BioAnalyzer: Plataforma Web para el Análisis de Ritmos Circadianos

1. Introducción

BioAnalyzer es una aplicación web creada para facilitar el análisis de ritmos circadianos en datos biológicos de forma sencilla y robusta. Surge como respuesta a la necesidad detectada en entornos académicos y de investigación de contar con una herramienta intuitiva que permita realizar análisis circadianos avanzados sin requerir conocimientos de programación, eliminando así barreras técnicas y económicas para la investigación biomédica.

Los objetivos principales de BioAnalyzer son: **(1) Dar acceso a la ritmometría**. De esta forma los investigadores sin formación en programación puedan analizar datos temporales de forma autónoma. **(2) Unificar funcionalidades dispersas en distintas herramientas en una sola plataforma web**, permitiendo cargar datos biológicos y obtener parámetros circadianos clave (mesor, amplitud, acrofase, etc.) junto con su significancia estadística. **(3) Ofrecer una experiencia de usuario intuitiva y multiplataforma**, accesible desde cualquier dispositivo con navegador web.

BioAnalyzer pretende transformar la manera en que se analizan las oscilaciones biológicas diarias, proporcionando una herramienta accesible, fiable y alineada con las buenas prácticas de desarrollo web.

2. Contexto

El desarrollo de BioAnalyzer se enmarca en un creciente interés científico por los ritmos circadianos y su impacto en la salud humana. Los ritmos circadianos son ciclos biológicos de aproximadamente 24 horas que sincronizan procesos fisiológicos y conductuales con el entorno. Factores propios de la vida moderna, como el trabajo por turnos, la exposición prolongada a luz artificial durante la noche, el uso intensivo de pantallas o los desfases horarios frecuentes, pueden alterar los ritmos fisiológicos normales, generando desajustes circadianos. Estas alteraciones de los ritmos diarios se han implicado en diversos trastornos de salud, incluyendo problemas de sueño, desórdenes metabólicos, complicaciones cardiovasculares e incluso afecciones neuropsiquiátricas. Por tanto, comprender y medir adecuadamente las oscilaciones biológicas circadianas se ha vuelto prioritario para muchos investigadores, dado que puede conducir a avances en la prevención y tratamiento de enfermedades vinculadas al reloj biológico.

A pesar de la importancia de la cronobiología, la evaluación estadística de la ritmicidad circadiana plantea un desafío considerable. Los métodos tradicionales de análisis (como el ajuste de curvas cosenoidales mediante regresión) son complejos y frecuentemente requieren conocimientos especializados de programación y estadística avanzada. Actualmente existen diversas herramientas y librerías en R o Python para realizar análisis circadianos (ritmometría), entre ellas paquetes basados en el modelo cosinor clásico. Sin embargo, la mayoría carece de ciertas funcionalidades integrales y emplea formatos de datos de entrada no unificados, obligando a los investigadores a combinar múltiples herramientas y realizar conversiones de formato engorrosas. Incluso los paquetes más recientes y versátiles, como *CosinorPy* en Python o *Kronos* en R, ofrecen mejoras, pero siguen requiriendo conocimientos de programación para su uso efectivo. Esta situación ha dejado a muchos biólogos y profesionales de la salud con una barrera de entrada alta para explotar plenamente sus datos temporales.

En este contexto científico-tecnológico, BioAnalyzer cobra sentido al ubicarse en la intersección entre la biología circadiana y el desarrollo web. Aprovechando los avances en tecnologías web, BioAnalyzer pretende eliminar o reducir las barreras mencionadas, proporcionando a la comunidad investigadora una herramienta en línea accesible globalmente. De esta forma, se contribuye a la utilidad de la ritmometría en salud: los investigadores, centros y laboratorios podrán analizar ritmos biológicos de manera más eficiente, lo que potencialmente acelerará descubrimientos sobre cómo los factores externos e internos modulan nuestros relojes biológicos. En suma, el proyecto se sitúa en un entorno de innovación interdisciplinar, donde convergen la demanda en biomedicina por mejores análisis circadianos y las posibilidades que brindan las aplicaciones web modernas.

