UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

INGENIERÍA DE COMPUTADORES

Mecanismos flexibles para seleccionar, relacionar y visualizar fenómenos astronómicos en el marco de la Sociedad Malagueña de Astronomía

Flexible mechanisms for selecting, relating and visualizing astronomical phenomena within the framework of the Malaga Astronomy Society

Realizado por
Jesús García Gutiérrez

Tutorizado por
Beatriz Barros Blanco
Alberto Castellón Serrano

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2025

Resumen

La Sociedad Malagueña de Astronomía (SMA) es una agrupación astronómica que, entre otras muchas actividades, se dedica a la divulgación científica y al estudio e investigación de meteoros. Esto es gracias a la Red de detección de Bólidos y Meteoros de la UMA y la SMA, que consta a día de hoy de veinticuatro estaciones a lo largo del territorio español (y algunas otras en instalación), y que además colabora con la Red Global BOOTES, gestionada por el Instituto de Astrofísica de Andalucía (IAA) y el Consejo Superior de Investigaciones Científicas (CSIC), usando también sus estaciones.

Todos estos informes acerca de las detecciones de meteoros se almacenan en una base de datos. En este trabajo se ha realizado una aplicación web basada en *workflows* para permitir a los científicos generar visualizaciones e informes sobre estos datos, con el fin de poder estudiarlos y hacer divulgación de los mismos de una manera más sencilla, eficiente y visualmente atractiva.

Como resultado se ha visto que la aplicación mejora en gran medida los procesos llevados a cabo por los científicos de la SMA, permitiéndoles, por ejemplo, generar y publicar las noticias de su página web [UMA/SMA(2017)] de forma automatizada y usando un solo software, eliminando procesos manuales como la elaboración de gráficas, la búsqueda manual en la base de datos, la subida de vídeos a YouTube o la generación de mapas con la trayectoria de los meteoros. Además de esto, los informes generados con esta aplicación permiten hacer una mejor divulgación científica del estudio de meteoros y de las actividades de la SMA, ya que estos son visualmente atractivos y fáciles de comprender para el público general.

Palabras clave: meteoros, workflows, visualización, informes, web

Abstract

The Malaga Astronomy Society (SMA) is an astronomy group that, among many other activities, is dedicated to scientific dissemination and the study and research of meteors. This is thanks to the UMA and SMA Fireball and Meteor Detection Network, which currently consists of twenty-four stations throughout Spain (with several others in the process of being installed), and which also collaborates with the BOOTES Global Network, managed by the Andalusian Institute of Astrophysics (IAA) and the Spanish National Research Council (CSIC), also using its stations.

All these reports on meteor detections are stored in a database. In this project, a workflow-based web application has been developed to enable scientists to generate visualizations and reports on this data, in order to study it and disseminate it in a simpler, more efficient, and visually appealing way.

As a result, the application has been shown to greatly improve the processes carried out by SMA scientists, allowing them, for example, to generate and publish news on their website [UMA/SMA(2017)] automatically and using a single software program, eliminating manual processes such as creating graphs, manually searching the database, uploading videos to YouTube, or generating maps with the trajectory of meteors. In addition, the reports generated with this application allow for better scientific dissemination of meteor studies and SMA activities, as they are visually appealing and easy for the general public to understand.

Key Words: Meteors, workflows, visualization, reports, web

Índice

1 Introducción

1.1. Motivación

La Sociedad Malagueña de Astronomía (SMA) estudia y registra en bolas de fuego, bólidos y meteoros [Madiedo Gil(2017)], fenómenos que se producen cuando pequeñas partículas procedentes de cometas o asteroides entran en la atmósfera terrestre a gran velocidad y se desintegran por fricción, dejando el rastro luminoso que llamamos meteoro o estrella fugaz [SEA(2022)].

Desde el año 2022, se dispone de una Base de Datos con todos los informes de todos los fenómenos registrados desde diferentes telescopios de la geografía española, resultado de un TFG [García Escobar(2023)] y el esfuerzo de varios miembros de la SMA. En resumen:

- Base de Datos con los informes de meteoros desde diversos telescopios de la red BOOTES.
- Página web sencilla que da acceso a los informes almacenados en la Base de datos sin relacionar nada ni aportar vistas resumidas o relacionadas para poder sacar conclusiones sobre los datos.

Los informes que se almacenan en la Base de Datos se generan a diario, disponiéndose de un gran registro histórico de varios años y su contenido ha ido evolucionando según las necesidades de los astrónomos y estudiosos del tema. Toda esta información es una gran fuente de conocimiento que necesita organizarse, relacionarse y visualizarse para que los científicos puedan realizar sus estudios e investigaciones. Se han detectado las siguientes carencias:

1. Disponer de mecanismos para extraer datos de forma visual desde la Base de Datos y poder relacionarlos.

2. Disponer de metáforas visuales que relacionen los datos en varias dimensiones.

3. Disponer de mecanismos flexibles para definir qué extraer de la base de datos y soluciones para mostrarlos.

Aportar estas mejoras a la página web, requiere, en primer lugar, de un mecanismo de acceso a la base de datos. Posteriormente, se necesita diseñar un lenguaje flexible de definición de flujos de trabajo o algo similar, cuyo objetivo será el procesamiento de los datos leídos de la base de datos para generar gráficos. Estos gráficos deben poder generarse fácilmente mediante una interfaz de usuario diseñada siguiendo principios de usabilidad y diseño centrado en el usuario.

1.2. Objetivos

El objetivo principal de este TFG es el procesamiento de los datos observados por los telescopios para obtener visualizaciones de los mismos que los investigadores de ese campo necesitan para interpretar los fenómenos astronómicos desde diferentes puntos de vista y para la divulgación científica del tipo de actividades que realiza la Sociedad Malagueña de Astronomía.

Este objetivo puede dividirse en tres subobjetivos principales:

1. Definir mecanismos que permitan construir plantillas de extracción de datos, conectadas a la base de datos y a la interfaz de usuario para automatizar procesos de representación.

2. Implementación de procesos de extracción de datos de la base de datos y su conexión con las plantillas definidas anteriormente.

3. Uso de los desarrollos anteriores para obtener visualizaciones sobre astronomía útiles para los estudiosos del tema y para su divulgación científica.

1.3. Tecnologías Utilizadas

El desarrollo del Generador de Informes de Observación de Meteoros se compone de un stack tecnológico diseñado para soportar tanto la gestión de datos científicos como las capacidades de generación de informes visualmente atractivos y fáciles de usar.

1.3.1. Tecnologías Frontend

La interfaz de usuario fue desarrollada utilizando **React 16.14.0** [React(2025)] como framework principal de JavaScript, proporcionando una arquitectura basada en *widgets* adecuada para construir aplicaciones científicas interactivas. La interfaz incorpora varias librerías, algunas de las más importantes son:

- **React Bootstrap 2.10.0**: Utilizada para crear componentes responsive y gestionar layouts.
- **Recharts 2.1.9**: Utilizada para la visualización de datos y generación de gráficos científicos.
- **React DnD 14.0.5**: Habilita la funcionalidad de arrastrar y soltar para la construcción de informes basados en widgets.
- **@react-google-maps/api 2.19.1**: Utilizada para visualización geográfica interactiva de ubicaciones de observatorios y trayectorias de meteoros.
- **html2canvas 1.4.1**: Utilizada para la exportación de las gráficas.
- **react-to-print 2.13.0**: Utilizada para la impresión y exportación de informes generados.

1.3.2. Tecnologías Backend

La arquitectura del lado del backend está construida sobre **Node.js** con el framework **Express.js 4.18.4**, proporcionando endpoints de API RESTful para acceso y manipulación de datos. Los componentes clave del backend incluyen, entre otros:

- **MySQL2 3.9.1**: Driver de base de datos para la conexión y ejecución de consultas SQL.
- **MongoDB 6.3.0**: Driver de base de datos Mongo para el almacenamiento de snapshots de informes en construcción.
- **Dotenv 16.0.3**: Gestión de variables de entorno para la configuración de la app.
- **Node-fetch 2.6.7**: Cliente HTTP para comunicación con APIs externas y servicios web.

- **CORS 2.8.5**: La configuración que permite la comunicación entre frontend y backend debido a que se ejecutan en puertos diferentes (Cross-Origin Resource Sharing).

- **Babel**: Habilita la retrocompatibilidad de JavaScript convirtiendo el código a versiones más antiguas para que se pueda ejecutar en navegadores o entornos que no soportan las últimas funciones.

1.3.3. Gestión de Base de Datos

El proyecto implementa una arquitectura de base de datos híbrida que combina:

- **MySQL 8.0** [MySQL(2025)]: Como sistema de gestión de la base de datos relacional principal de meteoros.

- **MongoDB 7.0** [MongoDB(2025)]: Como sistema de gestión de una base de datos para el almacenamiento de informes en construcción.

1.3.4. APIs y Servicios Externos

La integración con servicios externos potencia la funcionalidad de la plataforma:

- **YouTube Data API v3**: Para subida de videos.

- **Google OAuth 2.0**: Para la autenticación necesaria para acceder a las APIs de Google.

- **Google Maps JavaScript API**: Para la generación y visualización de mapas con marcadores y trayectorias.

1.3.5. Desarrollo y Despliegue

La arquitectura de la aplicación soporta despliegue contenerizado utilizando **Docker** y **Docker Compose**, asegurando entornos consistentes a través de las etapas de desarrollo y producción. Las herramientas de desarrollo incluyen **Nodemon** para desarrollo backend y **React Scripts** para la construcción del frontend.

1.4. Estructura del documento

La memoria está organizada en cinco capítulos. Después de esta introducción donde se describe la motivación, los objetivos y las tecnologías utilizadas, se incluye un capítulo donde se resumen algunas tareas y procesos de la SMA que pueden beneficiarse de los resultados de este proyecto junto con ejemplos del uso de la herramienta.

A continuación, en el capítulo 3 se describe la especificación y diseño del sistema, en el que se realiza un análisis de requisitos y se describe la arquitectura de la aplicación.

Para finalizar, en el penúltimo capítulo se detallan los ciclos en los que se ha llevado a cabo el desarrollo de la aplicación.

Como último capítulo, tenemos la conclusión en la que se analiza el trabajo realizado y se describen las posibles líneas futuras.

2

Tareas y Procesos de la SMA

2.1. Procesos de la Sociedad Malagueña de Astronomia (SMA)

La SMA [SMA(1975)], fundada en 1975, es una agrupación dedicada a diversas tareas, como son la divulgación de la astronomía, la monitorización de la contaminación lumínica y el estudio de meteoros. En este proyecto nos centramos en el estudio de meteoros, concretamente en la generación de informes en base a la detección de estos bólidos por la Red de detección de Bólidos y Meteoros de la Universidad de Málaga y de la SMA. Esta red está compuesta por 21 estaciones en funcionamiento y 7 en instalación [Castellón(2020)] [Castellón(2015)]. Además, utilizan las cámaras allsky de la Red Global BOOTES liderada por el Instituto de Astrofísica de Andalucía (IAA-CSIC) que es una red de observatorios astronómicos ubicados en diversos países: España, China, México, Chile, Sudáfrica y Nueva Zelanda [IAA-CSIC(1998)] [Tejada(2023)].

2.1.1. Generacion informes

Cuando la Red detecta un meteoro se registran todos los datos disponibles del mismo para la realización de diversos informes que dependen tanto de los datos disponibles como de la cantidad de estaciones que hayan captado el meteoro.

Los informes que se generan son los siguientes [García Escobar(2023)]:

- **Informe-Z:** Este informe sólo se genera si el meteoro es detectado por al menos dos observatorios. Este tipo de informe es el que contiene información más valiosa para el estudio del mismo.
- **Informe Radiante:** Este informe se genera siempre que se detecte al meteoro por un

observatorio y se utiliza para comprobar si el meteoro se debería haber detectado por otra estación para generar el Informe-Z si efectivamente ha sido detectado por otras estaciones. Existen dos tipos dependiendo de si se puede asociar al meteoro a alguna lluvia activa [NASA(2018)].

- **Informe de Fotometría:** Este informe sólo se genera si existe una imagen del meteoro. A partir de la trayectoria extraída de esa imagen se calcula una recta de regresión para obtener el brillo y la masa invertida en producirlo.

2.1.2. Generacion de visualizaciones

Los informes generados por la red BOOTES se almacenan en una Base de Datos que, desde el año 2023, es accesible por la web: http://astro.iaia.lcc.uma.es. La información que se muestra se extrae directamente y se muestra en tablas sin hacer un preprocesado para cálculo de indicadores o resumen de los datos. Mucho menos, se incluyen facilidades de visualización para ayudar a los científicos a entender la magnitud o importancia de los fenómenos astronómicos registrados.

Dado que el ojo humano se siente atraído por los colores, las formas y los patrones, la visualización de datos presenta una gran ventaja ya que capta nuestra atención y hace que captemos mejor el mensaje. Si podemos ver algo, lo interiorizamos rápidamente [Tableau(2025)]. El ejemplo perfecto es esta base de datos; mirando una tabla llena de información no podemos ver ninguna tendencia o patrón, pero en cambio, haciendo unos sencillos gráficos, podemos ver rápidamente tendencias y relaciones entre los datos.

Las visualizaciones pueden modelarse como un proceso de flujos de trabajo. Ese flujo estaría compuesto por la extracción de datos, los cálculos sobre ellos y la representación de imágenes o gráficas.

Así, una infografía [López Alonso(2015)] tiene los siguientes pasos:

- **Idea:** Cuando surge la primera propuesta de visualización hay que hacer una breve descripción de la misma y plantearse ciertas cuestiones para estudiar su viabilidad como

pueden ser:

  - ¿Qué utilidad tiene para el proyecto?
  - ¿Es una idea novedosa y/o atractiva?
  - ¿Tiene interés científico?

- **Investigación:** Este paso consiste en la documentación y búsqueda de datos, además de estudiar la viabilidad de realizar la idea con la información de la que disponemos.
- **Contenidos:** Aquí analizaríamos los datos y elegiríamos el tipo de gráfico más adecuado para nuestro caso de uso.
- **Estructura:** En este punto realizaríamos un boceto de cómo quedaría el gráfico.
- **Diseño Final:** Como paso final elaboraríamos el gráfico con la herramienta de diseño digital. Para finalizar se debe repasar el estilo (tipografía, colores, formas, etc) y se exportaría al formato deseado.

La herramienta nos ayuda a automatizar los dos últimos pasos, ya que genera automáticamente el gráfico en base a consultas SQL y en base al tipo de gráfico seleccionado.

Ejemplo de Generación de Visualizaciones con la herramienta A continuación se presenta un ejemplo de visualización usando la herramienta. En este ejemplo hemos usado el widget **DataChart** para, usando queries predefinidas a la Base de Datos, contar el número de meteoros en diferentes agregaciones temporales: por hora del día 'Figura 1' y por mes 'Figura 2'.

