ni la confidencialidad de los proyectos involucrados.

Finalmente, los datos procesados se almacenan en las tablas correspondientes de la base de datos, donde quedan estructurados y listos para su utilización. Los módulos posteriores del sistema acceden directamente a estas tablas para llevar a cabo las tareas de asignación, garantizando así una integración fluida y eficiente entre las distintas fases del proceso.

3.3.3. Estimación de tiempo

En esta fase del proyecto se lleva a cabo la estimación del tiempo requerido para completar las tareas. Esta estimación resulta especialmente relevante, ya que constituye un factor clave para la gestión equilibrada de la carga de trabajo entre los distintos empleados. Una correcta valoración temporal permite optimizar la planificación y mejorar la eficiencia en la asignación de recursos.

En la primera fase del proceso, las tareas se agrupan según el proyecto al que pertenecen. A continuación, se analiza cada tarea individualmente dentro de su respectivo proyecto. Si no existen tareas previas asociadas al mismo, el sistema omite dicho proyecto y continúa con el siguiente.

Cuando el empleado asignado a una tarea ya ha participado anteriormente en ese proyecto, la estimación del tiempo se calcula como la media de duración de todas las tareas que dicho empleado ha realizado en el mismo. En caso de que el empleado no haya trabajado previamente en el proyecto o no tenga tareas, la

estimación se basa en la media de duración de todas las tareas del proyecto en general.

Para las tareas aún pendientes de asignación, se utiliza igualmente la media de duración de las tareas del proyecto correspondiente como valor estimado, garantizando así una referencia coherente en ausencia de datos específicos por empleado.

Finalmente, se incorpora el campo `bien_estimado`, una variable booleana que indica si la estimación temporal realizada para una tarea se ajusta al tiempo real empleado, considerando un margen de error del 5 %.

3.3.4. Asignar habilidades a las tareas

En esta fase se emplea el primer modelo desarrollado con el objetivo de asignar habilidades a las tareas de los distintos proyectos. Para ello, se parte de un conjunto inicial de tareas previamente etiquetadas manualmente con las habilidades correspondientes, que sirven como base de aprendizaje. A partir de esta información, el modelo infiere y asigna las habilidades más adecuadas al resto de tareas, permitiendo así enriquecer el conjunto de datos con información relevante para las fases posteriores del sistema.

Las tareas pre-etiquetadas disponibles se presentan únicamente con su identificador y un array con las habilidades asociadas, por lo que es necesario realizar una consulta adicional a la base de datos para recuperar el contenido completo de cada tarea, específicamente la descripción textual.

Dado que el número de tareas pre-etiquetadas es limitado (con un total de solo 25 registros), se ha optado por aplicar técnicas de aumento de datos para ampliar el corpus de entrenamiento del modelo. Para ello, se utiliza la librería `nlpaug` [4], que permite realizar modificaciones controladas sobre los textos, tales como la sustitución por sinónimos, la inserción de ruido léxico o la reorganización de palabras. Estas técnicas contribuyen a generar nuevas muestras a partir de las existentes, mejorando la robustez y capacidad de generalización del modelo.

Para llevar a cabo la asignación de habilidades a cada tarea, se emplea un clasificador del tipo `OneVsRestClassifier`. Este enfoque consiste en entrenar un estimador independiente para cada una de las posibles habilidades, permitiendo que cada modelo se especialice en predecir la presencia o ausencia de una habilidad específica. Como resultado, el sistema genera un conjunto de salidas binarias (correspondientes a “sí” o “no”) que indican, para cada habilidad, si debe ser asignada a la tarea en cuestión. Finalmente, se asigna el conjunto de habilidades cuya predicción haya sido afirmativa. Este enfoque permite abordar el problema como una clasificación multietiqueta, ofreciendo una solución flexible y escalable.

Con el objetivo de llevar a cabo una comparación rigurosa entre distintos modelos, se ha desarrollado el notebook `Eleccion.de.modelos.ipynb`, disponible en el repositorio del proyecto. Este archivo permite evaluar el rendimiento de los modelos considerados a partir de un conjunto de métricas comunes, facilitando así

la selección de la opción más adecuada para el problema planteado. Los modelos comparados son los siguientes:

- **Logistic Regression:** el modelo de regresión logística es un clasificador lineal ampliamente utilizado para problemas de clasificación binaria y multiclase. Estima la probabilidad de pertenencia a una clase a través de la función sigmoide, lo que permite interpretar los resultados en términos probabilísticos. En este caso, se ha configurado con un número máximo de iteraciones de 1000 para asegurar la convergencia del modelo en conjuntos de datos más complejos [13][22].
- **Random Forest:** el clasificador Random Forest es un algoritmo de ensamblado basado en árboles de decisión, que combina múltiples árboles entrenados sobre subconjuntos del conjunto de entrenamiento mediante “bagging”. Esto permite mejorar la precisión y reducir el sobreajuste. Es especialmente útil en tareas donde existen relaciones no lineales y características irrelevantes [14][21].
- **Multinomial Naive Bayes:** el modelo Multinomial Naive Bayes se basa en el teorema de Bayes y está especialmente adaptado para datos discretos, como los conteos de palabras en textos. Es eficiente, rápido y suele ser competitivo en tareas de clasificación de texto, como análisis de sentimientos o filtrado de spam, a pesar de su fuerte suposición de independencia entre características [7][23].
- **Linear Support Vector Classifier:** LinearSVC es una implementación eficiente del clasificador de Máquinas de Vectores de Soporte (SVM) lineales, optimizado para tareas de clasificación de gran escala. A diferencia del algoritmo SVM estándar que utiliza un enfoque dual, LinearSVC utiliza una formulación primal basada en descenso del gradiente, lo que mejora la velocidad en grandes volúmenes de datos [9][24].
- **BERT for Sequence Classification:** este modelo se basa en la arquitectura BERT, optimizada aquí para tareas de clasificación multietiqueta. Se carga desde un directorio local y se ajusta al número de etiquetas del problema mediante el parámetro `problem_type`. Gracias a su capacidad de entender el contexto bidireccional del lenguaje, BERT ofrece un rendimiento robusto en tareas de clasificación de texto [5][10][11].