3. Justificación y motivación del proyecto

La elección de desarrollar BioAnalyzer está motivada por su carácter innovador y por la evidente carencia de soluciones integrales en el análisis circadiano dentro del sector biomédico. Antes del inicio de este proyecto, la investigación previa del equipo señalaba una limitación importante: aunque existen herramientas académicas para analizar ritmos (paquetes en R/Python basados en cosinor) no había una plataforma unificada, de uso sencillo, que integrase todas las funcionalidades necesarias. Esta necesidad quedó de manifiesto en la experiencia de un grupo de investigación de la Universitat Rovira i Virgili, donde se había creado un software de escritorio llamado *CircAnalyst* (registrado en 2024) para analizar ritmos circadianos. El éxito y utilidad de esa herramienta de escritorio planteó el siguiente paso lógico: migrar la tecnología a una plataforma web con navegación intuitiva y amigable. Así nace BioAnalyzer, con la ambición de ampliar el alcance de dicha solución al entorno web y hacerla accesible a cualquier investigador sin importar su plataforma o conocimientos técnicos.

Las necesidades detectadas que BioAnalyzer aborda son varias. En primer lugar, la comunidad científica demandaba una forma de realizar análisis circadianos sin tener que programar o manejar software estadístico complejo. Además, se buscaba unificar en una sola herramienta diversas funcionalidades que antes requerían múltiples programas: ajuste de curvas cosenoidales, estimación de parámetros circadianos, comparación entre grupos experimentales, visualización gráfica de resultados y exportación de informes. También era fundamental estandarizar el formato de los datos de entrada, ya que los investigadores trabajan con hojas de cálculo (Excel/CSV) y necesitaban cargar sus datos sin pasos intermedios complicados.

BioAnalyzer aporta una solución real a la comunidad científica satisfaciendo las necesidades anteriores. La plataforma se distingue por ofrecer en un solo lugar todas las capacidades de análisis circadiano que antes estaban dispersas o eran inaccesibles para muchos usuarios. En concreto, BioAnalyzer:

- **Reúne todas las funcionalidades clave** de la ritmometría en una única herramienta web, evitando tener que usar múltiples paquetes aislados.
- **Permite cargar los datos en un formato unificado** (archivos Excel estándar) para todos los tipos de análisis, simplificando el proceso inicial.
- **Facilita el análisis simultáneo de varios parámetros o conjuntos de datos**, algo útil cuando se quieren comparar ritmos de diferentes variables biológicas o grupos experimentales.
- Ofrece **una interfaz de usuario amigable e intuitiva**, de modo que los investigadores pueden realizar todo el proceso de análisis sin conocimientos de programación ni asistencia de personal especializado.

Gracias a estas características, BioAnalyzer cubre la limitación existente en el sector y representa un avance disruptivo, convirtiendo un procedimiento antes reservado a expertos en programación en una herramienta accesible para cualquier laboratorio. Esto tiene un impacto potencial considerable, ya que acelera y hace accesible la investigación en cronobiología. La motivación personal detrás del proyecto también radica en la oportunidad de aplicar las competencias adquiridas en el ciclo formativo a un problema real de carácter científico, integrando el desarrollo web con el análisis de datos biológicos. En definitiva, BioAnalyzer se justifica por su aporte innovador y por resolver una necesidad concreta en la comunidad investigadora, proporcionando una plataforma que mejora la eficiencia y alcance de los estudios sobre ritmos circadianos.

4. Análisis y diseño del proyecto

En esta sección se describen las decisiones tecnológicas tomadas para el diseño de BioAnalyzer, tanto en el frontend como en el backend, justificando cada elección en función de los requisitos del proyecto y las ventajas ofrecidas por dichas tecnologías.

Frontend: Se optó por React.js como librería principal para construir la interfaz de usuario. React facilita la creación de interfaces modernas y altamente interactivas mediante componentes reutilizables, logrando una experiencia de usuario fluida y reactiva. Además, React cuenta con una amplia comunidad y ecosistema, lo que garantiza robustez y facilidad de mantenimiento. Para el diseño visual y responsivo se incorporó Tailwind CSS, un framework CSS utilitario que permite un estilo responsive consistente y una rápida personalización de los elementos de la interfaz. Tailwind contribuye a lograr una apariencia profesional y coherente sin invertir excesivo tiempo en estilos desde cero y con shadcn como embellecedor . Junto con React, se usó React Hook Form para la gestión eficiente de formularios y validación de datos de entrada del usuario (por ejemplo, en los formularios de registro y carga de parámetros), lo que mejora la experiencia al proporcionar feedback inmediato de errores al usuario. Asimismo, para la representación gráfica de resultados se integró Chart.js a través de la biblioteca *react-chartjs-2*. Esta combinación permite generar gráficos interactivos y personalizables en el navegador, fundamentales para visualizar las oscilaciones circadianas de manera clara. Los gráficos resultantes (como curvas de ajuste cosinor y comparativas de ritmos) pueden incluso exportarse como imágenes, facilitando su inclusión en informes científicos.