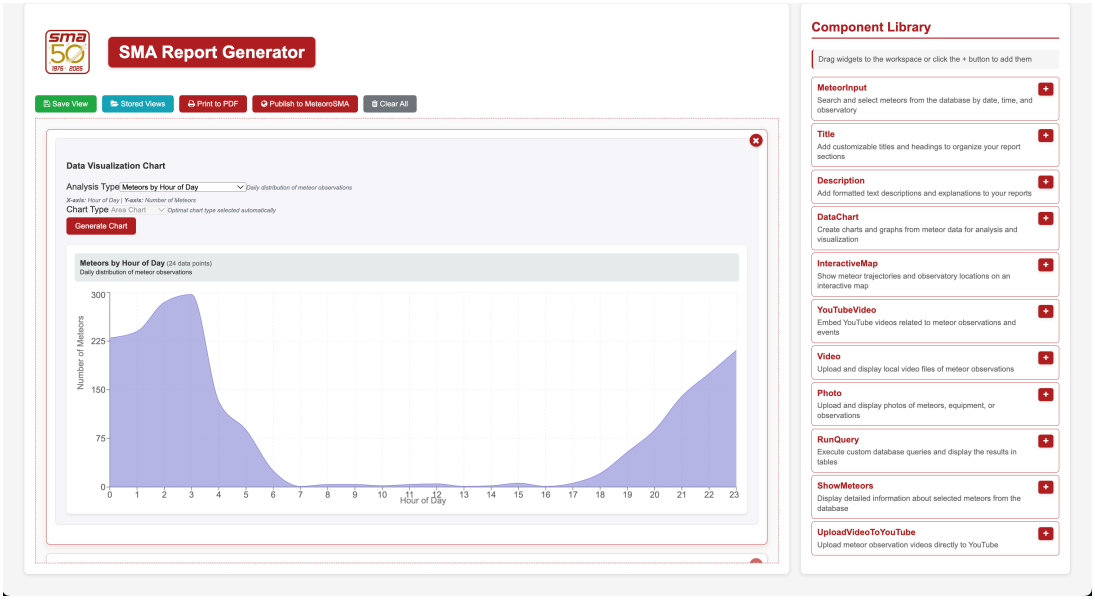

Figura 1: Visualización por Hora del Día

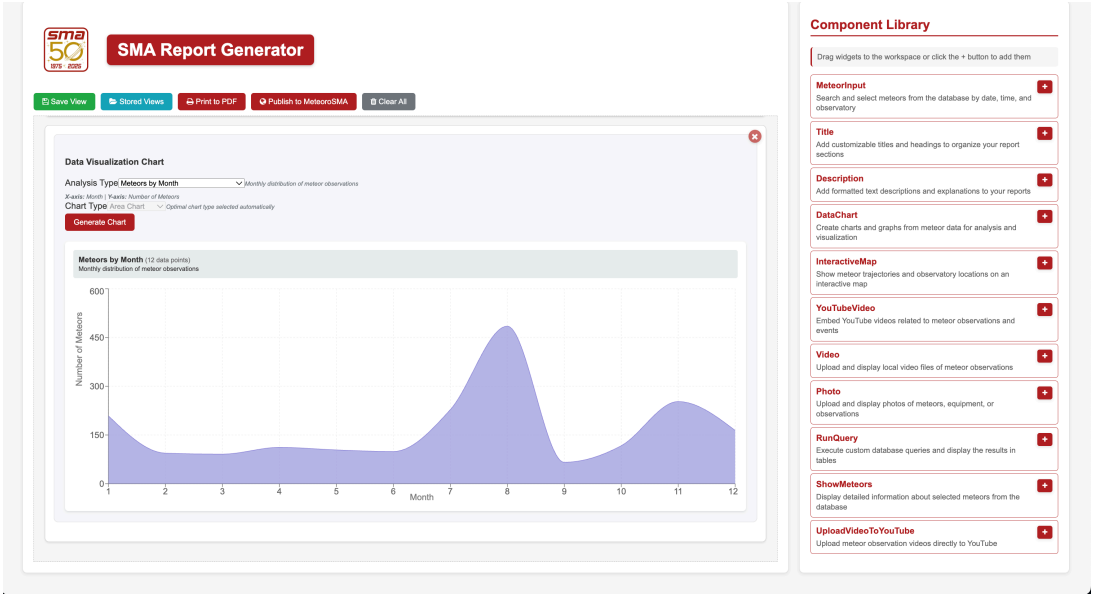

Figura 2: Visualización por Mes

La herramienta permite también otro tipo de gráficos, como de barras, líneas, puntos o columnas. Para definir estas visualizaciones existe un fichero de configuración llamado `analysisTypes.js` en el que se definen las queries a utilizar (a su vez definidas en el fichero `queries.js`), el tipo de gráfico, las etiquetas de los ejes y un flag para indicar si es un análisis sobre un sólo meteoro o un análisis genérico sobre toda la base de datos. La configuración se explica más detalladamente en 'Apéndice A. Manual de uso'.

```
"Photometric Analysis": [
    {
        key: "lightCurve",
        label: "Light Curve (Magnitude vs Time)",
        description: "Track brightness changes over time",
        xQuery: "meteorLightCurve",
        yQuery: "meteorLightCurve",
        xQueryGeneral: "lightCurve",
        yQueryGeneral: "lightCurve",
        xLabel: "Time (s)",
        yLabel: "Magnitude",
        chartType: "line",
        isXYPair: true
    },
    {
        key: "magnitudeDistance",
        label: "Magnitude vs Distance",
        description: "Brightness variation with distance",
        xQuery: "meteorMagnitudeDistance",
        yQuery: "meteorMagnitudeDistance",
        xQueryGeneral: "magnitudeDistance",
        yQueryGeneral: "magnitudeDistance",
        xLabel: "Distance (km)",
        yLabel: "Magnitude",
        chartType: "scatter",
        isXYPair: true
    }...
]
```

Figura 3: Ejemplo de tipos de análisis

Estos son ejemplos del resto de tipos de gráficos:

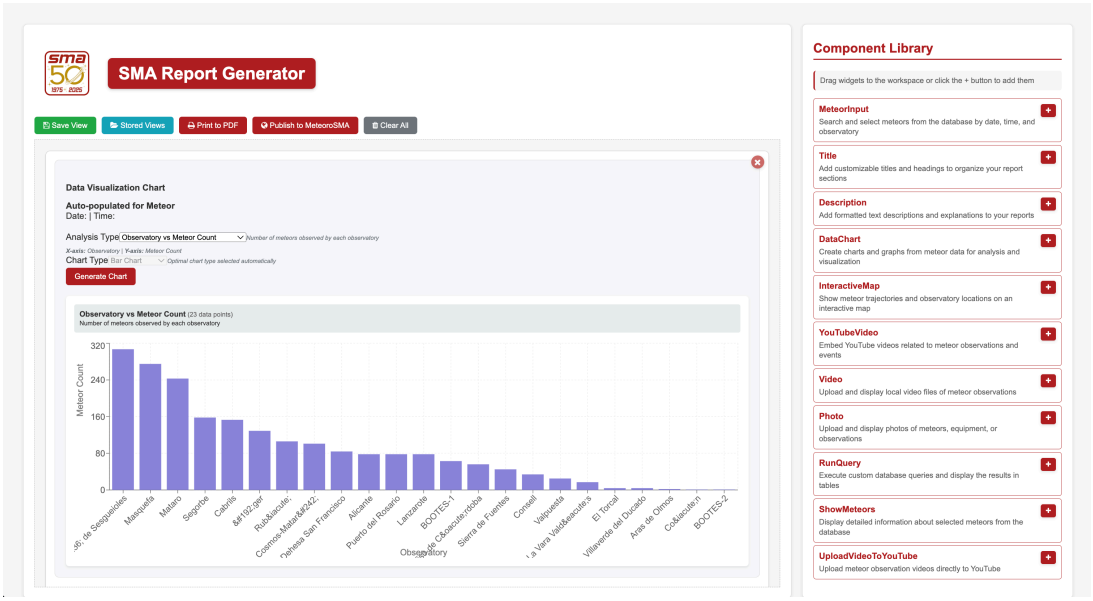

Figura 4: Gráfico de barras

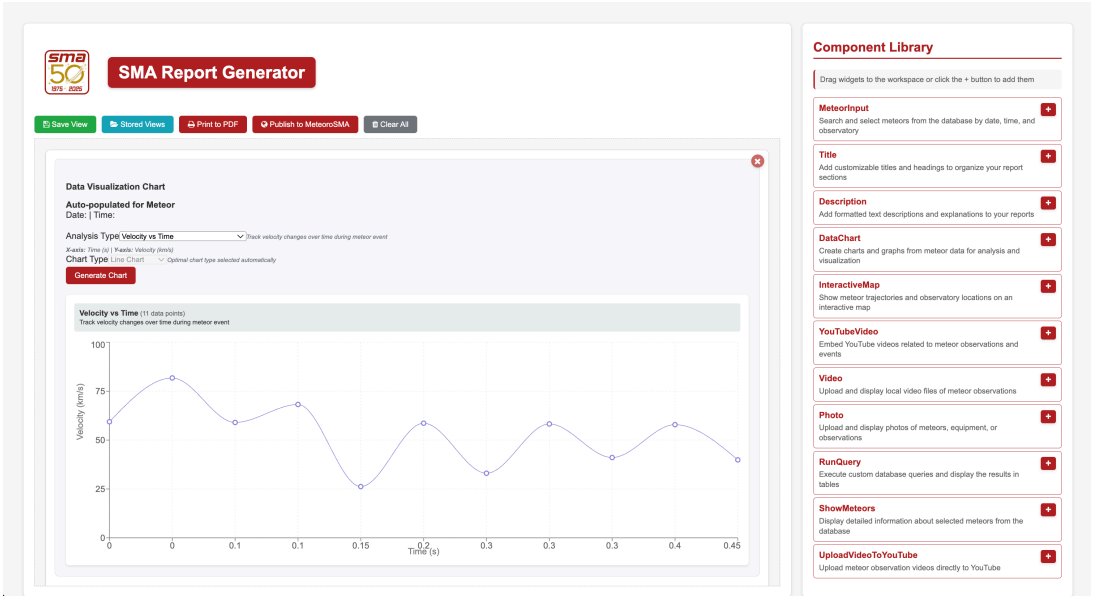

Figura 5: Gráfico de líneas

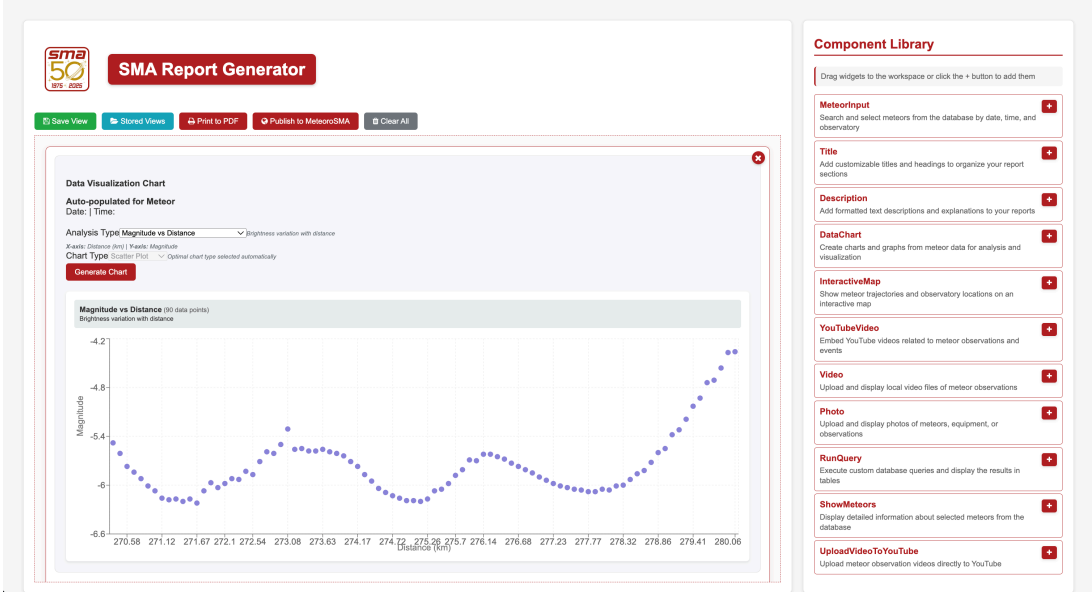

Figura 6: Gráfica de puntos

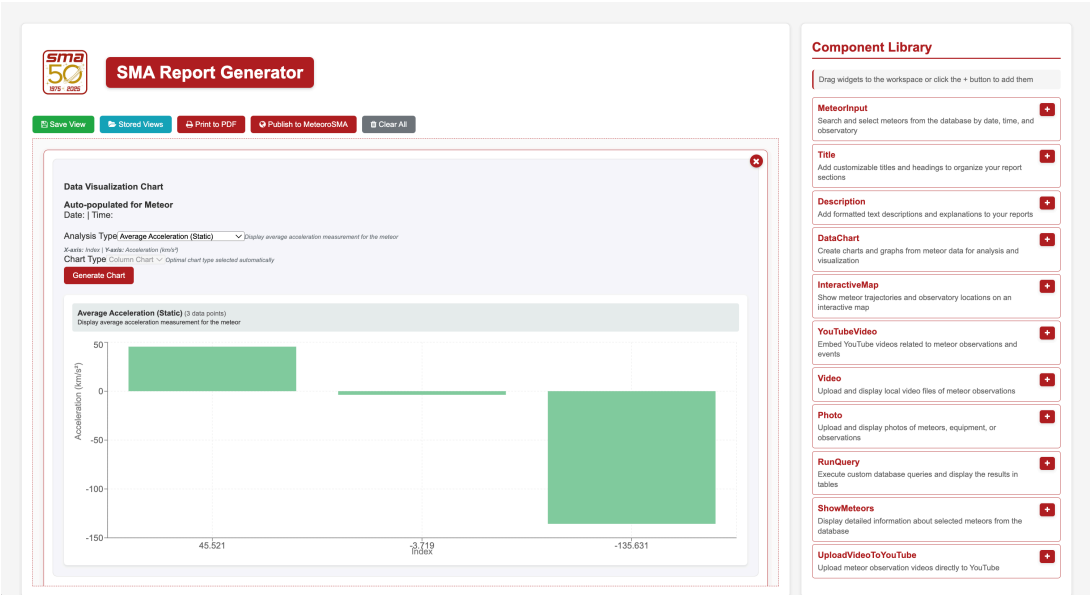

Figura 7: Gráfico de columnas

2.1.3. Generación de noticias

Cuando un meteoro tiene un gran interés científico, los miembros de la SMA publican una noticia en su página web sobre el mismo. Estas noticias suelen tener una serie de elementos comunes y otros opcionales que, dependiendo de los datos disponibles sobre el meteoro y del

criterio de la persona que publica la noticia, se añaden a estas [UMA/SMA(2017)].

Los elementos que pueden componer la noticia son:

- **Título:** El título que nombra la noticia
- **Texto descriptivo:** Son textos que suelen tener una estructura común en todas las noticias y describen el meteoro y su detección (fecha, hora y estaciones que lo detectaron). Gran parte de esta información se puede obtener de la base de datos.
- **Mapa:** Un mapa que contiene la trayectoria del meteoro, la altura del meteoro en el inicio y el fin de la trayectoria y las coordenadas geográficas de las estaciones cercanas que han detectado el meteoro.
- **Vídeos:** Vídeos del meteoro tomados por las diferentes estaciones y cualquier otro tipo de vídeo, uno de los más comunes es la representación de su órbita hecha con un programa externo.
- **Fotografías:** Fotografías del meteoro tomadas por las diferentes estaciones.
- **Gráficas:** La noticia puede contener gráficas con datos interesantes sobre el meteoro como por ejemplo su espectro.

El proceso de generación de estas noticias es el escenario perfecto para implementarse mediante una herramienta basada en workflows, en la que el usuario introduce los datos necesarios para la búsqueda del meteoro en la base de datos y elige qué elementos incluir en la noticia, los cuales se pueden auto-generar mediante búsquedas en la BD (mapa, textos genéricos, gráficas, etc). Una vez se auto-genera el esqueleto principal y los elementos genéricos que componen la mayoría de las noticias, el usuario puede modificar los elementos a su gusto antes de publicar la noticia.

Esto facilita enormemente la generación de noticias, ahorrando tiempo y centrando todos los pasos de la publicación de las mismas en una sola herramienta como, por ejemplo, la subida de videos a YouTube, la generación automática del mapa sin necesidad de usar aplicaciones como Google Earth, la generación de gráficas automáticas en base a consultas SQL predefinidas

(a las que el usuario puede añadir las que desee) o la publicación automática en la web de la SMA.

