

Desarrollo Backend para el Procesamiento y Centralización de Pruebas de Neuro-competencias para Neuro4B

Maximiliano Jaime Enrique Cornejo Meza

maximiliano.cornejom@usm.cl

Carlos Alten
Profesor Guía

Hernán Saavedra
Profesor Correferente

Resumen: La empresa Neuro4B se encarga de realizar pruebas de neuro-competencias para evaluar rendimiento y posibles reacciones de trabajadores frente a distintos ambientes, combinando el uso del programa Nesplora y un electroencefalograma para monitorear las ondas cerebrales del trabajador. El problema es que los datos generados no vienen en un formato adecuado para el análisis, lo cual obliga a realizar una extracción manual de una gran cantidad de datos, demorando horas y retrasando la entrega de resultados a los clientes. Frente a este problema, se propuso el desarrollo de una plataforma web que permita a los psicólogos subir los archivos obtenidos durante las pruebas para transformarlos a un formato utilizable y almacenarlos para cuando sean requeridos, aparte de generar un informe preliminar que facilite su análisis. Durante el desarrollo de este proyecto, se han llevado a cabo múltiples evaluaciones, entrevistando a posibles usuarios y considerando sus dificultades para utilizar la página para ir mejorándola. Gracias a este proyecto, el tiempo de extracción de datos se redujo de horas a minutos, disminuyendo el error humano y centralizando la información en un único lugar, mejorando la eficiencia del trabajo de los psicólogos y agilizando la entrega de resultados a clientes.

Palabras Clave: Plataforma web, Gestión de trabajadores, Extracción de datos, Centralización de información.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE
MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Desarrollo Backend para el Procesamiento y Centralización de Pruebas de Neuro-competencias para Neuro4B

Nombre del candidato(a): Maximiliano Jaime Enrique Cornejo Meza

Carrera / Grado: Ingeniería en Informática

Campus: José Miguel Carrera Departamento: ELINF

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Carlos Alten, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

- ☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 24 de julio de 2026 Firma:

Estudiante o Candidato(a):

Fecha: 24 de julio 2026 Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

1. Introducción

1.1. Contexto y antecedentes

En Neuro4B, se ha empezado a utilizar pruebas de neuro-competencias para monitorear el estado de los trabajadores y predecir su comportamiento frente a situaciones específicas, para así poder predecir cuándo se podría cometer un fallo al momento de realizar sus tareas, esto es particularmente relevante en contextos de acciones críticas que puedan poner en riesgo la seguridad o incluso la vida de personas, como al accionar maquinaria pesada en minas o al realizar acciones de carácter militar. [1]

De momento, estas pruebas se hacen con unos lentes de realidad virtual, en el cual se efectúa la prueba de Nesplora, que consiste en servir helados a múltiples clientes mientras se evalúan las capacidades de la persona con sus aciertos, fallos y con las ondas cerebrales que son medidas mediante un electroencefalograma. Esta prueba genera información de alto valor y permite predecir el cómo se comportaría la persona ante estímulos estresantes al momento de trabajar y se tiene planificado el escalarla a otras pruebas diferentes o incluso a ser tomada a tiempo real mediante artefactos de realidad aumentada.

Este proceso cuenta con la participación de 3 actores principales, los cuales son los siguientes:

- **Psicólogo:** Encargado de extraer los datos generados por la prueba de Nesplora y los resultados del EEG. También se encarga del análisis de datos y creación del informe final con los resultados de la evaluación.
- **Operario:** Encargado de ir físicamente a tomar las pruebas a los trabajadores.
- **Trabajador:** Persona a quién se le toma la prueba Nesplora mientras usa el EEG.

1.2. Definición del problema

El problema que actualmente enfrenta Neuro4B, es que al momento de realizar pruebas Nesplora, esta entrega los resultados en un formato no personalizable que no es adecuado para el análisis, entregando un PDF con mucha información que no aporta valor al psicólogo, provocando que, de forma previa a la evaluación de resultados, se deba realizar una extracción manual de los datos relevantes para el especialista. Esta tarea es agotadora y repetitiva, además de consumir una gran cantidad de tiempo a realizar un proceso no relacionado directamente con la labor del psicólogo, tardando hasta 3 horas de trabajo mecánico previo al análisis de cada trabajador, esto además perjudica gravemente el potencial del negocio, ya que perjudica los tiempos de entrega de resultados y les añade una gran carga de trabajo a los psicólogos. Además de que, al ser un proceso tan manual, la información queda descentralizada y sin un formato definido.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Figura 1. Tabla de Problemas identificados

Nº	Problema Identificado	Descripción
1	Sobrecarga de funciones para los psicólogos	Actualmente los psicólogos tienen demasiadas funciones, ya que aparte de analizar, se encargan de la extracción de datos.
2	Poca accesibilidad a la información	Al estar distribuidos los archivos por diferentes equipos y estar en diferentes formatos, se dificulta el acceso a los datos cuando son requeridos.
3	Tareas repetitivas	El extraer datos de los archivos generados por Nesplora es una tarea sumamente repetitiva que puede ser perfectamente automatizada.
4	Demora en recibir resultados	Al tardarse tanto en procesar los archivos, se demora en entregar resultados a los clientes.
5	Uso ineficiente del tiempo	Los psicólogos usan gran parte de su tiempo en una tarea que no está relacionada a su área de trabajo como tal.
6	Poca escalabilidad	Al estar gran parte del sistema basado en trabajo manual, es muy complicado escalarlo a un público más amplio.

Fuente: Elaboración propia

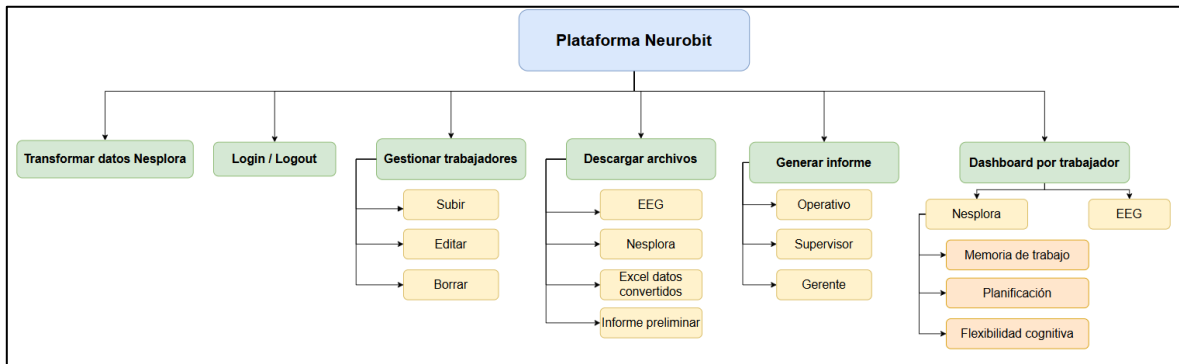
1.3. Breve descripción sobre propuesta de solución

1.3.1. Resumen de la propuesta

Nuestra solución propuesta sería una plataforma web multimodal, en la cual los psicólogos puedan subir los archivos obtenidos de las pruebas de neuro-capacidades y estos sean procesados de forma automática, con una estructura definida y en menos de 3 minutos. Además, la información quedará centralizada en una plataforma segura y organizada mediante un sistema de trabajadores donde cada uno contendrá los archivos relacionados a las pruebas realizadas, informes preliminares creados con los datos obtenidos para facilitar el diagnóstico de los psicólogos y un dashboard donde se podrá visualizar lo ocurrido durante las pruebas a través del tiempo. Para esto se implementará un módulo de reconocimiento óptico de caracteres (OCR) para la extracción de información, que se encargará de convertir las imágenes provenientes del PDF Nesplora a texto plano que luego será mostrado al usuario en formato Excel, es importante mencionar que se utilizará un módulo OCR ya que una cantidad importante de los datos a extraer vienen en gráficos donde los datos vienen sobrepuestos en las imágenes, por lo que es necesario implementar un mecanismo de visión por computadora para transformarlos. Por último, se contará con un registro histórico para poder recuperar datos eliminados o editados por los usuarios, para que, aunque un trabajador sea modificado, sea posible recuperar sus archivos e información previos en caso de ser requerido.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Figura 2. Árbol de funcionalidades



Fuente: Elaboración propia

1.3.2. Alcance y Límites del proyecto:

- El Backend de la plataforma web será diseñado inicialmente pensando en utilizar la menor cantidad de recursos de máquina posible.
- Los usuarios estimados para la plataforma web no superarán los 15 de forma simultánea.
- No será incluido un sistema de creación de usuarios mediante la plataforma web.
- El proyecto no se involucra en la forma de tomar las pruebas.
- No nos encargamos de buscar clientes.
- No cambiamos la forma de analizar los datos por parte del psicólogo.

1.4. Objetivos Generales y Específicos de la Tesina

1.4.1. Objetivo General:

Desarrollar e integrar un sistema Backend para la plataforma web NeuroBit, centrado en la extracción, gestión y centralización de datos sobre capacidades neurocognitivas para Neuro4B, reduciendo el tiempo de extracción a menos de 3 minutos y permitiendo la comunicación con el Frontend mediante APIs.

1.4.2. Objetivos específicos:

1. **Autenticar a los usuarios de la aplicación:** Debe permitir acceder a la plataforma web solamente cuando se ingrese un usuario con credenciales válidas.
2. **Extraer de forma automática la información relevante de los archivos Nesplora:** Se debe convertir un archivo proveniente de la prueba Nesplora y convertirlo en un archivo Excel estructurado y listo para ser utilizado.
3. **Almacenar la información de los trabajadores evaluados:** Se deben poder agregar, editar y eliminar trabajadores, guardando el nombre, cargo y empresa del trabajador aparte de la fecha y su identificador.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

4. **Almacenar los archivos de cada trabajador:** Se deben almacenar los archivos subidos asociados a cada trabajador mediante el ID, estos deben ser accesibles y descargables cuando sean requeridos.
5. **Generar un informe que ayude al análisis psicológico de los trabajadores:** Se debe poder crear un informe preliminar utilizando los archivos y datos que tenga el trabajador, este puede contar con diferentes plantillas, dirigidas a cargos operativos, de supervisión y de gerencia.
6. **Entregar datos listos para ser mostrados a los usuarios:** Se debe contar con múltiples funciones API que sean capaces de comunicar y entregar datos al Frontend, estas funciones incluyen funcionalidades como autenticar usuarios, entregar archivos y gestión de trabajadores.
7. **Procesar los datos del EEG y dejarlos listos para ser evaluados:** Se deben poder convertir los datos entregados por el encefalograma a información más entendible, utilizando las fórmulas entregadas por la empresa.
8. **Guardar el registro de cambios realizados en la aplicación:** Los trabajadores cuyos datos sean editados o eliminados, deben ser registrados en la base de datos interna del servidor, para no perder la información y ser recuperable.

1.5. Justificación del Proyecto

El desarrollo de este sistema Backend para Neuro4B se justifica debido a la necesidad de optimizar el uso de recursos de la empresa, tanto de tiempo como de personas. Al implementar este proyecto, ya no será necesario que los psicólogos usen una gran cantidad de horas extrayendo datos y podrán centrar su atención en tareas más propias de su cargo. El área de desarrollo Backend se encargará de todo lo relacionado al procesamiento, almacenaje y centralización de los archivos resultantes de las pruebas realizadas por Neuro4B.

1.5.1. Herramientas utilizadas:

- **Base de datos:** Almacenar trabajadores, cuentas de usuario y archivos.
- **API REST:** Facilitar la comunicación con el Frontend, entregar y recibir datos.
- **OCR:** Extraer información del archivo Nesplora.
- **GitHub:** Controlar la arquitectura del proyecto y facilitar trabajo cooperativo.

1.5.2. Beneficios:

- **Trabajo más rápido:** Acelerar el trabajo de los psicólogos, al obtener los documentos de forma más veloz.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- **Elimina error humano al extraer datos:** Garantizar tener información fiable al momento de analizar los datos.
- **Centralización:** Facilitar el acceso a la información requerida al tenerla toda en el mismo lugar.
- **Información consistente:** Al automatizar y darle un formato a la extracción de datos, esta queda en un formato único, dándole más orden a la información.
- **Mejor uso del tiempo:** Eliminar la necesidad de extraer la información de forma manual, permitiendo que los psicólogos se centren en tareas propias de su área.

1.6. Metodología

El desarrollo de este trabajo fue organizado mediante la metodología ágil Scrum [2], en la cual se divide el proyecto en varios ciclos de trabajo denominados como Sprint, estos tenían la duración aproximada de 1 semana y consistían en dividir las tareas totales a realizar en el proyecto en tareas más cortas y medibles, las cuales luego eran ingresadas al product-backlog y se iban distribuyendo por Sprint según fuera conveniente. Durante el transcurso de cada Sprint, se realizaron reuniones diarias con duración de 15 minutos para compartir avances y dificultades, para así ir monitoreando el avance de los integrantes y mejorando la comunicación entre las distintas áreas del proyecto. Cabe resaltar que se seleccionó la metodología Scrum frente a metodologías más tradicionales debido a la necesidad de ir implementando retroalimentación y mejoras a lo largo del proyecto para adaptarse a las necesidades de Neuro4B y construir una plataforma personalizada que cumpla con las expectativas.