Backend: En el lado servidor, BioAnalyzer está construido sobre Node.js junto con el framework Next.js. Node.js proporciona un entorno de ejecución Javascript en el servidor que destaca por su eficiencia y escalabilidad, mientras que Next.js complementa con un marco full-stack que soporta *Server-Side Rendering* (SSR) y la creación de una api rest. La elección de Next.js permitió estructurar el proyecto de forma unificada (frontend y backend en una misma base de código) y aprovechar SSR para mejorar los tiempos de carga inicial de la aplicación. Internamente, Next.js utiliza Express para gestionar las rutas y la lógica de negocio del backend, lo cual aporta flexibilidad a la hora de implementar las distintas API. Para manejar la subida de ficheros de datos se incorporó el middleware, Multer (xlsx), que simplifica la recepción y procesamiento de archivos en las rutas del servidor de manera segura. La autenticación de usuarios se implementó mediante JSON Web Tokens (JWT) en combinación con el servicio de autenticación de Firebase. En concreto, Firebase Authentication gestiona el registro/inicio de sesión de usuarios y la verificación de credenciales, mientras que JWT se emplea para generar tokens de sesión que son enviados por el cliente en cada petición, protegiendo así las rutas privadas de la API. Esta arquitectura de autenticación híbrida garantiza seguridad y escalabilidad: Firebase aporta una capa robusta de gestión de usuarios, y los JWT permiten al backend validar fácilmente cada petición. Para el almacenamiento de datos

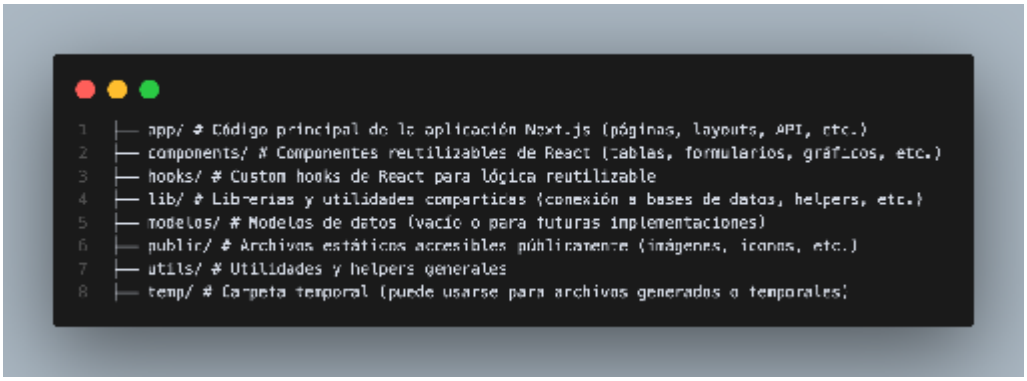
persistentes en cliente se optó por zustand y para gestión de datos se optó por una base de datos MongoDB manejada a través de Mongoose. MongoDB, al ser NoSQL, ofrece flexibilidad para almacenar estructuras de datos heterogéneas algo conveniente en este proyecto, pues los resultados de análisis circadiano pueden incluir una variedad de datos diferentes de esta forma facilita la escalabilidad horizontal de la aplicación.

Base de datos

Análisis de datos circadianos: Un aspecto central del diseño es cómo BioAnalyzer realiza los cálculos de ritmicidad. Para ello, se migraron al entorno JavaScript del backend ciertos algoritmos validados en entornos universitarios que originalmente estaban implementados en Python. En particular, se implementó el modelo cosinor clásico (ajuste de una función cosenoidal a los datos temporales) mediante una librería especializada de optimización no lineal llamada ml-levenberg-marquardt. Esta librería permite ajustar modelos sinusoidales utilizando el método de Levenberg-Marquardt, obteniendo parámetros como mesor (valor medio del ritmo), amplitud (variación máxima) y acrofase (fase o momento del pico máximo) de cada ritmo analizado. Adicionalmente, se incorporó la librería simple-statistics para calcular indicadores estadísticos complementarios de forma nativa en JS. Por ejemplo, se realizan pruebas no paramétricas *U* de Mann-Whitney para comparar si dos grupos difieren en sus ritmos (útil cuando se comparan, por ejemplo, ritmos circadianos de un grupo control vs. tratado).