Dado que los modelos tradicionales como Regresión Logística, Random Forest, MultinomialNB y LinearSVC no pueden procesar directamente texto en formato natural, es necesario convertir las entradas textuales en representaciones numéricas. Para ello, se emplea la técnica de vectorización TF-IDF mediante `TfidfVectorizer`, que transforma los textos en vectores de características ponderadas según la frecuencia de los términos en los documentos y en el corpus

completo [16]. Esta transformación permite que los algoritmos clásicos trabajen con texto de forma efectiva.

Por otro lado, el modelo basado en BERT requiere un tokenizador específico, `BertTokenizer`, que divide el texto en subpalabras y asigna identificadores compatibles con la arquitectura del modelo. Este tokenizador preserva el contexto y la semántica de los textos de entrada, lo cual es esencial para aprovechar al máximo la capacidad de representación contextual de BERT [25].

Además, para garantizar un entrenamiento equitativo y una evaluación más robusta del rendimiento de los modelos, se ha aplicado validación cruzada con *MultilabelStratifiedKfold*. Este método divide el conjunto de datos en k particiones o “folds” manteniendo la proporción de clases en cada subconjunto, lo cual es especialmente importante en problemas de clasificación desequilibrada. De este modo, cada modelo se entrena y evalúa varias veces sobre distintas combinaciones de datos, lo que reduce el sesgo y mejora la generalización del modelo [27].

Tras llevar a cabo las comparativas entre los distintos modelos, se ha seleccionado el algoritmo `RandomForestClassifier`, al ser el que mejor ha capturado las características relevantes del conjunto de datos y ha ofrecido un rendimiento más sólido en las métricas evaluadas.

Una vez identificadas las habilidades asociadas a cada tarea, se incorpora el campo *experiencia*, estimado a partir del tiempo invertido en su realización. En la versión actual, todas las habilidades reciben el mismo nivel de experiencia, asumiendo una distribución homogénea del esfuerzo entre ellas. Como posible línea de mejora, que se abordará en las conclusiones, se propone introducir ponderaciones diferenciadas para reflejar con mayor precisión el grado de implicación de cada habilidad en función de la tarea. En la base de datos, esta lógica se materializa mediante una estructura específica para el almacenamiento de habilidades. Cada entrada correspondiente a una habilidad incluirá tres elementos clave: el nombre de la habilidad, el nivel de experiencia obtenido a partir de la ejecución de una tarea concreta y la fecha en la que se actualizó dicha información.

3.3.5. Asignar habilidades a los empleados

La siguiente etapa del flujo de datos consiste en recuperar nuevamente todas las tareas almacenadas en la base de datos y agruparlas según el empleado que las haya realizado. A partir de esta agrupación, se generan los campos correspondientes a las habilidades identificadas, así como una estimación de su nivel, valorado en una escala del 1 al 10.

Para determinar el nivel, se combinan dos factores principales: la experiencia acumulada en cada habilidad (suma del tiempo invertido en cada habilidad) y la antigüedad del empleado desde su primera tarea registrada. A partir de estos valores, se calcula un nivel numérico normalizado en una escala de 1 a 10, que permite comparar el grado de dominio entre diferentes empleados y habilidades. El resultado final es una estructura de datos que puede ser almacenada y consul-

tada fácilmente en el sistema, y que será utilizada posteriormente para mejorar los procesos de asignación y recomendación dentro del modelo. Desde un enfoque lógico, se asume que, a mayor tiempo de permanencia en la empresa, mayor será el nivel de competencia adquirido por el empleado, como resultado de la experiencia acumulada en el desempeño de sus funciones.

Asimismo, se incorpora el campo `is_active`, cuya finalidad es indicar si un empleado se encuentra actualmente activo en los proyectos, es decir, si ha realizado tareas en un periodo reciente. En concreto, si un empleado no ha registrado actividad en un plazo superior a dos meses, se considera inactivo, y el valor de este campo se actualiza en consecuencia. Esta información resulta útil para evitar asignaciones a empleados que no estén disponibles de forma efectiva.

3.3.6. Asignar tareas a los empleados

Este módulo implementa un sistema de recomendación orientado a predecir, a partir del análisis de tareas históricas y del perfil de los empleados, cuáles son los candidatos más adecuados para asumir nuevas tareas. Para ello, se integran técnicas de procesamiento de lenguaje natural, ingeniería de características y aprendizaje automático.