Para apoyarnos durante este proceso utilizamos principalmente 3 herramientas cooperativas, las cuales serían:

- **Discord:** Para realizar las reuniones diarias y para compartir información de uso regular, como enlaces, archivos, imágenes o textos. Aparte aquí también nos reuníamos de forma espontánea al presentarse alguna dificultad inmediata, por ejemplo, problemas de conexión entre el Frontend y el Backend, discusiones sobre el cómo se deberían manejar los datos, etc.
- **Jira:** Se utilizaba principalmente para gestionar el product-backlog del proyecto e ir registrando avances.
- **GitHub:** Se utilizaba para compartir código y estructurarlo según fuera conveniente.

En este proceso, como el encargado principal del desarrollo Backend, era necesario que fuera reportando avances de forma continua para que la plataforma web cumpliera con los requisitos necesarios en todo momento. Principalmente estuve trabajando con la desarrolladora Frontend del proyecto, intercambiando información y planificando la arquitectura de software junto a los nuevos cambios a realizar.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

1.7. Breve descripción de la organización del informe en capítulos

1. **Introducción:** Visión general del proyecto, descripción del problema y solución propuesta.
2. **Marco teórico:** Base teórica del proyecto, historia, desarrollo y descripción del Backend dentro de la arquitectura de software. Tecnologías utilizadas y conceptos clave.
3. **Diseño e implementación:** Descripción detallada de la implementación del Backend en el proyecto. Construcción e integración con otros componentes del sistema.
4. **Conclusiones:** Aprendizajes, desafíos y conclusiones generales.
5. **Agradecimientos:** Reconocimiento de personas relevantes en el desarrollo del proyecto.
6. **Referencias:** Referencias de fuentes, sitios web y artículos citados durante el informe.
7. **Anexos:** Material complementario útil para comprender mejor el proyecto.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

2. Marco Teórico

2.1. Fundamentos del desarrollo Backend

El Backend en el desarrollo de software, se refiere a toda la parte del sistema que no es visible para el usuario final, pero que es fundamental para que funcione [3]. En esta parte del sistema se configuran todos los aspectos lógicos de una aplicación, como el procesamiento de datos, conversión de archivos, autenticación de usuarios, almacenamiento de información, etc. El Backend trabaja en conjunto con el Frontend, que es la parte en donde interactúa el usuario como tal, es decir, el Frontend en una aplicación se encarga de recibir una petición para luego mandarla al Backend para procesarla y conseguir una respuesta.

La programación Backend nace de la necesidad de crear aplicaciones más seguras, escalables y efectivas y su desarrollo ha evolucionado continuamente a través de los años. En la década de los 90 dominaron los programas creados con CGI y Perl, siendo Perl el lenguaje de programación más popular y CGI se encargaba de permitir la ejecución de programas desde un servidor web. Posteriormente, en la década del 2000, Java y PHP se volvieron más utilizados, ya que introdujeron nuevas herramientas que permitían un desarrollo más complejo. Luego, en la década del 2010, se popularizaron Ruby y Node.js debido a que ayudaban a acelerar el desarrollo. Y finalmente, en la década actual, es más popular utilizar microservicios y las aplicaciones nativas de nube, en donde lenguajes como Go y Rust adquieren una mayor relevancia, junto a los tradicionales Python y Java [4].

Dentro de la arquitectura de software, el Backend cumple el rol de asegurarse que la aplicación funcione, gestionando todos los elementos que no son accesibles para el usuario final y es aquí donde se configuran los aspectos lógicos de la aplicación, tales como:

- **Procesamiento de datos:** Recibir, procesar y entregar respuesta con los datos provenientes del Frontend.
- **Lógica de negocios:** Definir y aplicar reglas internas para mantener un correcto funcionamiento. [3].
- **Manejo de bases de datos:** Gestionar los datos agregados, modificados o eliminados de las bases de datos. Aparte de entregar datos según sea necesario.

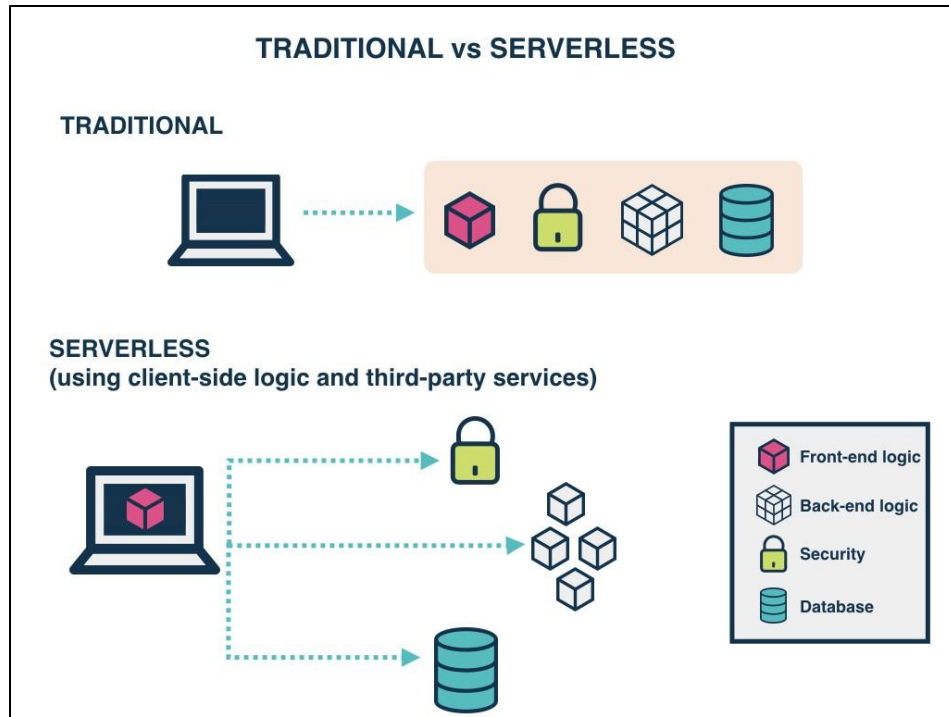
2.1.1. Tendencias dentro de la industria

1. **Arquitecturas Serverless:** En estas, el desarrollador no se encarga de administrar el servidor directamente, utilizando servicios en la nube que ejecutan funciones bajo demanda. Este tipo de arquitectura ayuda a reducir costes al pagar solo por el tiempo de ejecución, aparte de facilitar la escalabilidad de los servicios y ofrecer un despliegue más ágil con mantenimiento reducido [5]. El enfoque de la arquitectura serverless facilita en gran medida el despliegue de aplicaciones, permite una escalabilidad automática según los recursos requeridos por el sistema y facilita la mantención del proyecto, a cambio de tener un menor control sobre la infraestructura y limitaciones en el tiempo de ejecución [6], lo que podría generar problemas al momento de ejecutar tareas largas con alta exigencia

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

computacional. Por lo tanto, se decidió no optar por una arquitectura sin servidor para el desarrollo de la plataforma debido a que requerimos ejecutar procesos largos como el OCR.

Figura 3. Arquitectura Serverless vs Tradicional



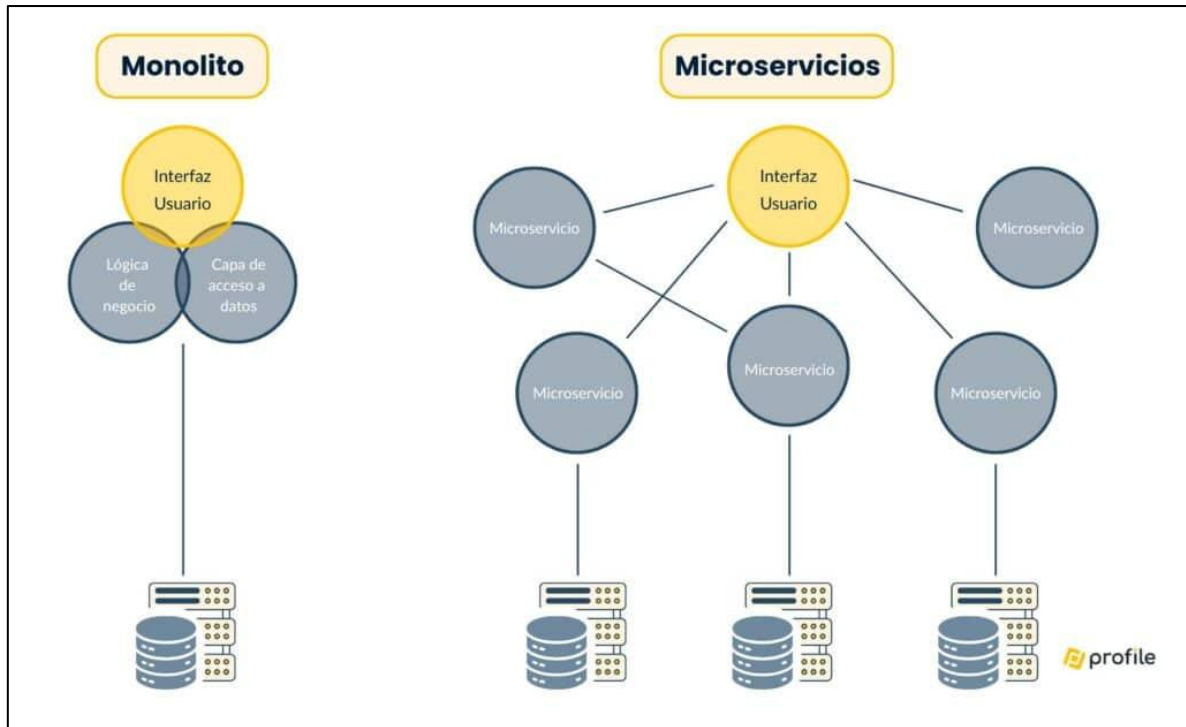
Fuente: Deorwine [7]

2. **Arquitectura de microservicios:** La aplicación no se construye como un bloque único y se divide en pequeños componentes independientes, donde cada uno de estos cumple una función específica y separada. Los microservicios ayudan al mantenimiento, evolución y mejoramiento continuo de los componentes del programa, ya que pueden actualizarse sin afectar al resto del sistema, ayudando a su vez a mejorar la escalabilidad. Esta arquitectura también ofrece una compatibilidad entre diferentes lenguajes de programación, ya que, al ser componentes diferentes, pueden comunicarse entre sí a través de una API que se encargue de traducirlos [8]. Esta arquitectura es especialmente útil en sistemas con mucha concurrencia de usuarios en donde la independencia de los módulos facilitaría el mantenimiento y la evolución del sistema. Los microservicios son utilizados en proyectos como plataformas de streaming, aplicaciones de comercio electrónico grandes o plataformas financieras, donde es clave el poder escalar los diferentes componentes del sistema sin afectar a toda la aplicación y hay muchas solicitudes de usuarios de forma continua. Aunque este tipo de arquitectura es útil para el escalamiento y la disponibilidad, en el proyecto actual, al tener una cantidad limitada de usuarios, la complejidad de

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

implementar microservicios podría generar más problemas en el desarrollo que los beneficios potenciales de modularizar tanto el proyecto.

Figura 4. Arquitectura Monolito vs Microservicios



Fuente: Profile [9]

3. **Automatización impulsada por IA:** Los procesos de inteligencia artificial están influyendo fuertemente en el desarrollo Backend al automatizar estrategias clave al momento de mejorar la consistencia del software, pudiendo revisar de manera más eficiente la codificación, el rendimiento y la integración continua. Permitiendo desarrollar softwares más rápidos y fiables [10]. Algunos ejemplos comunes del uso de la IA incluyen los asistentes de programación como el Copilot de GitHub o ChatGPT, que ayudan en la generación de código y resolución de problemas al momento de programar. Esto puede ser utilizado para facilitar en gran medida la corrección de errores o para encontrar posibles vulnerabilidades o fallos en el sistema, ayudando a realizar pruebas de funcionalidad más exhaustivas en menos tiempo. En el caso de Neurobit, se utilizó la inteligencia artificial como apoyo al momento de encontrar/solucionar fallos para agilizar el periodo de depuración.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

2.1.2. Usos del Backend

El Backend es utilizado en todo tipo de sistemas actuales, tales como aplicaciones web, sitios web, redes sociales, plataformas de comercio, aplicaciones financieras y similares. Esto debido a que es indispensable para procesar información, garantizar la seguridad del sistema y manejando las solicitudes provenientes del usuario mediante el Frontend.

> Ventajas

- **Separación de responsabilidades:** Ayudando a separar la capa de procesamiento lógico de la capa encargada de la interacción y experiencia del usuario.
- **Seguridad:** Permitiendo la autenticación de usuarios y encriptando datos sensibles.
- **Automatización de procesos:** Procesando información sin necesidad de interacción humana directa.

> Desventajas

- **Complejidad:** Produciendo una larga curva de aprendizaje [11].
- **Costos de arquitectura:** Dependiendo del sistema de base de datos y servicios, puede suponer un costo elevado a nivel de recursos.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

2.2. Arquitectura de software

2.2.1. Planificación Inicial de la arquitectura.

En el desarrollo de software, hay diversas arquitecturas para elegir, entre las cuales se encuentran la arquitectura monolítica, que se basa en desarrollar toda la aplicación en un único bloque de código, la arquitectura en capas, donde se divide el desarrollo en capas con responsabilidades separadas, y las arquitecturas por microservicios y Serverless que mencionamos en el punto anterior.

