Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Cuadro de mandos generalista para visualización de estadísticas de uso de una API Rest

Autor: Yosvany Sanabria, c200073

Tutor(a): Sergio Paraíso Medina

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Cuadro de mandos generalista para visualización de estadísticas de uso de una API Rest
Junio 2024

Autor: Yosvany Patrocinio Sanabria Viltres

Tutor: Sergio Paraíso Medina
Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este proyecto se enfoca en crear un sistema completo de análisis y visualización de datos, específicamente orientado a los datos de uso de aplicaciones. Con el creciente volumen de datos generado en la era digital, es fundamental contar con herramientas que permitan gestionar, analizar y visualizar estos datos de manera eficiente y efectiva. Se ofrece una solución novedosa que facilita el análisis y la visualización de grandes cantidades de datos. Abarca los desafíos asociados con la gestión y comprensión de conjuntos de datos extensos y complejos en entornos dinámicos.

La aplicación web que se ha creado utiliza tecnologías avanzadas como Angular para el frontend y Docker para el despliegue. Esta plataforma ofrece una interfaz de usuario simple y fácil de entender, diseñada para satisfacer las necesidades únicas de cada usuario y con una amplia gama de opciones de personalización. Los usuarios pueden cargar datos de una variedad de fuentes, como APIs REST y archivos CSV locales. Los datos pueden ser filtrados, seleccionados y procesados según las necesidades del usuario una vez cargados, lo que permite visualizarlos en formatos de tablas y gráficos interactivos.

La capacidad de la aplicación para descubrir patrones y tendencias ocultas en grandes volúmenes de datos es una característica clave que ayuda a las organizaciones y empresas a tomar decisiones informadas. Estas decisiones pueden mejorar significativamente sus productos y servicios, basándose en análisis detallados y visualizaciones claras de los datos. Además, el sistema está diseñado para ser fácilmente configurable e instalable, lo que facilita su integración en diversos entornos mediante el uso de Docker. Esto asegura que el despliegue y la gestión de la aplicación sean procesos simplificados y eficientes.

En resumen, este proyecto no solo proporciona una herramienta potente para el análisis y visualización de datos, sino que también establece una base sólida para futuras expansiones y adaptaciones, permitiendo su uso en una amplia gama de aplicaciones y sectores.

Abstract

This project focuses on creating a comprehensive data analysis and visualization system, specifically targeting application usage data. With the growing volume of data generated in the digital age, it is essential to have tools that efficiently and effectively manage, analyze, and visualize this data. The proposed solution facilitates the analysis and visualization of large datasets, addressing the challenges associated with managing and understanding extensive and complex datasets in dynamic environments.

The developed web application utilizes advanced technologies such as Angular for the frontend and Docker for deployment. This platform offers a simple and easy-to-understand user interface, designed to meet the unique needs of each user, with a wide range of customization options. Users can load data from various sources, such as REST APIs and local CSV files. Once loaded, the data can be filtered, selected, and processed according to the user's needs, allowing for visualization in table formats and interactive graphs.

The application's ability to uncover hidden patterns and trends in large volumes of data is a key feature that helps organizations and companies make informed decisions. These decisions can significantly improve their products and services, based on detailed analysis and clear data visualizations. Additionally, the system is designed to be easily configurable and installable, facilitating its integration into various environments through the use of Docker. This ensures that the deployment and management of the application are streamlined and efficient processes.

In summary, this project not only provides a powerful tool for data analysis and visualization but also establishes a solid foundation for future expansions and adaptations, allowing its use across a wide range of applications and industries.

Tabla de Contenidos

Tabla de Ilustraciones

1 Introducción

En la actualidad, nos encontramos en medio de una gran cantidad de datos generados por una variedad de fuentes, desde dispositivos móviles hasta sistemas comerciales complejos. Esta gran cantidad de información ofrece tanto oportunidades como obstáculos importantes. La abundancia de datos nos permite tomar decisiones informadas en una variedad de áreas, desde la investigación científica hasta la gestión empresarial, y extraer conocimientos valiosos. Sin embargo, a medida que aumenta la cantidad de datos, se necesitan herramientas y sistemas que puedan organizar, analizar y extraer valor de esta gran cantidad de datos.

En este contexto, este trabajo analiza la importancia de desarrollar una herramienta que permita la interpretación y visualización de datos de manera fácil y eficiente. La capacidad de poder comprender y tomar decisiones basadas en datos es esencial para el éxito de cualquier empresa u organización. Una ventaja significativa tanto en el ámbito académico como empresarial puede deberse a la capacidad de poder extraer información útil de grandes conjuntos de datos.

Contar con una herramienta que simplifique el proceso de análisis y visualización puede marcar la diferencia entre el éxito y el estancamiento, dada la gran cantidad de información disponible en la actualidad. Además, en un mundo donde el tiempo es valioso, es crucial tener la capacidad de acceder rápidamente a la información relevante y comprender su significado.

Este trabajo propone la creación de una solución novedosa para abordar los problemas de gestión y visualización de datos en entornos dinámicos y complejos. El objetivo principal de esta herramienta es brindar a los usuarios las herramientas necesarias para navegar a través del vasto mar de datos y descubrir conocimientos valiosos que impulsen el progreso y la innovación. Se busca brindar una experiencia de usuario mejorada y altamente personalizable utilizando un enfoque centrado en el usuario y aprovechando las últimas tecnologías disponibles.

Además, se espera que esta herramienta pueda adaptarse fácilmente a diversas fuentes de datos y brinde una mayor personalización y adaptabilidad para cada usuario. La implementación exitosa de esta solución no solo mejorará la eficiencia y la eficacia del manejo de datos, sino que también permitirá a los usuarios aprovechar al máximo la información disponible, lo que les permitirá tomar decisiones estratégicas y fundamentadas.

1.1 Lista de tareas y objetivos del proyecto

En este se punto se listarán las tareas y los objetivos del proyecto.

1.1.1 Lista de Tareas

Análisis y estado del arte de tecnologías (30 horas):

- Investigación sobre tecnologías de visualización de datos.
- Revisión de herramientas y bibliotecas relevantes en Angular para la creación de cuadros de mando.
- Estudio de tecnologías Docker para la distribución y despliegue de la aplicación.

Estudio de necesidades y requisitos (90 horas):

- Definir los requisitos funcionales y no funcionales del cuadro de mando.
- Documentar detalladamente los casos de uso y escenarios de usuario.

Desarrollo de trabajo (83 horas):

- Crear el diseño arquitectónico de la aplicación.
- Implementar el cuadro de mandos utilizando Angular para la interfaz de usuario.
- Integrar gráficos y visualizaciones para representar estadísticas de uso.

Pruebas y resultados (54 horas):

- Desarrollar casos de prueba para asegurar la funcionalidad y robustez.
- Ejecutar pruebas unitarias y de integración.
- Documentar detalladamente los resultados obtenidos durante las pruebas.

Memoria (40 horas):

- Elaborar la introducción, objetivos y justificación del Trabajo Fin de Grado.
- Describir detalladamente la metodología utilizada en cada fase del desarrollo.
- Presentar y analizar los resultados obtenidos.
- Concluir y proponer posibles mejoras.
- Incluir referencias bibliográficas y citas utilizadas en el trabajo.

1.1.2 Lista de Objetivos

Crear una Plataforma de Visualización de Datos Estadísticos

- Crear una plataforma que permita a los usuarios ver datos estadísticos de uso de manera fácil de entender utilizando tecnologías de visualización de datos avanzadas basadas en Angular.

Proveer de Estadísticas Configurables

- Implementar funcionalidades que permitan a los usuarios configurar las estadísticas que desean visualizar, incluyendo la posibilidad de filtrar datos.

Facilitar la Configuración e Instalación de la Aplicación

- Diseñar la aplicación para que sea fácilmente configurable e instalable, utilizando Docker para simplificar el proceso de despliegue y asegurar una fácil integración con otras aplicaciones.

1.2 Diagrama de Gant

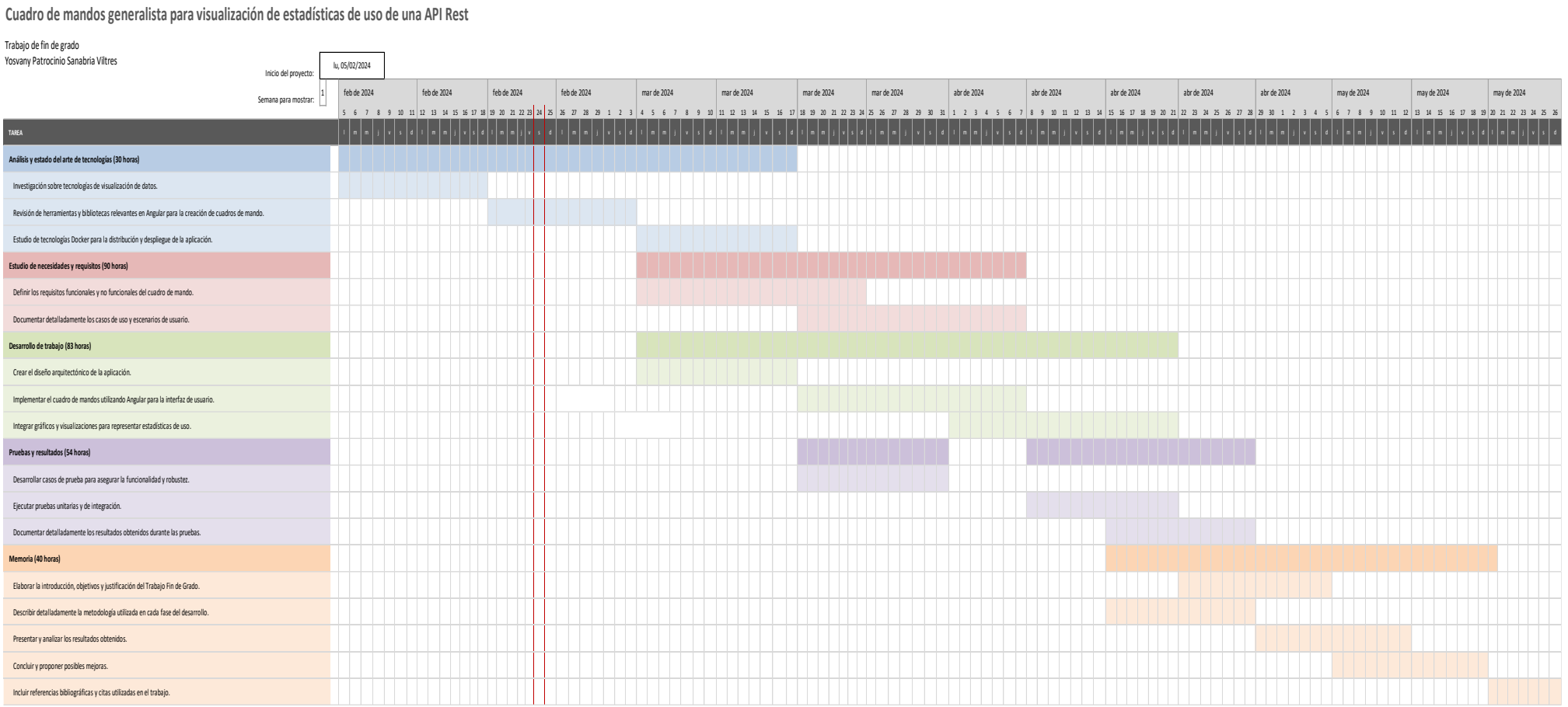

Imagen 1: Diagrama de Gantt de la planificación del proyecto

2 Estado del arte

2.1 Motivación y estado actual de las tecnologías

La visualización de estadísticas presenta numerosos desafíos en el entorno empresarial actual. Las empresas y desarrolladores que ofrecen APIs Rest a menudo tienen problemas para comprender cómo se utilizan sus servicios. Para recopilar y analizar datos, con frecuencia se ven obligados a recurrir a métodos manuales aburridos, lo que resulta en un proceso propenso a errores y que consume mucho tiempo y recursos.

La falta de herramientas automatizadas y eficientes para analizar el uso de las API Rest limita la capacidad de las empresas para optimizar sus servicios y responder de manera rápida a las necesidades de sus clientes. Además, la falta de una visualización clara y comprensible de las estadísticas de uso dificulta la toma de decisiones sobre la mejora y el mantenimiento de las APIs.

Debido a esta situación, es necesario crear un cuadro de mando que pueda visualizar las estadísticas de uso de las API Rest. El objetivo principal de este proyecto es proporcionar una solución automatizada completa que permita a los desarrolladores y empresas analizar con mayor facilidad el funcionamiento de sus APIs, detectar tendencias y tomar decisiones basadas en datos.

El objetivo principal de esta solución es permitir un análisis detallado del uso de una API Rest específica, así como un análisis visual y estadístico de cualquier conjunto de datos de uso. Esto ayudará a las empresas a comprender cómo se está utilizando su API y encontrar áreas de mejora y patrones de comportamiento en sus servicios.

Este cuadro de mandos proporcionará una interfaz fácil de usar e intuitiva que facilitará la exploración y comprensión de las métricas de uso de la API. Esto resultará en una mejor eficiencia operativa y una toma de decisiones estratégicas más inteligente y oportuna.

En resumen, hay una oportunidad para mejorar la gestión de servicios de API en el entorno empresarial actual mediante el desarrollo de un cuadro de mandos generalista para la visualización de estadísticas de uso de APIs Rest.