El proceso comienza con la extracción de datos relevantes desde una base de datos PostgreSQL. Estos datos incluyen información sobre las tareas realizadas, las habilidades de los empleados y su antigüedad en la organización. A partir de esta información, se normaliza la antigüedad y se calculan métricas de similitud entre las habilidades requeridas por cada tarea y las habilidades declaradas por los empleados, lo que constituye una base sólida para el cálculo de afinidad entre perfiles y tareas.

En cuanto al tratamiento del texto asociado a cada tarea, se emplea el modelo TF-IDF para convertir las descripciones en vectores numéricos de alta dimensión. No obstante, este tipo de representación puede ser redundante y contener ruido, ya que asigna peso a todos los términos del corpus sin distinción de relevancia contextual. Para abordar esta limitación, se aplica la técnica de Singular Value Decomposition (SVD), que permite reducir la dimensionalidad del espacio vectorial manteniendo la mayor parte de la información significativa. SVD descompone la matriz TF-IDF en componentes principales que capturan las principales variaciones semánticas del texto, seleccionando únicamente aquellos de mayor valor informativo [8]. Esto da lugar a representaciones vectoriales más compactas y eficientes, lo que no solo mejora el rendimiento computacional, sino que también reduce el riesgo de sobreajuste y mejora la capacidad de generalización del modelo.

Estas representaciones textuales, junto con variables adicionales como el número de habilidades implicadas, el estado de la tarea y la antigüedad del empleado, conforman el conjunto final de características utilizadas para entrenar un modelo de regresión. Para esta tarea, se ha empleado el algoritmo Gradient Boosting,

una técnica de aprendizaje supervisado basada en ensamblado (ensemble learning) que construye un modelo robusto a partir de la combinación secuencial de modelos débiles, típicamente árboles de decisión.

Gradient Boosting optimiza la predicción corrigiendo iterativamente los errores residuales de los modelos anteriores. En cada etapa, un nuevo árbol se ajusta para predecir el gradiente negativo de la función de pérdida, lo que permite refinar el modelo progresivamente. El resultado final es una suma ponderada de todos los árboles generados, capaz de capturar relaciones complejas y no lineales en los datos.

Este algoritmo destaca por su precisión y flexibilidad en tareas de regresión y clasificación, así como por su resiliencia ante datos heterogéneos y valores atípicos. Sin embargo, su rendimiento depende en gran medida de una correcta configuración de hiperparámetros, como la tasa de aprendizaje (learning rate), la profundidad de los árboles, el número de iteraciones (n_estimators) y los parámetros de regularización [6].

Una vez entrenado, el modelo permite predecir los empleados más adecuados para cada tarea, generando un ranking con los tres candidatos más prometedores según su puntuación estimada, bajo la limitación de que los empleados no pueden superar las 160 horas mensuales. Estos resultados se registran en la base de datos, actualizando las tareas con los empleados recomendados y la correspondiente fecha de modificación.

Este enfoque permite automatizar el proceso de asignación de tareas, contribuyendo a una gestión más eficiente y objetiva de los recursos humanos. Al basarse en datos históricos y criterios cuantificables, el sistema proporciona soporte a la toma de decisiones estratégicas dentro de la organización.

3.3.7. Escritura del excel

Finalmente, con el objetivo de facilitar la interpretación y análisis de los datos y resultados generados, se presenta la información en un formato más accesible que el almacenamiento directo en base de datos. Para ello, se genera un archivo Excel estructurado en cuatro hojas principales:

- **Tareas del mes actual:** contiene las tareas planificadas o ejecutadas en el periodo en curso.
- **Lista de empleados:** recoge información relevante sobre los empleados disponibles, incluyendo sus habilidades.
- **Lista de proyectos:** ofrece una visión general de los proyectos activos o registrados en el sistema.
- **Histórico de tareas:** incluye el conjunto completo de tareas disponibles, permitiendo su análisis longitudinal.

Este formato permite una consulta más cómoda por parte de los responsables de gestión, además de facilitar la incorporación de funcionalidades adicionales como filtros por proyecto, empleado o habilidades, entre otros, mejorando así la utilidad práctica del sistema para la toma de decisiones.

3.4. Tecnologías empleadas para el despliegue de la aplicación

Todas las partes del sistema mencionadas anteriormente, se despliegan utilizando contenedores de Docker. Para poder levantar el servicio se utiliza `docker-compose`, que permite la orquestación de contenedores para un fin concreto.

3.5. Interfaz de la aplicación

En este caso, la aplicación no cuenta con una interfaz gráfica convencional, ya que no está diseñada para ser utilizada mediante interacción visual directa por parte del usuario. Su funcionamiento se orienta principalmente a la automatización de procesos en segundo plano, sin requerir intervención manual a través de un entorno gráfico.

Los principales métodos de interacción con la aplicación son los siguientes:

- **Ejecución automática:** se contempla la utilización de un mecanismo de planificación temporal, como `crontab`, que permita ejecutar la aplicación de forma periódica (diaria, semanal, etc.). Esta modalidad representa el principal método de interacción, y está pensada para generar automáticamente registros de actividad (logs), que puedan ser analizados posteriormente para evaluar el comportamiento y rendimiento del sistema.
- **Ejecución manual:** se prevé la posibilidad de que determinados usuarios, como gestores de proyectos, puedan forzar la ejecución del sistema en momentos concretos, de manera independiente a la planificación automática. Esta opción resulta útil para casos excepcionales o pruebas puntuales de funcionamiento.