2.2. Implementacion de los flujos en la herramienta

La herramienta desarrollada implementa el concepto de workflows de una manera modular y flexible, permitiendo la creación de informes y visualizaciones de meteoros de forma automatizada.

2.2.1. Componentes del workflow

La herramienta incluye componentes especializados que pueden combinarse para crear workflows complejos:

- **MeteorInput:** Permite la búsqueda y selección de meteoros específicos por fecha, hora y observatorio.
- **Title:** Permite añadir un título y subtítulo
- **Description:** Genera textos descriptivos utilizando plantillas que se completan automáticamente con datos de la base de datos.
- **DataChart:** Genera gráficas automáticas basadas en consultas predefinidas (velocidad, aceleración, fotometría, etc.).
- **InteractiveMap:** Muestra mapas interactivos con las trayectorias de meteoros y ubicación de observatorios.
- **Video/Photo:** Gestiona la subida de contenido multimedia.
- **YouTubeVideo:** Permite incluir un vídeo de YouTube mediante su link.
- **UploadVideoToYouTube:** Automatiza la publicación de vídeos en YouTube.
- **RunQuery:** Ejecuta consultas SQL personalizadas y muestra los resultados.

2.2.2. Sistema de consultas predefinidas

La herramienta incluye más de 60 consultas SQL predefinidas organizadas en categorías, definidas en el fichero de configuración `queries.js`:

- **Análisis de trayectorias:** Velocidad, aceleración, distancia recorrida y tiempo de vuelo.
- **Análisis fotométrico:** Curvas de luz, magnitud y masa fotométrica.
- **Análisis de errores:** Precisión de las mediciones de velocidad, distancia y fotometría.
- **Análisis angular:** Velocidad angular y correlación con lluvias de meteoros.
- **Análisis temporal:** Patrones diarios y mensuales de detección.
- **Análisis por observatorio:** Comparativa entre estaciones y eficiencia de detección.

```
// ========== OBSERVATORY ANALYSIS QUERIES ==========
// Observatory average velocity
observatoryVelocity: "SELECT o.Nombre_Observatorio as observatory,
    AVG(iz.Velocidad_media) as velocity FROM Informe_Z iz
    JOIN Observatorio o ON iz.Observatorio_Numero = o.Numero
    WHERE iz.Velocidad_media IS NOT NULL GROUP BY o.Nombre_Observatorio
    ORDER BY velocity DESC;",
// Observatory meteor count
observatoryCount: "SELECT o.Nombre_Observatorio as observatory,
    COUNT(DISTINCT iz.Meteoro_Identificador) as meteor_count FROM Informe_Z
    iz JOIN Observatorio o ON iz.Observatorio_Numero = o.Numero
    GROUP BY o.Nombre_Observatorio ORDER BY meteor_count DESC;",
// Observatory altitude vs measurements
observatoryAltitude: "SELECT o.Altitud as altitude,
    COUNT(DISTINCT iz.Meteoro_Identificador) as meteor_count FROM Informe_Z
    iz JOIN Observatorio o ON iz.Observatorio_Numero = o.Numero WHERE o.Altitud
    IS NOT NULL GROUP BY o.Altitud ORDER BY altitude;",
```

Figura 8: Ejemplo de queries predefinidas

2.2.3. Motor de plantillas

La clase **TemplateService** permite la generación automática de contenido mediante plantillas que se rellenan con datos extraídos de la base de datos. Esto facilita la creación de textos descriptivos estándar para noticias, utilizando variables como **${meteorId}**, **${observatorio}**

o **${fecha}** que se sustituyen automáticamente. Estas plantillas, al igual que las queries mencionadas anteriormente, se definen en el fichero de configuración `templates.js`.

```
'meteor-detection': {
    name: 'Meteor Detection Report',
    template: 'A las ${hour} ha sido detectado un bólido por la Red de la Universidad de Málaga
        y de la Sociedad Malagueña de astronomía. En concreto fue grabado en vídeo por las estaciones de ${stations}.',
    variables: [
        { name: 'hour', query: 'getMeteorHour', field: 'hour' },
        { name: 'stations', query: 'getObservatoriesWithCredits', field: 'stations' }
    ]
}
```

Figura 9: Ejemplo de plantillas predefinidas

2.2.4. Interfaz de usuario

La interfaz web permite a los usuarios crear workflows de manera intuitiva mediante un sistema de **drag & drop**. Los usuarios pueden:

- Seleccionar componentes de una lista y arrastrarlos al área de trabajo 'Figura 10'.
- Configurar cada componente con parámetros específicos.
- Reordenar los componentes para definir el flujo del informe.
- Previsualizar el resultado en tiempo real.
- Exportar el informe completo a PDF
- Publicar el informe en una web Wordpress.

Existe también una funcionalidad que permite guardar los componentes que tengamos en ese momento en el área de trabajo 'Figura 11'. Es muy útil para guardar el trabajo en progreso o para guardar informes no publicados o no exportados a PDF. A la captura de un cierto momento del área de trabajo que guardamos la llamamos **view** o **vista**. Estas vistas se guardan por defecto en una base de datos **MongoDB**, pero para que la aplicación sea fácilmente desplegable y tolerante a errores, si esta base de datos no está disponible se hace un fallback automático al guardado de las vistas en el almacenamiento local del navegador, de manera totalmente transparente al usuario.

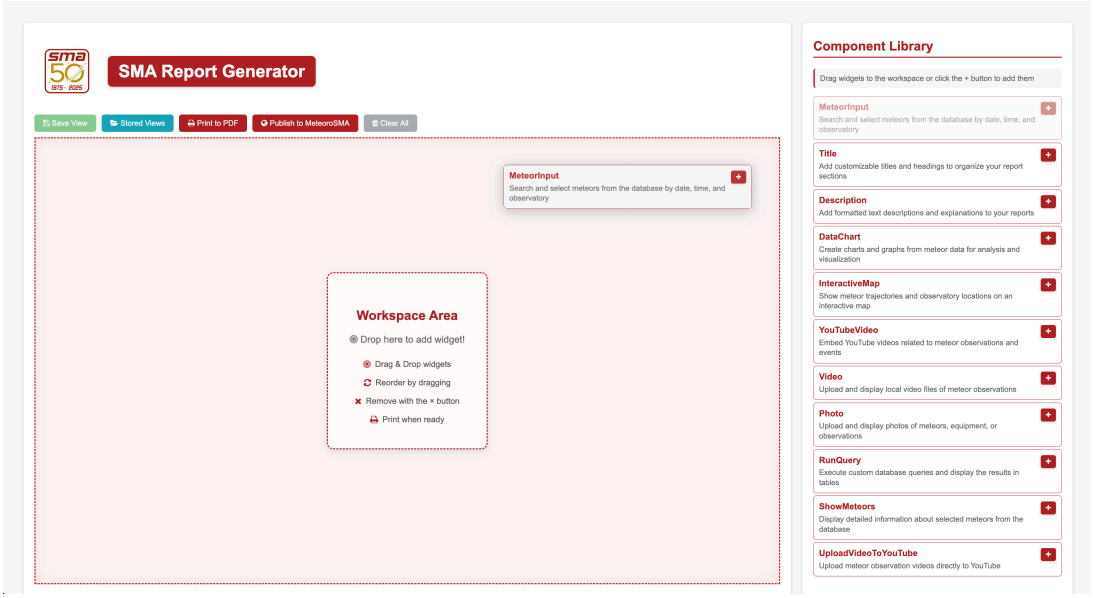

Figura 10: Funcionalidad de ’Drag & Drop’

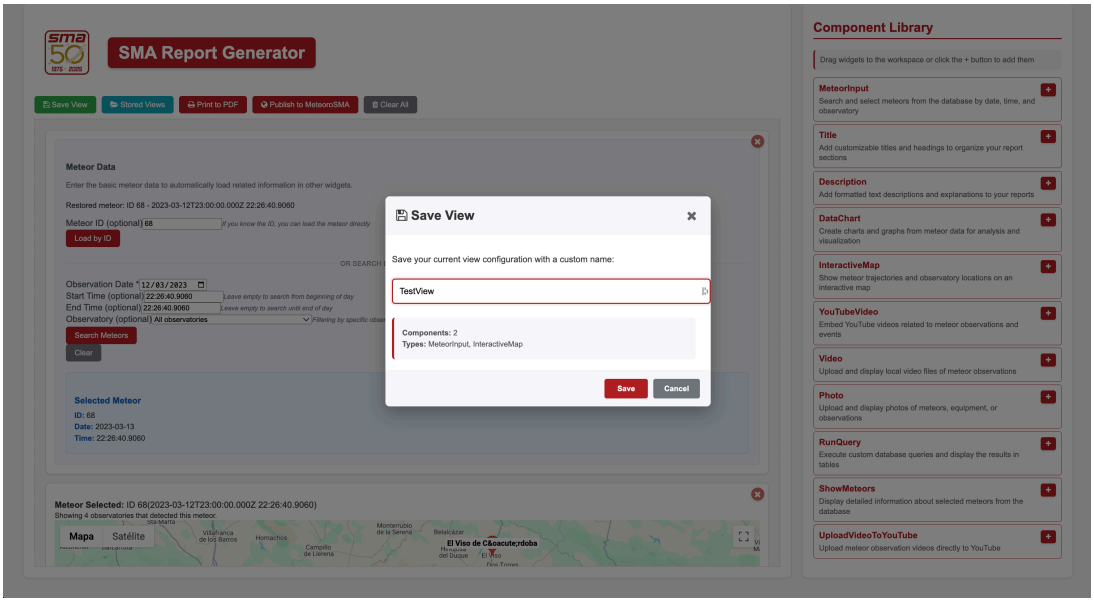

Figura 11: Funcionalidad para guardar Vistas

Estas vistas, posteriormente, se pueden cargar desde la base de datos.

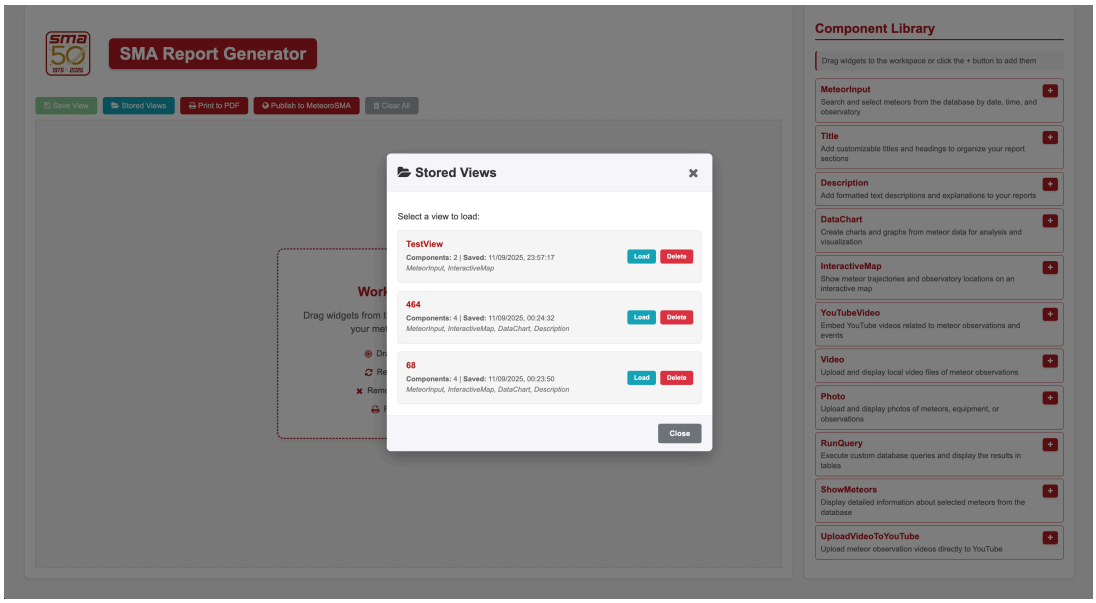

Figura 12: Funcionalidad para cargar Vistas

Como hemos comentado previamente, una vez terminado un informe o noticia, podemos exportarlo en formato PDF 'Figura 14'.

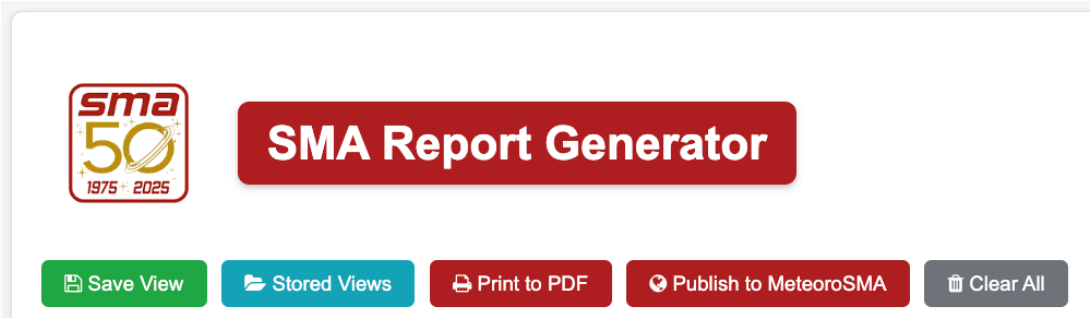

Figura 13: Botones de guardado, carga, exportación y publicación

También podemos publicar el informe directamente a la página web de Wordpress de la SMA 'Figura 15'.