Figura 5. Tabla de comparación de arquitecturas

Arquitectura	Ventajas	Desventajas	Adecuación al proyecto
Monolítica	Simplicidad de desarrollo e implementación inicial	Difícil de mantener y escalar, ya que cualquier cambio afecta a toda la aplicación	Media-baja, ya que dificulta expandir el sistema en caso de ser necesario
Por capas	Separación clara de responsabilidades, facilita el trabajo en equipo y el mantenimiento	Mayor planificación inicial y puede ser ineficiente en aplicaciones muy simples	Alta, balanceando simplicidad, orden y escalabilidad
Microservicios	Escalabilidad muy alta, al tener módulos independientes y alta disponibilidad	Requiere mayor infraestructura y complejiza el desarrollo.	Baja, es excesiva para un sistema con pocos usuarios
Serverless	Alta escalabilidad automática y se paga por uso	Costos elevados en tareas de procesamiento complejas	Baja, ya que el sistema requiere procesamiento continuo de datos

Fuente: Elaboración propia

Debido a esto, el sistema fue planificado en base a una arquitectura en capas [12], en donde se separa el funcionamiento de la aplicación en 3 diferentes capas.

- **Capa Frontend:** Capa encargada de la interacción con el usuario. Está armada utilizando React, una librería de JavaScript creada por Facebook orientado al desarrollo de interfaces de usuario [13].
- **Capa Backend:** Capa encargada del procesamiento de archivos, gestión de trabajadores, autenticación de usuarios, entre otros. Está creada utilizando el framework Django junto a Django-restframework para la generación de APIs que interactúan con el Frontend.
- **Capa de datos:** Capa encargada del almacenamiento de los datos. Inicialmente se pensó para ser desarrollada mediante una arquitectura mixta, mezclando una base de datos SQL junto a un sistema de archivos para guardar documentos PDF, Excel, etc. Esto nos permitiría evitar sobrecargar la base de datos y deja abierta la posibilidad de escalar al utilizar otros contenedores de almacenamiento en un futuro [14].

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Se seleccionó una arquitectura en capas debido a que permite separar las responsabilidades del proyecto en componentes independientes, permitiendo una mejor organización y permitiendo que si se debe cambiar de base de datos o de aplicación Frontend, se pueda realizar sin afectar de forma relevante al Backend, ya que al estar el Frontend en una aplicación React aparte comunicándose exclusivamente mediante APIs, permitiría el reemplazar totalmente ese componente, por ejemplo, a una aplicación Angular sin tener que cambiar el funcionamiento interno, y lo mismo ocurriría al momento de requerir migrar de base de datos.

Para el despliegue de la aplicación web, tanto el Frontend como el Backend se podrían encontrar contenidos en Docker, lo cual permitiría agilizar el despliegue al tener empaquetadas todas las dependencias [15], otra opción sería dejar todos los requerimientos para el funcionamiento de la aplicación en un servidor físico o en la nube mediante AWS, pero esto generaría problemas si fuera necesario el cambiar el método de despliegue de la plataforma.

2.2.2. Patrones de diseño

Al momento de desarrollar software, existen son una "solución general, reutilizable y aplicable a diferentes problemas de diseño de software" [16], en donde se utilizan conceptos ya existentes para basar la creación del programa, algunos de los más útiles en el contexto del proyecto son:

- **Observer:** Consiste en definir un mecanismo de notificación para cuando se realiza un cambio en cierto objeto [16]. Sería útil al momento de crear el registro de trabajadores al momento de modificarlos o para notificar al usuario cuando se termine de procesar un archivo.
- **Strategy:** Permite definir una familia de algoritmos en clases separadas y llamar al más conveniente en tiempo de ejecución [17]. Se podría utilizar al momento de extraer datos de archivos, llamando a diferentes métodos de extracción de datos dependiendo del archivo que se suba.
- **Facade:** Permite generar una interfaz simplificada con la que se pueda interactuar [17]. Sería muy útil al momento de trabajar con funciones API, ya que éstas pueden ser utilizadas por el Frontend sin interactuar directamente con funciones más complejas del Backend.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

2.3. Tecnologías y frameworks utilizados

Investigación de tecnologías

Para la realización del sistema Backend del proyecto, fue necesario explorar diversas alternativas para el desarrollo de software, para luego seleccionar las que mejor se adecuen a los requerimientos del proyecto, considerando que requerimos herramientas de procesamiento de archivos, autenticación de usuarios, almacenamiento de trabajadores, conexión con el Frontend realizado en React, entre otras.

- **Lenguaje de programación:** Las opciones que consideramos para seleccionar un lenguaje de programación fueron Python, C++ y Java, donde cada uno contaba con sus ventajas particulares. Python ofrece una gran cantidad de librerías para técnicas de procesamiento de archivos, simplificando en gran medida el análisis de datos [18]. C++ cuenta con un gran rendimiento y control de memoria, significando un menor tiempo de extracción de información por archivo, pero complejizando el desarrollo del software. Java es estable y robusto al momento de procesar grandes cantidades de datos, pero su fuerte no es la visión por computadora que buscamos para el proyecto. Debido a esto, consideramos que Python sería la mejor opción al minimizar el tiempo de desarrollo por las librerías que posee de procesamiento de archivos y al acortar el tiempo de desarrollo por su simpleza.
- **Framework:** Se compararon los frameworks de Django con Flask y Node.js con Express y se llegó a la conclusión de que, debido a su amplia cantidad de mecanismos integrados, Django era una mejor opción de desarrollo ya que integra opciones de seguridad, de gestión de usuarios y utiliza Python que es el lenguaje de programación por utilizar en el procesamiento de archivos del proyecto. Todo esto nos favorecía a tener una velocidad de desarrollo superior frente a las alternativas.
- **Base de datos:** Al momento de seleccionar base de datos, consideramos que la mayor parte de los limitados recursos que tenemos serán utilizados para el sistema de procesamiento de archivos, teniendo esto en cuenta, comparamos 3 sistemas de almacenamiento diferentes, uno sería un almacenamiento mixto utilizando la base de datos integrada de Django (SQLite), junto a un contenedor donde estarían los archivos de los trabajadores para no sobrecargar la base de datos, otra opción era optar por PostgreSQL, en la cual podríamos insertar directamente los archivos en la base de datos al ser más robusta y potente, pero requiriendo una cantidad de recursos fuera de nuestro alcance inicial, finalmente la última opción era MySQL, siendo más robusta que SQLite y menos pesada que PostgreSQL, siendo una muy buena opción si se aumentaran los recursos de la aplicación y se decidiera aumentar su alcance. Cabe destacar, que, aunque con una base de datos más potente se podrían almacenar los archivos de forma más directa, esto podría suponer problemas si se almacenan demasiados datos, ya que el peso de la BD aumentaría muchísimo [14], por lo que seguramente la mejor opción sería utilizar el almacenamiento mixto pero con una base de datos MySQL o PostgreSQL.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- **Autenticación y seguridad:** Para ingresar a la plataforma web, se debe contar con credenciales válidas ingresadas en el sistema. Las cuentas de usuario serán creadas y otorgadas por el equipo de desarrollo según sea solicitado. Al momento de ingresar a la plataforma, el sistema de usuarios de Django se encarga de validar y asegurar el ingreso solamente de usuarios verificados [19].

Tablas comparativas complementarias

Figura 6. Tabla de comparación de base de datos con ponderación

Característica	Ponderación	SQLite	PostgreSQL	MySQL
Facilidad de Configuración	10%	5	2	4
Ahorro y optimización de recursos	30%	5	3	4
Funcionamiento frente a alta concurrencia	20%	2	5	4
Escalabilidad	40%	2	5	4
Puntuación		3,2	3,8	4

Fuente: Elaboración propia

Figura 7. Tabla de comparación de lenguajes de programación

Característica	Ponderación	Python	C++	Java
Velocidad de ejecución	20%	3,5	5	4,5
Cantidad de librerías incluidas útiles para el proyecto	40%	5	4	3
Tiempo de desarrollo	10%	5	3	4
Adecuación al contexto del proyecto	30%	4	3,5	3
Puntuación		4,4	3,95	3,4

Fuente: Elaboración propia

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

3. Diseño e implementación

En este capítulo, se describirá el cómo fue desarrollado el Backend del proyecto NeuroBit, detallando la arquitectura utilizada y las funcionalidades implementadas, además de cómo interactúa con los demás componentes del proyecto. También se explicará el motivo de cada decisión de diseño, basándonos en lo explicado en el marco teórico y en los requerimientos de negocio, apoyándonos en recursos gráficos cuando sea conveniente.

3.1. Requerimientos del sistema

Los requerimientos son un punto clave para validar el correcto funcionamiento del Backend, ya que estos son los que especifican lo que debe lograr el sistema. A continuación, se mostrará un listado de los principales requerimientos funcionales que fueron considerados al momento de construir el Backend de NeuroBit.

3.1.1. Requerimientos Funcionales

- RF1: El sistema NeuroBit debe proporcionar al psicólogo la capacidad de iniciar sesión en la plataforma web mediante credenciales válidas.
- RF2: El sistema NeuroBit debe permitir al usuario autenticado cerrar su sesión en la plataforma web.
- RF3: El sistema NeuroBit debe otorgar al psicólogo la capacidad de registrar un trabajador en la plataforma, ingresando sus datos personales y los archivos asociados a las pruebas realizadas.
- RF4: El sistema NeuroBit debe proporcionar al psicólogo la capacidad de modificar la información y archivos asociados a un trabajador registrado en la plataforma.
- RF5: El sistema NeuroBit debe permitir al psicólogo el eliminar un trabajador registrado en la plataforma.
- RF6: Al momento de eliminar o editar un trabajador, el sistema NeuroBit debe crear un registro con sus datos y archivos.
- RF7: El sistema NeuroBit debe permitir al psicólogo el visualizar el listado de trabajadores registrados en la plataforma.
- RF8: Al registrar un nuevo trabajador, el sistema NeuroBit debe almacenar sus archivos en un directorio estructurado.
- RF9: El sistema NeuroBit debe proporcionar al psicólogo la capacidad de transformar un archivo Nesplora en un archivo utilizable para el análisis profesional.
- RF10: El sistema NeuroBit debe permitir al psicólogo el visualizar los datos de un trabajador a través de un Dashboard.
- RF11: El sistema NeuroBit debe proporcionar al psicólogo la capacidad de generar un informe preliminar con los datos extraídos de las pruebas de neuro competencias.
- RF12: El sistema NeuroBit debe permitir al psicólogo el descargar los archivos asociados a un trabajador.
- RF13: Al momento de recibir una solicitud pesada, el sistema NeuroBit debe registrarlo en una cola de procesos para evitar bloqueos y sobrecargas del sistema.
- RF14: El sistema NeuroBit debe integrar un sistema de extracción de datos OCR externo.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

3.1.2. Requerimientos no Funcionales

- RNF1: El sistema debe garantizar el acceso únicamente a usuarios previamente registrados mediante credenciales válidas.
- RNF2: El sistema debe ser capaz de soportar las solicitudes de al menos 15 usuarios simultáneos sin generar bloqueos generales en la plataforma, incluso cuando se esté realizando procesamiento pesado.
- RNF3: Las solicitudes básicas del sistema deben responder en menos de 3 segundos, excluyendo tareas pesadas de procesamiento como el procesamiento de archivos Nesplora o la generación de informes.
- RNF4: Las solicitudes exigentes de procesamiento, como la ejecución del OCR o generación de informes, deben realizarse en menos de 3 minutos.

3.2. Diseño de componentes

En esta sección se describirá el funcionamiento de los principales componentes del Backend de NeuroBit, explicando sus responsabilidades y el cómo interactúan entre sí dentro de la arquitectura del proyecto.

3.2.1. Arquitectura general del proyecto

Basándonos en lo mencionado en el marco teórico, se optó por trabajar en una arquitectura en capas, debido a su modularidad y que facilita el trabajo en equipo e implementación de mejoras a lo largo del desarrollo. Definimos las siguientes 3 capas para el proyecto:

- **Capa Frontend:** Construida con React en forma de plataforma web. Se encarga de la interacción con el usuario final.

Funciones principales:

- Ingreso de credenciales para la autenticación de usuario
- Subida de archivos Nesplora y EEG
- Vista de gestión de trabajadores
- Descarga de archivos (Nesplora, EEG, Informe, Nesplora-transformado)
- Visualización del Dashboard con datos EEG

- **Capa Backend:** Desarrollada con Django y Django-Restframework. Se encarga de la lógica de negocios, procesamiento de archivos, autenticación de usuarios, entre otras funciones que serán detalladas más adelante.

Funciones Principales:

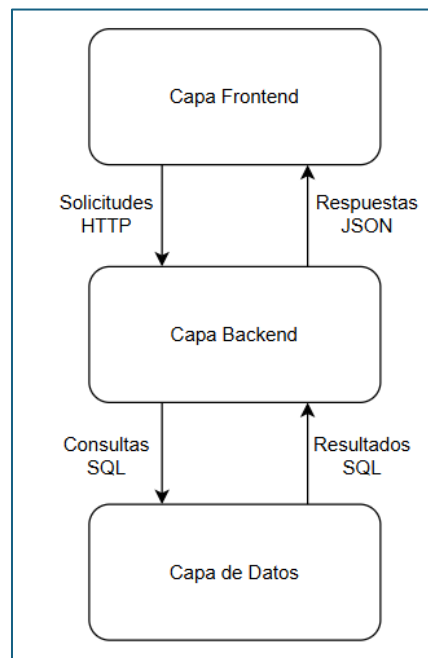
- Gestión de usuarios y autenticación
- Procesamiento OCR de archivos Nesplora
- Procesamiento de archivos EEG
- CRUD de trabajadores
- Generación de informes preliminares
- Exposición de APIs
- Manejo de cola de procesos para tareas pesadas

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- **Capa de Datos:** Encargada del almacenamiento de información, utilizando una base de datos MySQL para guardar datos como trabajadores, usuarios y Jobs, junto a un directorio estructurado para almacenar archivos XLSX, PDF, entre otros.

El Frontend se comunica con el Backend realizando solicitudes HTTP y recibiendo respuestas JSON y el Backend se comunica con la capa de datos mediante consultas SQL, como se puede observar en la siguiente figura.

Figura 8. Diagrama de capas



Fuente: Elaboración propia

3.2.2. Framework

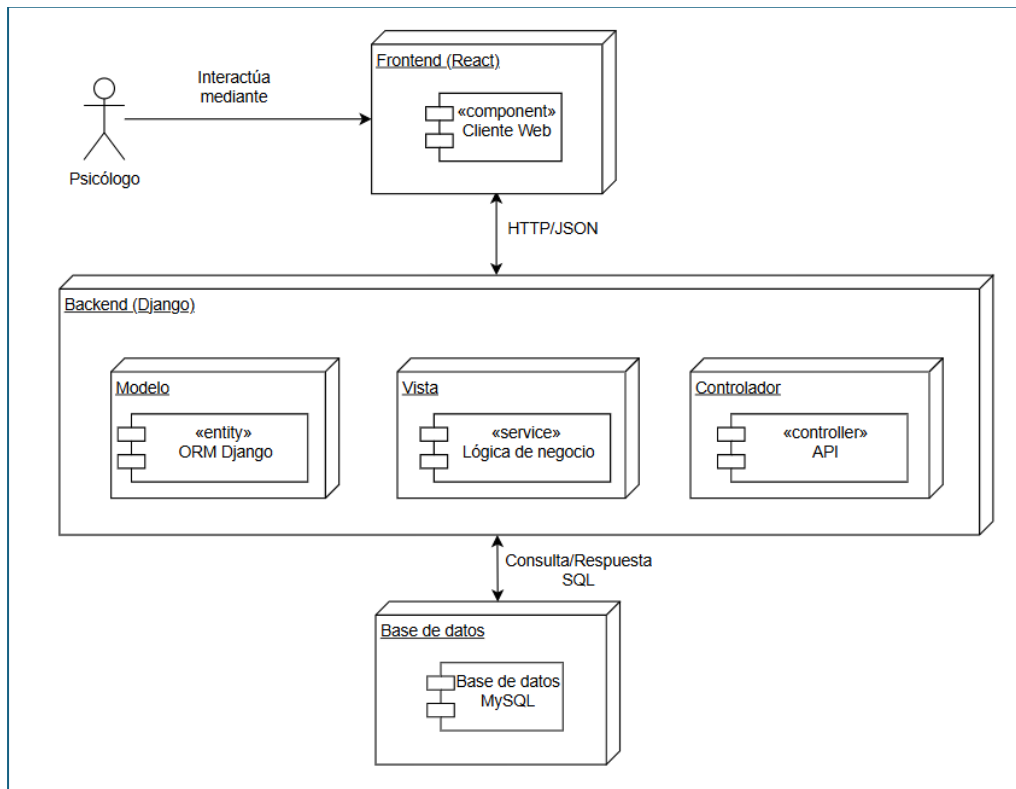
Django funciona internamente con un modelo MVT(Modelo-Vista-Template), pero en nuestro caso el Frontend está totalmente hecho con React, por lo que no ocupamos los templates de Django, significando que el Backend en este contexto funciona únicamente como API, manejando datos, lógica de negocio y exposición de Endpoints.

Provocando que, aunque siga funcionando como MVT a nivel de Framework, en este caso se comporta muy similar a un MVC.

- Modelo: Tablas de la base de datos (Usuarios, trabajadores, Jobs)
- Vista: Lógica funcional (Autenticación, procesamiento de archivos, etc.)
- Controlador: Endpoints REST expuestos mediante Django-Restframework que actúan como controladores al recibir y coordinar la lógica.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Figura 9. Diagrama de arquitectura



Fuente: Elaboración propia

3.2.3. Base de datos

Para guardar los datos dentro del proyecto NeuroBit, se implementó una arquitectura de almacenamiento mixta que combina una base de datos relacional (MySQL) y un sistema de directorios estructurado por trabajador. Esta decisión fue tomada considerando que necesitábamos almacenar datos estructurados de forma consistente y, además, el procurar no sobrecargar la base de datos con archivos pesados como PDF, Excel, CSV, etc.

Inicialmente optamos por utilizar la base de datos integrada de Django (SQLite), ya que esta cumplía con los requerimientos del proyecto, debido a su bajo consumo de recursos y su simplicidad. Pero a medida que se fue desarrollando el proyecto y fuimos integrando procesos más complejos como el sistema de colas para procesar archivos, empezaron a afectar las limitaciones de SQLite, ya que esta no permite la escritura simultánea de múltiples peticiones e imposibilita el dejar trabajos procesándose en segundo plano (lo cual era necesario para que se pudiera seguir utilizando la plataforma mientras se procesaban archivos). Esta limitante nos llevó a utilizar otra de las alternativas consideradas durante el marco teórico, la cual sería MySQL. La elección de esta base de datos se debe principalmente a que permite el trabajar con colas de procesos y no se rompe con múltiples peticiones simultáneas, a la vez que consume menos recursos que PostgreSQL, que era la otra alternativa que consideramos.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

3.2.4. Modelo relacional implementado

Para esta plataforma, necesitamos implementar 3 entidades que estarán asociadas a procesos clave del sistema y se almacenarán en la base de datos.

1. Usuario:

- > Entidad clave para la autenticación y almacenamiento de los usuarios de la plataforma.
- > Sus principales atributos son ID (generado automáticamente por Django), username, email y password (encriptada)

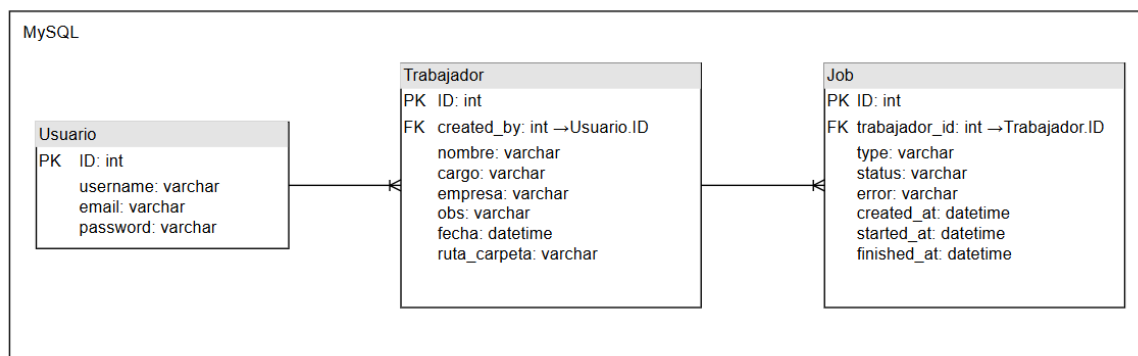
2. Trabajador:

- > Entidad central de la plataforma, representa a cada persona evaluada por Neuro4B y se utiliza para almacenar los archivos de las pruebas Nesplora y EEG
- > Sus principales atributos son ID, nombre, ruta_Carpeta, cargo, empresa, obs (observaciones), fecha y created_by(referencia al usuario creador)
- > Cada trabajador posee una referencia directa al directorio donde se encuentran sus archivos, la cual siempre se nombrará utilizando el nombre, ID y fecha
- > Dentro de su directorio, este puede tener hasta 4 archivos, los cuales siempre tendrán el mismo nombre y pertenecerán al trabajador asociado al directorio.

3. Job:

- > Encargada de implementar el sistema de cola de procesos (se explicarán más adelante en los componentes Backend)
- > Sus atributos principales son type, trabajador_ID, status y created_at, tiene más, los cuales serán explicados más adelante.

Figura 10. Diagrama Base de Datos



Fuente: Elaboración propia

3.2.5. Componentes Backend

El Backend de NeuroBit está dividido en diferentes módulos especializados que trabajan en conjunto para llevar a cabo todas las tareas de procesamiento, autenticación y

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

comunicación con las demás capas del proyecto, a continuación, se mostrarán los principales componentes del proyecto y su implementación real, explicando el funcionamiento de cada uno.

1. Módulo de autenticación:

La autenticación del proyecto está basada en el uso de JWT (JSON Web Tokens) y cookies de acceso, con esto permitimos el mantener una sesión persistente en el navegador sin exponer las credenciales en cada petición. Este módulo cumple con el objetivo específico 1.

Implementación

- Autenticación Usuario:

```
class CookieOrHeaderJWTAuthentication(JWTAuthentication):  
    def authenticate(self, request):  
        header = self.get_header(request)  
        if header is not None:  
            return super().authenticate(request)  
        raw_token = request.COOKIE.get('access')  
        if not raw_token:  
            return None  
        validated = self.get_validated_token(raw_token)  
        return (self.get_user(validated), validated)
```

Su funcionamiento se basa en primero verificar si viene un token de acceso en el header, en caso de no ser así, trata de obtener las cookies de acceso del navegador y autenticar, en caso de no tener ninguno de estos o rechazar la validación, se rechaza la conexión.

- Login Usuario:

Para el inicio de sesión, se llama a la siguiente clase:

```
class EmailLoginAPIView(APIView)
```

Esta clase se encarga de validar el email y la contraseña entregados por el Frontend, utilizando el sistema de autenticación de Django, en caso de ser correctos se generan los tokens de acceso y se devuelven para ser almacenados en las cookies del navegador y así asegurar una sesión persistente.

- Logout Usuario:

```
class LogoutAPIView(APIView):
```

Se encarga de cerrar la sesión actual del usuario, borrando las cookies del navegador y asegurando que termine la persistencia de la sesión.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

2. Módulo de gestión de trabajadores:

Este módulo se encarga de todo lo relacionado a la gestión de trabajadores, interactuando directamente con la base de datos y permitiendo crear, editar, borrar y obtener trabajadores. Este módulo cumple con el objetivo específico 3 y ayuda a cumplir el 8.

Implementación

La implementación se basa en la clase Trabajador, la cual está definido en los modelos de Django y se guarda en la base de datos MySQL, esta clase cuenta con los atributos

```
class Trabajador(models.Model):
    id = models.AutoField(primary_key=True)
    nombre = models.CharField(max_length=255, null=True, blank=True)
    ruta_Carpeta = models.CharField(max_length=500, null=True,
    blank=True)
    cargo = models.CharField(max_length=255, null=True, blank=True)
    empresa = models.CharField(max_length=255, null=True, blank=True)
    obs = models.TextField(max_length=1000, null=True, blank=True)
```

De estos datos, los que son ingresados por el usuario final son el nombre, cargo, empresa y observaciones, el resto se rellena de forma automática.

- **Crear Trabajador**

```
class CrearTrabajadorAPI(APIView):
```

> Se reciben los datos nombre, cargo, empresa y observaciones, junto a los archivos Nesplora y EEG (puede ser solo 1 o ambos)

> Se crea un directorio en donde se almacenan los archivos de los trabajadores, con el nombre en formato nombre + id + fecha

> Se crea el nuevo trabajador en la base de datos con sus respectivos campos, rellenando la fecha actual, el id y la ruta de la carpeta del trabajador

> Se guardan los archivos entregados en la carpeta del trabajador, renombrándolos para tener un formato único, el archivo Nesplora pasa a llamarse PDF.pdf y el EEG pasa a llamarse EEG.csv

> Se retorna un mensaje de éxito en caso de guardarse correctamente o un mensaje de error si es que falla

- **Editar Trabajador**

```
class EditarTrabajadorAPI(APIView):
```

> Se recibe la ID de un trabajador y se asocia a la PK de un trabajador en la BD

> Se clona el directorio con los archivos del trabajador y se guarda en una ruta donde se guardarán todos los registros de trabajadores

> Se edita el trabajador anterior

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

> En caso de modificarse el archivo Nesplora o el archivo EEG, se borra el informe y excel con los datos del Nesplora, ya que estarían desactualizados con los nuevos archivos subidos

- **Borrar Trabajador**

```
class EliminarTrabajadorAPI (APIView):
```

>Se recibe la ID de un trabajador

>Se clona el directorio del trabajador con sus archivos en el registro

>Se borra el trabajador y su carpeta de archivos

- **Listar Trabajadores**

```
Class TrabajadorListAPI (ListAPIView):
```

>Se entregan los trabajadores para que puedan ser mostrados en el Frontend

3. Módulo de gestión de archivos:

Este módulo administra todos los archivos asociados a un trabajador, los cuales son 4 en la versión actual del proyecto, pero se pueden ir añadiendo más en caso de ser necesario. Los archivos que puede tener un trabajador serían el archivo Nesplora (PDF.pdf), el archivo EEG (EEG.csv), el archivo C2PDF.xlsx (Nesplora transformado) y el Informe.docx (informe preliminar). Este módulo cumple con los objetivos específicos 4 y 6 por el almacenamiento de archivos, también cumple con los objetivos 2, 5 y 7 por el procesamiento y es parte del objetivo específico 8.