3.6. Resultados

3.6.1. Presentación de resultados

Tras la ejecución manual del sistema realizada el día 26/06/2025, se ha podido verificar en la base de datos la correcta inserción de los resultados generados, tal como se muestra en la siguiente figura.

Indicador	Cantidad
Total Tareas	19,320
Tareas con Habilidades	15,129
Tareas con Habilidades y Candidatos	11,181
Proyectos	23
Empleados	100

Tabla 3.1: Resumen de Tareas, Proyectos y Empleados

En esta ejecución, se han cargado un total de 19.320 tareas correspondientes a los 23 proyectos definidos en el proceso. De este conjunto, se han asignado habilidades a 15.129 tareas (filtrando las épicas, ya que no aportan información para la asignación), lo que representa un 78,31 % del total. A su vez, de las tareas con habilidades asignadas, 11.181 cuentan con una predicción del top 3 de candidatos más adecuados para su ejecución. Esto equivale al 73,90 % de las tareas con habilidades y al 57,87 % del total de tareas procesadas.

Estos resultados reflejan un correcto funcionamiento del sistema de asignación, tanto en la extracción de habilidades como en la recomendación de perfiles, dentro de los parámetros establecidos para esta fase de prueba.

Indicador	Cantidad
Número de habilidades diferentes	7

Tabla 3.2: Número de habilidades diferentes en la base de datos

Las tareas procesadas han sido etiquetadas con un total de siete habilidades distintas, entre las cuales se incluyen, entre otras, *SQL*, *Desarrollo* y *Reclamaciones_correo*. Este conjunto de etiquetas refleja la diversidad de competencias requeridas en los distintos proyectos analizados durante la ejecución.

Otra de las funcionalidades del sistema que debe ser evaluada es la estimación del tiempo requerido para la ejecución de las tareas, con el fin de determinar si el enfoque utilizado proporciona resultados adecuados.

Indicador	Cantidad
Bien estimado	508
Mal estimado	15024

Tabla 3.3: Número de habilidades diferentes en la base de datos

Para la elaboración de la Tabla 3.3, se han excluido aquellas tareas cuyo campo `timespent_real` presenta un valor igual a cero, ya que no permiten validar la precisión de las estimaciones. A partir del análisis visual de los resultados, se observa que, en la mayoría de los casos, el uso de medias como método de

estimación no proporciona una aproximación suficientemente precisa del tiempo necesario para completar las tareas. Por tanto, se considera conveniente explorar métodos alternativos que puedan mejorar la capacidad predictiva en esta dimensión del sistema.

Como se ha comentado previamente, los resultados de la ejecución se vuelcan en el excel. Un ejemplo extraído y anonimizado de este excel es el siguiente:

	D	E	I	J	K	L	M	N	P
	timespent_real	timespent_estimado	nombre_assignee	status_text	issue_type	nombre_candidato	nivel_candidato	habilidad	fecha_modificacion
19097	1	1.086956522	Brais Miguez	In Progress	Sub-task	Empleado1	7.514533719118628	SQL	2025-06-26 21:12:46
19122	1	1.086956522	Brais Miguez	Resolved	Sub-task	Empleado2	7.514533719118628	SQL	2025-06-26 21:12:46
23077	0	10.375	Brais Miguez	Open	Task	Empleado3	6.85	Desarrollo	2025-06-26 21:12:46
23080	2	10.375	Brais Miguez	Closed	Task	Empleado4	6.85	Desarrollo	2025-06-26 21:12:46
23424	0	0	Brais Miguez	To Do	Epic	Empleado5	2.18083685733363	Desarrollo	2025-06-26 21:12:46

Figura 3.3: Excel de salida

En la captura se pueden observar algunos de los campos descritos previamente, incluyendo el empleado recomendado para cada tarea (en este caso, se muestra únicamente un candidato) junto con el nivel asignado en función de su adecuación. Asimismo, se ha implementado un sistema de codificación por colores basado en el estado de cada tarea, lo que permite una visualización más clara y diferenciada de las distintas fases del proceso. Esta representación facilita la interpretación rápida del estado general del conjunto de tareas.

3.6.2. Evaluación de los resultados de la asignación

En este caso particular, no resulta sencillo establecer métricas objetivas que permitan evaluar de forma automática si los empleados recomendados son realmente los más adecuados para cada tarea. Debido a esta limitación, la validación de los resultados se ha llevado a cabo mediante la presentación de las asignaciones al cliente (en este caso, un gestor de proyectos), quien ha valorado la adecuación de las mismas en función de su conocimiento experto y del contexto organizativo.

A partir de estas sesiones de revisión, se acordó seleccionar un subconjunto representativo de datos correspondiente a las tareas registradas durante el mes de junio. Sobre este conjunto específico, el cliente ha confirmado que las asignaciones generadas por el sistema son coherentes y apropiadas en la mayoría de los casos, validando así la eficacia del modelo en un entorno real.