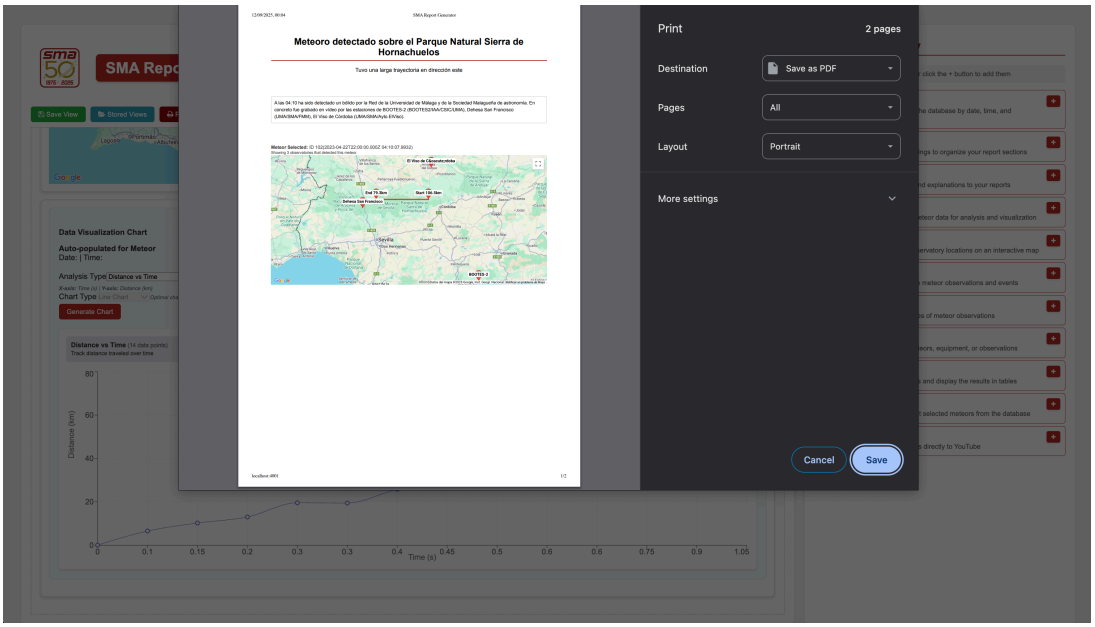

Figura 14: Funcionalidad para exportar a PDF

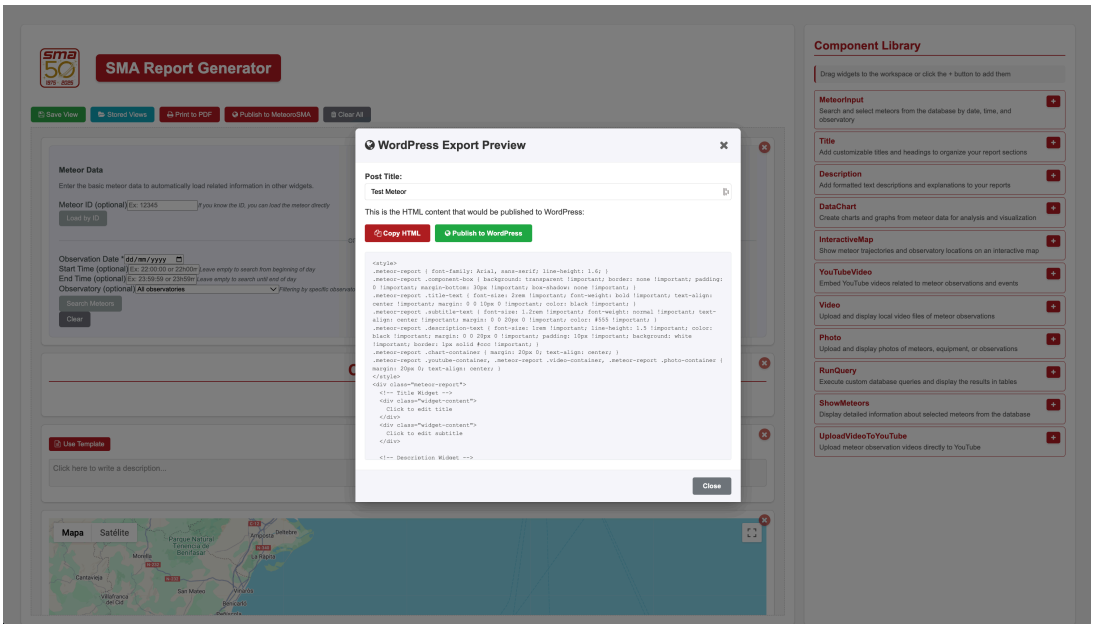

Figura 15: Funcionalidad para publicar en la web de la SMA

2.2.5. Ejemplo de workflow: Generación de noticia sobre un meteoro

En esta sección vamos a ver un ejemplo práctico de la generación de una noticia sobre un meteoro que se detectó el 22 de Abril de 2023. Un workflow típico para generar un informe completo de un meteoro seguiría estos pasos:

1. **Búsqueda:** Utilizamos el widget **MeteorInput** para localizar el meteoro. Podemos usar varios filtros: fecha, hora y observatorio. También podemos cargarlo directamente si conocemos su **ID** en la Base de Datos.

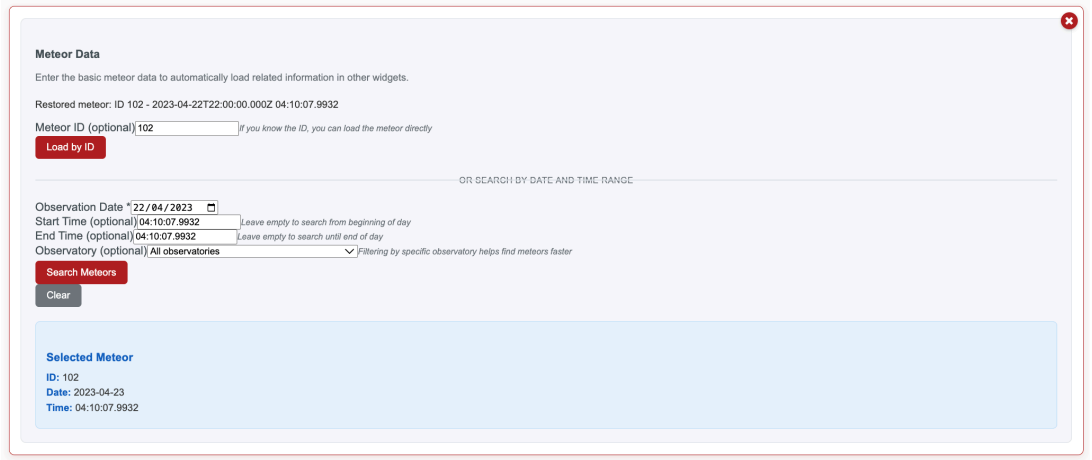

Figura 16: Configuración de widget MeteorInput

2. **Información básica:** Añadir widgets **Title** y **Description**. Para la descripción generada con **Description** podemos usar una de las plantillas predefinidas 'Figura 17' y modificarla a nuestro gusto, pudiendo añadir también links a otras páginas web.

Figura 17: Selección de plantilla predefinida

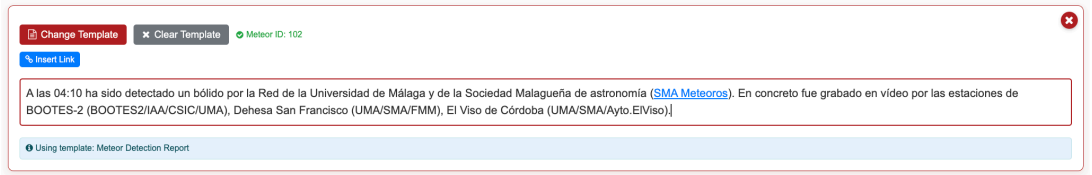

Figura 18: Título y descripción de la noticia

3. **Visualización espacial:** Incorporar widget **InteractiveMap** para mostrar la trayectoria y las estaciones que detectaron el meteoro.

Figura 19: Mapa con la trayectoria del meteoro

4. **Análisis científico:** Incluir múltiples widgets **DataChart** con diferentes análisis (velocidad, fotometría, etc.) en función del caso de uso.

Figura 20: Gráfica de ejemplo

5. **Publicación de vídeos:** Si necesitamos publicar algún vídeo a YouTube previo a incluirlo en el informe podemos utilizar **UploadVideoToYouTube** para su publicación automática.

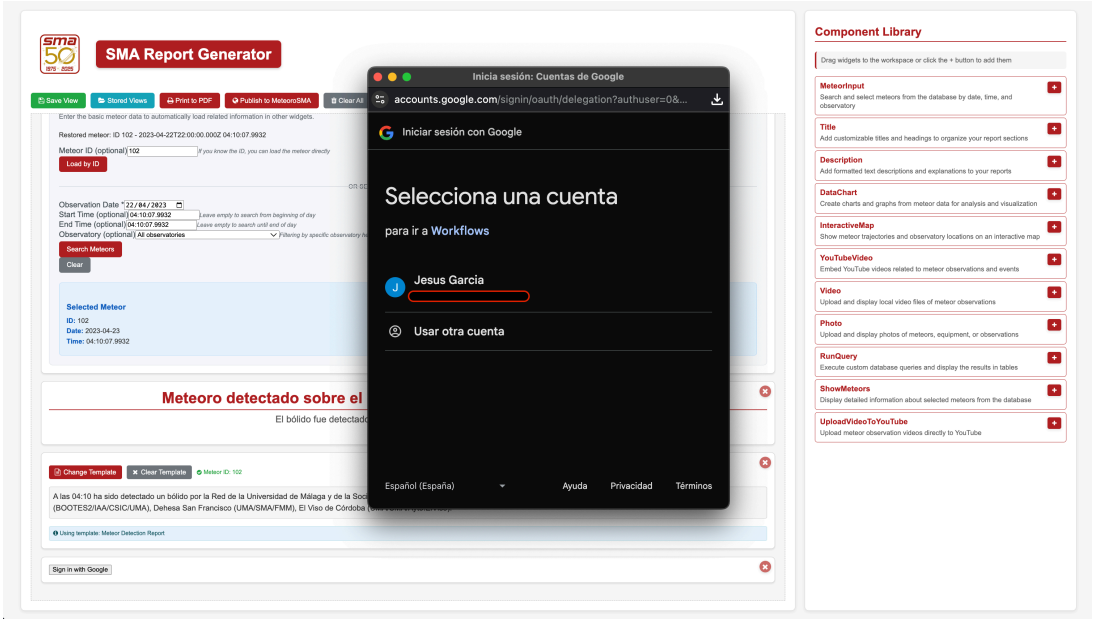

Figura 21: Subida de video a YouTube: Sign In

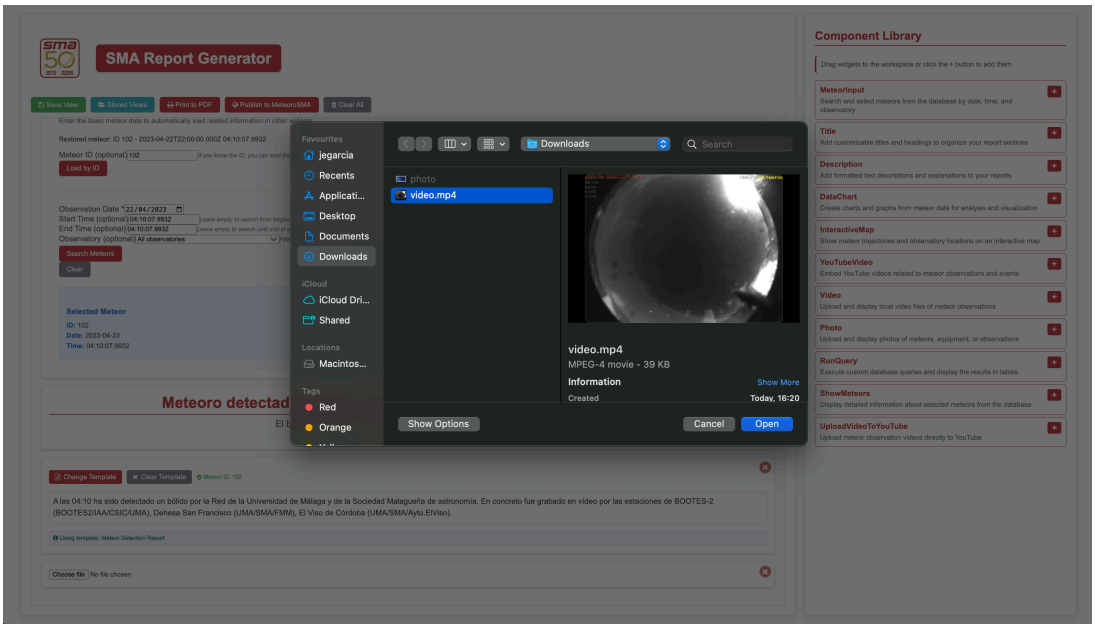

Figura 22: Subida de video a YouTube: Choose File

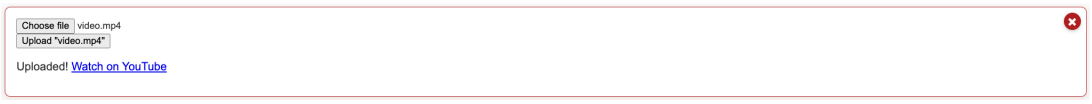

Figura 23: Subida de video a YouTube: Video Uploaded

6. **Contenido multimedia:** Añadir widgets **Photo**, **Video** y/o **YouTubeVideo** con imágenes y vídeos del meteoro.

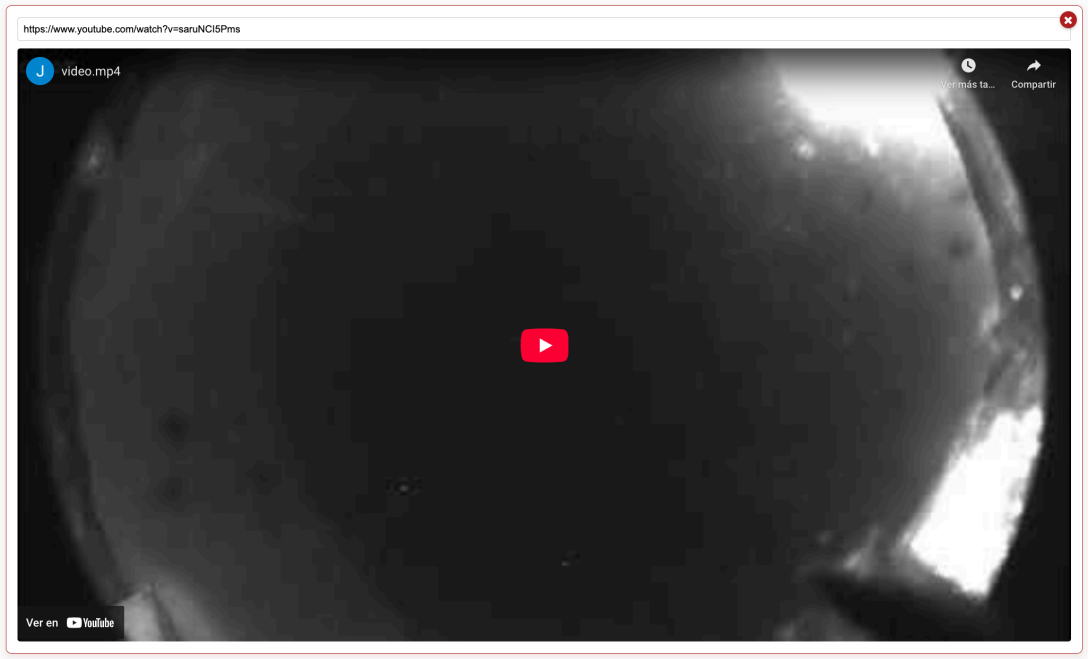

Figura 24: Subida de video desde YouTube

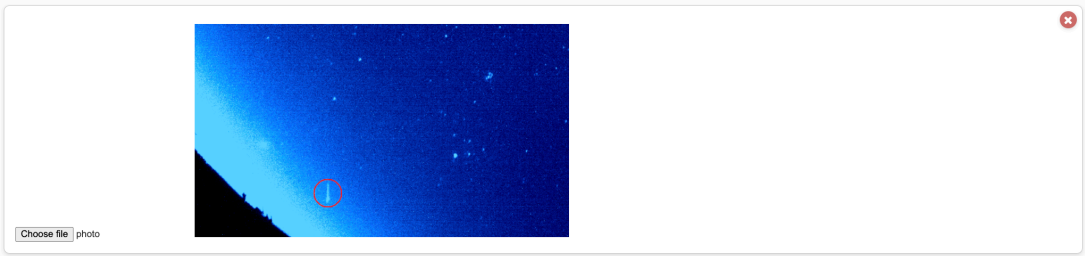

Figura 25: Subida de fotos

7. **Exportación a PDF:**

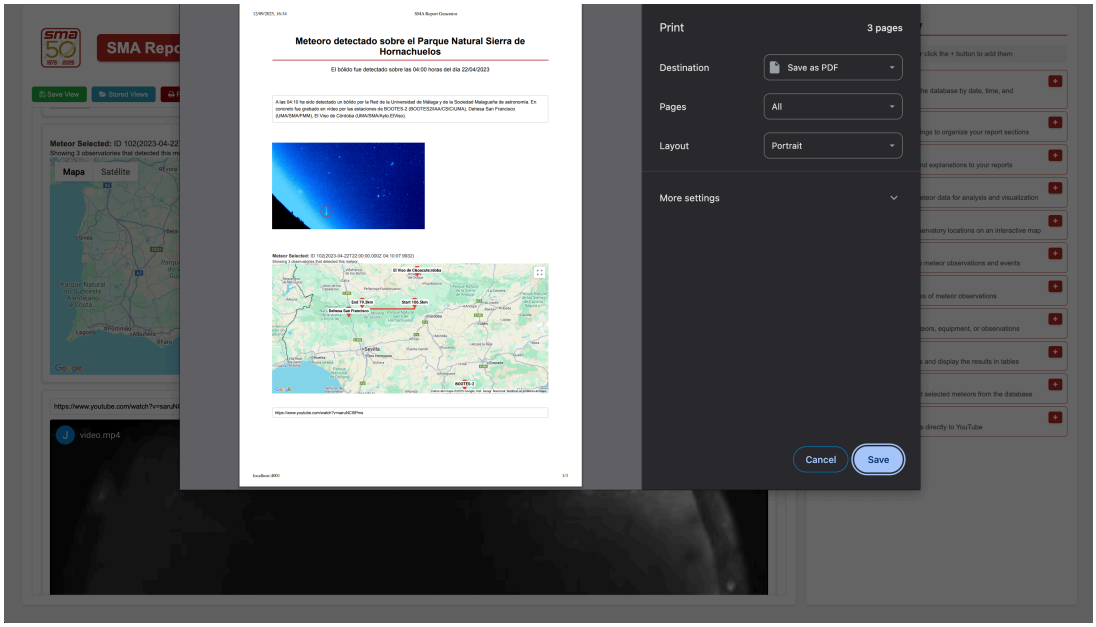

Figura 26: Exportación en formato PDF

Este enfoque modular permite tanto la creación de informes estándar automatizados como la personalización avanzada según las necesidades específicas de cada caso de estudio.