En cuanto a la **arquitectura general del sistema**, BioAnalyzer sigue una arquitectura clásica de tres capas: la capa de presentación (frontend React) interactúa con la capa de lógica de negocio (API Node/Express) a través de peticiones HTTP, y está a su vez interactúa con la capa de datos (base de datos MongoDB y módulo de análisis) para cumplir cada funcionalidad. En la práctica, el navegador del usuario carga una *Single Page Application* de React/Next.js que ofrece las vistas de la aplicación; cuando el usuario realiza acciones (ej. subir un archivo o solicitar un análisis), la app envía peticiones asíncronas al backend ya sean endpoint de nextjs o server actions (funciones que se ejecutan en servidor). El servidor procesa esas solicitudes – ya sea almacenando/retrayendo datos de MongoDB o ejecutando los cálculos circadianos – y responde con los resultados o confirmaciones adecuadas. Esta arquitectura orientada a servicios facilita la escalabilidad y mantenibilidad: por ejemplo, permitiría desplegar la API y la interfaz web por separado si fuera necesario, o incluso integrar en el futuro nuevos métodos de análisis como microservicios independientes para aplicaciones móviles .

En cuanto a la organización del proyecto, se ha estructurado siguiendo buenas prácticas de separación de responsabilidades. El código se distribuye en carpetas específicas: la carpeta app/ contiene la lógica principal de la aplicación Next.js, diferenciando claramente entre componentes y rutas que se ejecutan exclusivamente en el servidor ("server only") y aquellos que pueden compartirse entre el servidor y el cliente ("server/client"), aprovechando las capacidades híbridas de Next.js. Por ejemplo, las funciones que acceden directamente a la base de datos o gestionan lógica sensible se ubican en archivos o rutas marcadas como "server only", mientras que los componentes de interfaz y lógica reutilizable se encuentran en components/ y hooks/, respectivamente. Los custom hooks en la carpeta hooks/ encapsulan lógica tanto para el cliente como para el servidor, facilitando la reutilización y el mantenimiento del código. Además, la carpeta lib/ agrupa utilidades y funciones compartidas, y utils/ contiene helpers generales. Esta estructura modular permite escalar el proyecto fácilmente, mantener la seguridad y claridad en la separación de contextos de ejecución, y facilita la incorporación de nuevas funcionalidades o servicios en el futuro.