Implementación

- **Convertir archivo Nesplora**

```
class ConvertirPDFAPI (APIView):
```

>Se recibe el ID de un trabajador y se busca si tiene un archivo PDF Nesplora asociado

>En caso de tenerlo, se comprueba si tiene un archivo C2PDF.xlsx asociado

>Si no tiene un archivo C2PDF.xlsx, empieza la transformación de datos, se llama a las funciones

```
extract_data_from_pdf(pdf_path)  
generate_cognitive_report_excel(data)
```

>Estas funciones se encargan de ejecutar el OCR para extraer la información relevante del archivo y generar un Excel con los datos en formato utilizable

>Se guarda este archivo en el directorio del trabajador asociado

>Se entrega la descarga del archivo esto ocurre tanto si ocurrió conversión o si el archivo C2PDF.xlsx ya existía de antemano con el siguiente código

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

```
ReturnFileResponse(open(pdf_path, 'rb'), as_attachment=True,  
filename=os.path.basename(excel_path))
```

- **Descargar archivo Nesplora**

```
class DescargarPDFAPI (APIView):
```

> Se entrega la descarga del archivo Nesplora de la siguiente manera

```
return FileResponse(open(pdf_path, 'rb'), as_attachment=True,  
filename=os.path.basename(pdf_path))
```

- **Descargar archivo EEG**

```
class DescargarEEGTrabajadorAPI (APIView):
```

> Se entrega la descarga del archivo EEG

```
return FileResponse(open(pdf_path, 'rb'), as_attachment=True,  
filename=os.path.basename(eeg_path))
```

- **Convertir EEG a JSON**

Esta función es clave para implementar el Dashboard en el Frontend, ya que pasa los datos del archivo EEG a un formato listo para utilizar en la generación de gráficos, pasando de tener variables del estilo "ondas alpha", "ondas beta" a tener indicadores del estilo "atención", "estrés", entre otros.

```
CalcularIndicadoresEEGAPI (APIView):
```

> Requiere la ID de un trabajador para obtener su archivo EEG (si es que lo tiene)

> Utiliza fórmulas (entregadas por la empresa patrocinadora) de conversión de ondas cerebrales a emociones

> Se normalizan los resultados obtenidos a una escala de 0 a 100 para facilitar su visualización en el dashboard

> Se calcula el promedio de cada variable

> Transforma los datos a formato JSON

> Entrega la respuesta con un código de éxito si funcionó o un código de error si falló la conversión

- **Generar y descargar informe**

```
class DescargarInformeTrabajadorAPI (APIView):
```

> Funciona utilizando diferentes plantillas de word, las cuales serán rellenas con los datos provenientes de los archivos del trabajador

> Requiere que el trabajador tenga asociados los archivos necesarios para la plantilla deseada

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
 INGENIERÍA EN INFORMÁTICA

- > En caso de que la plantilla seleccionada necesite datos del Nesplora, si el trabajador no tiene un archivo C2PDF, intenta llamar a la función de convertir archivo Nesplora y si esta falla se retorna un código de error.
- > Los datos del EEG se transforman a un formato apropiado para la plantilla en caso de ser requeridos y retorna un código de error en caso de no existir.
- > Luego de verificar que los datos necesarios existen, se recorre un archivo Word y se van rellendo como muestra la siguiente figura

Figura 11. Plantilla para informe preliminar

Ítem	Puntaje	Interpretación
Planificación - Aciertos Globales	<<Planificacion Aciertos>>	<<Planificacion Aciertos interpretacion>>
PD1 - Etapa 1	<<Planificacion Aciertos PD1>>	<<Planificacion Aciertos PD1 interpretacion>>
PT1 - Etapa 1	<<Planificacion Aciertos PT1>>	<<Planificacion Aciertos PT1 interpretacion>>
PC1 - Etapa 1	<<Planificacion Aciertos PC1>>	<<Planificacion Aciertos PC1 interpretacion>>
PD2 - Etapa 2	<<Planificacion Aciertos PD2>>	<<Planificacion Aciertos PD2 interpretacion>>
Asignación Tiempo PD1	<<Planificacion Tiempo Asignación PD1>>	<<Planificacion Tiempo Asignación PD1 interpretacion>>
Asignación Tiempo PT1	<<Planificacion Tiempo Asignación PT1>>	<<Planificacion Tiempo Asignación PT1 interpretacion>>
Asignación Tiempo PC1	<<Planificacion Tiempo Asignación PC1>>	<<Planificacion Tiempo Asignación PC1 interpretacion>>
Asignación Tiempo PD2	<<Planificacion Tiempo Asignación PD2>>	<<Planificacion Tiempo Asignación PD2 interpretacion>>
Asignación Tiempo PT2	<<Planificacion Tiempo Asignación PT2>>	<<Planificacion Tiempo Asignación PT2 interpretacion>>
Asignación Tiempo PC2	<<Planificacion Tiempo Asignación PC2>>	<<Planificacion Tiempo Asignación PC2 interpretacion>>

Fuente: Equipo NeuroBit

- > Este informe queda guardado en el directorio de archivos del trabajador
- > Se entrega la descarga del archivo en formato Word, esto ocurre tanto si se ejecuta la generación de informe o si este ya existía de antes, en caso de ya existir el informe se descarga directamente sin pasar por la generación nuevamente

```
return FileResponse(open(destino_path, 'rb'), as_attachment=True,
filename=os.path.basename(destino_path))
```

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

4. Cola de procesos

Este módulo se encarga de administrar las tareas pesadas del sistema, en este caso el convertir archivo Nesplora y el generar informe, y se encarga de que vayan ejecutándose secuencialmente por orden de llegada (FIFO) en lugar de ejecutarse simultáneamente, ya que esto provoca bloqueos en el Backend, fallos de rendimiento general o que directamente se sobrecargue y se caiga.

Implementación

La implementación de la cola de procesos se basa en una clase llamada Job, la cual se encarga de almacenar el ID del trabajador asociado, el estado y el tipo de proceso que se va a ejecutar, como se ve en este extracto de código

```
class Job(models.Model):
    TYPE_CHOICES = [
        ("convertir_pdf", "Convertir PDF"),
        ("generar_informe", "Generar Informe"),
    ]
    type = models.CharField(max_length=32, choices=TYPE_CHOICES)
    trabajador_id = models.IntegerField()
    status = models.CharField(max_length=16, default="queued")
    created_at = models.DateTimeField(auto_now_add=True)
    started_at = models.DateTimeField(null=True, blank=True)
    finished_at = models.DateTimeField(null=True, blank=True)
    error = models.TextField(blank=True, default="")
```

Cada solicitud pesada queda registrada como un nuevo Job, el cual viene en estado base "queued", cuando se está ejecutando pasará a estado "running", y finalmente terminará en estado "done" o "error" según si se completa correctamente o no.

- **Añadir Job a la cola**

```
def enqueue_and_run(self, job_type, trabajador_id, runner_fn):
```

- > Se crea un nuevo Job en la base de datos con el tipo de tarea asignado (convertir Nesplora o generar informe), y el ID del trabajador asociado
- > Se inicia en estado "queued"
- > Se asigna la hora y fecha del momento en que fue creada
- > Se llama a la función para ejecutar siguiente Job (ver siguiente componente)

- **Ejecutar siguiente Job en cola**

- > Se verifica que no haya un Job en estado running, en caso de haberlo espera un tiempo y vuelve a comprobar

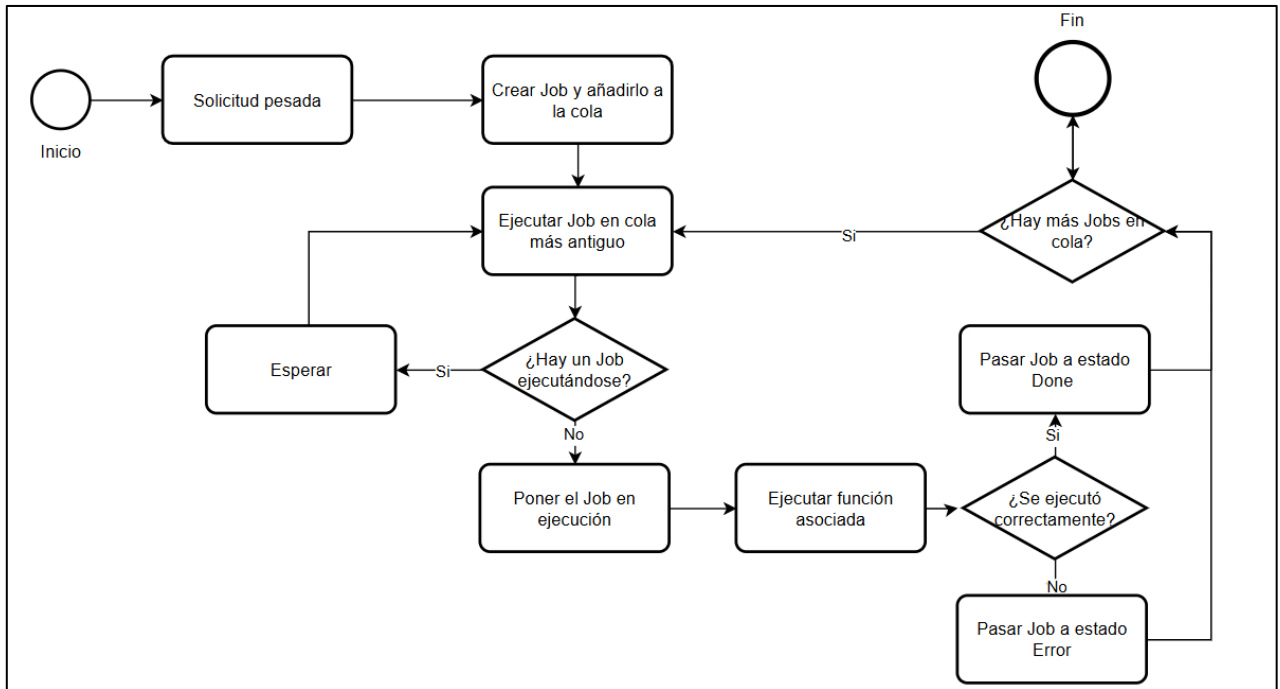
- > Si no existe un proceso activo, se busca el Job más antiguo en la base de datos

```
job = Job.objects.filter(status="queued").order_by("created_at").first()
```

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- > Se asigna el estado running
- > Se empieza ejecutar la función asociada a ese Job
- > Una vez termina pasa a estado done y trata de ejecutarse el siguiente Job en estado queued
- > En caso de no haber ninguno, termina la ejecución de la cola

Figura 12. Diagrama de flujo Jobs

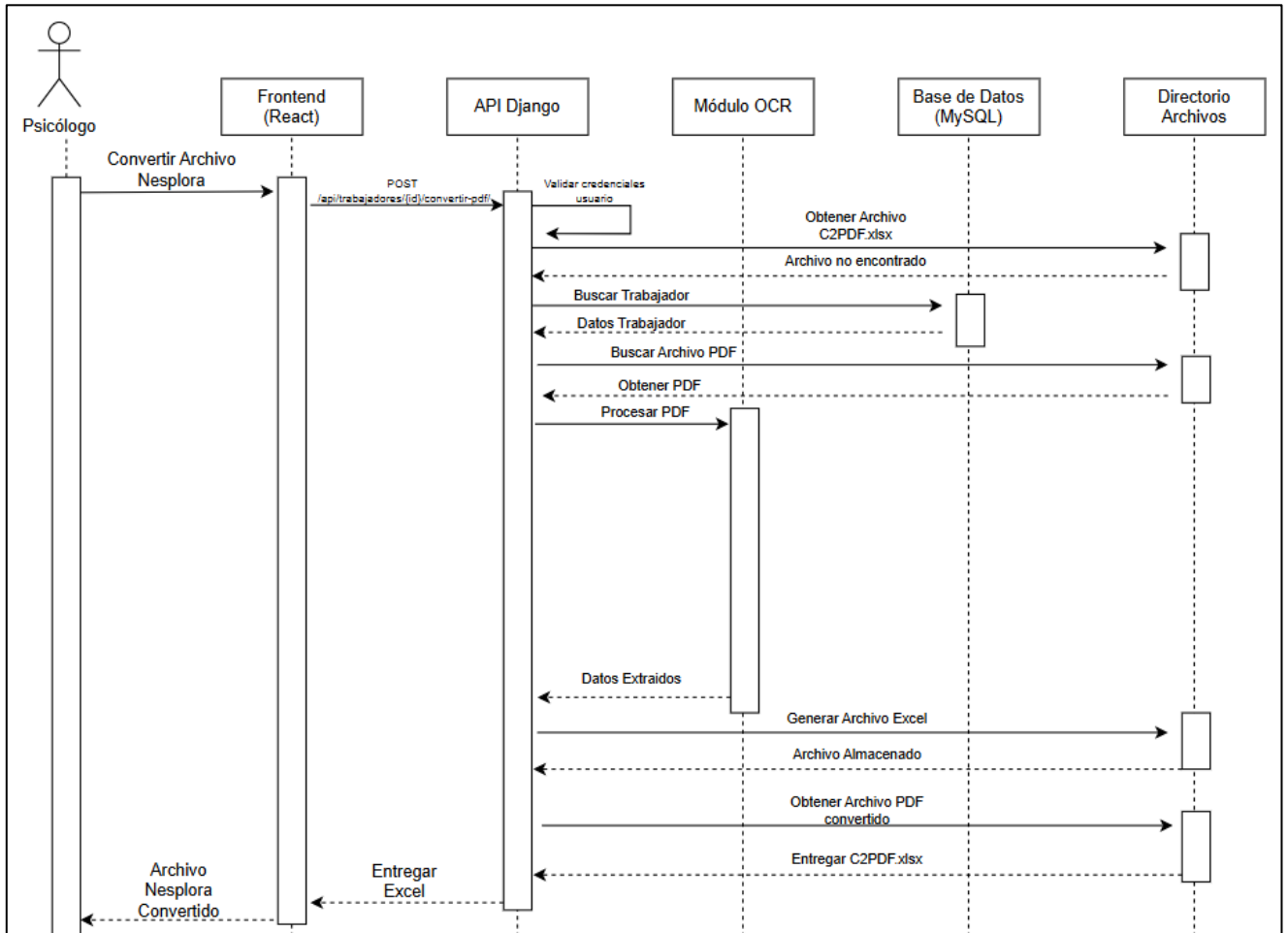


Fuente: Equipo NeuroBit

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
 INGENIERÍA EN INFORMÁTICA

Finalmente, el flujo completo de una petición normal para realizar una operación normal de la plataforma como sería el obtener un archivo Nesplora convertido a un formato listo para el análisis queda representado en la siguiente figura:

Figura 13. Diagrama de secuencia transformación archivo Nesplora



Fuente: Equipo NeuroBit

Cabe aclarar que, aunque no sale explícitamente en el gráfico, al ser una tarea pesada, se ejecutará la cola de procesos (Figura 12) para evitar que se ejecuten más de los permitidos de forma simultánea.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

3.2.6. Exposición de servicios

El Backend del proyecto expone todas las funcionalidades necesarias a través de una API REST construida con Django-Restframework, actuando como comunicación entre el Frontend y el Backend. Cada módulo descrito anteriormente se encuentra encapsulado y expuesto mediante uno o varios Endpoints, permitiendo que sean accedidos mediante su respectiva URL.

Se utilizan las prácticas habituales de diseño REST, utilizando métodos GET para obtener información, POST para crear o ejecutar acciones y DELETE para eliminar recursos. Es importante mencionar que cada uno de estos Endpoints autentifica al usuario con el módulo de autenticación presentado anteriormente, asegurando el que solo los usuarios con sesión iniciada puedan utilizarlos.

Figura 14. Tabla de Endpoints

Endpoint	Método	Descripción	Autenticación
/api/login/	POST	Iniciar sesión	No
/api/logout/	POST	Cerrar sesión	Si
/api/listar/	GET	Listar trabajadores	Si
/api/crear/	POST	Crear trabajador	Si
/api/eliminar/<pk>/	DELETE	Eliminar trabajador	Si
/api/editar/<pk>/	PUT/PATCH	Editar trabajador	Si
/api/trabajador/<pk>/descargar-pdf/	GET	Descargar PDF Nesplora	Si
/api/trabajador/<pk>/descargar-eeg/	GET	Descargar EEG	Si
/api/trabajador/<pk>/descargar-excel/	GET	Descargar C2PDF (Nesplora convertido)	Si
/api/descargar-informe/<pk>/	GET	Descargar informe	Si
/api/trabajadores/<pk>/convertir-pdf/	POST	Transformar archivo Nesplora	Si
/api/trabajador/<pk>/indicadores/	GET	Procesar EEG y devolverlo como JSON	Si

Fuente: Elaboración propia

Cada uno de estos Endpoints está vinculado a una clase API de las presentadas en los componentes y sus respuestas son entregadas en formato JSON o como archivos descargables según el contexto.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

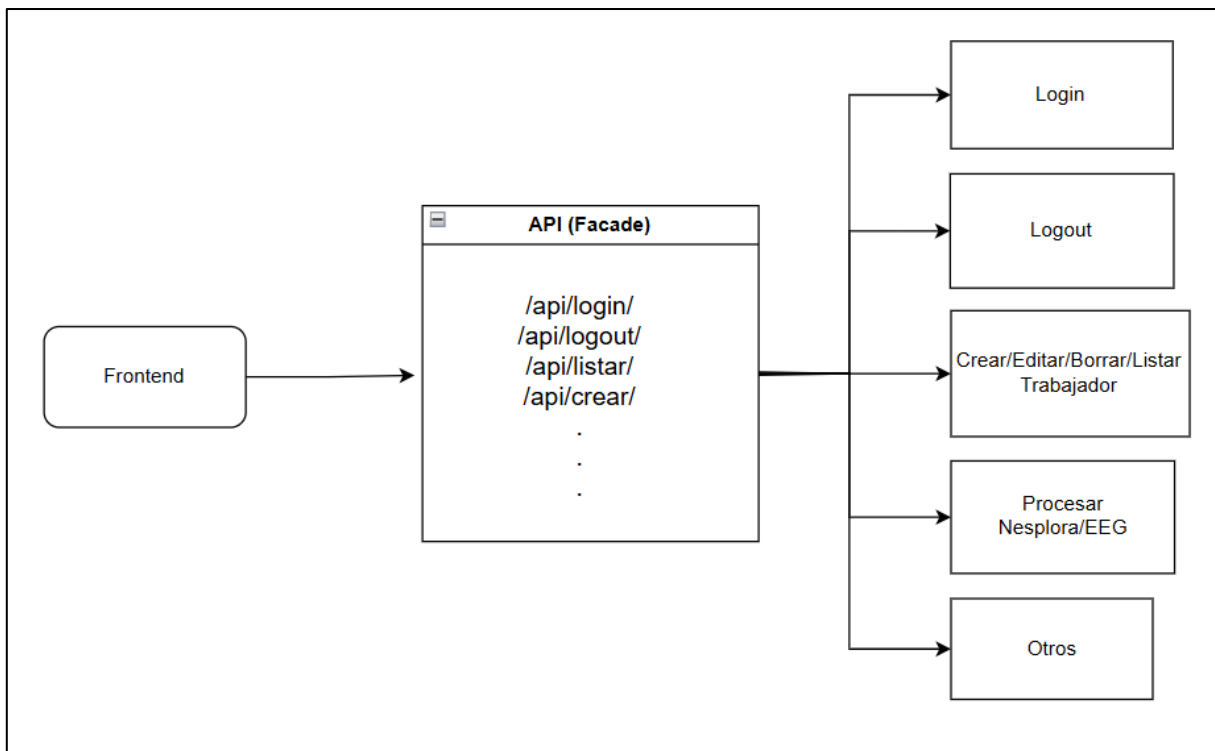
3.2.7. Patrones de diseño

El desarrollo del Backend de la plataforma NeuroBit fue realizado aplicando varios patrones de diseño y buenas prácticas para el desarrollo de software para así obtener un sistema modular y fácil de escalar, para esto nos basamos en algunos de los patrones descritos por Refactoring Guru [17].

- Patrón Facade:

Este patrón sugiere el crear una interfaz simple para interactuar con sistemas o funciones complejas, para así conseguir una forma más fácil de interactuar con el sistema sin necesidad de siquiera conocer los componentes del sistema como tal, en nuestro caso, las vistas API fueron diseñadas siguiendo este patrón, consiguiendo así que el Frontend pueda hacer peticiones http simples con datos mínimos y no tener que conocer cómo funciona el sistema internamente, ayudando enormemente a separar las responsabilidades y a trabajar en equipo de forma más eficiente, ya que es mucho más sencillo para el Frontend operar pensando en llamar un Endpoint específico con algunos datos mínimos que el interactuar directamente con el sistema de procesamiento de archivos u otros componentes del Backend.

Figura 15. Patrón Facade



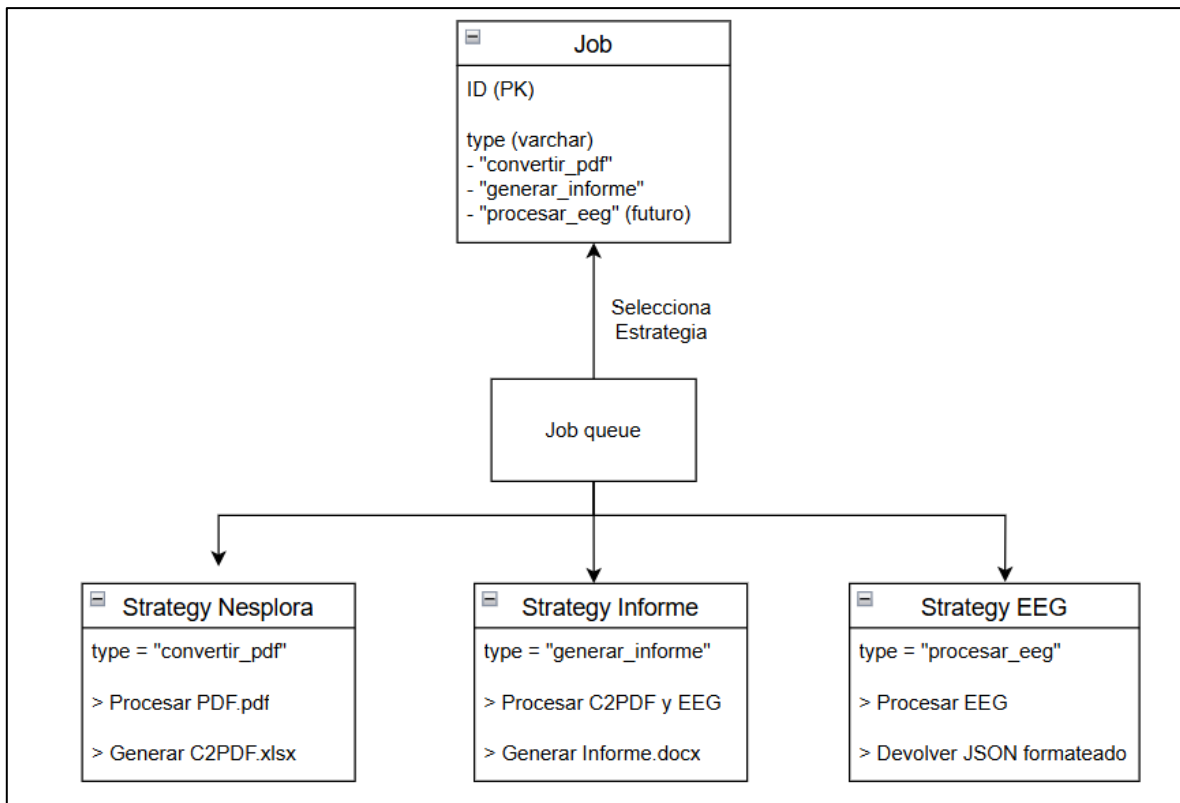
Fuente: Elaboración propia

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
 INGENIERÍA EN INFORMÁTICA

- Patrón Strategy

Este patrón hace referencia al definir una familia de algoritmos con un resultado "similar" pero con procesos diferentes y llamarlas según el contexto adecuado, en este caso, tanto procesar EEG, como procesar Nesplora y crear el informe, son 3 procesos que se encargan de tomar un archivo, realizarle operaciones y entregar un resultado, además de que las 3 son procesos de alto consumo de recursos que deben ser trabajadas mediante Jobs. Debido a la similitud del resultado esperado por estas funciones, podemos aplicar el patrón Strategy y, definir un campo llamo type en los Jobs, para así simplemente cambiar esa variable según el proceso que sea requerido en cada caso. Actualmente el procesar EEG no está incluido entre los posibles tipos del Job, pero se puede añadir perfectamente, al igual que añadir futuros procesamientos de nuevos archivos.

Figura 16. Patrón Strategy



Fuente: Elaboración propia

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- Patrón Command

Este patrón se refiere al encapsulamiento de una función o un proceso para utilizarlo como objeto en lugar de instrucción, es decir, en vez de ejecutarse al momento, este puede ser almacenado y ser ejecutado cuando sea requerido. Esto aplica perfectamente al uso de Jobs, ya que su existencia se basa en este concepto, al pasar las funciones de procesamiento pesado a ser parte de una entidad de la base de datos para ejecutarse cuando no haya otro en proceso, para así evitar sobrecargar al servidor.

3.2.8. Buenas prácticas generales

Además de utilizar los patrones de diseño mencionados anteriormente, se aplicaron diferentes buenas prácticas de desarrollo al proyecto, entre las cuales encontramos:

- Funciones acotadas y de responsabilidad única

Cada función del código cumple con su propia funcionalidad separada, promoviendo que el proyecto sea modular y su correcto funcionamiento, ya que ayuda a detectar errores y el realizar cambios sin modificar módulos no relacionados.

- Estandarización

Todos los archivos de los trabajadores dentro de sus directorios son renombrados a un formato estándar (PDF.pdf, EEG.csv, etc.), facilitando el orden y la reutilización del código.

- Uso correcto de códigos HTTP

Se manejan respuestas HTTP tales como 200, 400, 500, entre otros. según corresponda, para facilitar la depuración y rastreo de errores en el código.

3.2.9. Integración con el sistema general

Como se ha visto a lo largo de este capítulo, el Backend de NeuroBit con el resto del sistema se realiza mediante un sistema de comunicación desacoplado basado en consultas HTTP y respuestas en formato JSON, siguiendo el modelo de cliente-servidor. Esto permite que el Frontend acceda a las funcionalidades sin requerir acceso directo a la lógica interna del sistema. Cada módulo del Backend expone Endpoints específicos los cuales reciben solicitudes HTTP y entregan una respuesta estructurada.

A continuación, veremos ejemplos de los principales tipos de comunicación entre el Backend y el Frontend.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Figura 17. Tabla ejemplos comunicación Frontend y Backend

Funcionalidad	Request (Frontend)	Respuesta (Backend)
Autenticación	POST /api/login/ { "email": "usuario@correo.cl", "password": "*****" }	{ "message": "Inicio de sesión exitoso", "user": { "id": 1, "username": "admin" } }
Obtener trabajadores	GET /api/trabajadores/	[{ "id": 1, "nombre": "Nombre", "empresa": "NeuroBit", "cargo": "Operario", "ruta_Carpeta": "archivos/Nombre.1.2025-01-10/" },]
Conversión archivo Nesplora	POST /api/trabajadores/1/convertir-pdf/	{ "message": "Job creado correctamente", "job_id": 57, "status": "queued" }
Procesamiento archivo EEG	GET /api/trabajador/1/indicadores/	{ "atencion": 72, "estres": 41, "fatiga": 38, ... }

Fuente: Elaboración Propia

Este mismo formato sigue para el resto de las peticiones. Es importante mencionar que estos son ejemplos de peticiones válidas, ya que, si el usuario no estuviera autenticado, respondería con un código de error. En los casos de peticiones de descarga de archivos, se retornaría la descarga del archivo como tal y un código 200 (éxito) en caso de cumplirse correctamente.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

3.3. Detalles de la codificación y desarrollo

3.3.1. Metodología de trabajo

El desarrollo del proyecto fue dividido entre 6 diferentes miembros del equipo, cada uno con sus funciones y especializaciones, en donde a lo largo de 3 Sprints, dividimos todas las actividades necesarias para llevar a cabo la plataforma NeuroBit. En este caso, yo fui el encargado principal de desarrollar el Backend del proyecto y fui responsable de realizar la mayoría de las tareas asociadas a este, a continuación, veremos lo realizado en cada Sprint por los miembros del equipo. Todo esto fue documentado y repartido a través de JIRA, una plataforma de gestión de proyectos [20].

Durante este punto, se describirán los principales avances relacionados al Backend del proyecto separados por los diferentes Sprints realizados.

- **Sprint 1**

Durante el primer Sprint se establecieron las bases del sistema, creando las funcionalidades esenciales para el funcionamiento del Backend.

Tareas realizadas:

- Definición de la entidad Trabajador, diseñada para tener todos los campos requeridos para guardar toda la información relevante junto a la ruta donde estarán guardados sus archivos
- Implementación completa del CRUD de trabajadores
- Configuración de un sistema de almacenamiento de archivos asociados a los trabajadores
- Integración de la primera fase del OCR para la extracción de datos de la prueba Nesplora
- Creación de Endpoints para todas las funcionalidades creadas hasta el momento

- **Sprint 2**

El segundo sprint fue principalmente dedicado a implementar medidas de autenticación de usuarios y seguridad de la plataforma, definiendo el sistema de usuarios y la comprobación de credenciales. Adicionalmente se integró la generación de informes preliminares y se fue mejorando el sistema OCR para reducir fallos.

Tareas realizadas:

- Generación de la clase usuario
- Creación e implementación del módulo de autenticación
- Integración del módulo de generación de informes
- Reestructuración de gran parte de código del Backend para asegurar su escalabilidad y limpieza
- Creación del método para pasar el EEG a formato JSON

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- **Sprint 3**

El último Sprint se centró en implementar la ejecución secuencial de tareas pesadas, es decir, las operaciones de conversión del archivo Nesplora y la creación de informes. Esto es crucial para evitar bloqueos o fallos debido a el consumo excesivo de recursos de forma simultánea.

Tareas realizadas:

- Migración de SQLite a MySQL
- Creación de la entidad Job con los campos necesarios para poder implementar la cola de procesos
- Implementación de una cola FIFO para las tareas pesadas
- Integración de la generación de informes y la conversión de archivos Nesplora para ejecutarse de forma secuencial en vez de directamente cuando son llamados

3.3.2. Gestión de repositorio y gestión del código

Para separar responsabilidades y poder trabajar en paralelo, se creó una organización privada en GitHub, para centralizar todas las versiones del código con sus diferentes versiones. Este repositorio solo es accesible por miembros del equipo para evitar filtrar información relevante en caso de que encuentren el repositorio, pero de todas formas se evitó el subir archivos con nombres de trabajadores reales para no comprometer su privacidad.

En esta organización, separamos el proyecto en 3 repositorios principales:

1. **Repositorio Backend:** Aquí estaba todo el código correspondiente al Backend de NeuroBit, y posee varias ramas de trabajo separadas. La rama principal es la main, donde se subían las versiones ya testeadas y funcionales, y se utilizaba para desplegar el proyecto. Luego, tenemos una rama llamada release, donde se hacían las fusiones de ramas antes de subirlas al main y se testeaba que no diera error o incompatibilidades con los componentes anteriores. Y luego hay varias ramas donde los integrantes del equipo hacían cambios e incorporaban funcionalidades a medida avanzaba el proyecto antes de subirse a release y posteriormente a la rama principal. Se puede ver mejor en la Figura 18.
2. **Repositorio Frontend:** En este repositorio se subió el código correspondiente al Frontend de NeuroBit, se organizó de forma similar al Backend, pero aquí se trabajó con React en lugar de Django.
3. **Repositorio de Orquestación:** Este repositorio contenía el docker-compose, además de hacer referencia tanto al repositorio Backend, como al Frontend. Este repositorio era encargado de todo lo correspondiente al despliegue del proyecto.

Es importante mencionar que tanto el repositorio Backend, como el Frontend tenían sus correspondientes archivos Dockerfile, que eran necesarios para levantar el proyecto de forma cómoda y automatizada.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
 INGENIERÍA EN INFORMÁTICA

Figura 18. Repositorio Backend NeuroBit

Branch	Updated	Check status	Behind / Ahead	Pull request
main 	 6 months ago	✓ 1 / 1	Default	 ...
develop 	 5 months ago		0 12	 ...
feature/pytest 	 5 months ago		0 11	 ...  #9
feature/webhookTest 	 6 months ago		0 5	 ...  #8
hotfix 	 6 months ago	✓ 1 / 1	0 0	 ...
release 	 6 months ago	✓ 1 / 1	0 0	 ...
feature/queue-test 	 6 months ago		101 0	 ...
feature/seguridadBD 	 6 months ago		107 0	 ...
feature/paralel-ocr 	 6 months ago		59 0	 ...
feature/test_postgre 	 6 months ago		107 1	 ...
feature/pruebas-ocr 	 6 months ago		56 0	 ...

Fuente: Equipo NeuroBit

3.4. Integración con otros componentes

Además de la comunicación con el Frontend mediante API que ya fue explicado anteriormente, el Backend del proyecto también requiere integrarse con otro componente desarrollado por otro miembro del equipo, el cual es el módulo de procesamiento del archivo Nesplora ubicado en una carpeta nombrada como "procesar_archivo" y que contiene el sistema OCR como tal.

La importación es realizada mediante la importación interna de módulos Python, ya que "procesar_archivo" aunque es un programa independiente, se encuentra alojado en el repositorio general del Backend.

Este componente cuenta con varios sub-modulos especializadas, las cuales serían:

- data_extractor.py: Se encarga del OCR y extracción de datos del archivo Nesplora.
- excel_generator.py: Transforma los datos extraídos en un archivo Excel.
- pdf_procesor.py: Coordina procesamiento de tablas.
- tables.py: Define la estructura de datos y organiza la información.
- main.py: Programa para ejecutar el procesamiento.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Para integrar este programa con el Backend, se realizan importaciones de las funciones necesarias para realizar el procesamiento del archivo Nesplora y utilizarlas cuando sea necesario, las importaciones se ven de la siguiente manera:

```
from procesar_archivo.data_extractor import extract_data_from_pdf
from procesar_archivo.excel_generator import
generate_cognitive_report_excel
```

Una vez importadas, ya pueden ser utilizadas por el Backend, como ocurre cuando queremos convertir un archivo asociado a un trabajador, por ejemplo:

```
data = extract_data_from_pdf(pdf_path)
excel_path = generate_cognitive_report_excel(data, output_path)
```

Esto permite al Backend trabajar como coordinador de ejecución, mientras que el programa "procesar_archivo" funciona como el motor de procesamiento. Además, gracias a que las funciones de este programa están diseñadas para ser trabajadas de forma independiente, se realiza una implementación en el Backend limpia y sin romper el funcionamiento interno.

3.5. Pruebas y validación

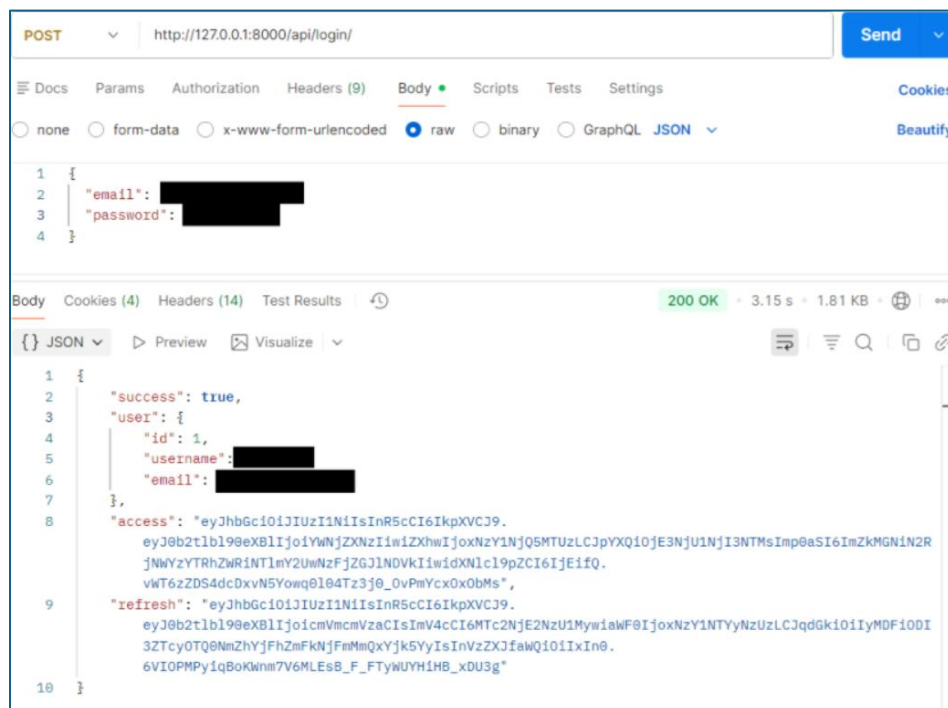
En esta sección describiremos las pruebas realizadas para verificar que el Backend NeuroBit cumple con los requerimientos planteados en la sección 3.1. Para comprobar que los Endpoints expuestos efectivamente cumplen con su función, se utilizó la herramienta Postman, la cual permite interactuar y probar el funcionamiento de servicios web [21]. Con Postman se revisaron los diferentes componentes Backend de forma manual para revisar su código HTTP retornado, la estructura de respuesta y la restricción de acceso a funciones protegidas mediante autenticación.

3.5.1. Estrategias de Testing aplicadas a los componentes

Se aplicó una revisión manual a cada uno de los Endpoints presentados anteriormente en la Figura 14 y se verificó que entregaban las salidas esperadas. A continuación, se muestran ejemplos del método de testing de algunos componentes.

The screenshot shows the Postman interface with a POST request to `http://127.0.0.1:8000/api/login/`. The request body is a JSON object: `{ "email": "admin@correo.cl", "password": "123456" }`. The response is a 400 Bad Request with a status bar indicating "400 Bad Request" in red. The response body is a JSON object: `{ "success": false, "error": "Credenciales inválidas" }`. The interface includes tabs for Docs, Params, Authorization, Headers (8), Body, Scripts, Tests, and Settings. The Body tab is selected, and the response is displayed in the bottom panel.

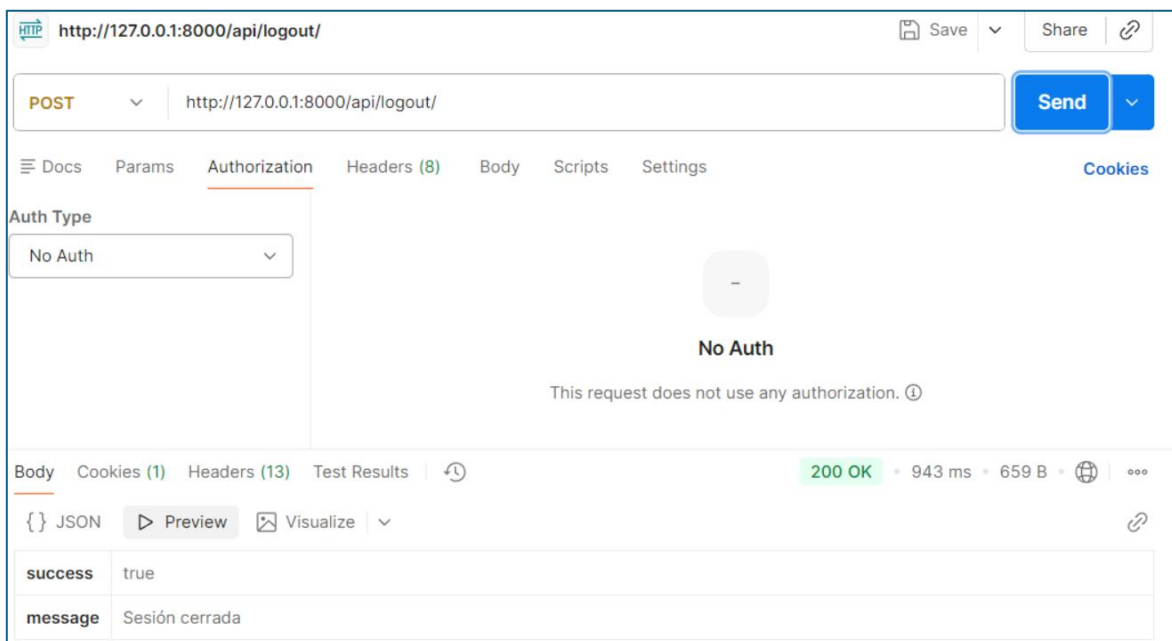
Figura 20. Login aprobado con credenciales censuradas



37

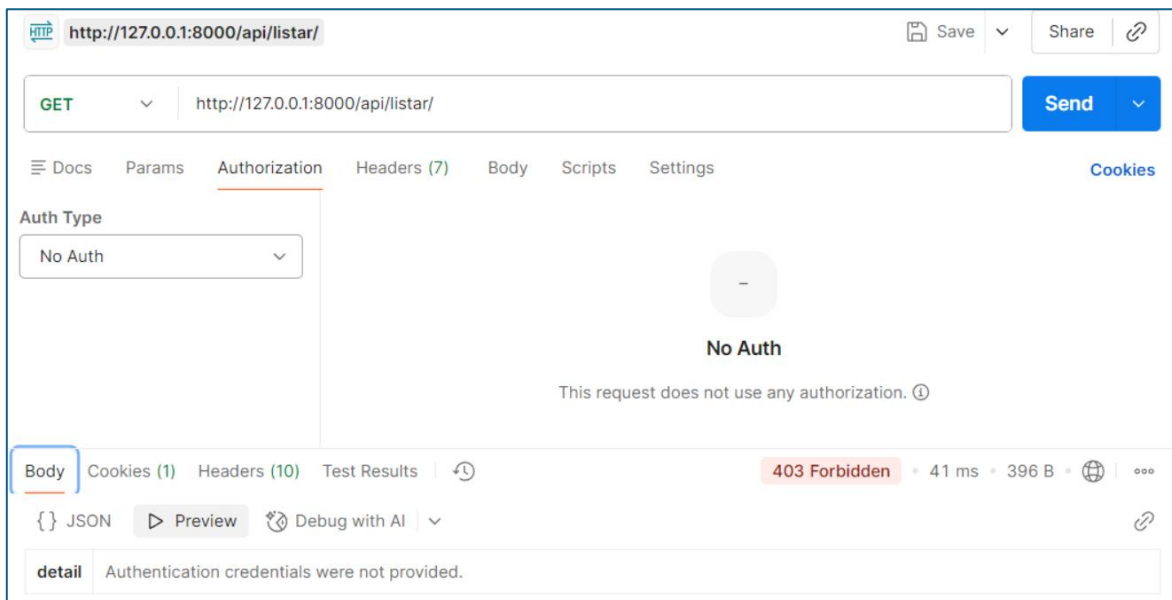
DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Figura 21. Logout



Fuente: Elaboración Propia

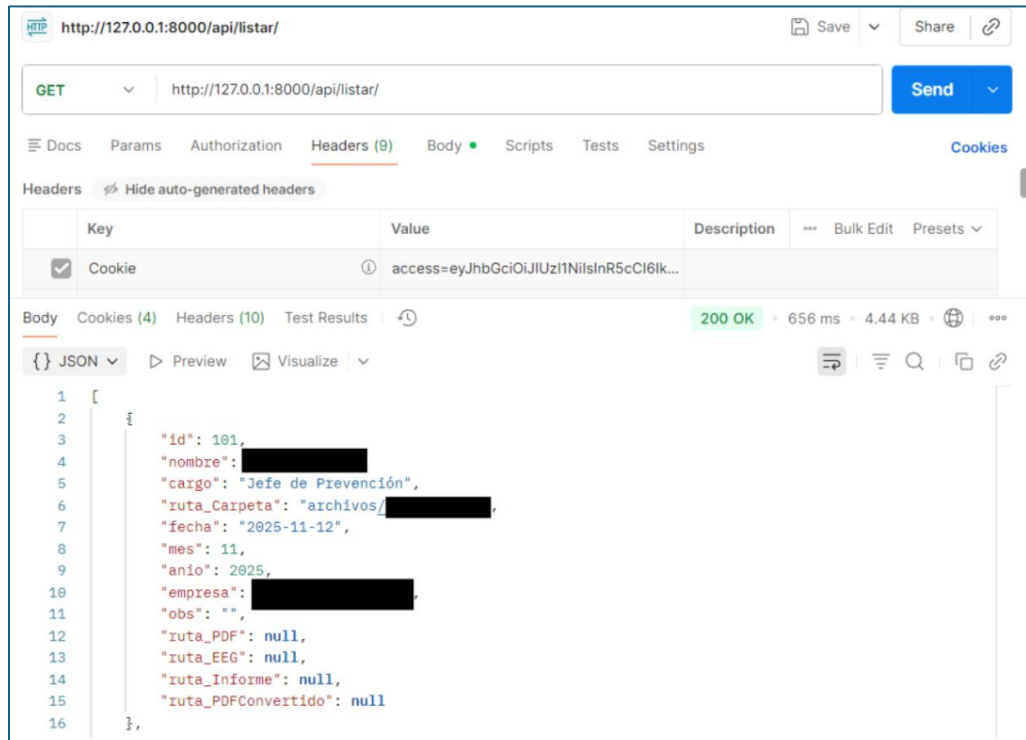
Figura 22. Listar trabajadores denegado por credenciales faltantes



Fuente: Elaboración Propia

Figura 23. Listar trabajadores exitoso, con datos personales censurados

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA



Fuente: Elaboración Propia

3.5.2. Resolución de bugs y optimización

Durante el desarrollo del proyecto se detectaron y corrigieron múltiples fallas, principalmente encontradas durante las pruebas utilizando Postman.

Principales tipos de problema abordados y sus soluciones:

- Bloqueos al ejecutar múltiples tareas pesadas simultáneamente
 - Esto ocurría al ejecutar varias conversiones de archivos Nesplora o generación de informes a la vez y fue solucionado implementando la cola de procesos.
- Estados inconsistentes al editar archivos
 - Los archivos Nesplora convertidos y los informes preliminares quedaban desactualizados al editar los PDF y EEG del trabajador, por lo que decidimos que cuando estos fueran modificados, se eliminaran tanto el informe como el Excel, ya que estos dependen directamente de los archivos del trabajador.
- Problemas de mantenibilidad
 - Esto ocurría al tener archivos con nombres muy diferentes entre los trabajadores y fue solucionado al estandarizar los archivos, renombrándolos a PDF.pdf, EEG.csv, informe.docx o C2PDF.xlsx según fuera necesario.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

4. Conclusiones

Durante el desarrollo del proyecto NeuroBit, experimentamos diversas complicaciones y desafíos que fuimos superando como equipo, como problemas al dividir las cargas de trabajo que siempre había miembros del equipo con tareas significativamente más pesadas y al principio fue complicado el organizarnos como tal. Pero mientras avanzábamos fuimos aprendiendo de estas fallas e íbamos mejorando el trabajo en equipo, perdiendo menos tiempo en las reuniones diarias, asignando mejor las tareas y en general trabajando mejor. Fue una experiencia exigente, pero pudimos llevarla a cabo exitosamente y puedo decir que todos los miembros del equipo cumplieron muy bien con sus roles y nos complementábamos bien. Si tuviera que cambiar algo, sería la organización inicial que tuvimos, ya que, de los primeros periodos del trabajo, hay muy pocos procesos documentados como tal y cuando los necesitábamos nos generaba problemas, pero creo que nos sirvió muy bien como experiencia de lo importante que es el trabajar de manera organizada y estructurada. También la separación de tareas es bastante mejorable, pero nada terrible que no se pueda mejorar. Ahora, a nivel de proyecto, la plataforma NeuroBit aporta significativamente al uso del tiempo de los psicólogos de Neuro4B, logrando que su trabajo sea mucho más rápido y que, por ende, estos puedan realizar las evaluaciones de neuro-competencias de forma más eficiente y evitar posibles incidentes que podrían poner en riesgo la seguridad e incluso la vida de múltiples personas, aportando un nivel de escalabilidad mucho mayor a un proceso muy limitado por la necesidad de interacción humana al extraer información. A nivel futuro, esta plataforma podría contar con más tipos de informes o documentos que procesar, además de crear nuevos tipos de informes para ayudar en las tareas de análisis de los psicólogos, también la plataforma podría ser utilizada con diferentes empresas o áreas con sus respectivos archivos en caso de desvincularnos con Neuro4B por algún motivo, esto sería bastante sencillo debido a que la infraestructura general ya estaría lista y si quitamos el tema de extracción de archivos Nesplora o EEG, queda una plataforma de gestión de archivos genérica que puede ser utilizada en diferentes contextos.

En cuanto a las limitaciones del desarrollo, la plataforma fue diseñada considerando los usuarios de Neuro4B y sus necesidades actuales, por lo que esta no está diseñada para enfrentarse a una cantidad masiva de peticiones y se tendría que trabajar en su optimización y ampliación de recursos en caso de que la empresa crezca. Además, el procesamiento de información está sujeto al formato actual de información esperada por la empresa, provocando que no soporte distintos formatos de pruebas de neuro-competencias, por lo que a futuro sería conveniente adaptar la plataforma para que se ajuste a distintos tipos de archivos. También sería interesante a futuro evolucionar la plataforma a una solución flexible que sea utilizable por distintas empresas y no solo por Neuro4B.

Trabajar con el equipo NeuroBit aportó mucho a mi desarrollo como profesional y me proporcionó experiencia importante para mi futuro como ingeniero informático, además de conocer personas diferentes de la industria como los patrocinadores y un primer vistazo al funcionamiento de equipamientos de análisis de ondas cerebrales e instrumentos de realidad virtual, con los que nunca había tenido contacto.

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

5. Agradecimientos

Quisiera agradecer a mi familia por el apoyo que me han dado durante toda mi vida, a mi pareja por estar siempre presente y acompañándome durante estos últimos años, fue un apoyo gigante para mí vida en general, también agradezco a todas las amistades que formé en mi vida universitaria, ya que me ayudaron mucho a desarrollarme como persona y como profesional. Por último, muchas gracias al equipo NeuroBit por ponerle tanto empeño al proyecto y aunque hubo varias complicaciones, agradezco que hayan trabajado tan bien en general.

6. Referencias

- [1] M. B. Steinborn, «frontiers,» 8 Mayo 2024. [En línea]. Available: <https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1351625/full>
- [2] C. Drumond, «atlassian,» [En línea]. Available: <https://www.atlassian.com/es/agile/scrum>
- [3] A. Ken, «gluo,» [En línea]. Available: <https://www.gluo.mx/blog/backend-que-es-y-para-que-sirve>
- [4] A. Priono, «nexalab,» [En línea]. Available: <https://nexalab.io/blog/development/popular-backend-languages/>
- [5] Shakers, The home of freeworking, «Shakers,» [En línea]. Available: <https://blog.shakersworks.com/tendencias-desarrollo-backend-empresas>
- [6] R. Ramos, «openmetal,» [En línea]. Available: <https://openmetal.io/resources/blog/serverless-computing-understanding-when-and-why-to-use-it/>
- [7] Deorwine, [En línea]. Available: <https://deorwine.com/blog/is-serverless-architecture-becoming-so-popular-for-app-development/>
- [8] C. Muñoz, «pragma,» [En línea]. Available: <https://www.pragma.co/es/blog/que-es-un-microservicio-conceptos-claves-y-ejemplos>
- [9] F. d. I. Torre, «Profile,» [En línea]. Available: <https://profile.es/blog/monolitos-vs-microservicios-que-arquitectura-elegir/>
- [10] I. Vyas, «citrusbug,» [En línea]. Available: <https://citrusbug.com/blog/backend-development-trends/>
- [11] P. Verma, «medium,» [En línea]. Available: <https://medium.com/@pooja.verma2171/pros-and-cons-of-front-end-and-back-end-development-43b6dc84ce4f>

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

- [12] O. Blancarte, «reactive programming,» [En línea]. Available: <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas>
- [13] A. Deyimar, «hostinger,» [En línea]. Available: <https://www.hostinger.com/es/tutoriales/que-es-react>
- [14] M. Orlov, «maxim orlov,» [En línea]. Available: <https://maximorlov.com/why-storing-files-database-bad-practice/>
- [15] inray, «OPC router,» [En línea]. Available: <https://www.opc-router.com/what-is-docker/>
- [16] M. Martínez, «profile,» [En línea]. Available: <https://profile.es/blog/patrones-de-diseno-de-software/>
- [17] Refactoring guru, «Refactoring guru,» [En línea]. Available: <https://refactoring.guru/es/design-patterns>
- [18] Anaconda, «Anaconda,» [En línea]. Available: <https://www.anaconda.com/topics/python-for-data-analysis>
- [19] django, «django,» [En línea]. Available: <https://docs.djangoproject.com/en/5.2/topics/auth/>
- [20] Atlassian, [En línea]. Available: <https://www.atlassian.com/es/software/jira/guides/getting-started/introduction>
- [21] Javier, «formadoresit,» [En línea]. Available: <https://formadoresit.es/que-es-postman-cuales-son-sus-principales-ventajas/>