2.3. Análisis de la solución

La solución desarrollada aborda las carencias identificadas en el sistema de gestión de datos astronómicos de la SMA mediante una arquitectura modular basada en workflows y widgets auto-contenidos que automatiza la extracción, procesamiento y visualización de datos meteorológicos.

La solución cumple satisfactoriamente los tres objetivos planteados:

1. **Plantillas de extracción de datos**: Se implementó un sistema robusto de consultas SQL predefinidas organizadas por categorías (análisis de trayectoria, fotometría, patrones temporales, correlaciones avanzadas) que son utilizadas por los diferentes widgets para extraer y mostrar la información de la base de datos.

2. **Procesos de extracción**: Se desarrolló una arquitectura completa de backend con las clases: `MySqlClient` para la gestión de conexiones a la BD, `WorkflowsController` para el enrutado de peticiones, y `WorkflowsExecutor` para implementar lógica de negocio y procesamiento de datos.

3. **Visualizaciones científicas**: Se creó una interfaz web con widgets especializados (`DataChart`, `InteractiveMap`, `RunQuery`, etc) que genera visualizaciones científicas especializadas y facilita la divulgación científica mediante exportación a PDF y auto publicación en la web de la SMA.

2.3.1. Ventajas de la Solución

La solución propuesta ofrece múltiples ventajas:

- **Usabilidad**: Interfaz intuitiva que permite a usuarios sin conocimientos técnicos profundos de bases de datos generar informes científicos completos.
- **Escalabilidad**: Arquitectura modular que facilita la incorporación de nuevos tipos de análisis, consultas, visualizaciones y widgets según evolucionen las necesidades científicas.
- **Mantenibilidad**: Código bien estructurado con separación clara de responsabilidades y documentación completa que garantiza la sostenibilidad del sistema.
- **Portabilidad**: Estrategia de contenerización con Docker que facilita el despliegue en diferentes entornos y la instalación por parte de usuarios finales.

3

Especificación y Diseño del Sistema

3.1. Análisis de Requisitos

El objetivo principal de la aplicación es proporcionar a los astrónomos de la SMA una herramienta intuitiva y eficiente para generar informes y visualizaciones de meteoros a partir de los datos almacenados en la base de datos. La aplicación debe abstraer la complejidad técnica de las consultas SQL y ofrecer una interfaz visual que permita crear workflows personalizados de análisis de datos.

3.1.1. Requisitos Funcionales

Los requisitos funcionales definen las funcionalidades específicas que debe proporcionar el sistema:

RF-01: Búsqueda y selección de meteoros

- El sistema debe permitir buscar meteoros por fecha específica
- Debe permitir filtrar por rango horario de detección del meteoro (hora de inicio y fin)
- Debe permitir filtrar por observatorio específico que detectó el meteoro
- Debe mostrar una lista de meteoros que coincidan con los criterios de búsqueda
- Debe permitir seleccionar un meteoro específico para análisis detallado

RF-02: Generación de visualizaciones

- El sistema debe generar gráficas automáticas basadas en consultas predefinidas

- Debe soportar múltiples tipos de análisis: velocidad, aceleración, fotometría, trayectorias, etc.
- Debe generar gráficas de correlación entre diferentes parámetros
- Debe permitir análisis temporal (patrones diarios y mensuales)
- Debe soportar análisis comparativo entre observatorios y meteoros

RF-03: Mapas interactivos

- El sistema debe mostrar mapas con las trayectorias de meteoros y la altitud en el inicio y fin de las mismas
- Debe visualizar la ubicación de los observatorios que detectaron el meteoro
- Debe mostrar las coordenadas de inicio y fin de la trayectoria
- Debe permitir interacción con los elementos del mapa (zoom, desplazamiento)

RF-04: Generación de contenido descriptivo

- El sistema debe generar automáticamente textos descriptivos usando plantillas
- Debe permitir la edición manual del contenido generado
- Debe permitir incluir links a webs externas en el texto

RF-05: Gestión de contenido multimedia

- El sistema debe permitir la subida de fotografías
- Debe permitir la subida de vídeos
- Debe permitir la publicación automática de vídeos en YouTube

RF-06: Composición de workflows

- El sistema debe proporcionar una interfaz simple basada en *drag & drop*
- Debe permitir añadir, reordenar y eliminar componentes del workflow
- Debe permitir configurar parámetros específicos de cada componente
- Debe propagar automáticamente la selección de meteoro entre componentes que lo necesiten
- Debe permitir previsualización en tiempo real del workflow

RF-07: Exportación y distribución

- El sistema debe permitir exportar informes completos a PDF
- Debe permitir publicar automáticament el informe en la web de la SMA
- Debe mantener el formato y diseño en las exportaciones
- Debe incluir todas las visualizaciones y contenido multimedia en las exportaciones

RF-08: Consultas personalizadas

- El sistema debe permitir ejecutar consultas SQL personalizadas
- Debe proporcionar consultas predefinidas organizadas por categorías para facilitar la generación de informes
- Debe mostrar los resultados de las consultas en formato tabular y gráfico
- Debe validar la sintaxis de las consultas antes de su ejecución

RF-09: Gestión de workflows

- El sistema debe permitir guardar el trabajo en progreso mediante un nombre único y descriptivo
- Debe permitir cargar el trabajo previamente guardado

- Debe mantener las configuraciones y el orden de los componentes del workflow guardado
- Debe proporcionar la interfaz necesaria para guardar, listar, cargar, borrar y sobreescribir workflows
- Debe persistir los workflows guardados en una base de datos
- Debe permitir el funcionamiento local mediante el almacenamiento del navegador en caso de no disponer de la base de datos
- Debe sincronizar el almacenamiento local con la base de datos en cuánto esta última vuelva a estar disponible

3.1.2. Requisitos No Funcionales

Los requisitos no funcionales especifican características de calidad y restricciones del sistema:

RNF-01: Usabilidad

- **Facilidad de uso:** La interfaz debe ser intuitiva para usuarios sin conocimientos técnicos de bases de datos
- **Curva de aprendizaje:** Un usuario nuevo debe poder crear un informe básico en menos de 15 minutos
- **Consistencia:** La interfaz debe mantener un diseño coherente en todos los componentes
- **Retroalimentación:** El sistema debe proporcionar feedback inmediato para todas las acciones del usuario

RNF-02: Rendimiento

- **Tiempo de respuesta:** Las consultas simples deben ejecutarse en menos de 2 segundos

- **Carga de datos:** La carga inicial de la aplicación debe completarse en menos de 5 segundos
- **Visualizaciones:** Las gráficas deben renderizarse en menos de 3 segundos

RNF-03: Compatibilidad

- **Navegadores:** Debe funcionar en Chrome, Firefox, Safari y Edge (versiones actuales)
- **Resoluciones:** Debe adaptarse a resoluciones desde 1024x768 hasta 4K

RNF-04: Escalabilidad

- **Datos:** Debe manejar eficientemente bases de datos con millones de registros de meteoros
- **Consultas:** Debe permitir añadir nuevas consultas predefinidas
- **Componentes:** La arquitectura debe permitir añadir nuevos tipos de widgets fácilmente

RNF-05: Confiabilidad

- **Disponibilidad:** El sistema debe estar disponible 24/7 con un uptime del 99 %
- **Recuperación:** Debe recuperarse automáticamente de errores de conexión
- **Validación:** Debe validar todos los datos de entrada antes del procesamiento
- **Manejo de errores:** Debe mostrar mensajes de error claros y descriptivos

RNF-06: Mantenibilidad

- **Código:** El código debe estar bien documentado y seguir estándares de desarrollo
- **Modularidad:** Los widgets deben ser independientes y reutilizables
- **Configuración:** Los parámetros de configuración deben ser externalizables
- **Logs:** El sistema debe generar logs detallados para diagnóstico

RNF-07: Seguridad

- **Validación:** Debe validar y sanitizar todas las entradas del usuario
- **SQL Injection:** Debe prevenir ataques de inyección SQL usando consultas parametrizadas
- **CORS:** Debe implementar políticas CORS apropiadas

RNF-08: Portabilidad

- **Despliegue:** Debe poder desplegarse usando Docker para facilitar la instalación
- **Base de datos:** Debe poder conectarse a diferentes instancias de MySQL
- **Configuración:** Debe permitir configuración flexible del entorno de desarrollo y producción

3.2. Arquitectura de la aplicación

Para comenzar el proyecto, como la idea principal giraba entorno al uso de Workflows, se investigaron las distintas tecnologías y librerías existentes para el desarrollo de aplicaciones basadas en ellos. Como primera opción se investigó WDL, un lenguaje específico para la definición de Workflows. Tras una pequeña investigación, fue descartado debido a que introduce mucha complejidad al proyecto. Para priorizar la simplicidad y la facilidad de programación y de conexión entre backend y frontend se decidió partir de la premisa de usar JavaScript tanto para el backend (Workflow) como para el frontend.

Una vez elegido el lenguaje JavaScript [Docs(2025)] para programar el backend, la investigación siguió por el camino de buscar librerías que pudieran facilitar la creación de estos Workflows. Se comenzó haciendo algunos tests usando el proyecto WorkflowJS [Vahid(2023)]. Al ser este proyecto propiedad de un usuario no disponía de mucha documentación o ejemplos, además introducía cierta complejidad que no era necesaria para nuestro caso de uso por lo que fue descartado.

La opción final fue desarrollar en JavaScript sobre Node.js para mantener el mismo lenguaje tanto en backend como en frontend. El desarrollo de los Workflows fue propio, sin uso de ninguna librería externa por simplicidad. Para la creación del API necesario para conectar el backend con el Frontend se utilizó el framework Express.js, que facilita la creación y el manejo de endpoints.

A continuación se describen las capas que componen la arquitectura de la aplicación.

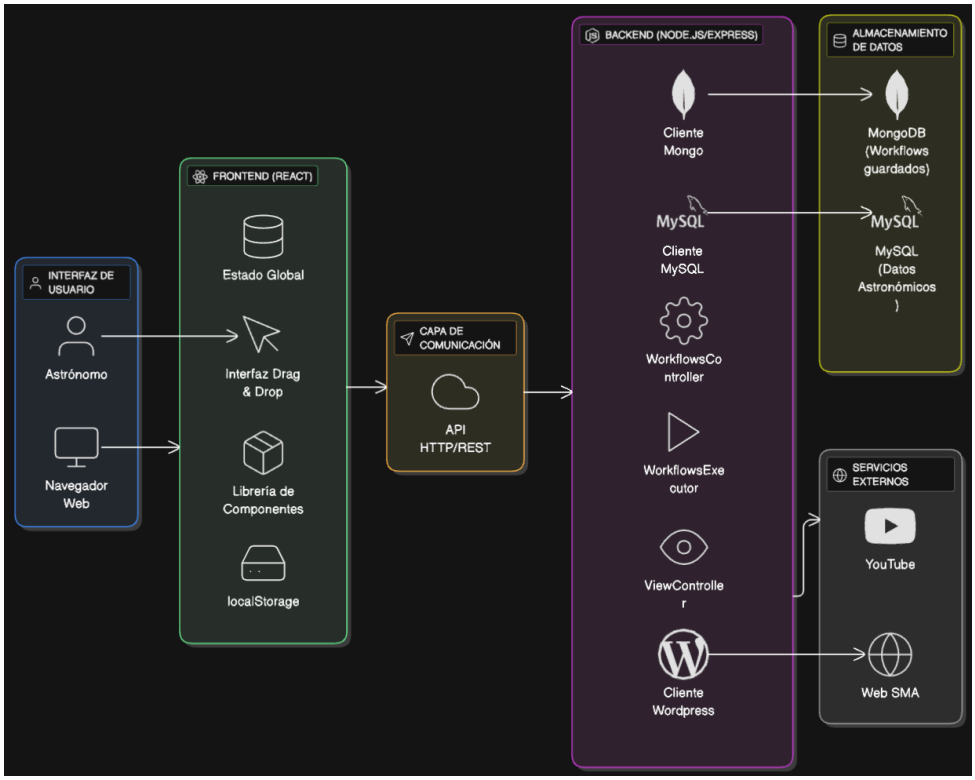

Figura 27: Diagrama de la Arquitectura de la aplicación

3.2.1. Arquitectura del Frontend

En cuanto a la interfaz de usuario se planteó hacer una aplicación basada en Widgets, pequeños componentes reutilizables con funciones definidas que se pudiesen combinar para crear la interfaz final. Para ello tendríamos una lista de componentes ocupando una pequeña parte de la pantalla y el resto sería una plantilla dónde se arrastrarían estos componentes para formar la interfaz final que define nuestra aplicación. Para hacer todo esto se eligió usar React

[by UXPin(2025)], una libería de JavaScript que facilita la creación de interfaces de usuario interactivas con funciones como 'Drag & Drop' y un diseño 'responsive'.