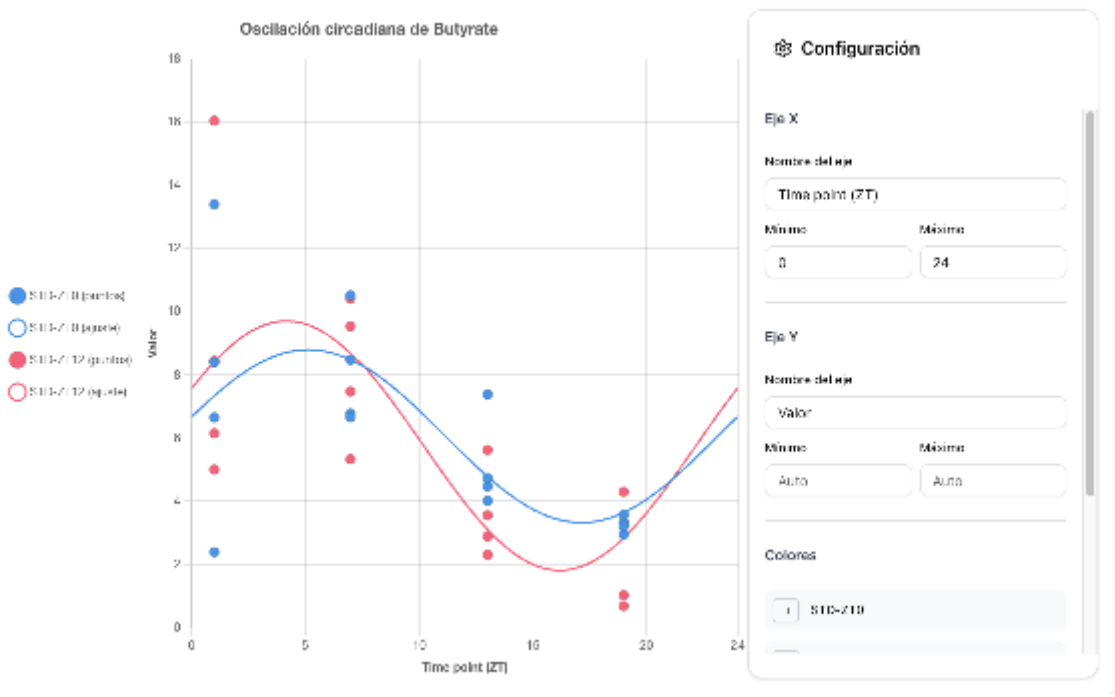
5. Desarrollo e implementación

El desarrollo de BioAnalyzer se llevó a cabo siguiendo una estrategia iterativa, abordando por etapas las distintas funcionalidades clave del sistema. En primer lugar, se configuró el entorno de proyecto con Next.js, estructurando la base de código para incluir tanto las páginas frontend como las rutas de API backend. Posteriormente se implementó el sistema de autenticación, integrando Firebase Authentication para el registro e inicio de sesión de usuarios, y estableciendo middleware en el backend para verificar los JSON Web Tokens en cada petición protegida. Una vez asegurada la capa de seguridad, el foco pasó a la implementación del análisis circadiano: se tradujeron los algoritmos desde Python a JavaScript, afinando el ajuste cosinor y validando resultados contra el script Python de referencia. En paralelo, se fueron construyendo los componentes de interfaz correspondientes a cada funcionalidad, utilizando React con componentes funcionales y estado manejado mediante hooks. Cada avance se probó con datos reales proporcionados por colaboradores del ámbito universitario, iterando sobre la interfaz para mejorar la usabilidad.

Adicionalmente, se implementó un middleware personalizado en Next.js para reforzar la seguridad y el control de acceso a las rutas de la aplicación. Este middleware intercepta cada petición entrante y verifica, mediante una consulta interna a la API de login, si el usuario dispone de una sesión autenticada válida. Si el usuario ya está autenticado y trata de acceder a rutas públicas como /login, /register o la raíz, es redirigido automáticamente al panel principal (/overview). Por el contrario, si intenta acceder a rutas privadas sin estar autenticado, el middleware lo redirige a la página de inicio de sesión. Esta lógica se apoya en la validación de una cookie de sesión y en la comprobación del estado de autenticación a través de la API, asegurando que solo los usuarios autorizados puedan acceder a las funcionalidades protegidas. Además, el middleware está configurado para excluir rutas públicas y recursos estáticos, optimizando así el rendimiento y la experiencia de usuario. Esta capa adicional de control contribuye a mantener la integridad y privacidad de los datos gestionados por la plataforma.

La arquitectura interna resultante consta de varios módulos bien definidos. Por un lado, el *frontend* de Next.js con React maneja la interacción con el usuario: formularios, botones y gráficos. Por otro lado, el *backend* expone endpoints REST y server actions para operaciones como autenticación, subida de archivos y ejecución de análisis. Cuando un usuario autenticado solicita un nuevo análisis, por ejemplo, el flujo es el siguiente: el archivo de datos es enviado al endpoint correspondiente, el backend lo guarda temporalmente y extrae sus datos a través de la librería xlsx para obtener las series temporales; luego se invoca al módulo de análisis cosinor en JavaScript, que ajusta el modelo circadiano a los datos y calcula los parámetros y estadísticos requeridos; finalmente, el backend guarda los resultados en la base de datos y retorna al frontend un objeto con los resultados numéricos y las gráficas generadas. El frontend recibe la respuesta y actualiza la interfaz para mostrar al usuario tanto los parámetros calculados (mesor, amplitud, acrofase, nivel de significación p, etc.) como las visualizaciones gráficas correspondientes. Este proceso típico

ejemplifica la coordinación entre los distintos componentes desarrollados.