2.2 Análisis de tecnologías y herramientas relevantes

Se utilizaron varias herramientas para el FrontEnd y el despliegue del sistema durante la fase de desarrollo de la aplicación.

Se utilizó una variedad de tecnologías y marcos en el front-end para crear una interfaz de usuario dinámica y atractiva. Entre las tecnologías usadas están HTML, CSS, JavaScript y typeScript. Además, se decidió utilizar un marco moderno como Angular, que ofreció una base sólida para el desarrollo de componentes reutilizables.

Por otro lado, se utilizaron tecnologías de contenedores como Docker para acelerar el despliegue del sistema. La herramienta permitió encapsular la aplicación y sus dependencias en contenedores independientes, lo que facilitó su distribución y ejecución en varios entornos.

En resumen, la combinación de estas herramientas y tecnologías en el desarrollo y despliegue de la aplicación permitió la creación de una solución robusta, flexible y fácilmente escalable, adaptada a las necesidades del proyecto y enfocada en brindar la mejor experiencia para los usuarios finales.

2.2.1 HTML

Este lenguaje ha evolucionado a lo largo de los años desde sus primeros pasos bajo el nombre de "Etiquetas HTML", alcanzando su forma más reciente con HTML 5, que se introdujo en 2017. Este lenguaje, que se basa en etiquetas, es responsable de establecer la disposición y el estilo del contenido web y sirve como enlace entre la intención del desarrollador y la experiencia del usuario final. Las etiquetas HTML, influenciadas por el estándar SGML (Standard Generalized Markup Language), proveen una base sólida para la creación de documentos web, garantizando una estructura coherente y accesible.

Para mostrar elementos de contenido como texto, imágenes y otros recursos en un navegador web, HTML utiliza "marcas". Estas marcas, también conocidas como etiquetas HTML, engloban una variedad de elementos especiales que van desde los básicos como <p> (párrafo) y <img> (imagen), hasta los más complejos como <article>, <section>, <header>, <footer> y muchos otros.

Cada elemento HTML tiene etiquetas que rodean su nombre, que comienzan con "<" y terminan con ">", para diferenciarlo del resto del texto del documento. Es posible escribir estas etiquetas en mayúsculas o minúsculas, sin que esto afecte su funcionamiento. Por ejemplo, la etiqueta " title" puede escribirse como "Title" o "TITLE".

Además, con Html se pueden usar otros recursos adicionales, como las hojas de estilo CSS (Cascading Style Sheets) y los scripts JavaScript, los cuales mejoran la interactividad y el diseño de las páginas web. Estos componentes funcionan juntos para crear experiencias web dinámicas y atractivas que satisfacen las necesidades y expectativas del usuario contemporáneo.

La capacidad de HTML para convertir el contenido bruto en experiencias visuales significativas es lo que lo hace valioso. Las siguientes imágenes ilustran cómo el código HTML se convierte en una página web, lo que demuestra su importancia en el mundo digital actual.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de HTML</title>
</head>
<body>
    <header>
        <h1>Encabezado de la Página</h1>
    </header>

    <nav>
        <ul>
            <li><a href="#">Inicio</a></li>
            <li><a href="#">Acerca de</a></li>
            <li><a href="#">Contacto</a></li>
        </ul>
    </nav>

    <article>
        <h2>Título del Artículo</h2>
        <p>Este es un párrafo de ejemplo dentro de un artículo.</p>
        <img src="ejemplo-imagen.jpg" alt="Ejemplo de Imagen">
    </article>

    <aside>
        <h3>Barra lateral</h3>
        <p>Este es un contenido de ejemplo dentro de una barra lateral.</p>
    </aside>

    <footer>
        <p>Pie de página &copy; 2024</p>
    </footer>
</body>
</html>
```

Imagen 2: Código HTML

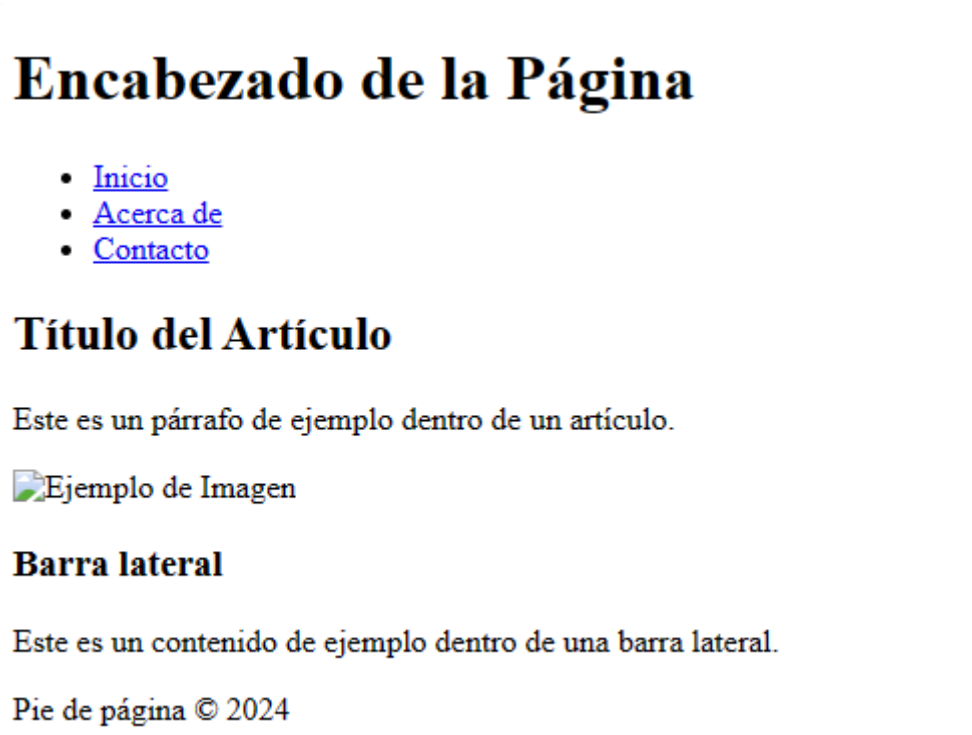

Imagen 3: Página Web resultado de código

2.2.2 CSS

Cascading Style Sheets (CSS) son un componente esencial del desarrollo web actual porque permiten la definición de estilos para la presentación de páginas HTML. Su función principal es distinguir el diseño visual de un sitio web de la estructura y el contenido. Esta separación es crucial para mejorar la accesibilidad y la flexibilidad del contenido, al tiempo que otorga un mayor control sobre la apariencia de la página.

La flexibilidad de CSS radica en su capacidad para aplicar estilos de manera global o selectiva a elementos específicos de una página web. Esto se logra creando reglas de estilo que especifican cómo se deben ver algunos elementos HTML. Por ejemplo, se pueden configurar colores, fuentes, márgenes, tamaños y posiciones para varios elementos, lo que permite personalizar la apariencia del sitio según las necesidades del diseño.

Esto significa que, dependiendo de su importancia y la especificidad de los selectores utilizados, una característica especificada en una regla de estilo puede ser sobrescrita por otra regla de estilo posterior. Cada regla de estilo de CSS consta de uno o más selectores y un bloque de declaración que contiene las propiedades y valores que se aplicarán a los elementos elegidos. Los selectores pueden dirigirse a elementos, clases, identificadores o relaciones entre elementos HTML específicos. Esto brinda una gran flexibilidad en la definición de estilos para varios componentes de una página web.

La comparación entre una página HTML básica y una versión mejorada con estilos CSS muestra cómo la aplicación de CSS a una página HTML puede cambiar significativamente su apariencia final. Para crear experiencias de usuario atractivas y funcionales, es esencial tener esta capacidad para personalizar la presentación del contenido web.

A continuación, veremos un ejemplo de cómo varia la apariencia de una página web con la aplicación de estilos CSS. Para ello usaremos como ejemplo el código de la *Imagen 2*.

```css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 20px 0;
}

nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

nav li {
    display: inline;
    margin-right: 20px;
}

nav a {
    text-decoration: none;
    color: #333;
}

article {
    background-color: #fff;
    padding: 20px;
    margin: 20px;
}

aside {
    background-color: #f4f4f4;
    padding: 20px;
    margin: 20px;
    border-radius: 5px;
}

footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 20px 0;
}
```

Imagen 4: CSS aplicado al código de la Imagen 2

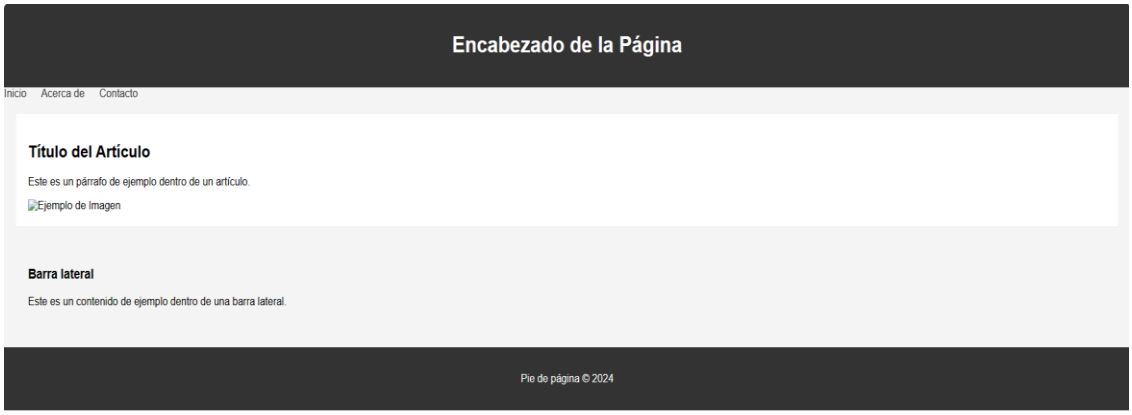

Imagen 5: Resultado de la aplicación del CSS al código de la Imagen 2.

2.2.3 JavaScript

JavaScript es un lenguaje de programación popular en el desarrollo web para mejorar la interactividad y la funcionalidad de las páginas. Desde su creación en la década de 1990, JavaScript ha evolucionado significativamente y se ha convertido en un pilar fundamental en la construcción de aplicaciones web modernas. A diferencia de HTML y CSS, que se centran en las estructuras y el estilo de las páginas web respectivamente, JavaScript se encarga de la lógica y el comportamiento dinámico del contenido.

La introducción de JavaScript permitió a los desarrolladores web crear sitios web más dinámicos y responsivos, lo que llevó al surgimiento de una variedad

de aplicaciones web interactivas, desde páginas de inicio simples hasta aplicaciones de una sola página complejas (SPA). Gracias a la popularización de entornos de ejecución como Node.js, JavaScript se ha vuelto crucial en el backend, además de su uso en el desarrollo frontend.

En la actualidad, JavaScript cuenta con una amplia variedad de frameworks y bibliotecas que facilitan el desarrollo de aplicaciones web complejas y eficientes. Frameworks como React.js, Angular y Vue.js han ganado una gran popularidad en la comunidad de desarrollo web, permitiendo a los desarrolladores crear interfaces de usuario interactivas y escalables de manera más eficiente.

A continuación, veremos un ejemplo de cómo usando JavaScript se pueden añadir funcionalidades a las páginas web y al mismo tiempo mejorar su apariencia visual, para ello usaremos como base el código de la *Imagen 2* añadiendo código JavaScript.

```
<script>
    const parrafo = document.getElementById('parrafo');
    const boton = document.getElementById('cambiarColor');

    boton.addEventListener('click', function() {
        if (parrafo.classList.contains('rojo')) {
            parrafo.classList.remove('rojo');
            parrafo.classList.add('azul');
        } else if (parrafo.classList.contains('azul')) {
            parrafo.classList.remove('azul');
            parrafo.classList.add('verde');
        } else {
            parrafo.classList.remove('verde');
            parrafo.classList.add('rojo');
        }
    });
</script>
```

Imagen 6: Código Java Script añadido al código de la Imagen 2.

Imagen 7: Resultado de añadir Código Java Script al código de la Imagen 2.

Imagen 8: Funcionalidad del botón añadido en la Imagen 8.

2.2.4 TypeScript

Debido a su capacidad para brindar una experiencia de codificación más robusta y segura, el lenguaje TypeScript, creado por Microsoft, se ha convertido en una opción popular para los desarrolladores web. TypeScript facilita el mantenimiento de grandes bases de código y reduce la probabilidad de errores durante el desarrollo al permitir la declaración de tipos estáticos.

La popularidad de este idioma se debe a una serie de características distintivas que ofrece, entre las que se encuentran:

- ***Tipado fijo:*** Los desarrolladores pueden especificar tipos de datos para variables, parámetros de funciones y otros elementos del código gracias al sistema de tipos estático que proporciona TypeScript. Antes de que se ejecuten los scripts en un navegador, esto ayuda a detectar errores comunes durante la fase de desarrollo.
- ***Aumento de la productividad****:* El tipado estático de TypeScript ofrece sugerencias de código más útiles y un autocompletado más preciso en los editores de texto.
- ***Refactorización Confiable:*** Al detectar y advertir sobre posibles problemas de tipo durante el proceso de desarrollo, TypeScript facilita la refactorización segura del código. TypeScript ayuda a mantener la integridad del sistema, por lo que se pueden realizar cambios en el código con mayor confianza.
- **Compatible con ECMAScript 6 y otros estándares**: Las características de ECMAScript 6, incluidas las clases, los módulos y las funciones de flecha, son compatibles con TypeScript. Esto permite a los desarrolladores utilizar las características más recientes del lenguaje JavaScript mientras mantienen la compatibilidad con navegadores de versión anterior.

2.2.5 Angular

En los últimos años, Angular, un marco de desarrollo web de código abierto creado por Google, se ha vuelto muy popular. Su combinación de herramientas integradas, escalabilidad y versatilidad lo convierten en una opción destacada para la creación de aplicaciones web modernas y dinámicas. A continuación, se enumeran algunos puntos destacados que demuestran la relevancia de Angular en la actualidad del desarrollo de aplicaciones web:

- **Popularidad y adopción:** Angular tiene una comunidad activa y es ampliamente utilizado por desarrolladores de todo el mundo. El gran número de aplicaciones web conocidas que se han construido con este marco demuestra su popularidad.

- **Arquitectura Modelo-Vista-Controlador (MVC):** Angular utiliza el patrón de diseño Modelo-Vista-Controlador (MVC), lo que facilita la

organización del código y la separación de preocupaciones. Esto mejora la escalabilidad y la mantenibilidad de las aplicaciones.

- **La arquitectura basada en componentes:** Angular se basa en una arquitectura orientada a componentes, lo que significa que las aplicaciones se construyen como una jerarquía de componentes que se pueden usar de nuevo y de nuevo. Esto facilita el desarrollo y la evolución de las aplicaciones al promover el modularidad y la reutilización del código.
- **Soporte para TypeScript:** Angular está escrito en TypeScript, una colección de JavaScript que agrega tipado estático y otras características avanzadas al lenguaje. TypeScript mejora la detección de errores y el mantenimiento del código en proyectos de gran tamaño.
- **Herramientas Integradas:** Angular ofrece una amplia gama de herramientas integradas que facilitan el desarrollo, incluido un poderoso sistema de interfaz de línea de comandos (CLI) para la generación de código, Pruebas y despliegue de aplicaciones.
- **Rendimiento y Optimización:** Angular ofrece características como El envío lento, la inclinación del árbol y la compilación AOT (Ahead of Time) que ayudan un optimizar el rendimiento de las aplicaciones y Mejorar la experiencia del usuario.

Imagen 9: Imagen logo Angular.

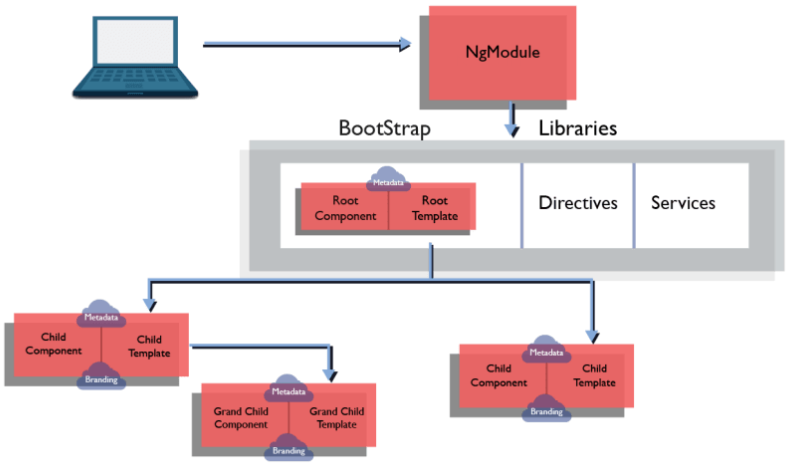

Imagen 10: Imagen Arquitectura Angular

2.2.6 Docker

Docker es una plataforma de desarrollo que simplifica el proceso de construcción, ejecución y gestión de aplicaciones sin la necesidad de configurar o mantener la infraestructura subyacente que normalmente se asocia con el desarrollo y despliegue de software. Docker permite a los usuarios crear entornos de desarrollo estandarizados y portátiles mediante la virtualización a nivel del sistema operativo en forma de contenedores.

Los contenedores de Docker encapsulan el software y todas sus dependencias en bibliotecas y archivos de configuración, lo que facilita la portabilidad y la ejecución consistente de aplicaciones en diferentes entornos. Estos contenedores están completamente aislados entre sí, lo que significa que pueden ejecutarse en el mismo host sin interferir unos con otros. Además, en comparación con las máquinas virtuales convencionales, los contenedores comparten el mismo núcleo del sistema operativo, lo que reduce el uso de recursos.

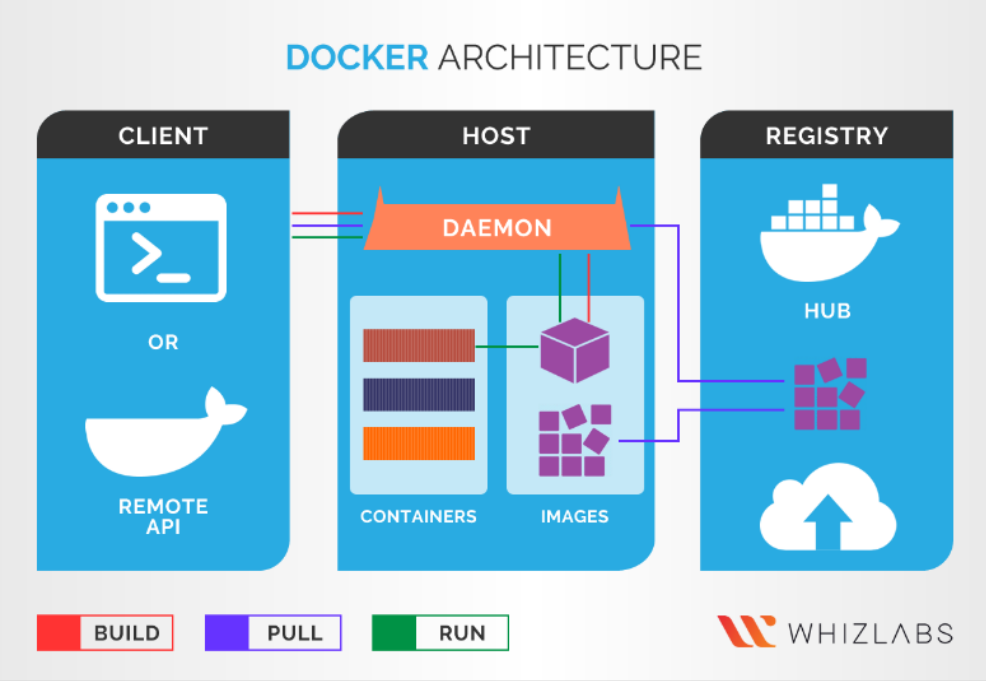

Imagen 11: Imagen Arquitectura Docker.

3 Requisitos del sistema

3.1 Identificación de requisitos

Para la realización de la aplicación se dispone de los siguientes requisitos.

R1: **Visualización de estadísticas**: La aplicación debe ser capaz de mostrar estadística de manera clara y comprensible, utilizando gráficos, tablas u otros elementos visuales.

R2: **Filtrado de datos**: Los usuarios deben poder filtrar los datos según diferentes criterios, criterios que se deben adaptar a la fuente de datos que se use.

R3: **Personalización de vistas**: Se debe permitir a los usuarios personalizar las vistas de las estadísticas según sus preferencias, como seleccionar qué métricas mostrar.

R4: **Interactividad**: La aplicación debe ser interactiva, permitiendo a los usuarios explorar los datos mediante acciones como hacer clic en elementos para obtener detalles adicionales, desplazarse por gráficos interactivos, etc.

R5: **Fuentes de datos**: La aplicación debe permitir al usuario poder cargar datos de distintas fuentes.

R6: **Interfaz de usuario intuitiva**: La aplicación debe tener una interfaz de usuario fácil de entender y utilizar, con una navegación clara y coherente.

3.2 Casos de uso

Teniendo en cuenta los requisitos anteriores tendremos en cuenta dos casos de uso uno por cada forma de cargar datos que permite la aplicación.

El Caso de Uso 1

"Carga de datos desde una API", describe la funcionalidad que permite al usuario cargar datos desde una API externa a la aplicación. Para llevar a cabo esta acción, el usuario debe tener las credenciales necesarias para acceder a la API y los datos deben estar en formato CSV.

El proceso comienza cuando el usuario inicia la creación de la imagen Docker utilizando el script proporcionado. Durante este proceso, el usuario tiene la opción de especificar las credenciales requeridas para acceder a la API desde la cual se descargarán los datos.

Una vez que se proporcionan las credenciales y se completa el proceso de creación de la imagen Docker, la aplicación intenta cargar los datos desde la API especificada. Si la carga de datos se realiza con éxito, el usuario puede seleccionar los datos específicos que desea visualizar del archivo CSV cargado y comenzar a visualizar estadística de estos datos.

Una vez cargados los datos con éxito, el usuario puede utilizar la funcionalidad de visualización de estadísticas para explorar y analizar los datos de manera personalizada, adaptándolos según sus necesidades específicas.

CASO DE USO 1	Carga de datos desde una API
DESCRIPCIÓN	El usuario puede cargar datos procedentes de una API externa.
PRE	• El usuario debe disponer de las credenciales necesarias para acceder a la API. • Los datos que se descargan de la API deben estar en formato CSV.
ACCIÓN	Al crear la imagen Docker con el script proporcionado se dispone de una sección en dicho script, para especificar las credenciales necesarias para descargar los datos de la Api.
POST	Una vez cargados los datos el usuario podrá seleccionar los datos que quiere mostrar del csv cargado, a continuación, poder visualizar estadísticas de estos datos, pudiendo adaptarla según sus necesidades

Tabla 1: Carga datos API

El Caso de Uso 2

"Carga de datos desde un CSV local", describe la funcionalidad que permite al usuario cargar datos desde un CSV local. Al iniciar la aplicación el usuario tiene la posibilidad de elegir si desea usar datos de csv que tenga de manera local en su ordenador.

Una vez seleccionada esta opción, el usuario podrá elegir un archivo entre sus archivos locales. Si la carga de datos se realiza con éxito, el usuario puede seleccionar los datos específicos que desea visualizar del archivo CSV cargado y comenzar a visualizar estadística de estos datos.

Una vez cargados los datos con éxito, el usuario puede utilizar la funcionalidad de visualización de estadísticas para explorar y analizar los datos de manera personalizada, adaptándolos según sus necesidades específicas.

CASO DE USO 1	Carga de datos desde un CSV local
DESCRIPCIÓN	• El usuario tiene la posibilidad de cargar datos procedentes de un csv almacenado de manera local.
PRE	• El usuario debe disponer de datos almacenados en formato csv almacenados de manera local.
ACCIÓN	• El usuario selecciona la opción de cargar datos desde una fuente local. • El sistema muestra un explorador de archivos para que el usuario seleccione el archivo CSV. • El usuario selecciona el archivo CSV. • El sistema carga los datos del archivo CSV.
POST	• El usuario puede seleccionar los datos a visualizar del CSV cargado. • El usuario puede visualizar estadísticas de estos datos y adaptar la visualización según sus necesidades

Tabla 2: Carga datos locales

El Caso de Uso 3

La aplicación dispone de un apartado para visualizar los datos de manera grafica. En este apartado el usuario podrá seleccionar tanto los datos que quiere visualizar como el formato d gráfico que más se adapte a sus necesidades.

CASO DE USO 3	Visualización de Datos en Gráficos
DESCRIPCIÓN	• El usuario puede visualizar los datos cargados en forma de gráficos.
PRE	• Los datos deben estar cargados en la aplicación.
ACCIÓN	• El usuario selecciona los datos que desea visualizar. • El usuario elige el tipo de gráfico (barra, línea, pastel, etc.). • El sistema genera el gráfico basado en los datos seleccionados.
POST	• El usuario puede ver los gráficos generados y modificar los parámetros de visualización.

Tabla 3: Visualización de Datos en Gráficos

El Caso de Uso 4

La aplicación dispone de un apartado para la visualización de los datos en formato tabla.

CASO DE USO 4	Visualización de Datos en Forma de Tabla
DESCRIPCIÓN	• El usuario puede visualizar los datos cargados en forma de tabla.
PRE	• Los datos deben estar cargados en la aplicación.
ACCIÓN	• El usuario selecciona la opción de visualizar datos. • El sistema muestra los datos en una tabla con todas las columnas disponibles.
POST	• El usuario puede ver todos los datos organizados en filas y columnas.

Tabla 4: Visualización de Datos en Forma de Tabla

El Caso de Uso 5

El usuario podrá genera un gran número de combinaciones de filtros, de manera que pueda encontrar los datos en la tabla de manera fácil. Contará con 3 filtros para filtrar por un elemento de la columna que desee y además dispondrá de un campo para introducir texto que también podría aplicar a la columna deseada.

CASO DE USO 5	Filtrado de Datos en la Tabla
DESCRIPCIÓN	• El usuario puede filtrar los datos en la tabla por diversos parámetros.
PRE	• Los datos deben estar cargados en la aplicación. • La tabla de datos debe estar visible.
ACCIÓN	• El usuario selecciona la opción de filtrar datos. • El usuario define los parámetros de filtrado (por ejemplo, rango de fechas, categorías, valores específicos, etc.). • El sistema aplica los filtros y actualiza la tabla para mostrar solo los datos que cumplen con los parámetros definidos.
POST	• El usuario puede ver los datos filtrados en la tabla.

Tabla 5: Filtrado de Datos en la Tabla

El Caso de Uso 6

El usuario dispone de 3 tablas en formato reducido donde podrá seleccionar una columna de los datos mostrados donde podrá visualizar estos datos de manera simple.

CASO DE USO 6	Selección de Columnas para Visualización
DESCRIPCIÓN	• El usuario puede seleccionar columnas específicas para visualizar en la tabla.
PRE	• Los datos deben estar cargados en la aplicación. • La tabla de datos debe estar visible.
ACCIÓN	• El usuario selecciona la opción de selección de columnas. • El sistema muestra una lista de todas las columnas disponibles. • El usuario marca las columnas que desea visualizar. • El sistema actualiza la tabla con los datos de esa columna.
POST	• El usuario podrá visualizar los datos de la columna seleccionada en una tabla aparte.

Tabla 6: Selección de Columnas para Visualización

4 Desarrollo

4.1 Arquitectura Aplicación

La arquitectura de la aplicación desarrollada para este proyecto sigue un enfoque modular y escalable que facilita el mantenimiento y la expansión futura de la aplicación. La aplicación está diseñada siguiendo los principios del framework Angular, que promueve la creación de aplicaciones web dinámicas y de una sola página (Single Page Application - SPA). En la siguiente imagen se muestra en forma de resumen la arquitectura de la aplicación.

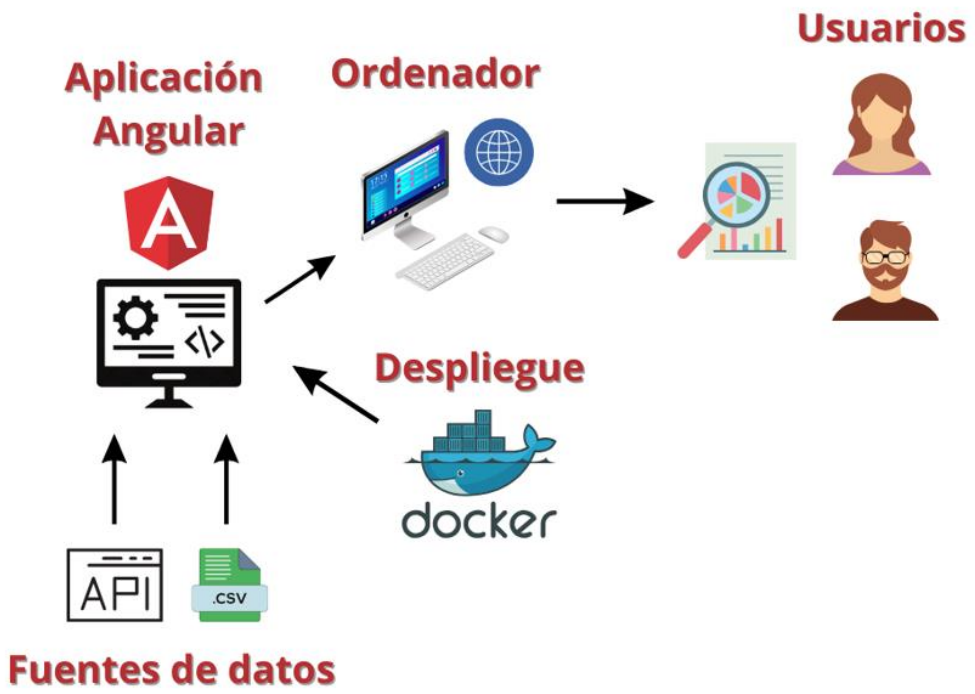

Imagen 12: Arquitectura de la aplicación

La arquitectura de la aplicación consta de los siguientes elementos principales:

Componentes Angular: La aplicación está dividida en componentes que son reutilizables y modulares. Cada componente se encarga del manejo de una parte especifica de la aplicación, aunque dichos componentes pueden depender en cierta medida unos de otros. Por ejemplo, se tienen componentes para carga de datos, para acceso a la aplicación, muestreo de estadística, filtrado de datos, ect.

- **Servicios:** Se utilizan servicios para encapsular la lógica de negocio y compartir funcionalidades entre diferentes componentes. Por ejemplo, se puede tener un servicio dedicado a la gestión de la carga de datos desde la API, otro para el procesamiento de datos y otro para la generación de estadísticas.

- **Módulos:** La aplicación se organiza en módulos de Angular, que agrupan componentes relacionados y servicios asociados. Estos módulos pueden ser funcionales, como un módulo para la carga de datos, o de características, como un módulo para la visualización de estadísticas.

- **Plantillas y Estilos:** Las plantillas HTML y los estilos CSS se utilizan para definir la apariencia y el diseño de la interfaz de usuario. Angular permite la separación de la estructura del documento (HTML) y los estilos (CSS) de la lógica de la aplicación, lo que facilita el mantenimiento y la personalización. Cada componente de Angular dispone de su propia hoja de estilos, aunque se dispone de una hoja de estilos global lo que puede acceder cualquier componente.

La arquitectura de la aplicación también incluye un módulo dedicado a la carga de datos desde una API externa proporcionada por el usuario. Este módulo se integra con los componentes existentes y los servicios de la aplicación para permitir al usuario realizar la carga de datos de manera eficiente y segura.

El módulo de carga de datos desde la API incluye:

- **Servicio de conexión API:** Se ha implementado un servicio para gestionar la comunicación con las APIs proporcionadas por los usuarios. Este servicio se encarga de enviar las solicitudes HTTP necesarias para obtener los datos desde la API y procesar las respuestas recibidas.

- **Procesamiento de datos:** Una vez que se obtienen los datos de la API, se realiza un proceso de procesamiento para estructurar y almacenar los datos de manera adecuada en la aplicación. Esto puede incluir la transformación de los datos en un formato estándar, la eliminación de datos duplicados o corruptos, entre otros.

- **Interfaz de usuario para la carga de datos:** Se proporciona una interfaz de usuario intuitiva que guía al usuario a través del proceso de carga de datos, mostrando mensajes de estado y notificaciones en caso de errores o problemas durante el proceso.

4.2 Flujo de pantallas

La aplicación cuenta 4 pantallas diferentes:

1. **Pantalla Inicio de la aplicación:** Permite al usuario especificar si desea utilizar una fuente de datos alternativa a la API especificada durante la creación de la imagen Docker.

Imagen 13: Imagen Pantalla Inicio.

2. **Pantalla selección de datos:** Permite seleccionar que datos del csv cargado se desean mostrar.

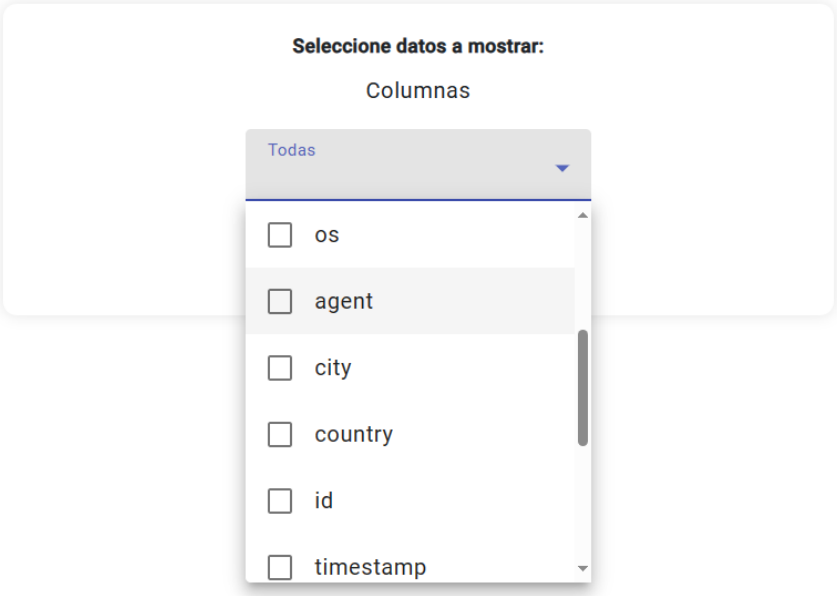

Imagen 14: Imagen Pantalla selección de datos.

3. **Pantalla de Visualización de Datos:** En esta pantalla, los usuarios podrán ver los datos cargados desde la API de manera estructurada y organizada. La información se presenta de forma clara y legible, utilizando tablas, gráficos o cualquier otro formato que facilite la comprensión de los datos. Los usuarios pueden navegar por los diferentes conjuntos de datos y aplicar filtros o búsquedas para encontrar la información específica que están buscando.

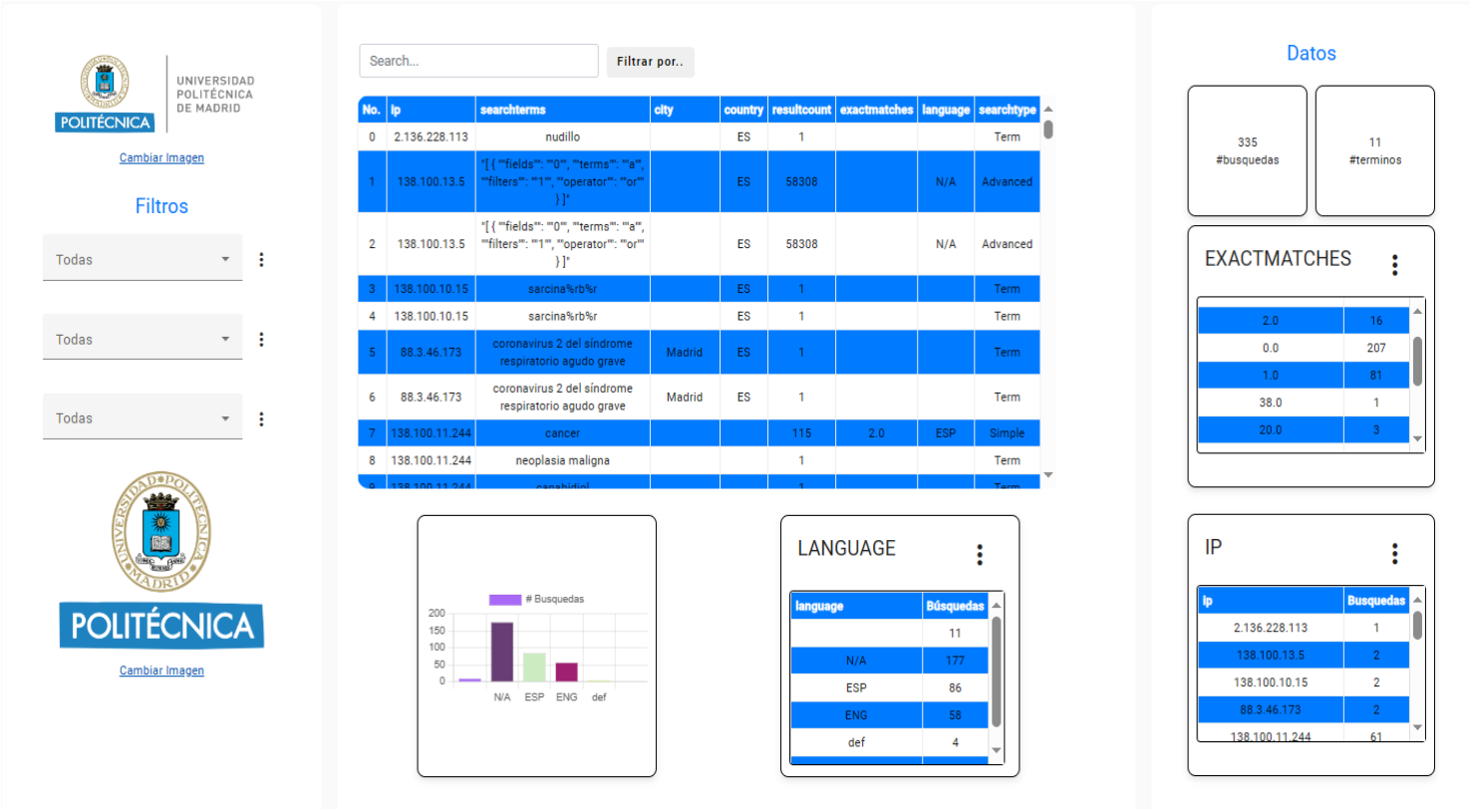

Imagen 15: Imagen Pantalla visualización de datos.

4. **Pantalla de Estadísticas Gráficas:** En esta pantalla, los usuarios pueden visualizar estadísticas gráficas generadas a partir de los datos cargados desde la API. Se utilizan diversos tipos de gráficos, como gráficos de barras, gráficos circulares, gráficos de líneas, entre otros, para representar visualmente la información de manera intuitiva y comprensible.

Imagen 16: Imagen Pantalla Estadística Gráfica.

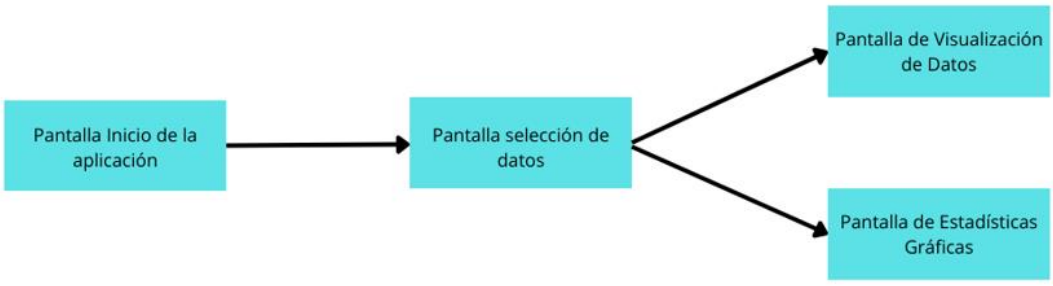

Imagen 17: Imagen Flujo Pantalla

4.3 Implementación de componentes

En el desarrollo de la aplicación, se han creado varios componentes clave que desempeñan un papel fundamental en la experiencia del usuario y en la funcionalidad general de la aplicación. Entre los componentes más importantes se encuentran:

- **Componente filtro de datos:** Este componente permite filtrar los datos de la tabla, permitiendo establecer al usuario una gran variedad de

combinaciones de filtros los cuales se adaptan a cada fuente de dato cargada.

```
<div style="margin-bottom:30px ;">
  <img src="{{ imageUrl2 }}" alt="Uploaded Image" width="300" height="125" >
  <label for="fileInput2" class="change-text">Cambiar Imagen</label>
  <input type="file" id="fileInput2" accept="image/*" (change)="handleFileInput2($event)" style="display:none;">
</div>

<h1 style="text-align: center;color: #007bff;">Filtros</h1>

<div class="Tipo-Busqueda">
  <h3 style="text-align: left; margin-left: 33px;color: #007bff;">{{nombreFiltro1}}</h3>
  <mat-form-field>
    <mat-label>Todas</mat-label>
    <mat-select [formControl]="toppingsTipoBusquedas" multiple >
      @for (topping of TipoList; track topping) {
        <mat-option [value]="topping" (click)="emitTipoBusquedaChange()">{{topping}}</mat-option>
      }
    </mat-select>
  </mat-form-field>
  <button mat-icon-button [matMenuTriggerFor]="menu1">
    <mat-icon>more_vert</mat-icon>
  </button>
  <mat-menu #menu1="matMenu">
    <ng-container *ngFor="let titulo of titulos;let i = index;">
      <button mat-menu-item *ngIf="mapaPosiciones.includes(i)" (click)="filtro1=i"(click)="emitFiltro1Changes(filtro1)">{{titulo}}</button>
  </ng-container>
  </mat-menu>
</div>

<div class="Ubicacion">
  <h3 style="text-align: left; margin-left: 33px; color: #007bff;">{{nombreFiltro2}}</h3>
  <mat-form-field>
    <mat-label>Todas</mat-label>
    <mat-select [formControl]="toppingsIdiomas" multiple>
      @for (topping of tipoFiltro2; track topping) {
        <mat-option [value]="topping" (click)="emitIdiomasChanges()">{{topping}}</mat-option>
      }
    </mat-select>
  </mat-form-field>
  <button mat-icon-button [matMenuTriggerFor]="menu2">
  <mat-icon>more_vert</mat-icon>
</button>
<mat-menu #menu2="matMenu">
  <ng-container *ngFor="let titulo of titulos;let i = index;">
    <button mat-menu-item *ngIf="mapaPosiciones.includes(i)" (click)="filtro2=i"(click)="emitFiltro2Changes(filtro2)">{{titulo}}</button>
</ng-container>
</mat-menu>
</div>
```

Imagen 18: Imagen Código componente filtros de datos.

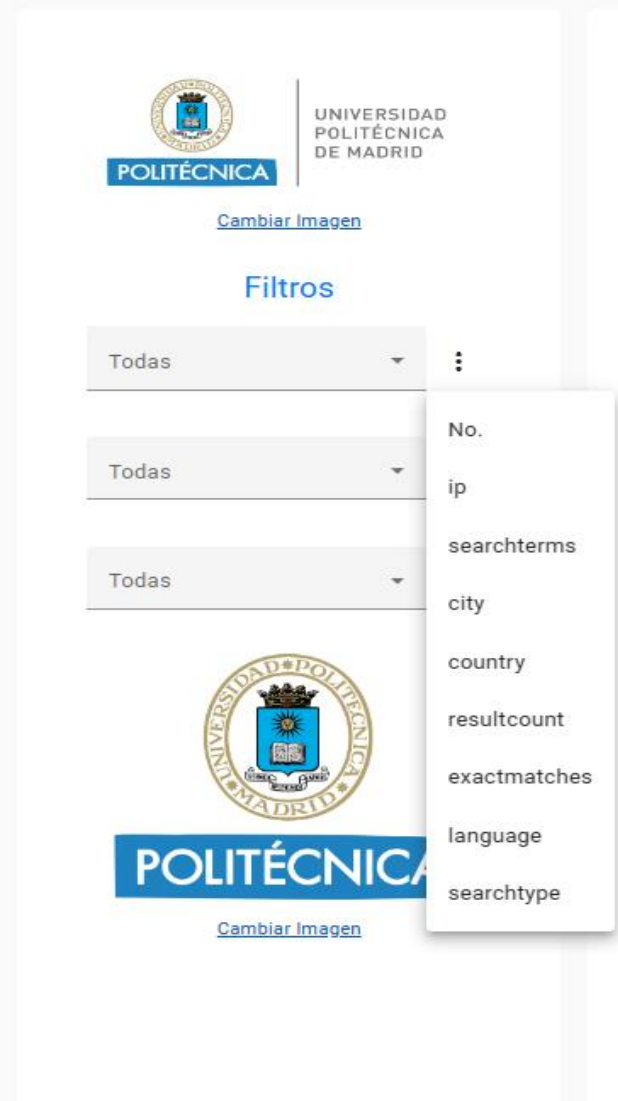

Imagen 19: Imagen visualización componente filtros de datos.

- **Componente de Visualización de Datos en formato tabla**: Este componente se encarga de mostrar los datos cargados de manera clara y comprensible para el usuario en formato tabla.

```
<div class="table-top">

<div class="search">
    <input type="text"  [formControl]="name"   class="form-control " placeholder="Search..." >
</div>
<div class="filtro">
    <button mat-button [matMenuTriggerFor]="buscadores"  class="btn-filtro">{{textFiltro}}</button>
    <mat-menu #buscadores="matMenu">
        <ng-container *ngFor="let pos of mapaPosiciones;let o=index;">
            <a mat-menu-item (click)="filtro=pos"  (click)="textFiltro=titulos[pos]">{{titulos[pos]}}</a>
        </ng-container>
    </mat-menu>
</div>
</div>

<table class="table-date">
    <thead>
        <tr>
            <ng-container *ngFor="let csvData of titulos;let i = index;">
                <th *ngIf="mapaPosiciones?.includes(i)" class="titulos"  >{{csvData}}</th>
            </ng-container>
        </tr>
    </thead>
    <tbody>
        <ng-container *ngFor="let csvData of datos | filterIdioma:[filtro1,tipoBusquedas] |
        pipeTipoBusqueda:[filtro2,Idiomas]|filtro3:[filtro3,tiposFiltro3] |filtroTabla:[filtro,name.value];let i = index;">
        <tr>
            <ng-container *ngFor="let data of csvData;let o=index;">
                <td *ngIf="mapaPosiciones?.includes(o)">
                    <span>{{data}}</span>
                </td>
            </ng-container>

        </tr>
        </ng-container>
    </tbody>


</table>
```

Imagen 20: Código componente Tabla de datos.

Search... Filtrar por..

No.	ip	searchterms	city	country	resultcount	exactmatches	language	searchtype
0	2.136.228.113	nudillo		ES	1			Term
1	138.100.13.5	"[{ ""fields"": ""0"", ""terms"": ""a"", ""filters"": ""1"", ""operator"": ""or"" }]"		ES	58308		N/A	Advanced
2	138.100.13.5	"[{ ""fields"": ""0"", ""terms"": ""a"", ""filters"": ""1"", ""operator"": ""or"" }]"		ES	58308		N/A	Advanced
3	138.100.10.15	sarcina%rb%r		ES	1			Term
4	138.100.10.15	sarcina%rb%r		ES	1			Term
5	88.3.46.173	coronavirus 2 del síndrome respiratorio agudo grave	Madrid	ES	1			Term
6	88.3.46.173	coronavirus 2 del síndrome respiratorio agudo grave	Madrid	ES	1			Term
7	138.100.11.244	cancer			115	2.0	ESP	Simple
8	138.100.11.244	neoplasia maligna			1			Term
9	138.100.11.244	canabidiol			1			Term

Imagen 21: Imagen visualización componente Tabla de datos.

- **Componente datos tabla**: Este componente permite seleccionar alguna de las columnas de datos que se muestra para obtener información personalizada de dicha columna demás de proporcionar datos generales de tabla de datos.

```
<div class="dates">
    <h1 style="color: #007bff">Datos</h1>
    <div class="contenedor">
        <div class="busquedas">{{numeroBusquedas}}<br>#busquedas </div>
        <div class="terminos">{{numeroTerminos}}<br>#terminos</div>
    </div>

    <div class="rankingBusquedas">
        <div class="contenedor-flex">
            <div>
                <h1>{{titulos[selectorTabla3] | uppercase}}</h1>
            </div>
            <div class="selector">
                <button mat-icon-button [matMenuTriggerFor]="menu2">
                    <mat-icon>more_vert</mat-icon>
                </button>
                <mat-menu #menu2="matMenu">
                    <ng-container *ngFor="let titulo of titulos; let i = index;">
                        <button mat-menu-item *ngIf="mapaPosiciones.includes(i)"
                            (click)="selectorTabla3=i; cargaDatosTabla3(selectorTabla3)">
                            {{ titulo }}
                        </button>
                    </ng-container>
                </mat-menu>
            </div>

        </div>
        <div class="tabla_busquedas">
            <table>
                <thead>
                    <tr>
                        <th>{{titulos[selectorTabla3]}}</th>
                        <th>Busquedas</th>
                    </tr>
                </thead>
                <tbody>
                    <tr *ngFor="let dato of terminosUnicos; let i = index">
                        <td>{{dato}}</td>
                        <td>{{cantidadTerminos[i]}}</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
    <br>
```

Imagen 22: Imagen código componente datos tabla.

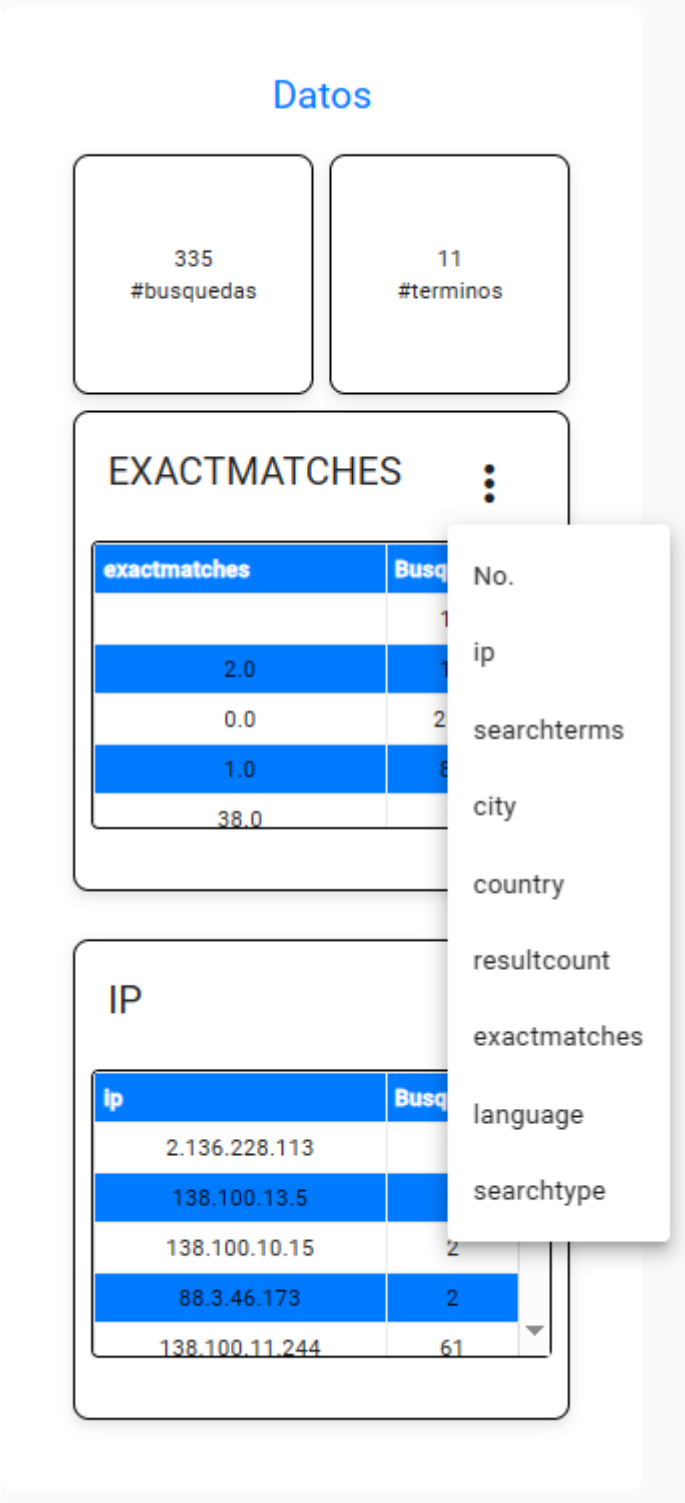

Imagen 23: Imagen visualización componente datos tabla.

- **Componente de Estadísticas Gráficas:** Este componente proporciona una visualización gráfica de las estadísticas y métricas relevantes derivadas de los datos cargados. Utiliza diferentes tipos de gráficos, como barras, líneas o pastel, para mostrar tendencias, patrones y comparaciones de manera efectiva.

```
<div class="grid-container" style="text-align: center;">
  <h1 class="mat-h1">Dashboard</h1>
  <div class="card-container">
    <div class="card">
      <div class="card-header">
        <div style="font-size: 25px;">
          {{filtro}}
        </div>
        <div class="filtro" style="position: absolute; right: 0;margin-right: 80px;">
          <button mat-button [matMenuTriggerFor]="buscadores1"  class="btn-filtro">{{textFiltro}}</button>
          <mat-menu #buscadores1="matMenu">
              <ng-container *ngFor="let pos of tipoGraficas;let o=index;">
                  <a mat-menu-item (click)="filtro=pos"  (click)="tipoGrafico=selectGrafico(pos)">{{pos}}</a>
              </ng-container>
          </mat-menu>
      </div>
        <div class="selector">
          <button mat-icon-button [matMenuTriggerFor]="menu4">
            <mat-icon>more_vert</mat-icon>
          </button>
          <mat-menu #menu4="matMenu">
            <ng-container *ngFor="let titulo of titulos; let i = index;">
              <button mat-menu-item *ngIf="mapaPosiciones.includes(i)"
                (click)="selectorTabla1=i; cargaDatosTabla1(selectorTabla1)">
                {{ titulo }}
              </button>
            </ng-container>
          </mat-menu>
        </div>
      </div>
      <div class="card-body" style="text-align: center;">
        <app-graficos [busquedaPaises]="busquedaPaises" [paises]="paises" [tipoGrafico]="tipoGrafico"></app-graficos>
      </div>
    </div>
    <div class="card">
      <div class="card-header">
        <div style="font-size: 25px;">
          {{filtro2}}
        </div>
        <div class="filtro" style="position: absolute; right: 0;margin-right: 80px;">
          <button mat-button [matMenuTriggerFor]="buscadores2"  class="btn-filtro">{{textFiltro}}</button>
          <mat-menu #buscadores2="matMenu">
              <ng-container *ngFor="let pos2 of tipoGraficas;let o=index;">
                  <a mat-menu-item (click)="filtro2=pos2"  (click)="tipoGrafico2=selectGrafico(pos2)">{{pos2}}</a>
              </ng-container>
          </mat-menu>
      </div>
```

Imagen 24: Imagen código componente estadística gráfica.

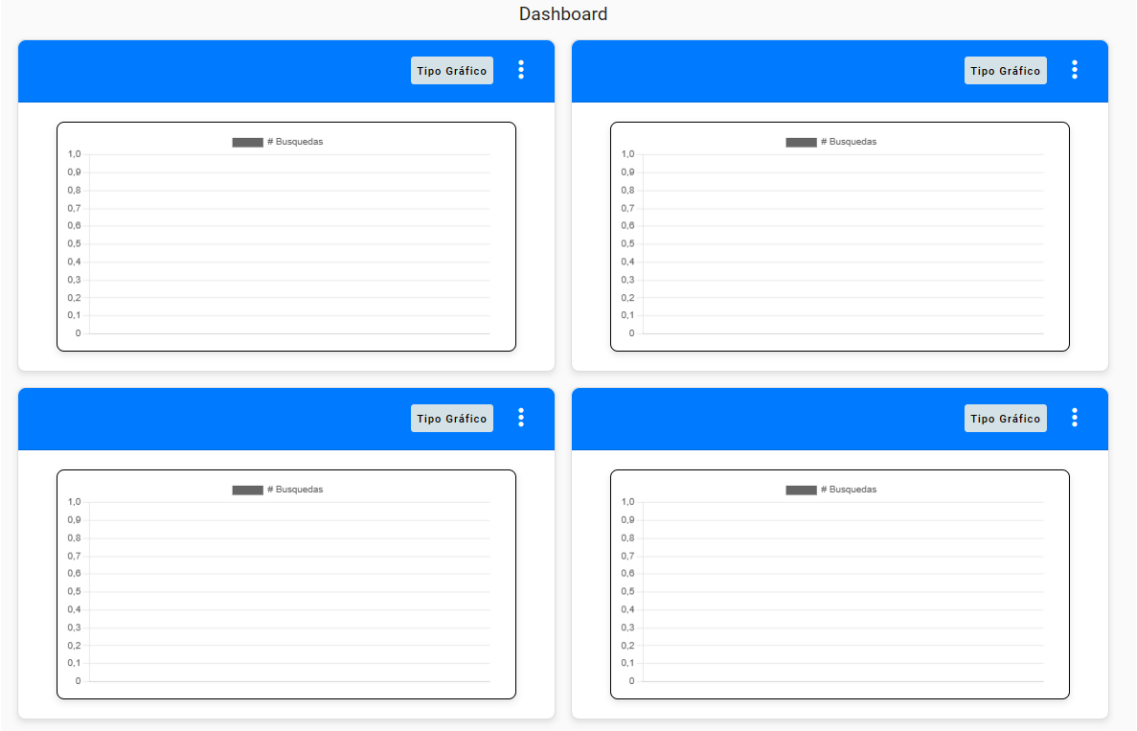

Imagen 25: Imagen visualización componente estadística gráfica.

4.4 Flujo de Información entre componentes

Este apartado describe el flujo de información entre los diferentes componentes de la aplicación, ilustrando cómo los datos se cargan, procesan y luego se muestran en tablas y gráficos. La figura muestra el proceso completo, desde la carga inicial de datos hasta la visualización.

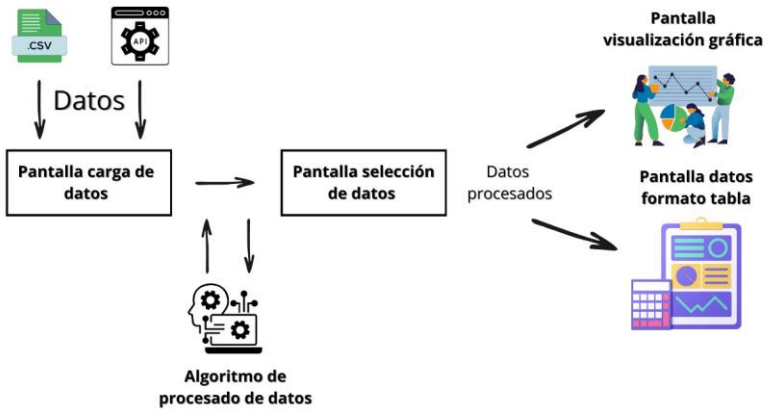

Imagen 26: Imagen Flujo de datos.

Pantalla de Carga de Datos

El flujo de información comienza en la Pantalla de Carga de Datos, donde el usuario tiene la posibilidad de cargar datos procedentes de dos fuentes principales:

- **Archivo CSV:** El usuario selecciona un archivo CSV desde su dispositivo local.
- **API Rest:** El usuario ingresa las credenciales y la URL de la API desde la cual se obtendrán los datos.

Ambas fuentes de datos proporcionan la entrada necesaria para iniciar el algoritmo en cargado de procesar los datos.

Pantalla de Selección de Datos

La Pantalla de Selección de Datos aparece después de que los datos se cargan en la aplicación. El usuario puede ver una vista previa de los datos en esta pantalla y elegir las columnas de interés para el análisis posterior.

Algoritmo de Procesado de Datos

El Algoritmo de Procesado de Datos es esencial para la limpieza y integración de datos. Este algoritmo es responsable de:

- Procesar los datos brutos obtenidos desde el CSV o la API.
- Transformar y preparar los datos para su análisis, asegurando que están en el formato adecuado.
- Almacenar los datos procesados en una estructura adecuada para su consulta y visualización.

Pantalla de Visualización en Formato Tabla

Los datos procesados son enviados a la Pantalla de Datos en Formato Tabla, donde el usuario puede:

- Visualizar los datos en forma de tabla.
- Aplicar filtros a los datos para centrarse en subconjunto específico de estos datos.

Pantalla de Visualización Gráfica

De forma Paralela, los datos procesados también son enviados a la Pantalla de visualización gráfica. En esta pantalla, el usuario puede:

- Visualizar los datos en diferentes tipos de gráficos, tales como gráficos de barras, líneas, o pastel.
- Los gráficos se actualizan dinámicamente basados la selección de datos, proporcionando una representación visual clara y comprensible de las tendencias y patrones ocultos en los datos.

4.5 Integración con fuentes de datos

La integración de la aplicación con diferentes fuentes de datos se describirá en este apartado. Para que la aplicación procese, analice y muestre datos de manera efectiva, es necesaria esta integración de datos. Este proceso no solo permite a los usuarios acceder a la información almacenada localmente, sino que también facilita la obtención de datos en tiempo real de servicios externos.

Las fuentes de datos locales y las API Rest son las dos categorías principales. La aplicación ha sido diseñada para manejar de manera efectiva tanto los desafíos como las oportunidades que presentan cada una de estas fuentes.

4.5.1-Fuente de datos locales

Los usuarios pueden cargar y administrar datos almacenados en sus propios dispositivos o redes internas gracias a la integración con fuentes de datos locales. La capacidad de cargar datos de manera local permite a los usuarios poder cargar datos de manera fácil y rápida. Además, este método es especialmente útil para las organizaciones que manejan grandes cantidades de datos sensibles.

Proceso de integración:

- **Carga de archivos:** La aplicación permite que los usuarios carguen archivos CSV desde su sistema local. A través de una interfaz de carga fácil de entender, los usuarios pueden seleccionar estos archivos. Este proceso evita problemas de compatibilidad y garantiza una rápida disponibilidad al permitir que los datos sean accesibles directamente desde la máquina del usuario.
- **Preprocesamiento:** Los datos pasan por un proceso de preprocesamiento después de ser cargados, donde se validan y limpian. La verificación de la estructura del archivo y el manejo de valores nulos o inconsistentes son parte de esto. Para garantizar que los datos sean precisos y útiles.
- **Visualización:** Los usuarios pueden elegir qué datos quieren visualizar y cómo quieren presentarlos utilizando las múltiples opciones de gráficos y tablas que ofrece la aplicación. Esto les permite analizar y extraer valor de los datos. La flexibilidad de visualización permite que los usuarios adapten la presentación de los datos a sus necesidades, lo que facilita la mejor comprensión y toma de decisiones.

4.5.2-API-Rest

La integración con la API Rest permite a la aplicación acceder a datos externos en tiempo real, lo que es necesario para trabajar con datos actualizados y en constante cambio.

Proceso de integración:

- **Autenticación:** Los usuarios deben ingresar sus credenciales para acceder a los datos de una API Rest. Esto garantiza que solo los usuarios autorizados puedan acceder a los datos, lo que protege los datos de accesos no autorizados y preserva su integridad.
- **Extracción de datos:** La aplicación puede solicitar la API Rest para extraer datos una vez configurada. Los datos deben enviarse en formato csv. Estos son procesados para que se puedan usar en la aplicación. Los datos se obtienen de manera eficiente y precisa gracias a este proceso de extracción.
- **Preprocesamiento:** Los datos pasan por un proceso de preprocesamiento después de ser cargados, donde se validan y limpian. La verificación de la estructura del archivo y el manejo de valores nulos o inconsistentes son parte de esto. Para garantizar que los datos sean precisos y útiles para el análisis posterior, este paso es crucial.
- **Visualización:** Los datos de la API Rest se almacenan temporalmente y se pueden visualizar y analizar, al igual que los datos locales. Los usuarios pueden personalizar los dashboards para mostrar las métricas y estadísticas más relevantes para sus necesidades. La personalización de la visualización permite que los usuarios adapten la presentación de los datos a sus necesidades, lo que facilita la comprensión y la toma de decisiones.

4.6 Despliegue Docker

El despliegue de un proyecto de software se refiere a todas las acciones necesarias para ponerlo en funcionamiento y hacerlo accesible para su uso. En el caso de este proyecto, se ha optado por utilizar Docker para facilitar el despliegue y la gestión de los contenedores virtuales.

El despliegue de este proyecto implica la ejecución de un contenedor virtual, con una imagen especifica.

El proyecto tiene un archivo DockerFile, que especifica las dependencias necesarias y el sistema operativo requerido para ejecutar la imagen en un contenedor.

El uso de Docker simplifica el proceso de despliegue al encapsular cada componente del proyecto en contenedores independientes, lo que garantiza la portabilidad y la consistencia del entorno de ejecución en diferentes sistemas operativos y entornos de desarrollo. Esto facilita la configuración y el mantenimiento del sistema, además de mejorar la escalabilidad y la disponibilidad del proyecto.

4.6.3 Integración con otras herramientas o servicios

Para la gestión de las variables de entorno que se especifican al crear la imagen Docker, en Angular se empleó la librería **@ngx-env/builder** [12], esta liberia se debe incluir en el proyecto angular usan usando el mandato:

-ng add @ngx-env/builder

Al incluir dicha librería en el proyecto se incluirá el fichero TypeScript **env.d.ts** la carpeta /src de proyecto este fichero te permite especificar variables de entorno las cuales se tienen que especificar también en el fichero **.env** de la raíz del proyecto.

```typescript
// Define the type of the environment variables.
declare interface Env {
  readonly NODE_ENV: string;
  // Replace the following with your own environment variables.
  // Example: NGX_VERSION: string;
  [key: string]: any;
  readonly NG_APP_API:string;
  readonly NG_APP_PASSWORD:string;
}

// Choose how to access the environment variables.
// Remove the unused options.

// 1. Use import.meta.env.YOUR_ENV_VAR in your code. (conventional)
declare interface ImportMeta {
  readonly env: Env;
}

// 2. Use _NGX_ENV_.YOUR_ENV_VAR in your code. (customizable)
// You can modify the name of the variable in angular.json.
// ngxEnv: {
//   define: '_NGX_ENV_',
// }
declare const _NGX_ENV_: Env;

// 3. Use process.env.YOUR_ENV_VAR in your code. (deprecated)
declare namespace NodeJS {
  export interface ProcessEnv extends Env {}
}

```

Imagen 27: Imagen contenido del fichero env.d.ts.

```
# Set environment variables
ENV NG_APP_API=http://138.100.11.174/dptm/searchdata
ENV NG_APP_PASSWORD=R5RV0KFxJpQy5bDHbOiDS5KV
```

Imagen 28: Imagen contenido del fichero .env.

Al incluir estas variables de entorno es estos dos ficheros ya se puede acceder a ellas desde cualquier punto de la aplicación, pero lo correcto es añadirlas al entorno de la aplicación en el cual se divide en entorno de producción y desarrollo.

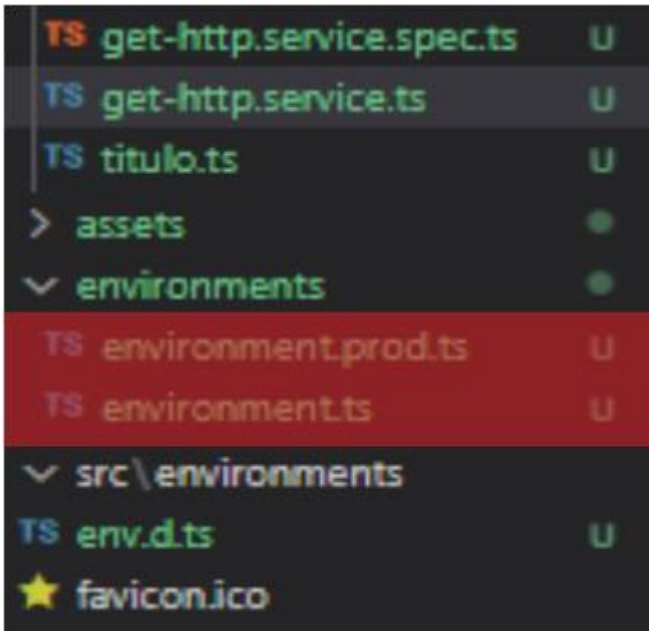

Imagen 29: Imagen fichero para las variables entorno de la App.

```
1  // environment.prod.ts
2  export const environment = {
3      production: true,
4      apiUrl: import.meta.env.NG_APP_API,
5      password: import.meta.env.NG_APP_PASSWORD
6    };
```

Imagen 30: Imagen inclusión de variables de entorno en la App.

Una vez las variables están incluidas en el entorno de la aplicación ya se pueden integrar en los distintos servicios de la aplicación. En este caso se integraron el servicio que es encarga de realizar la petición HTTP a la API que especifica el usuario para descargar los datos.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { environment } from '../environments/environment';


@Injectable({
  providedIn: 'root'
})
export class GetHTTPService {

  constructor(private http:HttpClient) { }

  apiUrl = `${environment.apiUrl}`;
  apiKey = `${environment.password}`;



  public get(url:string){

    const headers = new HttpHeaders({
      'Content-Type': 'application/json',
      'apikey': 'R5RV0KFxJpQy5bDHbOiDS5KV'
    });

    return this.http.get(this.apiUrl, { headers: headers });

  }
}

```

Imagen 31: Imagen código servicio de descarga de datos desde la API.

En la *Imagen 31 se* puede apreciar el modo de acceder a las variables de entorno de la aplicación.

5 Pruebas, conclusiones y líneas futuras.

5.1 Pruebas

Este apartado describe los métodos y procedimientos utilizados para realizar pruebas en la aplicación desarrollada. Para garantizar que la aplicación funcione correctamente y cumpla con los requisitos establecidos, las pruebas son esenciales. La carga de datos y la visualización de cada fuente de datos se presentan en dos casos de pruebas bien explicado

5.1.1 Caso de Prueba 1

Carga y Visualización de Datos Locales en Formato CSV.

Objetivo:

- Verificar que la funcionalidad de carga de datos desde un archivo CSV local, visualización en formato de tabla y gráfico se ejecute correctamente.

Precondiciones:

- El archivo CSV debe estar disponible en el dispositivo local.
- El usuario debe tener acceso a la aplicación y a la funcionalidad de carga de datos.

Pasos:

- Abrir la aplicación: El usuario inicia la aplicación web.
- Al acceder a la aplicación seleccionar la opción de carga de datos de forma local.
- Seleccionar archivo CSV: El usuario selecciona el archivo CSV desde su dispositivo local.
- Cargar datos: El usuario carga el archivo CSV en la aplicación.
- Verificar carga de datos: El sistema confirma la carga exitosa del archivo CSV.
- Visualización en tabla: El usuario selecciona la opción para visualizar los datos en formato de tabla.
- Verificar tabla: El sistema muestra una tabla con los datos cargados desde el CSV. El usuario verifica que los datos están correctamente organizados y completos.
- Visualización en gráfico: El usuario selecciona la opción para visualizar los datos en formato de gráfico.
- Verificar gráfico: El sistema genera un gráfico con los datos del CSV. El usuario verifica que el gráfico representa correctamente los datos y es interactivo.

Criterios de éxito:

- El archivo CSV se carga correctamente en la aplicación.
- La tabla muestra todos los datos del CSV sin errores y permite una navegación fluida.
- El gráfico se genera correctamente y representa los datos del CSV de manera precisa e interactiva.

Criterios de fallo:

- El archivo CSV no se carga o se carga incorrectamente.
- La tabla muestra datos incompletos, desorganizados o con errores.
- El gráfico no se genera correctamente o no representa los datos adecuadamente.

5.1.2 Caso de Prueba 2

Carga y Visualización de Datos desde una API Rest

Objetivo:

- Verificar que la funcionalidad de carga de datos desde una API Rest, visualización en formato de tabla y gráfico se ejecute correctamente.

Precondiciones:

- El usuario debe disponer de las credenciales necesarias para acceder a la API Rest.
- La API Rest debe estar disponible y operativa.
- El formato de los datos que se descargan de la API debe ser compatible con la aplicación (formato csv).

Pasos:

- Abrir la aplicación: El usuario inicia la aplicación web.
- Al acceder a la aplicación la opción por defecto es usar la API-Rest especificada en el momento de crear la imagen docker.
- Verificar carga de datos: El sistema confirma la carga exitosa de los datos desde la API Rest.
- Visualización en tabla: El usuario selecciona la opción para visualizar los datos en formato de tabla.
- Verificar tabla: El sistema muestra una tabla con los datos cargados desde la API Rest. El usuario verifica que los datos están correctamente organizados y completos.
- Visualización en gráfico: El usuario selecciona la opción para visualizar los datos en formato de gráfico.
- Verificar gráfico: El sistema genera un gráfico con los datos de la API Rest. El usuario verifica que el gráfico representa correctamente los datos y es interactivo.

Criterios de éxito:

- Los datos se cargan correctamente desde la API Rest en la aplicación.
- La tabla muestra todos los datos de la API Rest sin errores y permite una navegación fluida.
- El gráfico se genera correctamente y representa los datos de la API Rest de manera precisa e interactiva.

Criterios de fallo:

- Los datos no se cargan o se cargan incorrectamente desde la API Rest.
- La tabla muestra datos incompletos, desorganizados o con errores.
- El gráfico no se genera correctamente o no representa los datos adecuadamente.

5.2 Resultados

En este capítulo trataremos las distintas posibles ampliaciones o mejoras que puede tener las aplicaciones. También abordaremos los resultados obtenidos tras el desarrollo y prueba de la aplicación.

Análisis de los resultados

El desarrollo del sistema de análisis y visualización de datos ha permitido implementar una herramienta sólida y flexible que cumple con los objetivos planteados al inicio del proyecto. Los principales resultados obtenidos son:

- **Carga y manejo eficiente de datos:** Las pruebas demostraron que la aplicación es capaz de cargar y manejar de manera eficientes un gran volumen de datos en tiempo real, garantizando la integridad y la velocidad en el procesamiento de dichos datos.
- **Interfaz intuitiva y personalizable:** La interfaz de usuario basada en Angular brinda una experiencia de usuario fácil de entender y permite a los usuarios ajustar las visualizaciones según sus necesidades. Los usuarios pueden crear dashboards que incluyan tablas y gráficos que muestren las métricas más relevantes para ellos.
- **Visualización de estadísticas y tendencias:** La aplicación ayuda a los usuarios a encontrar patrones y tomar decisiones basadas en datos al mostrar estadísticas y tendencias. Las visualizaciones son accesibles y claras, lo que permite un análisis exhaustivo y comprensible.
- **Despliegue con Docker:** La distribución y el despliegue de la aplicación se han vuelto más fáciles con Docker. Se ha demostrado que la creación de contenedores independientes para aplicaciones web es efectiva, lo que garantiza el modularidad y la facilidad de mantenimiento del sistema.

En general, los resultados obtenidos validan la funcionalidad y la utilidad de la herramienta desarrollada, demostrando que cumple con los requisitos establecidos y ofrece un valor significativo a los usuarios.

5.3 Líneas futuras

Aunque los módulos implementados en las aplicaciones, cumple con el objetivo del proyecto la aplicación tiene mucho potencial de mejora. En este capítulo se propondrán algunas posibles mejoras o ampliaciones que se podrían llevar a cabo en la aplicación:

- **Integración con más fuentes de datos:** Ampliar la capacidad de la aplicación para obtener datos desde otras fuentes, como bases de datos SQL/NoSQL y otros servicios web. Este punto es muy interesante de cara posibles mejoras ya que la posibilidad de extraer datos de distintas fuentes hace que la aplicación sea un producto más flexible y robusto.
- **Mejoras en la visualización:** Incorporar más tipos de gráficos y herramientas de visualización avanzadas, como diagramas de flujo, mapas de calor y gráficos interactivos. En este punto también se puede realizar mejoras sustanciales ya que actualmente la aplicación puede ofrecer un numero de graficas limitadas para visualizar datos.
- **Optimización del rendimiento:** Continuar optimizando el rendimiento de la aplicación para manejar aún mayores volúmenes de datos y mejorar la experiencia del usuario.
- **Seguridad y autenticación:** Mejorar las funcionalidades de seguridad y autenticación para asegurar que los datos estén protegidos y solo accesibles por usuarios autorizados. Actualmente la seguridad de la aplicación radica en las credenciales que especifica el usuario para acceder a los datos, por este motivo crear un sistema de autenticación para acceder a la aplicación puede ser muy interesante de cara a futuro.

Estas propuestas buscan continuar el desarrollo de la aplicación, haciendo que sea aún más robusta, flexible y valiosa para los usuarios, asegurando que pueda seguir adaptándose a las necesidades emergentes en el campo del análisis y visualización de datos.

5.4 Conclusiones

El sistema de análisis y visualización de datos desarrollado se ha convertido en una herramienta eficiente y adaptable capaz de manejar grandes cantidades de datos. A lo largo del proyecto, se han logrado varios objetivos clave que contribuyen a su éxito:

Objetivo 1: Crear una plataforma que permita visualizar datos estadísticos

Se ha desarrollado una solución integral que permite a los usuarios analizar y visualizar de manera eficiente grandes cantidades de datos. La aplicación proporciona visualizaciones claras y precisas en forma de tablas y gráficos, lo que facilita la interpretación de la información y el descubrimiento de patrones y tendencias ocultos.

Objetivo 2: Asegurar que las estadísticas sean configurables

La plataforma brinda una amplia gama de personalización y adaptabilidad, lo que permite a los usuarios seleccionar y filtrar datos, seleccionar columnas específicas y ajustar las visualizaciones según sus necesidades. Esta capacidad de personalización garantiza que la herramienta sea útil en una variedad de situaciones y tipos de análisis.

Objetivo 3: Facilitar la configuración e instalación de la herramienta

El uso de tecnologías modernas como Docker para el despliegue y Angular para el frontend ha simplificado la configuración e instalación. La herramienta garantiza un despliegue eficiente y un rendimiento óptimo, y puede integrarse fácilmente en una variedad de entornos. Además, la modularidad de la arquitectura facilita las futuras expansiones y actualizaciones sin invertir mucho esfuerzo.

El cuadro de mandos implementado, que permite una visualización detallada de las estadísticas de uso de una API Rest, será muy beneficioso para el Grupo de Investigación Biomédica y otros desarrolladores. La versatilidad de la aplicación destaca, ya que soporta la carga de datos desde APIs Rest y archivos CSV locales. Esta flexibilidad permite que los usuarios integren una variedad de fuentes de datos según sus necesidades particulares. La interfaz de usuario, diseñada con un enfoque en la usabilidad, facilita la selección, filtrado y personalización de datos al presentarlos en formatos de tablas y gráficos, lo que los hace más fáciles de entender y analizar.

La integración y el despliegue eficientes requieren el uso de tecnologías modernas como Angular para el desarrollo front-end y Docker para el despliegue. Esto garantiza que la herramienta brinde una experiencia de usuario consistente y escalable y funcione de manera óptima en diversos entornos. La arquitectura modular del sistema no solo facilita su expansión y actualización, sino que también permite adaptaciones futuras sin modificar significativamente el sistema. En resumen, esta herramienta es una herramienta poderosa y adaptable que ayuda a las organizaciones a tomar decisiones informadas basadas en análisis de datos claros y bien estructurados.

6 Análisis de Impacto

Este trabajo se ajusta al Objetivo de Desarrollo Sostenible (ODS) número 9: "Industria, Innovación e Infraestructura". Este objetivo fomenta la innovación y promueve la industrialización inclusiva y sostenible.

Promueve la innovación: Este proyecto implica el desarrollo de una solución innovadora para analizar y visualizar grandes cantidades de datos de manera eficiente. Al ofrecer una plataforma para los usuarios que les permite tomar decisiones basadas en datos de forma sencilla, por lo que promueve la innovación en el ámbito de la gestión y comprensión de datos.

Infraestructura tecnológica: La plataforma web utiliza tecnologías de vanguardia, por lo que contribuye a la construcción de infraestructuras infraestructuras tecnológicas sólidas y avanzadas. Esto es fundamental para que las empresas y empresas tengan acceso a herramientas modernas que les permita hacer una gestión eficiente de sus datos.

Uso eficiente de los recursos: Esta herramienta está orientada a permitir que las organizaciones puedan tomar mejores decisiones basadas en datos, por lo tanto, esto les permitirá optimizar sus recursos y procesos. Esto les conducirá a tener una mayor eficiencia y sostenibilidad en sus operaciones.

7 Bibliografía

- [1] HTML: Lenguaje de etiquetas de hipertexto | MDN (mozilla.org)
- [2] CSS | MDN (mozilla.org)
- [3] JavaScript | MDN (mozilla.org)
- [4] https://www.typescriptlang.org/docs/
- [5] https://devblogs.microsoft.com/typescript/
- [6] https://angular.io/docs
- [7] https://material.angular.io/components/categories
- [8] https://docs.docker.com/
- [9] https://docs.docker.com/engine/reference/commandline/docker/
- [10] https://docs.docker.com/get-started/overview/
- [11] https://www.docker.com/products/docker-desktop/
- [12] https://www.npmjs.com/package/@ngx-env/builder

8 Anexo

8.1 Anexo A: Manual de despliegue.

8.1.1 Configuración del entorno de despliegue

Configurar el entorno de despliegue es fundamental para asegurar que el proyecto ejecute de manera correcta. Para ello se debe seguir una serie de pasos.

- **Preparación del servidor:** Se selecciona el servidor o la infraestructura en la que se desplegará el proyecto. Se asegura que el servidor tenga los recursos suficientes para ejecutar los contenedores de Docker de manera eficiente.

- **Instalación de Docker:** Se instala Docker en el servidor siguiendo las instrucciones proporcionadas por el proveedor de Docker para el sistema operativo correspondiente [11].

- **Configuración del Firewall y Puertos:** Se configuran los firewalls y los puertos del servidor para permitir el tráfico necesario para el proyecto, como el tráfico HTTP/HTTPS para el proyecto web.

- **Gestión de Variables de Entorno:** Se gestionan las variables de entorno necesarias para la configuración del proyecto, en este caso se deben especificar las credenciales de acceso a la API de la cual se descargarán los datos. Estas variables deben ser especificadas en el DockerFile del proyecto.

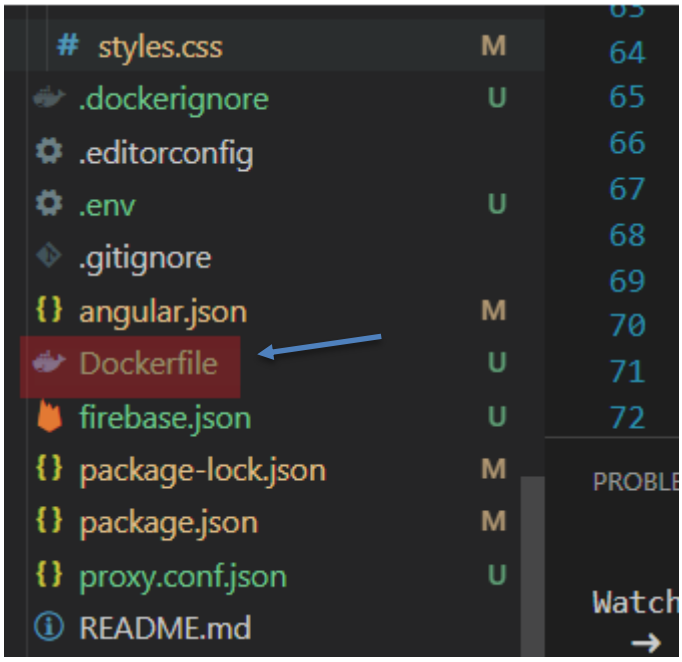

Imagen 32: Imagen Localización DockerFile

```
FROM node:alpine

WORKDIR /usr/src/app

COPY . /usr/src/app

RUN npm install -g @angular/cli

RUN npm install

# Set environment variables
ENV NG_APP_API=http://138.100.11.174/dptm/searchdata
ENV NG_APP_PASSWORD=

CMD ["ng", "serve", "--host", "0.0.0.0"]
```

Imagen 33: Imagen Variables de Entorno.

8.1.2 Proceso de despliegue.

El proceso de despliegue de la aplicación consta de dos sencillos pasos los cuales de describen a continuación.

- **Creación del contenedor:** Una vez se tiene configurado de manera correcta el DockerFile se debe proceder a la creación de contendor. Esto se debe hacer usando una terminal como puede ser un PowerShell, usando el mandato:

  docker build -t angular-docker

  -**angular-docker** es el nombre de la imagen del contenedor que se está ejecutando.

Imagen 34: Imagen creación imagen Docker.

- **Ejecución Contenedor:** Una vez se tiene configurado de manera correcta el DockerFile se debe proceder a la creación del contendor. Esto se debe hacer usando una terminal como puede ser un PowerShell, usando el siguiente mandato:

  docker run -p 4201:4200 angular-docker

  -**4201** es el puerto del host al que se mapea el puerto del contenedor.
  -**4200** es el puerto dentro del contenedor donde se ejecuta la aplicación.
  -**angular-docker** es el nombre de la imagen del contenedor que se está ejecutando.

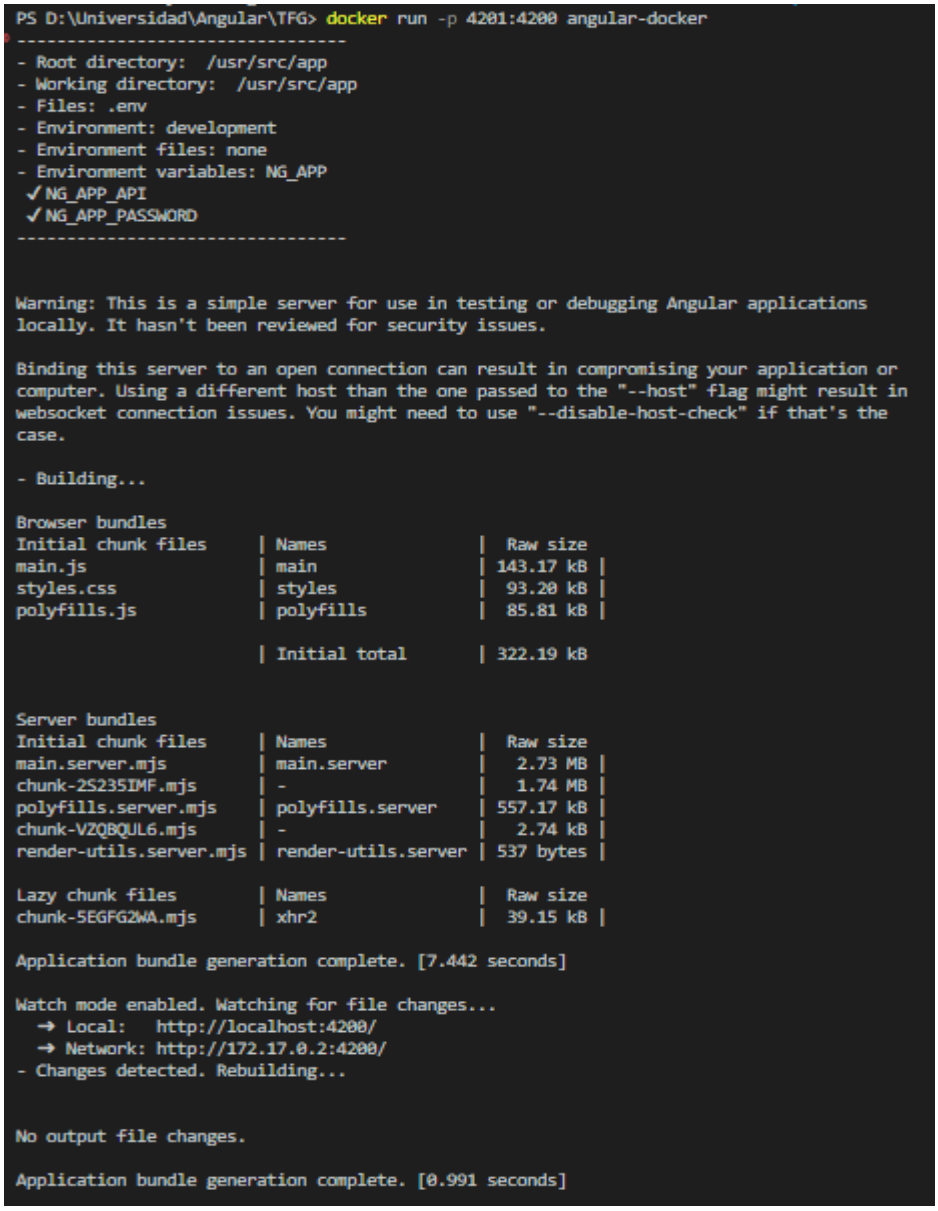

Imagen 35: Imagen Ejecución del contendor Docker.

8.2 Anexo B: Informe de originalidad Turnitin

Yosvany

INFORME DE ORIGINALIDAD

8%	8%	1%	%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	oa.upm.es Fuente de Internet	2%
2	repo.undip.ac.id Fuente de Internet	<1%
3	www.coursehero.com Fuente de Internet	<1%
4	www.slideshare.net Fuente de Internet	<1%
5	publications.theseus.fi Fuente de Internet	<1%
6	www.seidor.com Fuente de Internet	<1%
7	prezi.com Fuente de Internet	<1%
8	Nuno Moutinho. "Sharing Information in a Virtual Community of Crowdfunding: The case of Kickstarter", Repositório Aberto da Universidade do Porto, 2014. Publicación	<1%

9	cloud.google.com Fuente de Internet	<1 %
10	docs.bmc.com Fuente de Internet	<1 %
11	experienceleague.adobe.com Fuente de Internet	<1 %
12	repositorio.puce.edu.ec Fuente de Internet	<1 %
13	www.jove.com Fuente de Internet	<1 %
14	d-nb.info Fuente de Internet	<1 %
15	www.corsicaferries.com Fuente de Internet	<1 %
16	www.mendoza.gov.ar Fuente de Internet	<1 %
17	Babeș-Bolyai University Publicación	<1 %
18	manualzilla.com Fuente de Internet	<1 %
19	www.gti.ssr.upm.es Fuente de Internet	<1 %
20	www.techtitute.com Fuente de Internet	<1 %

21	www.urllinking.com Fuente de Internet	<1 %
22	1library.co Fuente de Internet	<1 %
23	aws.amazon.com Fuente de Internet	<1 %
24	blogthinkbig.com Fuente de Internet	<1 %
25	byte.mkm-pi.com Fuente de Internet	<1 %
26	divertida98876.articlesblogger.com Fuente de Internet	<1 %
27	documents.mx Fuente de Internet	<1 %
28	efeagro.com Fuente de Internet	<1 %
29	explainer.visual-paradigm.com Fuente de Internet	<1 %
30	prodcd.lincolnelectric.com Fuente de Internet	<1 %
31	underc0de.org Fuente de Internet	<1 %
32	ww.idg.es Fuente de Internet	<1 %

33	www.ciberespacio.com.ve Fuente de Internet	<1 %
34	www.hectorgomis.com Fuente de Internet	<1 %
35	www.jardineriaon.com Fuente de Internet	<1 %
36	www.scanbit.net Fuente de Internet	<1 %
37	"Newspapers collection management: printed and digital challenges", Walter de Gruyter GmbH, 2008 Publicación	<1 %
38	Alex Andrés Santamaría Philco. "Un marco de soporte para el ciclo de vida de la eParticipación enriquecido con gestión de confianza", Universitat Politecnica de Valencia, 2020 Publicación	<1 %
39	dev.to Fuente de Internet	<1 %
40	ec.europa.eu Fuente de Internet	<1 %
41	es.slideshare.net Fuente de Internet	<1 %
42	hdl.handle.net Fuente de Internet	

		<1 %
43	idoc.pub Fuente de Internet	<1 %
44	laplatavive.com Fuente de Internet	<1 %
45	qiduocomunica.com Fuente de Internet	<1 %
46	search.ndltd.org Fuente de Internet	<1 %
47	view.genial.ly Fuente de Internet	<1 %
48	webware.hypermart.net Fuente de Internet	<1 %
49	worldcat.org Fuente de Internet	<1 %
50	www.computerworld.es Fuente de Internet	<1 %
51	www.koreascience.or.kr Fuente de Internet	<1 %
52	www.uhu.es Fuente de Internet	<1 %
53	transportesynegocios.wordpress.com Fuente de Internet	<1 %

54	www.timetoast.com Fuente de Internet	<1 %

Excluir citas	Apagado	Excluir coincidencias	Apagado
Excluir bibliografía	Apagado		

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Sun Jun 02 23:48:14 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)