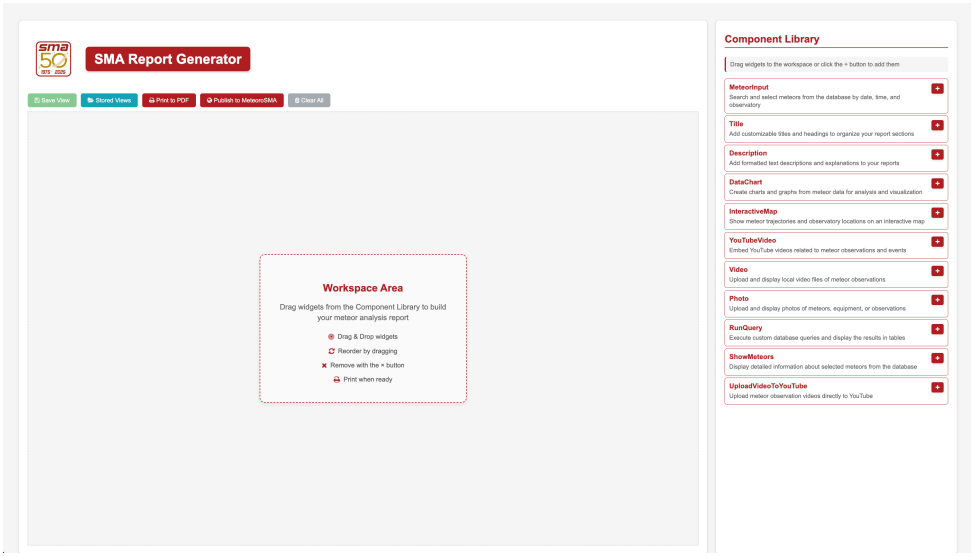

Figura 28: Interfaz de Usuario

Arquitectura Basada en Componentes

- **Componente Principal (App):** Actúa como contenedor principal y gestiona el estado global de la aplicación, incluyendo la selección de meteoros y la configuración del workflow
- **Librería de Componentes:** Área lateral que contiene una lista con todos los widgets disponibles. Cada widget tiene su nombre y descripción
- **Área de Trabajo:** Zona principal donde se construye el workflow mediante *drag & drop* de widgets
- **Botones de acción:** Diferentes botones para gestionar el guardado de vistas, la exportación del informe y la limpieza del área de trabajo

Gestión del Estado La aplicación utiliza el estado de React para coordinar la comunicación entre componentes:

- **Estado Global:** Mantiene información compartida como el meteoro seleccionado

- **Propiedades Descendentes:** Los datos se pasan desde el componente principal hacia los widgets hijos
- **Callbacks Ascendentes:** Los eventos y cambios se comunican desde los componentes hijos hacia el componente principal si es necesario, por ejemplo desde el widget **MeteorInput**
- **Identificadores Únicos:** Cada componente del workflow tiene un ID único para facilitar su gestión

Sistema de Drag & Drop La funcionalidad de arrastrar y soltar se implementa usando la librería `react-dnd`:

- **Elementos Arrastrables:** Cada widget de la librería puede ser arrastrado al área de trabajo
- **Zona de Colocación:** El área de trabajo acepta componentes y permite reordenarlos
- **Feedback Visual:** Indicaciones visuales durante el proceso de arrastre para mejorar la experiencia de usuario
- **Persistencia del Orden:** El orden de los componentes se mantiene en el estado para preservar la estructura del workflow

Comunicación con el Backend El frontend se comunica con el backend a través de una capa de servicios HTTP:

- **Cliente Axios:** Maneja todas las peticiones HTTP al backend
- **Endpoints RESTful:** Utiliza los endpoints definidos en el `WorkflowsController`
- **Manejo de Errores:** Captura y presenta errores de red y del servidor

3.2.2. Arquitectura del Backend y Patrones de Diseño

El backend implementa una arquitectura en capas combinada con varios patrones de diseño que garantizan la separación de responsabilidades, la reutilización de código y la facilidad de mantenimiento.

Arquitectura en Tres Capas y Patrón MVC El sistema sigue el patrón clásico de tres capas físicamente distribuidas, implementando el patrón MVC adaptado:

- **Capa de Presentación (Vista):** React ejecutándose en el navegador del cliente. Los componentes renderizan datos y capturan interacciones del usuario.
- **Capa de Lógica de Negocio (Controlador + Modelo):** Node.js/Express ejecutándose en el servidor, dividida en subcapas:
  - **Servidor Express:** Middleware para CORS, parseo JSON y gestión de errores
  - **WorkflowsController:** Enrutamiento HTTP, validación de parámetros y formateo de respuestas
  - **WorkflowsExecutor:** Lógica específica de tratamiento de datos, procesamiento de consultas complejas y transformación de datos
  - **Utilidades:** Motor de plantillas (`TemplateService`), repositorio de consultas (`queries.js`), de tipos de análisis de datos (`analysisTypes.js`) y de plantillas predefinidas (`templates.js`)
  - **MySqlClient:** Abstracción de la comunicación con la Base de Datos MySQL
  - **MongoClient:** Abstracción de la comunicación con la Base de Datos MongoDB
  - **WordpressClient:** Cliente para la publicación de informes a la web de la SMA (o cualquier otra web Wordpress)
  - **ViewsController y ViewsService:** Gestión de workflows (también llamados vistas) guardados
- **Capa de Datos:** Servicios MySQL y MongoDB, comúnmente ejecutándose en contenedores Docker

Esta separación física permite escalar cada capa independientemente y facilita el despliegue en diferentes entornos.

Patrón Observer (Auto-Propagación) La funcionalidad más innovadora del sistema es el mecanismo de auto-propagación implementado en el frontend. Cuando un usuario selecciona un meteoro en el componente `MeteorInput`, esta selección se propaga automáticamente a todos los demás componentes del workflow que requieren datos específicos de meteoro.

Este patrón funciona mediante:

- **Estado Global:** React mantiene el estado del meteoro seleccionado a nivel de aplicación
- **Observadores:** Cada componente que depende de datos de meteoro se suscribe a cambios en esta selección
- **Notificación Automática:** Cuando cambia la selección, todos los componentes dependientes se actualizan automáticamente sin intervención manual
- **Lazy Loading:** Los componentes solo cargan datos cuando hay una selección válida, optimizando el rendimiento

Patrón Template Method El `TemplateService` implementa el patrón Template Method para la generación de contenido dinámico:

- Define la estructura general de consultas y textos descriptivos usando la sintaxis `${variable}`
- Permite personalización mediante la sustitución de variables específicas del meteoro seleccionado
- Mantiene consistencia en el formato mientras permite flexibilidad en el contenido
- Facilita la reutilización de plantillas para diferentes tipos de informes

3.3. Modelo de datos

El almacenamiento de los datos ya estaba definido previamente a este proyecto mediante una base de datos MySQL, por lo que la aplicación se ha desarrollado en base a esto [García Escobar(2023)]. El sistema utiliza una base de datos MySQL diseñada específicamente para almacenar y gestionar datos astronómicos de observaciones de meteoros. La estructura está organizada en torno a tres entidades centrales que permiten un análisis científico completo de los fenómenos meteorológicos.

Se ha incluido una base de datos MongoDB para el guardado de definiciones de workflows.

3.3.1. Entidades Principales

Meteoro La tabla *Meteoro* constituye el núcleo central del sistema, almacenando los datos básicos de cada observación: identificador único, fecha y hora de ocurrencia. Esta tabla actúa como punto de referencia para todas las demás entidades del sistema.

Observatorio La tabla *Observatorio* contiene información detallada de las estaciones de observación, incluyendo: coordenadas geográficas, metadatos (nombre, descripción, créditos...), etc.

Lluvia La tabla *Lluvia* define las lluvias de meteoros conocidas, con información sobre períodos de actividad, velocidades características y nomenclatura astronómica estándar.

3.3.2. Informes de Análisis Científico

El sistema genera tres tipos principales de informes de análisis:

Informe-Z Contiene los análisis de trayectoria más completos, incluyendo:

- Coordenadas astronómicas del radiante
- Cálculos de velocidad, aceleración y trayectoria
- Análisis de error y precisión estadística
- Ecuaciones paramétricas del movimiento
- Elementos orbitales calculados

Informe Fotometria Almacena análisis fotométricos especializados:

- Curvas de luz y análisis de magnitud
- Cálculos de masa fotométrica

Informe Radiante Contiene análisis específicos del radiante:

- Asociaciones con lluvias conocidas
- Velocidades angulares y distancias angulares
- Análisis de agrupación de trayectorias

3.3.3. Datos de Soporte y Detalle

Datos de Trayectoria Tres tablas especializadas almacenan diferentes aspectos de las trayectorias obtenidas del análisis de las observaciones.

3.3.4. Diagrama Entidad-Relación

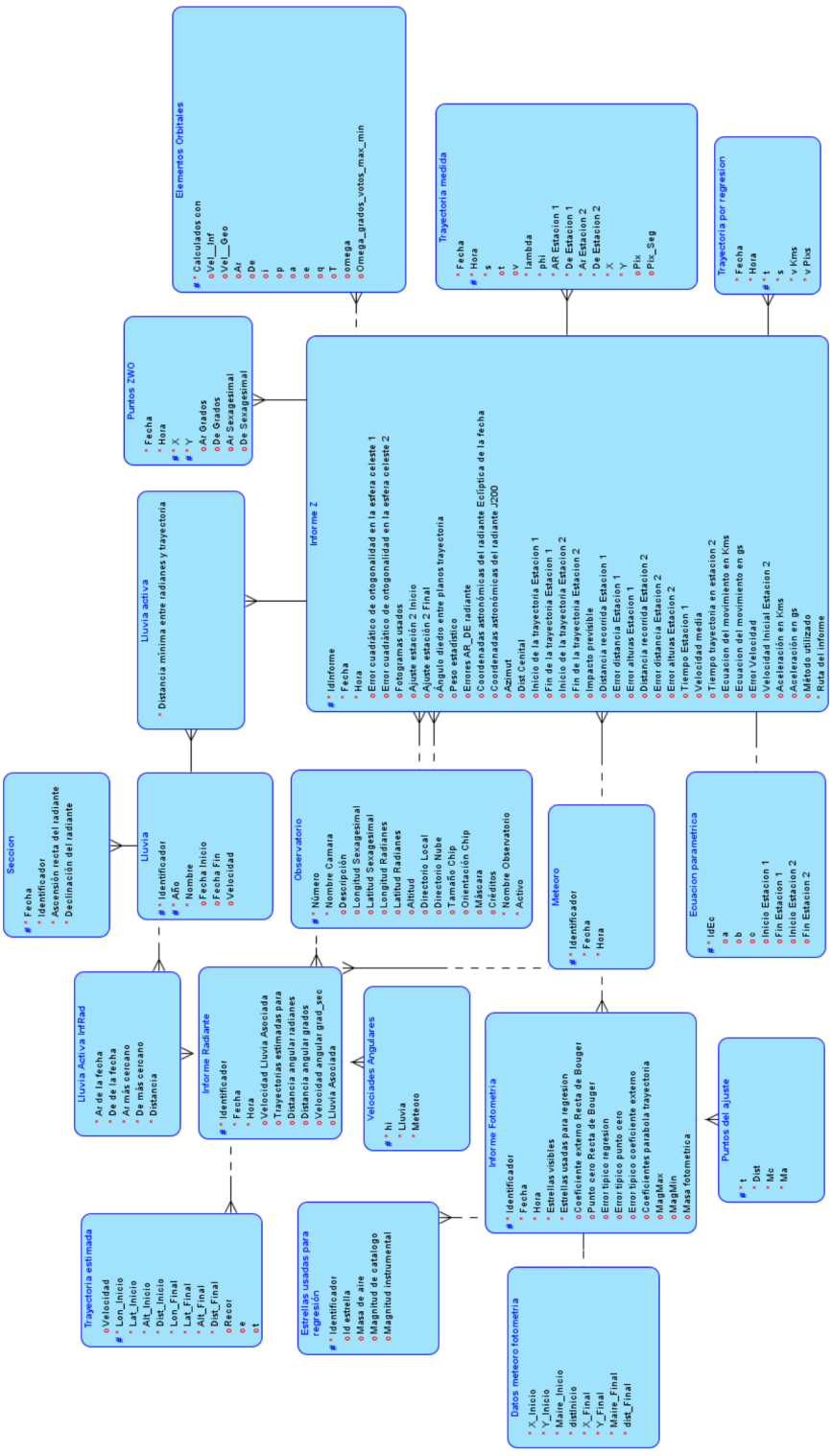

Figura 29: Modelo conceptual de la Base de Datos

4

Ciclos de Desarrollo

4.1. Metodología de Desarrollo

Para el desarrollo de este proyecto se adoptó la Metodología Ágil, que permitió una aproximación iterativa e incremental al desarrollo del sistema. Esta metodología se caracterizó por la división del trabajo en ciclos cortos de desarrollo (sprints), cada uno enfocado en la entrega de funcionalidades específicas [Posgrado(2023)].

La metodología ágil facilitó la adaptación continua a los requisitos emergentes del proyecto, permitiendo ajustes en el diseño y funcionalidad basados en el feedback obtenido durante cada ciclo de desarrollo. Esta aproximación resultó especialmente valiosa dado el carácter exploratorio del proyecto, donde la comprensión de los requisitos y las mejores soluciones técnicas se fueron refinando progresivamente a través de la experimentación y validación con datos reales.

Cada ciclo de desarrollo siguió un patrón consistente: planificación de objetivos específicos, desarrollo de funcionalidades, pruebas exhaustivas y evaluación de resultados, lo que garantizó la calidad y robustez del sistema final mientras se mantenía la flexibilidad necesaria para incorporar mejoras y optimizaciones identificadas durante el proceso de desarrollo.

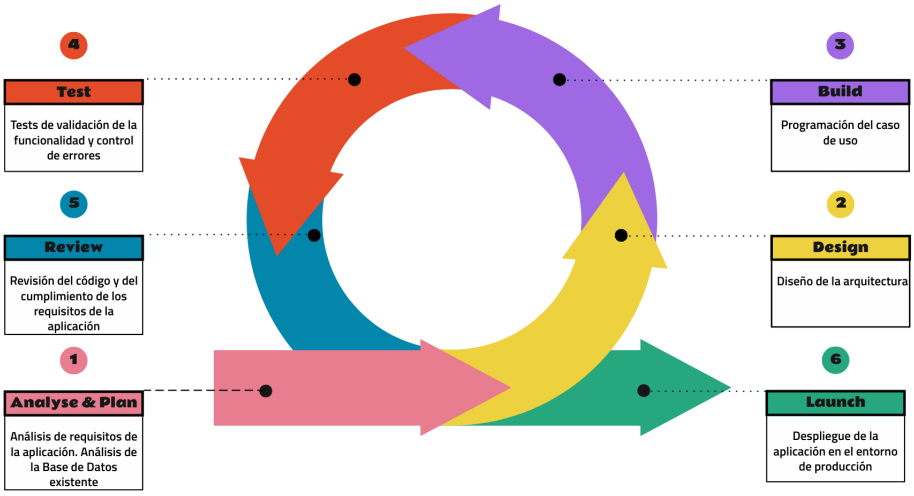

Figura 30: Ciclos de la Metodología Ágil

4.2. Ciclo 1: Análisis de la Base de Datos

Como comienzo de este proyecto se hizo un análisis de la base de datos de la que disponíamos con el objetivo de comprender los datos con los que trabajar y las relaciones entre las distintas entidades que se generan cuando hay una detección de un meteoro.

Durante este ciclo se estudió en detalle la estructura de la base de datos MySQL que almacena los datos astronómicos de observaciones de meteoros. Se analizaron las entidades principales (Meteoro, Observatorio, Lluvia), los diferentes tipos de informes científicos (Informe-Z, Informe Fotometría, Informe Radiante), y las relaciones entre todas estas entidades.

Este análisis permitió identificar los datos disponibles y diseñar las consultas SQL necesarias para extraer información relevante para los diferentes tipos de visualizaciones y análisis que la aplicación debía proporcionar. La estructura detallada del modelo de datos se describe en la sección 'Modelo de datos' del anterior capítulo.

Como resultado de este análisis se generó un conjunto de consultas SQL predefinidas organizadas por categorías temáticas, que posteriormente se integrarían en el sistema de workflows para facilitar la generación automática de informes y visualizaciones.

4.3. Ciclo 2: Programación de un primer caso de uso

En este segundo ciclo se abordó la implementación de la infraestructura básica del sistema, desarrollando dos componentes fundamentales: la conectividad con la base de datos y una interfaz de usuario inicial para validar la arquitectura propuesta.

4.3.1. Desarrollo del Cliente de Base de Datos

Se implementó la clase `MySqlClient` como abstracción principal para la conectividad con la base de datos MySQL. Esta clase proporciona un pool de conexiones optimizado y métodos estandarizados para la ejecución de consultas:

- **Pool de conexiones**: Se configuró un pool de hasta 10 conexiones concurrentes para mejorar el rendimiento y la gestión de recursos
- **Métodos de consulta**: Se implementaron dos métodos principales: `runQuery()` para consultas SQL directas con parámetros, y `runPredefinedQuery()` para consultas predefinidas organizadas por categorías temáticas
- **Gestión de errores**: Se incorporó manejo robusto de errores y conexiones, con reintentos automáticos y logging detallado

Como complemento se desarrolló el módulo `queries.js`, que centraliza más de 60 consultas SQL predefinidas organizadas en categorías como análisis de trayectorias, fotometría, análisis temporal y correlaciones avanzadas. Este enfoque facilita el mantenimiento y reutilización de consultas complejas.