A continuación, se describen las **funcionalidades clave** implementadas en BioAnalyzer, junto con sus flujos de usuario asociados:

1. **Registro e inicio de sesión:** Un nuevo usuario puede crear una cuenta proporcionando sus datos (correo electrónico y contraseña, principalmente). Tras el registro, puede iniciar sesión para acceder a las demás funcionalidades. El sistema valida las credenciales con Firebase Auth y, si son correctas, genera un token JWT que se almacena en el cliente para autenticar las siguientes operaciones. La interfaz de esta sección es simple e intuitiva, con notificaciones de error en caso de credenciales inválidas.

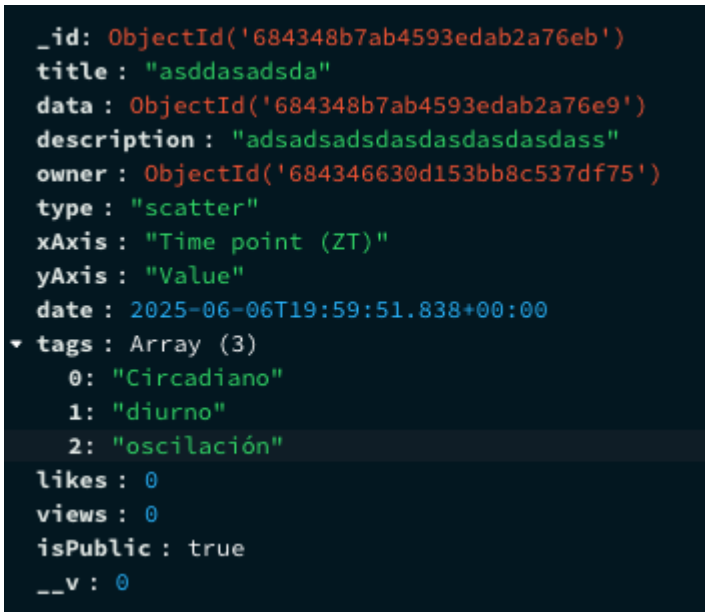
2. **Carga de datos:** Una vez autenticado, el usuario accede a su panel donde puede subir archivos de datos en formato Excel. La aplicación proporciona un formulario de carga que permite seleccionar el archivo desde el sistema de archivos local. Al subirlo, el backend verifica que el formato del fichero y la estructura de los datos sean correctos en este caso que exista columnas de tiempo y respectivos grupos. Si hay algún error de formato, se informa al usuario con mensajes claros para que pueda corregirlo. En caso exitoso, los datos se almacenan y se procede al siguiente paso de configuración.

3. **Configuración del análisis:** Tras cargar los datos, BioAnalyzer permite configurar los parámetros del análisis circadiano. Desde la interfaz web, el usuario puede seleccionar qué columna de datos quiere analizar (en caso de haber múltiples variables), el tipo de análisis a realizar y opciones adicionales. También se le ofrecen opciones para personalizar los gráficos resultantes como títulos, nombres de ejes, colores de las curvas y seleccionar el período de la oscilación a ajustar (por defecto 24 horas para ritmos circadianos). Esta etapa de configuración fue diseñada para ser lo más intuitiva posible, con explicaciones breves junto a cada opción para guiar a usuarios que no estén familiarizados con los términos.

4. **Ejecución del análisis y visualización de resultados:** Al completar la configuración, el usuario inicia el análisis con un botón dedicado. En ese momento, el backend procesa los datos como se describió antes (ajustando el modelo circadiano y realizando los cálculos estadísticos). Durante unos instantes, el usuario ve un indicador de progreso mientras se ejecuta el algoritmo. Una vez terminado, la

aplicación muestra los resultados de forma interactiva: se renderizan gráficos dinámicos donde la curva cosenoidal ajustada aparece superpuesta a los datos originales, permitiendo evaluar visualmente el grado de ajuste. Además, se presenta una tabla resumida con los valores numéricos obtenidos: mesor, amplitud, acrofase y el valor p de significación del ritmo (por ejemplo, indicando si el patrón diario es estadísticamente significativo). La interfaz de resultados soporta interactividad: es posible pasar el cursor por los puntos del gráfico para ver valores exactos, ocultar o mostrar ciertas series, hacer zoom sobre una región de tiempo, etc., gracias a las capacidades de Chart.js.

5. **Gestión y exportación de resultados:** BioAnalyzer almacena el historial de análisis realizados por cada usuario si así lo desea. Esto significa que un investigador puede realizar múltiples análisis (por ejemplo, distintas variables o distintos archivos de datos) y luego acceder a ellos en una sección de mis análisis. Desde allí es posible exportar los resultados de interés: la plataforma permite descargar un informe en PDF que incluye los gráficos y valores numéricos, apto para documentación o anexos de publicaciones científicas. Alternativamente, también se ofrece la exportación de los datos numéricos y de los gráficos a formato de imagen PNG/SVG para que el usuario pueda incorporarlos manualmente en sus informes. Esta funcionalidad de exportación fue importante en el desarrollo, ya que añade valor práctico para el científico que suele necesitar evidencias visuales y numéricas para reportar.



Cabe mencionar que durante la implementación se pusieron en práctica diversas buenas prácticas de desarrollo: uso de control de versiones (Git) para el código, realización de pruebas con casos reales para validar cada módulo, y documentar el código fuente especialmente en las funciones complejas de análisis matemático. Además, se tuvo presente la seguridad en todo momento (por ejemplo, sanitizando entradas, protegiendo contra inyecciones, asegurando la comunicación con HTTPS en despliegues). Como resultado, el sistema desarrollado no solo cumple con los requisitos funcionales previstos, sino que también está construido sobre una base sólida y mantenible, lista para escalabilidad futura.

6. Requisitos para el uso de BioAnalyzer

Para poder instalar, ejecutar y contribuir al desarrollo de BioAnalyzer en un entorno local, es necesario cumplir con los siguientes requisitos previos:

- **Node.js**: Versión 18 o superior. Es el entorno de ejecución necesario para Next.js y el backend de la aplicación.
- **pnpm**: Gestor de paquetes recomendado para instalar las dependencias del proyecto. Se puede instalar globalmente con `npm install -g pnpm`.
- **Git**: Para clonar el repositorio y gestionar el control de versiones.
- **MongoDB**: Es necesario tener acceso a una instancia de MongoDB (local o en la nube) para el almacenamiento de datos.
- **Cuenta de Firebase**: Para la autenticación de usuarios, se requiere configurar un proyecto en Firebase y obtener las credenciales correspondientes.
- **Navegador web moderno**: Para acceder a la interfaz de usuario (se recomienda Google Chrome, Firefox o Edge).

Instalación básica (modo desarrollador)

1. Clonar el repositorio:

```
git clone <url-del-repositorio>
cd analisis-circadianos
```

2. Instalar las dependencias:

```
pnpm install
```

3. Configurar las variables de entorno necesarias (MongoDB, Firebase, etc.) en un archivo `.env.local` siguiendo el ejemplo proporcionado en el repositorio.

Ejemplo de archivo `.env.local`

```
MONGODB_URI=mongodb+srv://usuario:contraseña@cluster.mongodb.net/bioanalyz
er

# Claves de Firebase
FIREBASE_API_KEY=tu_api_key
FIREBASE_AUTH_DOMAIN=tu_auth_domain
FIREBASE_PROJECT_ID=tu_project_id
FIREBASE_STORAGE_BUCKET=tu_storage_bucket
FIREBASE_MESSAGING_SENDER_ID=tu_sender_id
FIREBASE_APP_ID=tu_app_id
```

4. Iniciar la aplicación en modo desarrollo:

```
pnpm dev
```

5. Acceder a la aplicación desde el navegador en `http://localhost:3000`

> **Nota:** Si solo se desea usar la aplicación como usuario final, basta con acceder al enlace de despliegue que en este caso a sido desplegado en vercel, conectandose al github y vinculadno el repositorio si quiere probar el analizador en la carpeta raiz tuene un xlsx de prueba para realizarlo, ademas de que si quieres probar las funcionalidades de guardar analisis deberia de iniciar sesion

7. Conclusiones y valoración personal

En conclusión, BioAnalyzer representa una solución innovadora y necesaria en el campo del análisis de ritmos circadianos. A lo largo de este proyecto se ha logrado desarrollar una plataforma web funcional que integra con éxito tecnologías de desarrollo modernas (React, Node, etc.) con algoritmos científicos avanzados proporcionados por la universidad para ofrecer una herramienta de uso sencillo pero científicamente rigurosa. El resultado es una aplicación accesible, robusta y escalable que elimina muchas de las barreras técnicas previamente existentes, ahora cualquier investigador con un navegador web puede cargar sus datos y obtener un análisis circadiano completo, sin instalar software especializado anticuados ni depender de terceros con habilidades de programación. Esto facilita la investigación multidisciplinar, permitiendo que equipos de biólogos, médicos y otros científicos aprovechen técnicas de ritmometría en sus proyectos con mínima fricción.

Desde el punto de vista técnico y personal, el desarrollo de BioAnalyzer ha supuesto un reto sobretodo en la integrar algoritmos matemáticos complejos en un entorno web requirió profundizar tanto en conocimientos de programación como en conceptos científicos a pesar de tener apoyo de la universidad en este tema (entender la estadística detrás de la ritmometría para poder implementarla correctamente). Entre las principales dificultades superadas se incluyen la adaptación del modelo cosinor al lenguaje JavaScript asegurando la precisión numérica, la gestión eficiente de archivos de datos relativamente grandes en el navegador y el servidor, y la conjugación de múltiples herramientas como puede ser firebase y nextjs en despliegue con vercel y en una arquitectura coherente.

El proyecto cumple con los objetivos planteados inicialmente y sienta una base sólida para futuras mejoras. Entre las posibles ampliaciones a considerar se encuentra la incorporación de nuevos métodos de análisis circadiano (por ejemplo, métodos no paramétricos o algoritmos basados en transformadas wavelet para detectar ritmicidad e incluso comparaciones de analisis), así como la extensión de la plataforma a un público más amplio mediante la internacionalización de la interfaz a varios idiomas como puede ser la libreria de react-i18n. Otra mejora factible sería evolucionar BioAnalyzer hacia un servicio en la nube de tipo *SaaS*, donde múltiples usuarios puedan ejecutar análisis concurrentemente con diferentes niveles de suscripción, garantizando sostenibilidad y actualización continua de la herramienta. Igualmente, se podría explorar la integración con dispositivos o plataformas de salud para analizar ritmos circadianos en tiempo real, lo que abriría la puerta a aplicaciones clínicas.

La realización de BioAnalyzer no solo permitió consolidar los conocimientos adquiridos en el ciclo formativo (desde la programación frontend hasta la administración de servidores, bases de datos y diseño), sino que también ofreció la oportunidad de contribuir en un ámbito científico de actualidad que a nivel tecnológico está bastante abandonado. Ver funcionar la plataforma con datos reales y obtener resultados coherentes con los esperados ha reafirmado la motivación detrás del proyecto. Como desarrollador, este trabajo ha mejorado mis habilidades para planificar y abordar proyectos fuera de mi zona de confort, comunicándome con expertos en dominio biotecnológicos. También aprendí a equilibrar consideraciones de rendimiento, usabilidad y rigor científico, que son lecciones valiosas para futuros emprendimientos profesionales.

8. Bibliografía y recursos

- Luboš Molčan. *Time distributed data analysis by Cosinor.Online application*. bioRxiv 805960 (preprint), 2019. doi: 10.1101/805960.
- Soliz-Rueda, J. R., et al. *Gut microbiota and eating behaviour in circadian syndrome*. **Trends in Endocrinology & Metabolism**, 36(1): 15–28, 2025.
- Moškon, M. *et al.* **(2020)**. *CosinorPy: a Python package for cosinor-based rhythmometry*. BMC Bioinformatics 21:485.
- Bastiaanssen, T. F. S. *et al.* **(2023)**. *Kronos: A computational tool to facilitate biological rhythmicity analysis*. Preprint en bioRxiv, doi: 10.1101/2023.04.21.537503.
- Refinetti, R. **(2016)**. *Refining the cosinor model for the analysis of biological rhythms: An overview.* **Journal of Circadian Rhythms** 14(1): 1. (Artículo de revisión sobre métodos de ritmometría).
- **React.js** – Documentación oficial (biblioteca JavaScript para construir interfaces de usuario).
- **Node.js** – Documentación oficial (entorno de ejecución JavaScript del lado servidor).
- **Express** – Documentación oficial (framework web para Node.js).
- **Tailwind CSS** – Documentación oficial (framework CSS utilitario para diseño web responsivo).
- Documentación de **Firebase Authentication** (servicio de autenticación de Firebase) y guía de **JSON Web Tokens (JWT)** – recursos oficiales en línea.
- **Mongoose** (base de datos) - https://mongoosejs.com/docs/guide.html