Capítulo 4

Pruebas

Este capítulo se centra en la descripción y ejecución de las pruebas realizadas antes del despliegue final del sistema, con el objetivo de verificar su comportamiento frente a posibles errores o situaciones no contempladas. Dado que muchas de estas pruebas son comunes a varios de los scripts que conforman el proyecto, se agruparán y ejecutarán de forma conjunta en un único entorno de prueba. No obstante, aquellas pruebas específicas que afecten únicamente a un componente concreto del sistema se presentarán de forma individual en secciones separadas, permitiendo un análisis detallado de cada caso particular.

4.1. Pruebas generales

La primera prueba general tiene como objetivo evaluar la capacidad del sistema para gestionar situaciones en las que se produce una caída de la base de datos durante la ejecución de alguno de sus componentes. Este escenario permite comprobar el nivel de tolerancia a fallos y la gestión de errores críticos, así como garantizar que el sistema responde de forma controlada ante la pérdida temporal de conectividad con la base de datos.

P - 01	Prueba de caída de la base de datos
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que el servicio de la base de datos deje de estar disponible durante la ejecución.
Metodología de la prueba	La prueba consistirá en forzar la caída de la base de datos y ver cómo el sistema reacciona.
Salida esperada	Se espera que el sistema informe del error y finalice la ejecución.
Estado	Superada

Tabla 4.1: Prueba de caída del servicio de base de datos

```

+-----+
| Ejecutando script leer_datos.py |
+-----+

Faltan una o más tablas. Ejecutando script 'Tablas.sql'...

✓ Tablas creadas desde 'Tablas.sql'.

Procesando 1 proyectos de Jira...
  Procesando proyecto 1: https://bisdev.jira.com/rest/api/3/search?jql=project=BISDEVMAN

✓ Datos insertados correctamente en la base de datos.

+-----+
| Ejecutando script estimacion_tiempos.py |
+-----+

✓ Datos de tareas y empleados cargados correctamente
✓ Datos de tareas y empleados cargados correctamente

Estimando tiempos para tareas
  Estimando tiempos para el proyecto cbb8b62c2587f8b800b2901e0921a7026da0c3ebae849cd5138440d8b710f426 (1/1)

✓ Cambios efectuados en base de datos

+-----+
| Ejecutando script asignar_habilidades_tareas.py |
+-----+

✗ Error al conectar con la base de datos: (psycopg2.OperationalError) could not translate host name "postgres" to address:
Name or service not known

(Background on this error at: https://sqlalche.me/e/20/e3q8)

+-----+
| Error en asignar_habilidades_tareas.py. Abortando ejecución. |
+-----+

```

Figura 4.1: Evidencia de superación de la P-01

Esta prueba arrojará los mismos resultados previamente descritos, independientemente del momento exacto en el que se produzca la caída de la base de datos durante la ejecución del sistema.

4.2. Pruebas de la lectura de datos

Durante la ejecución de este módulo, pueden surgir diversos errores que comprometen el correcto funcionamiento del sistema. A continuación, se enumeran los tres principales escenarios de fallo identificados:

- La base de datos no está creada, por lo que no se pueden insertar los datos.
- El enlace no es correcto y, por tanto, no se pueden leer datos.
- El proyecto del que se han intentado obtener las tareas no existe o el usuario que hace la invocación no tiene permisos para verlas.

P - 02	Prueba de existencia de la base de datos
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que se intenten guardar datos en la base cuando no está creada.
Metodología de la prueba	La prueba consistirá en ver la salida de la ejecución con la base de datos no creada, comprobando la reacción.
Salida esperada	Se espera que el sistema genere de forma automática la base de datos y cree las tablas que son necesarias para la correcta ejecución.
Estado	Superada

Tabla 4.2: Prueba de conectividad de la base de datos

```

root@5909cc597a0b:/app# ./ejecucion_total.sh

+-----+
| Ejecutando script leer_datos.py |
+-----+

Faltan una o más tablas. Ejecutando script 'Tablas.sql'...

Tablas creadas desde Tablas.sql.

```

Figura 4.2: Evidencia de superación de la P-02

P - 03	Prueba de error en el enlace de conexión
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que el enlace de la API de la que extraer tareas no sea correcto.
Metodología de la prueba	La prueba consistirá en forzar al sistema a utilizar una cadena que no es correcta
Salida esperada	Se espera que el sistema informe del error y finalice la ejecución.
Estado	Superada

Tabla 4.3: Prueba de error en el enlace


```

root@5909cc597a0b:/app# ./ejecucion_total.sh

+-----+
| Ejecutando script leer_datos.py |
+-----+

Las tablas ya existen.

🔍 Procesando 1 proyectos de Jira...
  📄 Procesando proyecto 1: https://error.jira.com/rest/api/3/search?jql=project=BISDEVMAN
    Error: 404, {"errorMessage": "Site temporarily unavailable", "errorCode": "OTHER"}

❌ Se han producido demasiados errores, abortando la ejecución.

+-----+
| Error en leer_datos.py. Abortando ejecución. |
+-----+

```

Figura 4.3: Evidencia de superación de la P-03

P - 04	Prueba de obtener tareas de un proyecto que no existe
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que el proyecto del que extraer tareas no exista o que el usuario invocador no tenga acceso.
Metodología de la prueba	La prueba consistirá en forzar al sistema a utilizar un proyecto que no existe.
Salida esperada	Se espera que el sistema informe del error y finalice la ejecución.
Estado	Superada

Tabla 4.4: Prueba de conectividad de la base de datos

```

+-----+
| Ejecutando script leer_datos.py |
+-----+

Las tablas ya existen.

🔍 Procesando 1 proyectos de Jira...
  📄 Procesando proyecto 1: https://bisdev.jira.com/rest/api/3/search?jql=project=ERROR
    Error: 400, {"errorMessages":["The value 'ERROR' does not exist for the field 'project'."], "warningMessages": []}

❌ Se han producido demasiados errores, abortando la ejecución.

+-----+
| Error en leer_datos.py. Abortando ejecución. |
+-----+

```

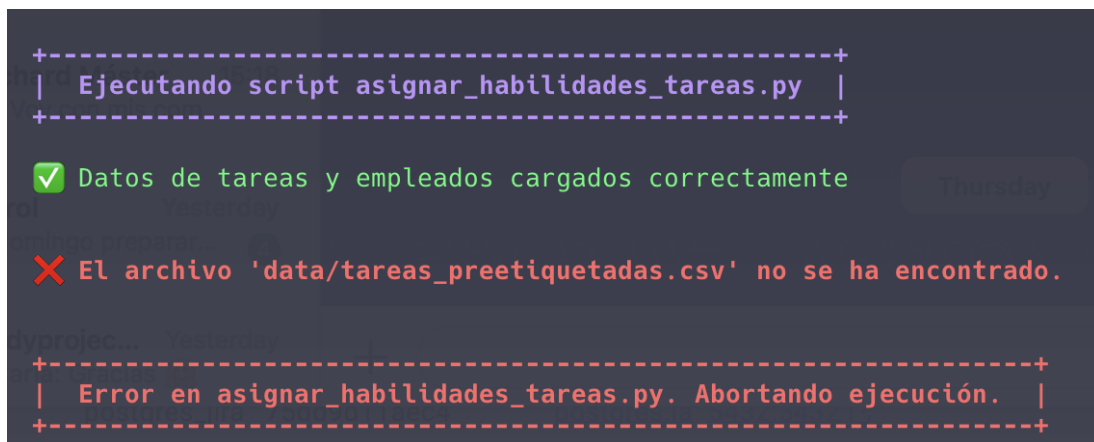
Figura 4.4: Evidencia de superación de la P-04

4.3. Pruebas en la asignación de habilidades a las tareas

El principal error asociado a esta fase consiste en que las tareas pre-etiquetadas no se encuentren registradas en la base de datos o que el archivo que las contiene no esté disponible en el sistema. Ambos casos impiden la correcta ejecución del proceso de entrenamiento y asignación de habilidades, al no contar con el conjunto de referencia necesario.

P - 05	Prueba de que no se encuentre el fichero de tareas pre-etiquetadas
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que el fichero necesario para entrenar el modelo, no se encuentre.
Metodología de la prueba	La prueba consistirá en eliminar el archivo y ver cómo reacciona el sistema.
Salida esperada	Se espera que el sistema informe del error y finalice la ejecución.
Estado	Superada

Tabla 4.5: Prueba de no encontrar el fichero de tareas



```

+-----+
| Ejecutando script asignar_habilidades_tareas.py |
+-----+

✅ Datos de tareas y empleados cargados correctamente Thursday

❌ El archivo 'data/tareas_preetiquetadas.csv' no se ha encontrado.

+-----+
| Error en asignar_habilidades_tareas.py. Abortando ejecución. |
+-----+

```

Figura 4.5: Evidencia de superación de la P-05

P - 06	Prueba de que las tareas pre-etiquetadas no estén en la base de datos
Descripción	Esta prueba consistirá en comprobar como reacciona el sistema en caso de que intente asignar las habilidades pre-etiquetadas a tareas que no se encuentren en la base de datos.
Metodología de la prueba	La prueba consistirá en no cargar las claves de las tareas y ver cómo reacciona el sistema.
Salida esperada	Se espera que el sistema informe del error y finalice la ejecución.
Estado	Superada

Tabla 4.6: Prueba de no encontrar las claves en la base de datos

```

+-----+
| Ejecutando script asignar_habilidades_tareas.py |
+-----+

✓ Datos de tareas y empleados cargados correctamente

✗ No se han encontrado todas las claves en la base de datos

+-----+
| Error en asignar_habilidades_tareas.py. Abortando ejecución. |
+-----+

```

Figura 4.6: Evidencia de superación de la P-06

Capítulo 5

Conclusiones y posibles ampliaciones

El presente Trabajo de Fin de Máster ha abordado el reto de optimizar la gestión de recursos humanos en entornos empresariales dinámicos, desarrollando un sistema capaz de asignar de manera eficiente las tareas a los empleados en función de sus habilidades y experiencia, con el objetivo último de mejorar la productividad y viabilidad de los proyectos.

A lo largo del proyecto, se ha demostrado la importancia de integrar datos históricos y actuales en la toma de decisiones sobre la asignación de recursos, lo que permite no solo una mejor adecuación entre empleados y tareas, sino también la identificación temprana de necesidades adicionales de personal y la detección de posibles ineficiencias o sobrecargas de trabajo.

El sistema desarrollado se apoya en técnicas avanzadas de análisis de datos y aprendizaje automático, permitiendo extraer información relevante de plataformas como Jira, estimar tiempos de ejecución y crear perfiles detallados tanto de tareas como de empleados. Especial mención merece el proceso de asignación de habilidades mediante modelos de machine learning, donde se ha validado la eficacia del algoritmo Random Forest para la clasificación de habilidades a partir de descripciones textuales, así como el uso de Gradient Boosting para la recomendación de candidatos a tareas concretas.

La estructura modular del sistema, apoyada en una base de datos relacional y una arquitectura flexible, facilita tanto la consulta y visualización de resultados como la posible ampliación futura del sistema. Así, queda abierta la puerta a la inclusión de nuevas variables, como preferencias personales, dinámicas de equipo o aspectos motivacionales, que en la versión actual se han considerado fuera del alcance del trabajo.

No obstante, se identifican ciertas limitaciones en la solución propuesta, principalmente derivadas de la imposibilidad de considerar factores subjetivos o éticos en la asignación automática de tareas, así como la necesidad de seguir perfeccionando la estimación del nivel de experiencia por habilidad de cada empleado.

Se señala, como posible mejora futura, la incorporación de ponderaciones diferenciadas según el grado de implicación de cada habilidad en las tareas, lo que permitiría una gestión aún más personalizada y ajustada a la realidad del entorno laboral.

En conclusión, el sistema desarrollado supone un avance significativo hacia la optimización de la gestión de recursos humanos apoyada en datos objetivos, y representa una base sólida sobre la que seguir construyendo soluciones cada vez más inteligentes, flexibles y adaptadas a las necesidades cambiantes de las organizaciones.

5.1. Ampliaciones futuras

A partir de las conversaciones mantenidas con un gestor de proyectos, se han identificado una serie de mejoras potenciales orientadas a incrementar el valor añadido del sistema y a optimizar su usabilidad para los usuarios finales. Estas propuestas tienen como objetivo adaptar la herramienta a las necesidades reales del entorno profesional y facilitar su integración en los flujos de trabajo habituales:

- **Ponderación de habilidades:** en la versión actual del sistema, la experiencia asociada a cada tarea se calcula en función del tiempo total invertido, asignando el mismo nivel de experiencia a todas las habilidades implicadas. Este enfoque puede resultar limitado, ya que no todas las habilidades requieren el mismo grado de dedicación dentro de una misma tarea. Como consecuencia, pueden producirse asignaciones subóptimas, al no reflejar con precisión la relevancia o el peso específico de cada habilidad en el desarrollo de la tarea. Incorporar un mecanismo de ponderación permitiría mejorar la calidad de las recomendaciones generadas por el sistema.
- **Mejora en la presentación de resultados:** en la versión actual del sistema, los resultados generados por el módulo de asignación se presentan en formato Excel. Si bien este formato facilita la portabilidad y permite operaciones básicas de filtrado, puede no ser el más adecuado para una interpretación ágil y visual de los datos. En este sentido, se considera de interés el desarrollo de un panel interactivo en Power BI u otra herramienta de visualización avanzada, que permita representar los resultados de forma dinámica y facilitar la toma de decisiones por parte de los usuarios finales.
- **Ampliación del conjunto de tareas pre-etiquetadas:** incrementar el número de tareas con habilidades previamente asignadas permitiría entrenar el modelo con un volumen de datos mayor, lo que, en términos generales, contribuiría a mejorar su capacidad predictiva y la calidad de los resultados obtenidos. No obstante, esta mejora implica una limitación importante: el etiquetado de nuevas tareas requiere un proceso manual de revisión y

asignación de habilidades, lo cual demanda tiempo y recursos humanos especializados.

- **Optimización del sistema de planificación de tareas:** en la versión actual del sistema, la planificación de los distintos procesos se gestiona mediante un script en Bash, lo cual resulta adecuado para entornos de prueba o despliegues sencillos. Sin embargo, con vistas a su integración en un entorno empresarial, se considera conveniente sustituir este enfoque por herramientas más robustas y escalables, como Apache Airflow u otras plataformas especializadas en la orquestación de flujos de trabajo. Esta mejora permitiría una gestión más flexible, trazable y automatizada de las ejecuciones periódicas del sistema.
- **Incorporación de atributos adicionales en el proceso de asignación:** en la versión actual del sistema, las asignaciones se basan fundamentalmente en el tiempo dedicado a la tarea y en el nivel de habilidad del empleado. No obstante, la inclusión de atributos adicionales (como el conocimiento específico del área temática, la experiencia previa en proyectos similares o la trayectoria profesional en otras organizaciones) permitiría enriquecer significativamente el proceso de toma de decisiones. Esta ampliación del conjunto de variables contribuiría a generar asignaciones más precisas y alineadas con el perfil real de cada empleado.
- **Incorporación de un calendario laboral:** en la versión actual del sistema no se contempla la posibilidad de ausencias programadas o imprevistas, como vacaciones, bajas médicas o días festivos. Como mejora, se propone integrar un calendario laboral que permita tener en cuenta la disponibilidad real de los empleados. De este modo, el sistema evitaría asignar tareas a personas que, por diversos motivos, no estén en condiciones de ejecutarlas en un periodo determinado, lo que incrementaría la fiabilidad y efectividad de las asignaciones.
- **Mejora del algoritmo de estimación de tiempos:** en la versión actual del sistema, la estimación del tiempo necesario para completar una tarea se basa en medias históricas a nivel de proyecto o de empleado. Si bien este enfoque proporciona una aproximación inicial, se ha comprobado que, en muchos casos, no refleja con precisión la duración real de las tareas. Como línea de mejora, se propone desarrollar un modelo predictivo más avanzado, basado en técnicas de aprendizaje automático, que integre múltiples variables relevantes (tipo de tarea, complejidad textual, habilidades requeridas, entre otras). Esto permitiría obtener estimaciones más ajustadas y realistas, contribuyendo así a una planificación más eficiente y fiable de los recursos.

Bibliografía

- [1] Atlassian. Jira cloud platform rest api v3. <https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/#about>, 2024. Accedido el 11 de junio de 2025.
- [2] Atlassian. Jira software. <https://www.atlassian.com/software/jira>, 2024. Acceso el 7 de junio de 2025.
- [3] Docker Inc. Docker: Empowering app development for developers. <https://www.docker.com>, 2024. Accedido el 11 de junio de 2025.
- [4] Edward Ma. nlpaug: Data augmentation for nlp. <https://github.com/makcedward/nlpaug#augmenter>, 2024. Accedido el 11 de junio de 2025.
- [5] GeeksforGeeks. Explanation of bert model in nlp. <https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/>, 2024. Accedido el 11 de junio de 2025.
- [6] GeeksforGeeks. Gradient boosting in machine learning. <https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/>, 2024. Accedido el 11 de junio de 2025.
- [7] GeeksforGeeks. Multinomial naive bayes in machine learning. <https://www.geeksforgeeks.org/machine-learning/multinomial-naive-bayes/>, 2024. Accedido el 11 de junio de 2025.
- [8] GeeksforGeeks. Singular value decomposition (svd). <https://www.geeksforgeeks.org/singular-value-decomposition-svd/>, 2024. Accedido el 11 de junio de 2025.
- [9] GeeksforGeeks. Support vector machine (svm) algorithm. <https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/>, 2024. Accedido el 11 de junio de 2025.
- [10] GeeksforGeeks. Understanding bert in nlp. <https://www.geeksforgeeks.org/understanding-bert-nlp/>, 2024. Accedido el 11 de junio de 2025.

- [11] Hugging Face. Bert model — transformers documentation. https://huggingface.co/docs/transformers/model_doc/bert, 2024. Accedido el 11 de junio de 2025.
- [12] IBM. Scheduling jobs with tivoli workload scheduler. <https://www.ibm.com/docs/es/taddm/7.3.0?topic=products-scheduling-jobs-tivoli-workload-scheduler>, 2024. Accedido el 11 de junio de 2025.
- [13] IBM. What is linear regression? <https://www.ibm.com/think/topics/linear-regression>, 2024. Accedido el 11 de junio de 2025.
- [14] IBM. What is random forest? <https://www.ibm.com/think/topics/random-forest>, 2024. Accedido el 11 de junio de 2025.
- [15] JetBrains. Pycharm: the python ide for professional developers. <https://www.jetbrains.com/pycharm/>, 2024. Accedido el 11 de junio de 2025.
- [16] KeepCoding. ¿qué es el algoritmo tf-idf vectorizer? <https://keepcoding.io/blog/que-es-el-algoritmo-tf-idf-vectorizer/>, 2024. Accedido el 11 de junio de 2025.
- [17] Kenneth Reitz et al. Requests: Http for humans. <https://pypi.org/project/requests/>, 2024. Accedido el 11 de junio de 2025.
- [18] Microsoft. Visual studio code. <https://code.visualstudio.com>, 2024. Accedido el 11 de junio de 2025.
- [19] Overleaf. Overleaf: Online latex editor. <https://www.overleaf.com/>, 2024. Accedido el 11 de junio de 2025.
- [20] Postman Inc. Postman: Api platform for building and using apis. <https://www.postman.com>, 2024. Accedido el 11 de junio de 2025.
- [21] scikit-learn developers. sklearn.ensemble.randomforestclassifier. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>, 2024. Accedido el 11 de junio de 2025.
- [22] scikit-learn developers. sklearn.linear_model.linearregression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html, 2024. Accedido el 11 de junio de 2025.
- [23] scikit-learn developers. sklearn.naive_bayes.multinomialnb. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html, 2024. Accedido el 11 de junio de 2025.

- [24] scikit-learn developers. sklearn.svm.linearsvc. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>, 2024. Accedido el 11 de junio de 2025.
- [25] TensorFlow. text.berttokenizer. https://www.tensorflow.org/text/api_docs/python/text/BertTokenizer, 2024. Accedido el 11 de junio de 2025.
- [26] The PostgreSQL Global Development Group. Postgresql: The world's most advanced open source relational database. <https://www.postgresql.org>, 2024. Accedido el 11 de junio de 2025.
- [27] Trent B. iterative-stratification. <https://github.com/trent-b/iterative-stratification>, 2024. Accedido el 11 de junio de 2025.