4.3.2. Interfaz de Usuario Inicial

Se desarrolló una interfaz web basada en React que implementa un sistema de widgets modulares con funcionalidad de arrastrar y soltar (*drag & drop*). Los componentes principales

incluyen:

- **Sistema de componentes modulares**: Se definió una arquitectura basada en widgets reutilizables, cada uno encapsulando una funcionalidad específica (gráficos, mapas, consultas, etc.)
- **Funcionalidad drag & drop**: Utilizando React DnD, se implementó la capacidad de arrastrar componentes desde una biblioteca y organizarlos dinámicamente en el área de trabajo
- **Widgets iniciales**: Se crearon componentes básicos como `Title`, `Description` y `DataChart` con datos de prueba para validar la arquitectura
- **Exportación de informes**: Se integró funcionalidad para exportar el contenido generado tanto en formato PDF como HTML estático

Este ciclo estableció las bases técnicas del sistema, validando la viabilidad de la arquitectura modular propuesta y confirmando la correcta integración entre el frontend React y el backend.

4.4. Ciclo 3: Ampliación de Widgets

El tercer ciclo se centró en el desarrollo de funcionalidades avanzadas de los widgets, implementando el sistema de auto-propagación de datos y la ampliación significativa del catálogo de componentes disponibles.

4.4.1. Componente MeteorInput y Sistema de Auto-Propagación

Se desarrolló el componente `MeteorInput` como elemento central del sistema de workflows, funcionando como fuente de datos para todos los demás widgets. Sus características principales incluyen:

- **Búsqueda flexible de meteoros**: Permite buscar meteoros por ID específico o mediante criterios combinados (fecha, rango horario, observatorio)
- **Sistema de selección**: Presenta resultados de búsqueda en listas interactivas que permiten seleccionar el meteoro de interés

- **Auto-propagación de datos**: Una vez seleccionado un meteoro, el componente notifica automáticamente a todos los widgets dependientes a través del sistema de estado compartido de React

4.4.2. Desarrollo del Backend de Workflows

Se implementó un sistema robusto de controlador y ejecutor para manejar las peticiones del frontend:

- **WorkflowsController**: Maneja el enrutado de todas las peticiones relacionadas con workflows, incluyendo búsqueda de meteoros, ejecución de consultas y obtención de datos de observatorios
- **WorkflowsExecutor**: Encapsula la lógica de negocio y coordina las operaciones con la base de datos, incluyendo búsquedas complejas con múltiples criterios y soporte para consultas parametrizadas

4.4.3. Ampliación del Catálogo de Widgets

Se desarrollaron múltiples widgets especializados y se mejoraron los existentes:

- **DataChart**: Componente para visualización de datos mediante gráficos de líneas y dispersión
- **ShowMeteors**: Widget para mostrar listas detalladas de meteoros
- **InteractiveMap**: Mapa interactivo que visualiza ubicaciones de observatorios y trayectorias de meteoros utilizando Google Maps API
- **RunQuery**: Componente para ejecución de consultas SQL personalizadas con presentación de resultados en formato tabular
- **Componentes multimedia**: `Video`, `Photo`, `YouTubeVideo` para integración de contenido multimedia en los informes

Cada widget fue diseñado siguiendo principios de modularidad y reutilización.

4.5. Ciclo 4: Validación y Pruebas

El cuarto ciclo se dedicó a la validación exhaustiva del sistema desarrollado, implementando diferentes niveles de pruebas para garantizar la robustez, fiabilidad y correcta funcionalidad de todos los componentes.

4.5.1. Pruebas de Conectividad y Base de Datos

Se realizaron pruebas sistemáticas de la infraestructura de datos:

- **Validación de consultas SQL**: Se verificó la correcta ejecución de todas las consultas predefinidas en el módulo `queries.js`, asegurando que devuelven resultados válidos y estructurados
- **Pruebas de rendimiento**: Se evaluó el comportamiento del pool de conexiones bajo diferentes cargas de trabajo, confirmando la gestión eficiente de recursos
- **Manejo de errores**: Se validaron los mecanismos de recuperación ante fallos de conexión, timeouts y consultas malformadas

4.5.2. Pruebas de Funcionalidad Frontend

Se ejecutaron pruebas de la interfaz de usuario:

- **Funcionalidad drag & drop**: Se validó el correcto funcionamiento del sistema de arrastrar y soltar componentes, incluyendo reorganización dinámica y persistencia de posiciones
- **Auto-propagación de datos**: Se verificó que la selección de meteoros en `MeteorInput` se propaga correctamente a todos los widgets dependientes
- **Responsividad**: Se probó la interfaz en diferentes resoluciones y dispositivos para garantizar una experiencia de usuario consistente
- **Validación de formularios**: Se confirmó el correcto funcionamiento de validaciones en tiempo real y manejo de estados de error

4.5.3. Pruebas de Integración

Se realizaron pruebas end-to-end para validar el flujo completo de trabajo:

- **Flujos de búsqueda**: Se verificó el funcionamiento completo desde la búsqueda de meteoros hasta la visualización de datos en diferentes widgets
- **Generación de informes**: Se validó la funcionalidad de exportación en PDF, confirmando la preservación del formato y contenido
- **Integración con APIs externas**: Se probó la conectividad con Google Maps API y YouTube API
- **Compatibilidad de navegadores**: Se verificó el funcionamiento en diferentes navegadores web (Chrome, Firefox, Safari, Edge)

4.5.4. Validación con Datos Reales

Se realizaron pruebas utilizando datos reales de la base de datos astronómica:

- **Casos de uso científicos**: Se validó la aplicación con flujos de trabajo típicos de análisis astronómico, desde la selección de meteoros hasta la generación de informes científicos
- **Precisión de cálculos**: Se verificó la exactitud de los cálculos y visualizaciones generadas por los widgets

Este ciclo confirmó la estabilidad y fiabilidad del sistema, identificando y corrigiendo errores menores antes del despliegue final y verificando que los requisitos de la aplicación se cumplen.

4.6. Ciclo 5: Empaquetado y Despliegue de la aplicación

El quinto y último ciclo se enfocó en la preparación de la aplicación para entornos de producción, implementando estrategias que faciliten la instalación, mantenimiento y escalabilidad del sistema.

4.6.1. Contenerización con Docker

Se desarrolló una estrategia de contenerización multi-servicio utilizando Docker:

- **Contenedor Backend**: Se creó un Dockerfile para el servicio backend basado en Node.js 18, que incluye la instalación de dependencias, configuración del entorno y exposición del puerto 3000 para la API REST
- **Contenedor Frontend**: Se implementó un build multi-etapa que primero compila la aplicación React y posteriormente sirve los archivos estáticos mediante un servidor Nginx
- **Contenedores Base de Datos**: Se configuró un servicio MySQL 8.0 con inicialización automática de esquemas y datos de prueba mediante scripts SQL ubicados en el directorio `db-init`. Se configuró también un servicio MongoDB para el guardado de workflows.

4.6.2. Orquestación con Docker Compose

Se diseñó un archivo `docker-compose.yml` que define la arquitectura completa del sistema:

- **Servicios coordinados**: Cuatro servicios configurados con dependencias entre sí asegurando el orden de inicio controlado (base de datos MongoDB, base de datos MySQL, backend y frontend)
- **Volúmenes persistentes**: Configuración de almacenamiento persistente para datos de la base de datos
- **Variables de entorno**: Gestión centralizada de configuración mediante archivos de entorno que facilitan el despliegue en diferentes sistemas

4.6.3. Estrategia de Despliegue

Se implementó un proceso de despliegue simplificado que permite la instalación rápida del sistema:

- **Despliegue con un comando**: Mediante `docker-compose up`, se inicia toda la infraestructura necesaria de forma automática

- **Inicialización automática**: Los scripts de inicialización de base de datos se ejecutan automáticamente en el primer arranque, creando el esquema completo y poblando datos de ejemplo
- **Configuración flexible**: El sistema permite personalizar parámetros como puertos, credenciales de base de datos y ubicaciones de archivos mediante variables de entorno
- **Monitorización**: Se incluyó un endpoint de health check que permite verificar el estado de todos los servicios

4.6.4. Documentación de Despliegue

Se generó documentación comprensiva que incluye:

- **Guía de instalación**: Instrucciones paso a paso para instalación del sistema
- **Troubleshooting**: Soluciones a problemas comunes de instalación y configuración

Este ciclo culminó con la validación del proceso de despliegue en múltiples entornos, confirmando la portabilidad y facilidad de instalación del sistema desarrollado, lo que garantiza su adopción efectiva por parte de los usuarios finales de la SMA.

5
Conclusiones

5.1. Conclusiones

En este Trabajo Fin de Grado se ha desarrollado exitosamente una aplicación para el análisis y visualización de datos astronómicos que aborda las carencias identificadas en el sistema de gestión de datos astronómicos de la Sociedad Malagueña de Astronomía (SMA). El proyecto ha logrado crear una herramienta integral que facilita el análisis científico y la divulgación de fenómenos meteorológicos mediante una interfaz intuitiva y funcionalidades avanzadas de visualización.

El desarrollo ha cumplido satisfactoriamente los objetivos planteados al inicio. El objetivo principal era el procesamiento de los datos astronómicos para su análisis y divulgación por parte de los científicos, y este se ha realizado mediante la implementación de un sistema completo que procesa datos de la base de datos MySQL de la SMA y los transforma en visualizaciones científicas especializadas. El sistema permite a los investigadores acceder a múltiples perspectivas de análisis (trayectorias, fotometría, patrones temporales, correlaciones) y facilita la divulgación científica a través de informes multimedia exportables.

Uno de los subobjetivos era la implementación de mecanismos para construir plantillas de extracción de datos, conectadas a la base de datos y a la interfaz de usuario, para lo cual se implementó un sistema robusto de plantillas de consultas SQL predefinidas organizadas por categorías temáticas en el módulo `queries.js`.

Además de estas queries predefinidas, se han definido una serie de análisis científicos específicos en el módulo `analysisTypes.js` que pueden combinar varias de ellas para la representación de información interesante en el widget **DataChart**.

Otros widgets usan estas queries para permitir la automatización de estos procesos de re-

presentación.

Relacionado a lo expuesto anteriormente, otro de los objetivos era la implementación de procesos de extracción de datos de la base de datos y su conexión con las plantillas definidas anteriormente. Para satisfacerlo se desarrolló una arquitectura completa de backend que implementa los procesos de extracción de datos mediante las clases explicadas anteriormente: **MySqlClient**, **WorkflowsController** y **WorkflowsExecutor**.

La conexión con las plantillas se realiza a través del sistema de auto-propagación, donde la selección de un meteoro en `MeteorInput` activa automáticamente la ejecución de las consultas predefinidas en todos los widgets dependientes.

Para finalizar el repaso de los objetivos que se plantearon, el último consistía en el uso de los desarrollos anteriores para obtener visualizaciones útiles para los astrónomos y para su divulgación científica. Para ello se creó una interfaz web completa basada en React con sistema de widgets modulares que genera visualizaciones especializadas:

- **Widgets de descripción**: Para incluir títulos y descripciones sobre lo incluido en el informe. Las descripciones pueden generarse automáticamente gracias al motor de plantillas, que rellena variables en el texto mediante consultas a la base de datos.

- **Widget DataChart**: Genera gráficos de diferentes tipos (líneas, barras, área, columnas y puntos) con los diferentes análisis científicos comentados anteriormente.

- **Widget InteractiveMap**: Integra Google Maps API para visualizar ubicaciones de observatorios y trayectorias de meteoros con coordenadas precisas, facilitando el análisis geográfico de estos fenómenos.

- **Widget RunQuery**: Permite la ejecución de una query definida por el usuario a la base de datos y muestra los resultados en una tabla.

- **Widgets multimedia**: Incluyen capacidades de subida de fotografías, vídeos y publicación automática en YouTube para documentación completa de fenómenos meteorológicos.

- **Sistema de exportación**: Permite generar informes completos en formato PDF, manteniendo el formato y diseño para facilitar la divulgación científica. Además se permite la exportación directa del informe o noticia generados a la web de la SMA.

El sistema de **drag & drop** permite a los usuarios construir workflows personalizados sin conocimientos técnicos avanzados, democratizando el acceso a las herramientas de análisis astronómico. La funcionalidad de auto-propagación de datos optimiza la experiencia de usuario, reduciendo la complejidad operativa y permitiendo que los investigadores se concentren en el análisis científico en lugar de en la manipulación técnica de datos.

5.1.1. Metodología de Desarrollo

La adopción de la **Metodología Ágil** resultó fundamental para el éxito del proyecto, permitiendo una aproximación iterativa e incremental que se adaptó perfectamente al carácter exploratorio del desarrollo. Los cinco ciclos de desarrollo (análisis de base de datos, programación del primer caso de uso, ampliación de widgets, validación y pruebas, y empaquetado y despliegue) facilitaron la adaptación continua a requisitos emergentes y la validación progresiva de funcionalidades con datos reales.

5.1.2. Impacto Científico y Social

Como resultado de este TFG, se ha obtenido una herramienta que:

- **Facilita la investigación astronómica**: Los astrónomos de la SMA pueden ahora generar informes científicos completos y visualizaciones especializadas de forma eficiente, reduciendo significativamente el tiempo necesario para el análisis de datos.

- **Mejora la divulgación científica**: La capacidad de generar informes multimedia con mapas interactivos, gráficos especializados y contenido audiovisual facilita la comunicación de resultados científicos al público general.

- **Democratiza el acceso a datos científicos**: La interfaz intuitiva permite que investigadores sin conocimientos técnicos profundos de bases de datos puedan acceder y analizar la información astronómica disponible.

- **Establece un marco escalable**: La arquitectura modular permite la incorporación futura de nuevos tipos de análisis y visualizaciones según evolucionen las necesidades científicas.

5.2. Líneas Futuras

El proyecto establece una base sólida para futuras mejoras y extensiones que podrían incluir:

- **Integración de inteligencia artificial**: Implementación de algoritmos de machine learning para detección automática de patrones en los datos meteorológicos y predicción de fenómenos.
- **Análisis en tiempo real**: Desarrollo de capacidades para procesar y visualizar datos de meteoros en tiempo real durante eventos astronómicos.
- **Multi-idioma**: Extensión del sistema para incluir la posibilidad de cambiar el idioma.
- **Móvil y aplicaciones nativas**: Desarrollo de aplicaciones móviles para facilitar el acceso y uso del sistema desde dispositivos portátiles.
- **Análisis estadístico avanzado**: Incorporación de herramientas estadísticas más sofisticadas para análisis de grandes volúmenes de datos históricos.

5.3. Reflexión Final

En conclusión, este proyecto ha logrado crear una solución que transforma la forma en que la Sociedad Malagueña de Astronomía gestiona, analiza y presenta sus datos meteorológicos. La combinación de tecnologías modernas de desarrollo web, principios de usabilidad centrada en el usuario y metodologías ágiles de desarrollo ha resultado en una herramienta que establece un nuevo estándar para la gestión de datos astronómicos en la organización.

El sistema desarrollado representa una contribución significativa tanto al campo de la ingeniería de software como a la investigación astronómica, demostrando cómo las tecnologías de la información pueden facilitar y potenciar el trabajo científico. La arquitectura modular y

la documentación realizada garantizan su sostenibilidad y evolución futura, asegurando que la SMA pueda continuar beneficiándose de esta herramienta en el futuro.

Referencias

[UMA/SMA(2017)] UMA/SMA, Red de detección de bólidos y meteoros de la uma y la sma, 2017. URL: http://meteoros.astromalaga.es.

[Madiedo Gil(2017)] J. Madiedo Gil, Meteoros y meteoritos, Marcombo, 2017.

[SEA(2022)] SEA, Lluvia de estrellas, 2022. URL: https://www.sea-astronomia.es/glosario/lluvia-de-estrellas.

[García Escobar(2023)] I. García Escobar, Diseño, desarrollo y puesta en marcha de una base de datos para el estudio y análisis de lluvias de meteoros, riuma (2023).

[React(2025)] React, React api reference, 2025. URL: https://es.react.dev/reference/react.

[MySQL(2025)] MySQL, Mysql 8.4 reference manual, 2025. URL: https://dev.mysql.com/doc/refman/8.4/en/.

[MongoDB(2025)] MongoDB, Mongodb docs, 2025. URL: https://www.mongodb.com/docs/manual/.

[SMA(1975)] SMA, Sociedad malageña de astronomia, 1975. URL: https://www.astromalaga.es/.

[Castellón(2020)] A. Castellón, La uma dispone de una red de detección de meteoros y bólidos junto con la sociedad malagueña de astronomía, 2020. URL: https://www.uma.es/sala-de-prensa/noticias/la-uma-dispone-de-una-red-de-deteccion-de-meteoros-y-bolidos-junto-con-la-sociedad-

[Castellón(2015)] A. Castellón, La red de seguimiento de bólidos y meteoritos de la sociedad malagueña de astronomía, 2015. URL: http://hdl.handle.net/10630/9957.

[IAA-CSIC(1998)] IAA-CSIC, Bootes, 1998. URL: https://bootesnetwork.com/es/.

[Tejada(2023)] P. Tejada, Bootes la red de telescopios robóticos española en la que participa la uma, 2023. URL: https://www.malagahoy.es/malaga/BOOTES-telescopios-roboticos-participa-UMA_0_1770124796.html.

[NASA(2018)] NASA, ¿qué es una lluvia de meteoritos?, 2018. URL: https://spaceplace.nasa.gov/meteor-shower/sp/.

[Tableau(2025)] Tableau, Ventajas y desventajas de la visualización de datos, 2025. URL: https://www.tableau.com/visualization/what-is-data-visualization#advantages-disadvantages.

[López Alonso(2015)] R. López Alonso, Como realizar infografías, 2015. URL: https://estudio-grafico.blogspot.com/2015/07/como-realizar-infografias.html.

[Docs(2025)] M. W. Docs, Guía de javascript, 2025. URL: https://developer.mozilla.org/es/docs/Web/JavaScript/Guide.

[Vahid(2023)] Vahid, Workflowjs, 2023. URL: https://github.com/vhidvz/workflow-js.

[by UXPin(2025)] S. by UXPin, Why use react?, 2025. URL: https://www.uxpin.com/studio/blog/why-use-react/.

[Posgrado(2023)] I. E. D. Posgrado, Qué son las metodologías ágiles y tipos más comunes, 2023. URL: https://iep.edu.es/que-son-metodos-agiles-tipos-mas-comunes/.

[Teixeira and James(2021)] M. Teixeira, G. James, Sqltools docs, 2021. URL: https://vscode-sqltools.mteixeira.dev/en/home#features.

Apéndice A
Manual de uso

A.1. Acceso a la Aplicación

1. Abrir el navegador web
2. Navegar a la URL proporcionada por el administrador (o `http://localhost:4000`)
3. La aplicación se cargará mostrando la interfaz principal

A.2. Componentes de la Interfaz Principal

La aplicación consta de tres áreas principales:

- **Librería de Widgets** (Panel lateral derecho): Contiene todos los widgets disponibles para agregar al informe
- **Constructor de Informes** (Área central): Donde se construye el informe organizando widgets
- **Panel de Control** (Área superior): Contiene opciones de exportación, publicación y guardado del informe

A.3. Flujo de Trabajo Científico

A.3.1. Proceso Centrado en un Meteoro

Normalmente queremos analizar un meteoro en concreto:

1. **Selección de Meteoro**: Usar el widget `MeteorInput` para búsqueda específica
2. **Auto-población**: Los widgets se rellenan automáticamente con datos relacionados
3. **Personalización**: Agregar widgets adicionales según necesidades científicas
4. **Exportación**: Generar el informe final en formato PDF o publicar a la web de la SMA

A.3.2. Proceso Genérico sobre los Meteoros disponibles

Si queremos hacer un análisis general de la base de datos no necesitamos usar el widget `MeteorInput`. Podemos utilizar el resto de widgets para elaborar nuestro informe al igual que en el anterior proceso, a excepción del widget `InteractiveMap` que está especializado para usarse sobre un sólo meteoro.

A.3.3. Widget MeteorInput (Punto de inicio recomendado)

El widget **MeteorInput** es el punto de partida recomendado para crear informes:

Opciones de búsqueda:

- **Identificador**: Búsqueda directa por ID de meteoro
- **Fecha**: Búsqueda por fecha exacta
- **Rango temporal**: Búsqueda dentro de un período usando hora de inicio y fin
- **Observatorio**: Filtrar por observatorio específico

Resultados de búsqueda:

- Los resultados aparecen como **tarjetas interactivas** con detalles del meteoro
- Cada tarjeta muestra: ID, Fecha, Hora, Observatorio
- **Al hacer click en cualquier tarjeta** se selecciona ese meteoro

A.4. Widgets Especializados

A.4.1. Widgets de Visualización de Datos

Widget DataChart

- Crea gráficos de líneas, dispersión, barras, área y columnas dependiendo del análisis seleccionado
- **Se auto-rellena** cuando se selecciona un meteoro
- **Opciones de consulta integrales** organizadas por categorías de análisis científico

Widget InteractiveMap

- Muestra ubicaciones de observatorios y trayectorias de meteoros
- **Se auto-rellena** cuando se selecciona un meteoro
- Muestra ubicaciones de observatorios para el meteoro seleccionado
- Muestra trayectorias con coordenadas precisas de inicio/fin
- Muestra las alturas de inicio/fin de la trayectoria

A.4.2. Widgets de Consulta de Base de Datos

Widget RunQuery

- Permite ejecutar consultas personalizadas de base de datos
- Introducir consultas SQL en el área de texto
- Hacer clic en 'Run Query' para ejecutar y ver resultados
- Los resultados se muestran en formato de tabla

A.4.3. Widgets de Texto y Multimedia

Widgets de Texto

- **Widget Title**: Permite crear encabezados para el informe
- **Widget Description**: Permite agregar contenido de texto explicativo. Permite también usar plantillas predefinidas que se rellenan dependiendo del meteoro seleccionado en **MeteorInput**

Widgets Multimedia

- **Widget YouTubeVideo**: Permite incrustar videos de YouTube mediante su link
- **Widget Photo**: Permite agregar fotografías al informe
- **Widget Video**: Permite cargar archivos de video locales

- **Widget UploadVideoToYouTube**: Nos da la facilidad para subir vídeos locales a YouTube

A.5. Organización del Informe

A.5.1. Organización de Widgets

1. **Arrastrar para reordenar**: Hacer click y arrastrar cualquier widget para moverlo hacia arriba o abajo
2. **Eliminar widgets**: Hacer clic en el botón 'X' en la esquina superior derecha
3. **Múltiples instancias**: Se pueden agregar múltiples instancias del mismo tipo de widget

A.5.2. Mejores Prácticas para la Estructura del Informe

1. **Comenzar con MeteorInput**: Usar como herramienta principal de selección de datos
2. **Agregar contexto**: Usar widgets Title y DescriptionText para proporcionar antecedentes
3. **Presentar datos**: Usar widgets DataChart e InteractiveMap
4. **Mostrar datos en bruto**: Incluir widgets RunQuery para información detallada si es necesario
5. **Agregar multimedia**: Incluir fotos o videos para hacer el informe más atractivo

A.6. Guardado, carga y borrado de Workflows

A.6.1. Guardado

1. Hacemos click en el botón 'Save View'
2. Introducimos un nombre para nuestro workflow (no debe existir ya en los workflows guardados, de lo contrario se nos mostrará un aviso para que introduzcamos otro nombre)
3. Se nos mostrará una notificación del correcto guardado

A.6.2. Carga y borrado

1. Hacemos click en el botón 'Stored Views'
2. Nos aparecerá una lista de los workflows guardados, con su nombre, el número y nombre de widgets que lo componen y la fecha en que se guardó
3. Para cargar uno haremos click en 'Load' y se nos abrirá un diálogo que nos permite guardar nuestro workflow actual antes de cargar
4. Desde este mismo panel podemos hacer click en 'Delete' para borrar un workflow. Se nos abrirá un diálogo de confirmación

A.7. Exportación del Informe

A.7.1. Imprimir a PDF

1. Hacemos clic en el botón 'Print to PDF' en el panel de control superior
2. Se abrirá el diálogo de impresión del navegador
3. Seleccionamos 'Guardar como PDF' como destino
4. Elegimos la configuración de impresión
5. Hacemos click en 'Imprimir' para generar el PDF

A.7.2. Publicar en la web

1. Hacemos click en el botón 'Publish to MeteoroSMA en el panel de control superior
2. Se mostrará el HTML a exportar en crudo
3. Añadimos un título para el post
4. Hacemos click en 'Publish to WordPress' para hacer efectiva la publicación

Apéndice B
Setup del entorno de desarrollo

Esta sección describe el proceso completo para configurar un entorno de desarrollo local del Generador de Informes de Observación de Meteoros, incluyendo la instalación de dependencias, configuración del proyecto y flujo de trabajo para desarrollo.

B.1. Prerrequisitos del Entorno de Desarrollo

Para configurar un entorno de desarrollo completo se requieren las siguientes herramientas:

Software Requerido

- **Docker Desktop**: Versión 20.10+ con Docker Compose 2.0+ para containerización de la base de datos
- **Node.js**: Versión 14.x o superior con npm 6.x+ para desarrollo JavaScript
- **Git**: Para control de versiones y clonado del proyecto
- **Editor de código**: Visual Studio Code recomendado con extensiones para React y Node.js

Herramientas Opcionales de Desarrollo

- **Extensión 'SQLTools' de Visual Studio Code [Teixeira and James(2021)]**: Herramienta de gestión e inspección de la base de datos MySQL
- **Postman**: Para pruebas de endpoints de la API
- **MongoDB Compass**: Herramienta de gestión e inspección de la base de datos MongoDB

B.2. Configuración del Entorno de Desarrollo

Paso 1: Clonado del Repositorio

```
git clone https://github.com/Jesus21G/sma-workflows.git
cd tfg
```

Paso 2: Configuración de la Base de Datos El proyecto utiliza MySQL containerizado para el desarrollo:

```
# Iniciar solo los servicios MySQL y MongoDB
docker-compose -f docker-compose-dev.yml up
```

Si se quiere levantar la aplicación completa (incluyendo backend y UI) en Docker containers (docker-compose up) se recomienda utilizar el siguiente comando para hacer la build del UI y que no use versiones previas cacheadas:

```
docker-compose up --build ui
```

La base de datos se inicializa automáticamente con los scripts en `db-init/`:

- `01_astro.create.mysql.sql`: Creación de tablas
- `02_astro.insert.mysql.sql`: Datos de ejemplo
- `03_astro.constraints.mysql.sql`: Restricciones y relaciones

Paso 3: Configuración del Backend

```
# Copiar el fichero env.example a un fichero .env y configurar
    según nuestro entorno de desarrollo
cp env.example .env

# Navegar al directorio del backend
cd src
```

```
# Instalar dependencias Node.js
npm install

# Ejecutar en modo desarrollo con auto-recarga
npm run devel
```

El servidor backend se ejecutará en `http://localhost:3000` con las siguientes características:

- Recarga automática de código con Nodemon
- Transpilación ES6+ con Babel
- API REST con endpoints en `/api/workflows/*`

Paso 4: Configuración del Frontend En una nueva terminal:

```
# Navegar al directorio del frontend
cd ui

# Copiar el fichero env.example a un fichero .env y configurar
    según nuestro entorno de desarrollo
cp env.example .env

# Instalar dependencias React
npm install

# Iniciar servidor de desarrollo
npm start
```

El servidor frontend se ejecutará en `http://localhost:4000` con:

- Hot reloading automático
- Proxy configurado hacia el backend

B.3. Configuración de APIs Externas

Google Maps API

1. Crear proyecto en Google Cloud Console
2. Habilitar Google Maps JavaScript API
3. Generar clave API
4. Configurar en ambos ficheros `.env` mencionados anteriormente (`GOOGLE_MAPS_API_KEY` y `REACT_APP_GOOGLE_MAPS_API_KEY`)

YouTube Data API (Opcional)

1. Habilitar YouTube Data API v3 en Google Cloud Console
2. Crear credenciales OAuth 2.0
3. Configurar en ambos ficheros `.env` mencionados anteriormente (`REACT_APP_YOUTUBE_API_KEY`, `REACT_APP_YOUTUBE_CLIENT_ID` y `YOUTUBE_API_KEY`). La `YOUTUBE_API_KEY` suele ser la misma clave que la anterior `GOOGLE_MAPS_API_KEY`.

B.4. Estructura del Proyecto para Desarrollo

```
tfg/
    src/                        # Backend Node.js
        main.js                 # Punto de entrada
        package.json           # Dependencias backend
        db-client/             # Clientes MySQL y MongoDB
        utils/                 # Utilidades y consultas
        workflows/             # Controladores API
        controllers/           # Controlador para guardado de workflows
        services/              # Clase con funcionalidad para guardado de workflows
    ui/                         # Frontend React
        src/
```

```
        App.js              # Componente principal
        components/         # Widgets y componentes
        utils/              # Utilidades frontend
        config/             # Fichero con registro de endpoints de la API
        services/           # Motor de plantillas y servicio para guardado de workflo
    package.json            # Dependencias frontend
    public/                 # Archivos estáticos
db-init/                     # Scripts de inicialización DB MySQL
docker-compose.yml           # Configuración Docker
docs/                        # Documentación del proyecto
```

B.5. Flujo de Trabajo de Desarrollo

Desarrollo de Backend

1. **Endpoints API**: Agregar nuevos endpoints en `src/workflows/WorkflowsController.js`
2. **Consultas de BD**: Definir consultas en `ui/src/utils/queries.js`
3. **Testing**: Probar endpoints con Postman o herramientas similares

Desarrollo de Frontend

1. **Widgets**: Crear nuevos widgets en `ui/src/components/`
2. **Drag and Drop**: Integrar con React DnD para funcionalidad de arrastre
3. **Styling**: Utilizar CSS modules o styled-components para estilos

B.6. Herramientas de Desarrollo

Scripts NPM Disponibles Backend (`src/`):

```
npm run start      # Producción
npm run devel      # Desarrollo con auto-recarga
```

Frontend (`ui/`):

```
npm start          # Servidor de desarrollo
npm run build      # Build de producción
npm test           # Ejecutar tests
npm run eject      # Eject de Create React App
```

Debugging y Desarrollo

- **Backend**: Debugging con Node.js inspector
- **Frontend**: React DevTools, console.log, Chrome DevTools
- **Base de Datos**: MySQL Workbench y MongoDB Compass para inspección de datos
- **API**: Postman para testing de endpoints REST

B.7. Solución de Problemas de Desarrollo

Problemas Comunes de Configuración

- **Puerto ocupado**: Cambiar puertos en package.json o matar procesos existentes
- **Dependencias**: Eliminar `node_modules` y reinstalar con `npm install`
- **Base de datos**: Verificar que Docker está ejecutándose y el contenedor MySQL o MongoDB está activo

Este entorno de desarrollo proporciona un flujo de trabajo completo para el desarrollo, testing y debugging del Generador de Informes de Observación de Meteoros de la SMA, con recarga automática y herramientas modernas de desarrollo web.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA