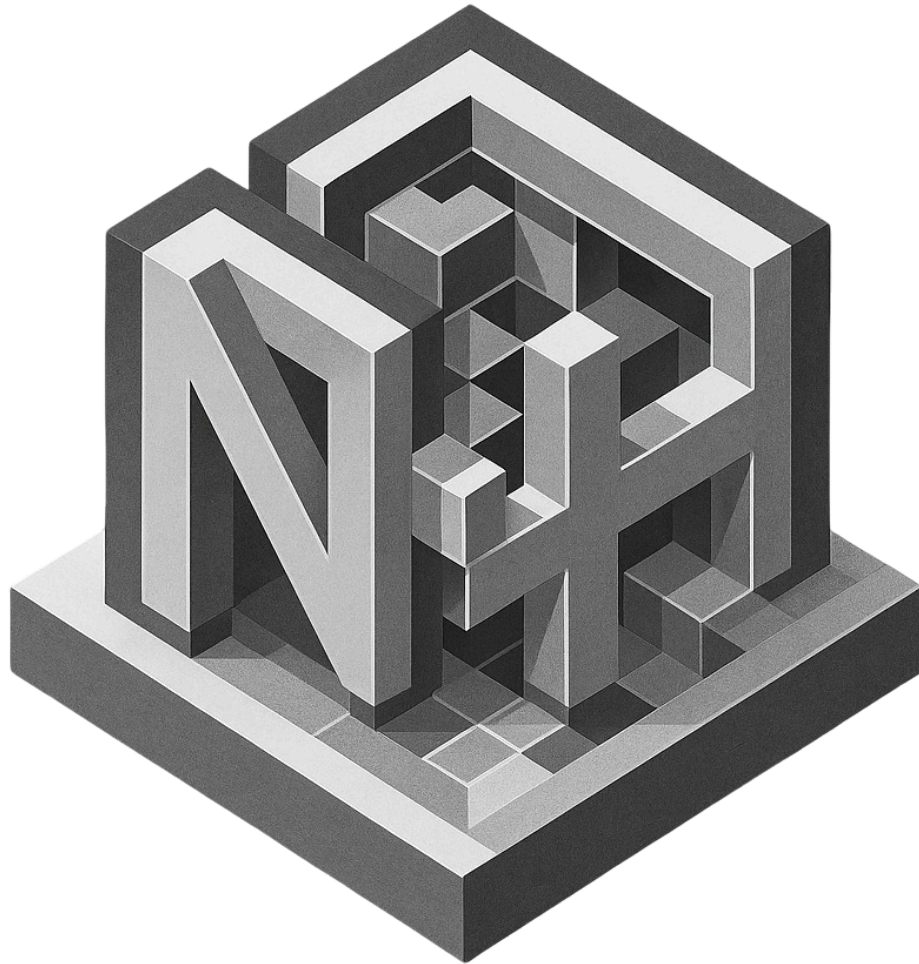


NewsHub



Proyecto Desarrollo de Aplicaciones Web


Javier Cruz Rojas y Manuel Villar Navarro

CIFP Zonzamas

Mayo 2025



| | |
|---|-----------|
| 1. Resumen..... | 5 |
| 2. Fase de análisis..... | 5 |
| 2.1 ¿Qué es NewsHub?..... | 5 |
| 2.2 Objetivos del proyecto..... | 5 |
| 3. Análisis de requisitos del Software..... | 6 |
| 3.1 Identificación del público objetivo..... | 6 |
| 3.2 Requisitos de la audiencia..... | 6 |
| Perfil Socioeconómico..... | 7 |
| Edad..... | 7 |
| Nivel de estudios..... | 8 |
| Género..... | 8 |
| Consumo de noticias..... | 8 |
| Temas de interés..... | 8 |
| Satisfacción con los medios..... | 9 |
| Verificación de noticias (cómo el usuario consulta la veracidad de noticias)..... | 9 |
| Sugerencias..... | 9 |
| 3.3 Documentar Requisitos..... | 9 |
| Requisitos Funcionales..... | 9 |
| Contraste de noticias:..... | 9 |
| Análisis estadístico:..... | 10 |
| Verificación y trazabilidad:..... | 10 |
| Personalización por intereses:..... | 10 |
| Alertas configurables:..... | 10 |
| Requisitos No Funcionales..... | 10 |
| Usabilidad:..... | 10 |
| Rendimiento:..... | 10 |
| Seguridad y privacidad:..... | 10 |
| Escalabilidad:..... | 11 |
| Requisitos de Contenido..... | 11 |
| Diversidad de fuentes informativas:..... | 11 |
| Actualización constante:..... | 11 |
| Transparencia informativa:..... | 11 |
| Evitar saturación y fake news:..... | 11 |
| 3.4 Priorizar requisitos..... | 11 |
| Alta prioridad..... | 11 |
| Media prioridad..... | 11 |
| Baja prioridad..... | 11 |
| 3.5 Diagramas casos de uso..... | 12 |
| Actores..... | 12 |
| Casos de uso..... | 12 |
| 4. Fase de diseño..... | 12 |
| 4.1 API..... | 12 |
| Arquitectura..... | 13 |
| Funcionamiento..... | 14 |

| | |
|---|-----------|
| Tecnologías y dependencias implicadas en la API..... | 14 |
|  Node.js / JavaScript..... | 14 |
|  Python..... | 15 |
| - OS: Módulo para interactuar con el sistema operativo y archivos..... | 15 |
| 4.2 Diagramas de clases..... | 15 |
| 4.3 Diagrama del flujo de la aplicación..... | 16 |
| 4.4 Modelo E/R..... | 17 |
| 4.5 Estructura de la Base de Datos..... | 17 |
| Periódicos..... | 17 |
| Temas..... | 17 |
| Claves..... | 18 |
| Noticias..... | 18 |
| Grupos..... | 19 |
| Noticias-Temas..... | 19 |
| Noticias-Claves..... | 20 |
| Grupos-Noticias..... | 20 |
| 4.6 Diseño..... | 20 |
| Uso Ideal..... | 21 |
| Aplicación en Pantallas..... | 21 |
| Pantalla de diseño..... | 21 |
| Maquetas Figma..... | 21 |
| Prototipo HTML + CSS..... | 22 |
| Pantalla de inicio actual..... | 22 |
| 5. Codificación..... | 22 |
| Stack Tecnológico..... | 22 |
| Backend (API)..... | 22 |
| Frontend..... | 22 |
| Preparación del entorno de proyecto..... | 22 |
| Requisitos del sistema..... | 22 |
| 1. Actualice la máquina..... | 23 |
| 2. Clone el repositorio..... | 23 |
| 3. Elimine el fichero package-lock.json (para evitar problemas)..... | 23 |
| 4. Instale las dependencias de node..... | 23 |
| 5. En el directorio /api cree el fichero '.env'..... | 23 |
| 6. Inicie la API..... | 23 |
| 6.Pruebas unitarias..... | 24 |
| 7. Conclusiones..... | 23 |
| 8. Glosario..... | 25 |
| A..... | 25 |
| B..... | 25 |
| C..... | 25 |
| D..... | 25 |
| E..... | 25 |
| F..... | 25 |

| | |
|--------|----|
| J..... | 25 |
| N..... | 26 |
| M..... | 26 |
| O..... | 26 |
| P..... | 26 |
| S..... | 26 |
| T..... | 26 |
| W..... | 26 |

1. Resumen

"NewsHub es una aplicación web innovadora que utiliza inteligencia artificial y web scraping para analizar y comparar el sesgo ideológico en noticias de diferentes medios. Su objetivo es promover el consumo crítico de información mediante herramientas de contraste estadístico y verificación. Desarrollada con Node.js, Python y MySQL, la plataforma busca combatir la desinformación y ofrecer transparencia mediática."

2. Fase de análisis

2.1 ¿Qué es NewsHub?

NewsHub es una **Aplicación Web innovadora** cuyo objetivo principal es **recopilar, analizar y representar información** sobre noticias y temas de diferentes fuentes de periódicos, tomando en cuenta la ideología o sentimiento de cada medio. La aplicación permite a los usuarios comparar cómo diferentes medios de comunicación cubren la misma noticia o tema, identificando el posicionamiento de los medios y proporcionando un análisis sobre el sesgo ideológico presente en las informaciones.

El nombre *NewsHub* nace de la idea de convertirse en un **centro neurálgico ("hub") de noticias**, donde se unen diversas fuentes informativas. Así como un hub conecta diferentes rutas en un sistema, NewsHub reúne múltiples visiones periodísticas en un solo lugar, facilitando la comparación, el análisis y la reflexión crítica sobre la información que consumimos. En resumen, queremos que nuestra aplicación sea una **centralita de noticias**.

Nuestro propósito es **facilitar el acceso a múltiples perspectivas sobre una misma noticia**, permitiendo a los usuarios obtener una visión más objetiva y equilibrada de los eventos. De esta manera, el proyecto busca contribuir a una mejor comprensión de cómo los medios informan y, potencialmente, reducir el impacto del sesgo mediático a la vez de reducir el tiempo de búsqueda sobre noticias.

2.2 Objetivos del proyecto

A. **Recopilar información** sobre noticias de diversas fuentes periodísticas, con especial enfoque en diferentes ideologías y enfoques informativos. Para ello, haremos **web scraping** manual de diversos medios nacionales.

- ❖ Periódicos de **izquierdas**: El País y El Plural.
- ❖ Periódicos de **derechas**: ABC y LibertadDigital.

B. **Identificar el sentimiento** en cada noticia a través de un análisis de las fuentes y sus tendencias (por ejemplo, de izquierda, derecha, o centrado).

Además, indagaremos cómo los periódicos tratan diversos temas en función de sus ideales.

- C. **Facilitar la comparación** de la cobertura de una misma **noticia o tema** entre **diferentes medios**, brindando un análisis claro sobre las diferencias y similitudes en la información. Los periódicos se clasifican por diversos parámetros (ideológicos, público objetivo, inversores, etc.).
- D. **Promover el acceso a la información de manera objetiva**, ayudando a los usuarios a reconocer y cuestionar los sesgos presentes en los medios de comunicación. Para ello, utilizaremos herramientas de análisis e Inteligencia Artificial para interpretar la información de la manera más objetiva posible.

3. Análisis de requisitos del Software

3.1 Identificación del público objetivo

La idea de NewsHub es llegar a las **grandes masas de la sociedad**, aunque reconocemos que la búsqueda proactiva de información suele ser realizada por un grupo minoritario.

Nuestro objetivo es tratar de que **independientemente del perfil**, en este caso sea una buena experiencia como cómoda el consultar noticias, haciendo que el acceso a la misma se haga con tan pocos recursos necesarios como son un dispositivo inteligente y conexión a internet.

3.2 Requisitos de la audiencia

Nuestra plataforma se basa en el contraste de información de diversos periódicos, realizando un análisis estadístico y estudios sobre sus noticias. Tras esta primera fase, identificamos temas comunes en los periódicos y aplicamos los análisis para representar la realidad de los medios, permitiendo al usuario observar si una noticia está sesgada y facilitando su posicionamiento crítico.

Tras un análisis del público objetivo, hemos identificado los siguientes patrones en cuanto a preferencias y preocupaciones:

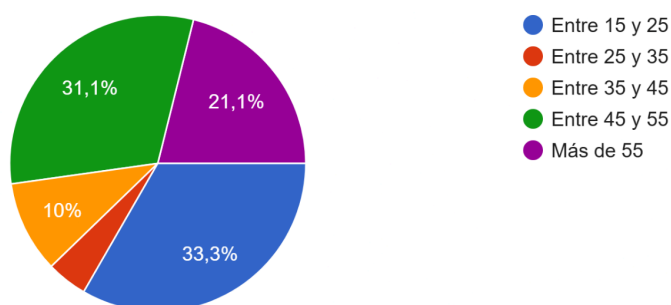
- **Consumo de noticias:** Frecuencia, medios preferidos, tiempo dedicado.
- **Temas de interés:** Economía, política, salud, etc.
- **Satisfacción con los medios:** Críticas comunes como publicidad excesiva, noticias falsas, falta de imparcialidad.
- **Verificación de noticias:** Métodos de verificación y confianza en la información.
- **Perfil socioeconómico:** Edad, situación laboral, nivel de ingresos, nivel de estudios.

Perfil Socioeconómico

Edad

Edad

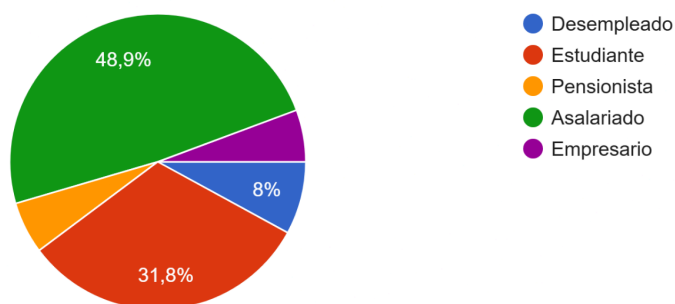
90 respuestas



Situación Laboral

Situación Laboral

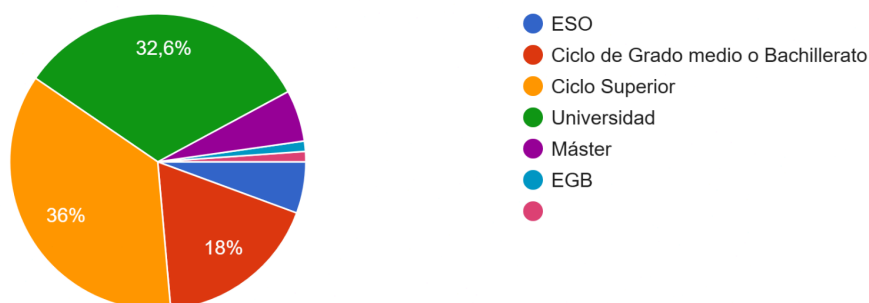
88 respuestas



Nivel de estudios

Nivel de estudios

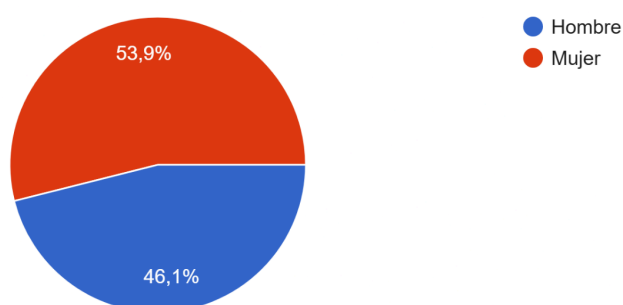
89 respuestas



Género

Género

89 respuestas



Consumo de noticias

- **Casi el 75% de las personas encuestadas se consideran informadas** con respecto a la actualidad.
- Dentro de este tercer cuartil, el consumo de noticias es realizado **diariamente**.
- Entre los medios más consumidos se encuentran: **Televisión, Periódicos Digitales y Redes Sociales**.
- En la métrica de ‘tiempo destinado a informarse’, sí que hay mayor diversidad de hábitos. Siendo los más destacados **entre 10 y 20 minutos y más de 20 minutos**.

Temas de interés

Los asuntos relevantes de los usuarios son: **Actualidad Local, Salud y Política**. Mientras que en el opuesto tenemos: **Prensa Amarilla, Historia y arqueología** además de **Medio Ambiente**.

Satisfacción con los medios

El 70% de las personas considera estar satisfecha con la forma en la que consumen noticias. Sin embargo hay aspectos que le resultan molestos tales como: **‘fake news’**, **‘publicidad excesiva’** y **‘escasa imparcialidad’**.

Es por ello que si bien están satisfechos, son reacios a las noticias que consumen, es decir, desconfían de las noticias consumidas. Para solventar este vacío los usuarios optan por contrastar diversas fuentes de noticias.

Volviendo a las molestias que tienen los consumidores con respecto a la forma de consumir noticias, vemos que $\frac{4}{5}$ partes de los mismos admite haber encontrado fake news en los medios frecuentados para buscar noticias.

Verificación de noticias (cómo el usuario consulta la veracidad de noticias)

- Métodos de verificación: (contrastar con otros medios, consultar fuentes oficiales, uso de redes sociales).
- Confianza en la información consumida.

Sugerencias

Por último y no menos importante, varios encuestados nos han indicado que verían con buenos ojos:

- ❖ “Mejorar los medios de comunicación , podamos tener más información y que sean noticias verdaderas.”
- ❖ “Que aporten más información y seguridad en noticias.”

3.3 Documentar Requisitos

Requisitos Funcionales

Los requisitos funcionales describen lo que el sistema debe ofrecer a los usuarios, considerando sus necesidades y hábitos identificados en la fase de análisis:

Contraste de noticias:

- La plataforma debe permitir a los usuarios comparar cómo diferentes medios tratan un mismo tema. Esto responde a la necesidad de contrastar fuentes para combatir la desconfianza en los medios y evitar fake news. Ejemplo: Un usuario puede comparar cómo distintos periódicos cubren una noticia política y observar las diferencias ideológicas.

Análisis estadístico:

- Se debe ofrecer al usuario estadísticas visuales y claras sobre la cobertura mediática: qué temas se tratan más, presencia de palabras clave, enfoque según la fuente, etc. Esto permitirá evidenciar patrones, sesgos y cobertura desigual.

Verificación y trazabilidad:

- La aplicación debe enlazar a las fuentes originales y permitir al usuario consultar artículos relacionados para verificar la veracidad y profundidad del contenido.

Personalización por intereses:

- Los usuarios deben poder seleccionar sus temas de interés prioritarios (como política, salud o actualidad local), según los resultados de la encuesta, y filtrar la información que reciben.

Alertas configurables:

- Ofrecer la posibilidad de configurar alertas sobre temas importantes, lo cual responde al patrón de consumo diario de noticias en secciones breves (10-20 minutos).

Requisitos No Funcionales

Estos requisitos aseguran que la plataforma funcione correctamente y ofrezca una buena experiencia al usuario:

Usabilidad:

- Interfaz clara, simple e intuitiva, accesible desde diferentes dispositivos. Esto permite que incluso usuarios con poco tiempo puedan informarse rápidamente (tiempo medio de consulta entre 10 y 20 minutos).

Rendimiento:

- Debe tener tiempos de respuesta bajos y manejar de forma eficiente la recopilación, comparación y visualización de noticias.

Seguridad y privacidad:

- Garantizar que los datos de usuario estén protegidos, especialmente si se implementan funciones como personalización o alertas.

Escalabilidad:

- La plataforma debe poder ampliarse para incluir más periódicos, más usuarios o nuevas funcionalidades sin comprometer el rendimiento.

Requisitos de Contenido

Estos requisitos se enfocan en la naturaleza y calidad de la información ofrecida:

Diversidad de fuentes informativas:

- Incluir medios con distintas ideologías (izquierda, derecha, centro) y formatos (digital, televisivo, redes sociales) para cubrir los hábitos de consumo detectados.

Actualización constante:

- Las noticias deben renovarse con frecuencia, permitiendo al usuario estar informado en tiempo real, algo esencial para quienes consumen noticias diariamente.

Transparencia informativa:

- Indicar claramente la fuente de cada noticia y el medio al que pertenece, ayudando a los usuarios a evaluar su confianza en ella.

Evitar saturación y fake news:

- Mediante filtros y sistemas de validación, se debe reducir el contenido basura o duplicado, y priorizar noticias verificadas, en línea con las quejas de los usuarios sobre publicidad excesiva y desinformación.

3.4 Priorizar requisitos

Alta prioridad

- Contraste de noticias y verificación de fuentes.
- Análisis estadístico y visualización de datos.

Media prioridad

- Diversidad de fuentes.
- Obtener noticias en tiempo real.

Baja prioridad

- Escalabilidad y rendimiento.

- Personalización de temas de interés y alertas de noticias.
- Diversidad de fuentes.

3.5 Diagramas casos de uso

Actores

- Usuario: Persona que consume noticias y utiliza la plataforma para contrastar información.

Casos de uso

- Contrastar noticias url: El usuario mediante una URL puede consultar el posicionamiento ideológico de una noticia.
- Verificar noticias: El usuario puede verificar la veracidad de una noticia contrastando con otras fuentes.
- Gestionar fuentes (Administrador): El administrador puede agregar, eliminar o actualizar las fuentes de las noticias.

4. Fase de diseño

Antes de empezar con el desarrollo del proyecto, es conveniente tener bien estructurado tanto los requisitos, arquitectura del proyecto, como las necesidades de nuestros usuarios para prevenir imprevistos e inconsistencia.

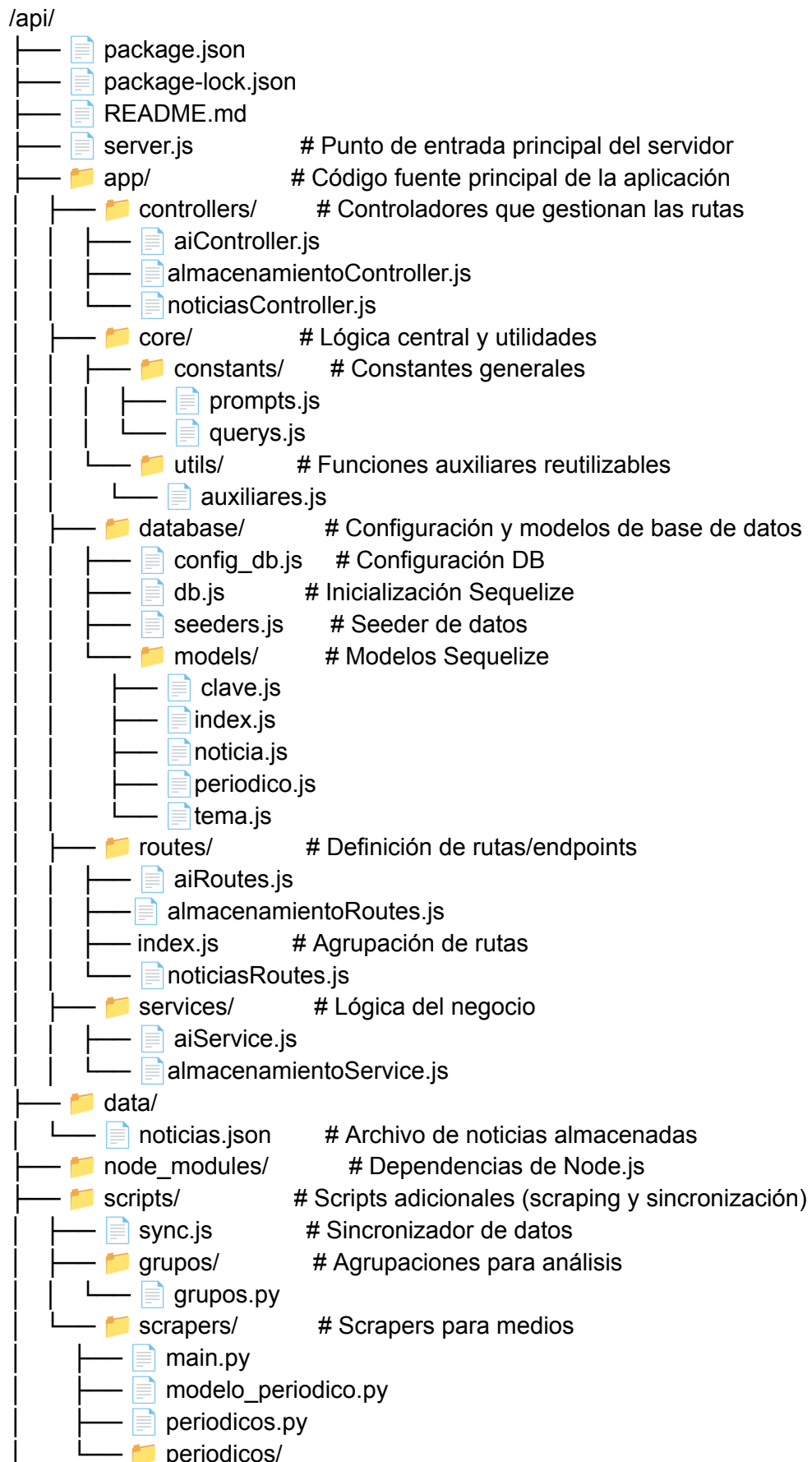
4.1 API

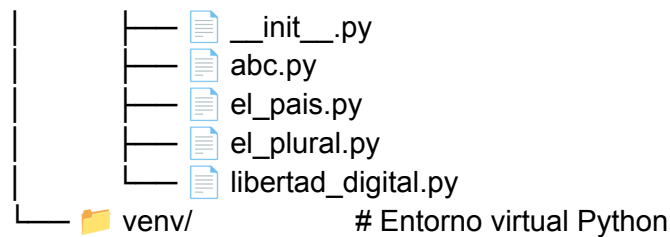
A la hora de realizar el proyecto nos enfrentamos al problema del tratamiento y obtención de datos, llegamos a la conclusión de que era indispensable recopilar noticias por nuestra propia cuenta. Como las APIs existentes son muy limitadas en cuanto a contenido y precio, determinamos que la mejor manera de sostener el proyecto es dividiéndolo funcionalmente en dos el mismo:

- Por un lado la web (apartado visual) que ofrece el resultado de todo el trabajo del backend.
- Para realizar tanto el trabajo con la IA, como web scraping, grupos y clasificación de noticias. Hemos decidido crear una API para gestionarla. Centrando la salida de datos en endpoints controlados por nosotros.

Las ventajas que nos ofrece esto es liberar al frontend de líneas de código para las interacciones con el backend (PHP). Con ello logramos también una escalabilidad del proyecto puesto que separamos las funciones de la misma, permitiendo bajo la misma API realizar múltiples programas del mismo ámbito.

Arquitectura





Funcionamiento

Para que la api funcione correctamente tiene que seguir estos pasos:

1. Iniciar el servidor express con 'express.js'.
2. Ejecutar el fichero 'main.py' ubicado en /scripts, /scraping.
3. Comprueba que se te ha creado en /data el fichero 'noticias.json' y que en su interior se encuentren noticias.
4. Ejecutas el endpoint: tuip/api/ia/analyze mediante una solicitud HTTP post.
5. Ejecuta otro endpoint: tuip/api/almacenamiento/verificar-palabras-claves mediante una solicitud HTTP post.
6. Compruebe nuevamente que en el fichero 'noticias.json' que todos los campos están rellenos de cada uno de las noticias. Además de que las palabras claves sean números.
7. En caso de cumplir con todo lo que te he dicho previamente. Entonces ejecute el último endpoint para guardar las noticias en la bbdd: tuip/api/noticias/almacenar-bbdd.

Tecnologías y dependencias implicadas en la API

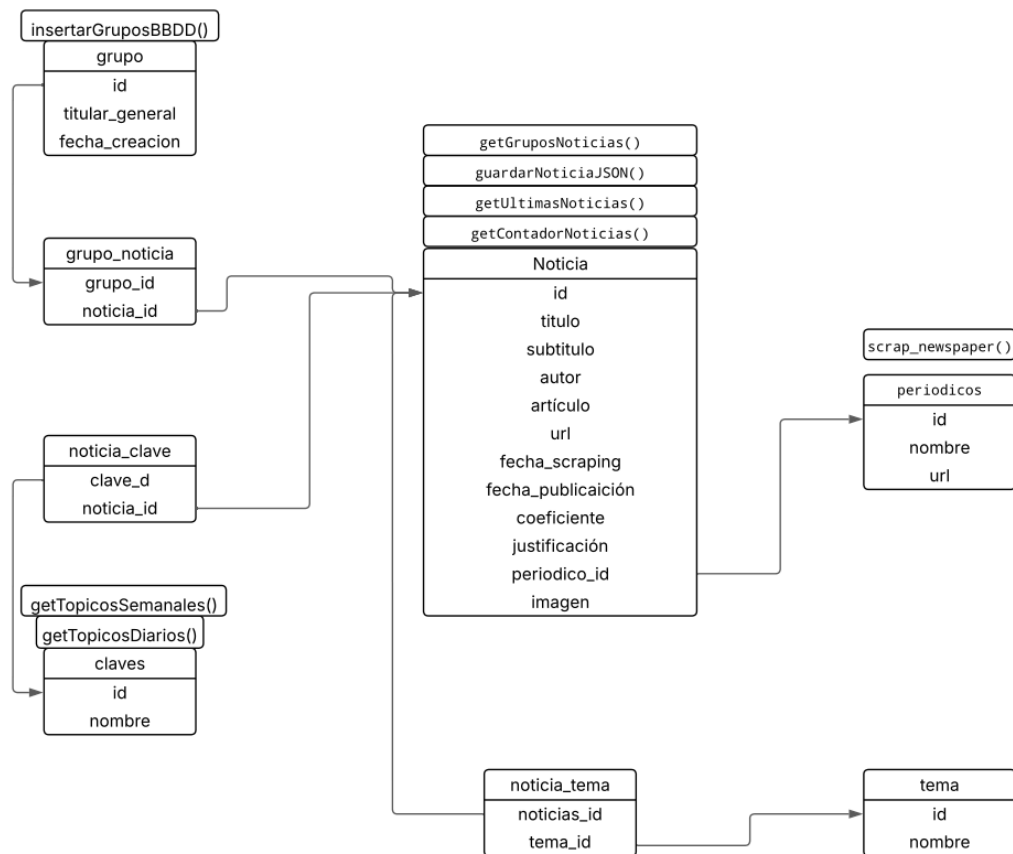
Node.js / JavaScript

- **Express:** Framework backend de Node.js para crear APIs RESTful.
- **Axios:** Cliente HTTP para realizar peticiones a otros servicios.
- **Cors:** Middleware que permite compartir recursos entre distintos orígenes.
- **Dotenv:** Carga variables de entorno desde un archivo .env.
- **Mysql2:** Cliente de MySQL compatible con Sequelize.
- **Sequelize:** ORM para manejar bases de datos relacionales desde Node.js.
- **Umzug:** Herramienta para la gestión de migraciones con Sequelize.
- **Jest:** Framework de testing para JavaScript.
- **SuperTest:** Librería para testear peticiones HTTP.
- **Nodemon:** Herramienta que reinicia el servidor automáticamente al detectar cambios.

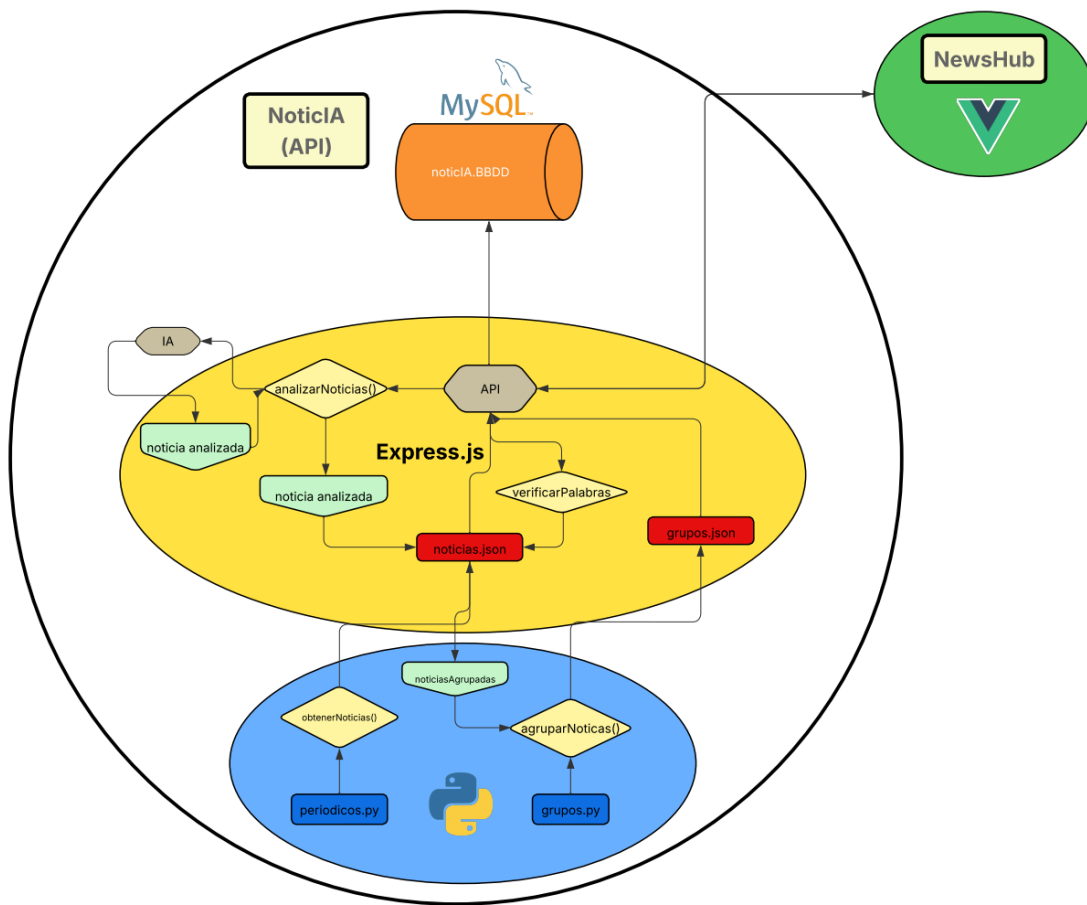
Python

- **AioHTTP:** Framework asíncrono para realizar peticiones HTTP.
- **Asyncio:** Librería estándar para programación asíncrona.
- **Requests:** Librería para hacer peticiones HTTP de forma sencilla.
- **BeautifulSoup:** Librería para parsear y extraer datos de HTML/XML.
- **Sequence Matcher:** Herramienta para comparar similitud entre textos (parte del módulo dif lib).
- **Defaultdict:** Diccionario con valores por defecto (desde collections).
- **Traceback:** Módulo para obtener trazas de errores detalladas.
- **Sys:** Módulo para manejar argumentos del sistema y salida de errores.
- **OS:** Módulo para interactuar con el sistema operativo y archivos.

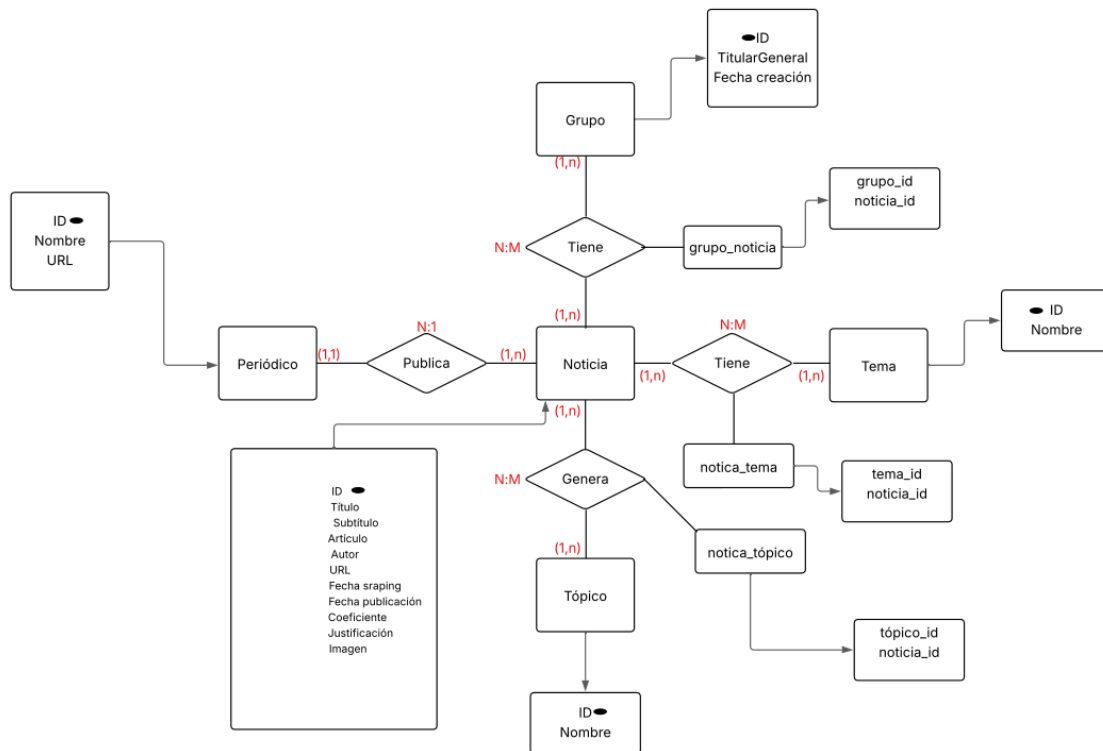
4.2 Diagramas de clases



4.3 Diagrama del flujo de la aplicación



4.4 Modelo E/R



[08]

4.5 Estructura de la Base de Datos

Periódicos

```
CREATE TABLE IF NOT EXISTS periodicos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  url VARCHAR(255)
);
```

```
module.exports = (sequelize, DataTypes) => {
  ...
  const Periodico = sequelize.define('Periodico', {
    id: {
      type: DataTypes.INTEGER,
      autoIncrement: true,
      primaryKey: true
    },
    nombre: {
      type: DataTypes.STRING(50),
      allowNull: false
    },
    url: {
      type: DataTypes.STRING(255)
    }
  }, {
    tableName: 'periodicos',
    timestamps: false
  });

  Periodico.associate = function(models) {
    this.hasMany(models.Noticia, { foreignKey: 'periodico_id' });
  };

  return Periodico;
};
```

Temas

```
CREATE TABLE IF NOT EXISTS temas (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(100) NOT NULL  
);
```

```
module.exports = (sequelize, DataTypes) => {  
  ...  
  const Tema = sequelize.define('Tema', {  
    id: {  
      type: DataTypes.INTEGER,  
      autoIncrement: true,  
      primaryKey: true  
    },  
    nombre: {  
      type: DataTypes.STRING(50),  
      allowNull: false  
    }  
  }, {  
    tableName: 'temas',  
    timestamps: false  
  });  
  
  Tema.associate = function(models) {  
    Tema.belongsToMany(models.Noticia, { through: 'noticias_temas', foreignKey: 'tema_id' });  
  };  
  
  return Tema;  
};
```

Claves

```
CREATE TABLE IF NOT EXISTS claves (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  palabra VARCHAR(100) NOT NULL  
);
```

```
module.exports = (sequelize, DataTypes) => {  
  ...  
  const Clave = sequelize.define('Clave', {  
    id: {  
      type: DataTypes.INTEGER,  
      autoIncrement: true,  
      primaryKey: true  
    },  
    nombre: {  
      type: DataTypes.STRING(50),  
      allowNull: false  
    }  
  }, {  
    tableName: 'claves',  
    timestamps: false  
  });  
  
  Clave.associate = function(models) {  
    this.belongsToMany(models.Noticia, { through: 'noticias_claves', foreignKey: 'clave_id' });  
  };  
  
  return Clave;  
};
```

Noticias

```
CREATE TABLE IF NOT EXISTS noticias (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  titulo VARCHAR(255) NOT NULL,  
  subtítulo TEXT,  
  periodico_id INT,  
  autor VARCHAR(255),  
  artículo TEXT,  
  url VARCHAR(255),  
  fecha_publicacion DATETIME,  
  fecha_scraping DATETIME,  
  coeficiente VARCHAR(255),  
  justificacion TEXT,  
  temas VARCHAR(255),  
  claves TEXT,  
  FOREIGN KEY (periodico_id) REFERENCES periodicos(id)  
);
```

```
module.exports = (sequelize, DataTypes) => {  
  ...  
  const Noticia = sequelize.define('Noticia', {  
    id: {  
      type: DataTypes.INTEGER,  
      autoIncrement: true,  
      primaryKey: true  
    },  
    titulo: {  
      type: DataTypes.STRING(255),  
      allowNull: false  
    },  
    subtítulo: {  
      type: DataTypes.TEXT,  
      allowNull: true  
    },  
    autor: {  
      type: DataTypes.STRING(100)  
    },  
    artículo: {  
      type: DataTypes.TEXT  
    },  
    url: {  
      type: DataTypes.STRING(255)  
    },  
    fecha_publicacion: {  
      type: DataTypes.DATE  
    },  
    fecha_scraping: {  
      type: DataTypes.DATE  
    },  
    imagen: {  
      type: DataTypes.TEXT  
    },  
    coeficiente: {  
      type: DataTypes.STRING  
    },  
    justificacion: {  
      type: DataTypes.TEXT  
    },  
    periodico_id: {  
      type: DataTypes.INTEGER,  
      references: {  
        model: 'periodicos',  
        key: 'id'  
      }  
    }  
  }, {  
    tableName: 'noticias',  
    timestamps: false  
  });  
  
  Noticia.associate = function(models) {  
    Noticia.belongsTo(models.Periodico, { foreignKey: 'periodico_id' });  
    Noticia.belongsToMany(models.Tema, { through: 'noticias_temas', foreignKey: 'tema_id' });  
    Noticia.belongsToMany(models.Clave, { through: 'noticias_claves', foreignKey: 'clave_id' });  
    Noticia.belongsToMany(models.Grupo, { through: 'grupo_noticia', foreignKey: 'noticia_id', otherKey: 'grupo_id' });  
  };  
  
  return Noticia;  
};
```

Grupos

```
CREATE TABLE IF NOT EXISTS grupos (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  titular_general TEXT NOT NULL,  
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

```
module.exports = (sequelize, DataTypes) => {  
  ...  
  const Grupo = sequelize.define('Grupo', {  
    id: {  
      type: DataTypes.INTEGER,  
      autoIncrement: true,  
      primaryKey: true  
    },  
    titular_general: {  
      type: DataTypes.TEXT,  
      allowNull: false  
    },  
    fecha_creacion: {  
      type: DataTypes.DATE,  
      defaultValue: DataTypes.NOW  
    }  
  }, {  
    tableName: 'grupos',  
    timestamps: false  
  });  
  
  Grupo.associate = function(models) {  
    Grupo.belongsToMany(models.Noticia, {  
      through: 'grupo_noticia',  
      foreignKey: 'grupo_id',  
      otherKey: 'noticia_id'  
    });  
  };  
  
  return Grupo;  
};
```

Noticias-Temas

```
CREATE TABLE IF NOT EXISTS noticia_temas (  
  noticia_id INT,  
  tema_id INT,  
  PRIMARY KEY (noticia_id, tema_id),  
  FOREIGN KEY (noticia_id) REFERENCES noticias(id),  
  FOREIGN KEY (tema_id) REFERENCES temas(id)  
);
```

Noticias-Claves

```
CREATE TABLE IF NOT EXISTS noticia_claves (  
  noticia_id INT,  
  palabra_id INT,  
  PRIMARY KEY (noticia_id, palabra_id),  
  FOREIGN KEY (noticia_id) REFERENCES noticias(id),  
  FOREIGN KEY (palabra_id) REFERENCES claves(id)  
);
```

Grupos-Noticias

```
CREATE TABLE IF NOT EXISTS grupo_noticia (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  grupo_id INT NOT NULL,  
  noticia_id INT NOT NULL,  
  FOREIGN KEY (grupo_id) REFERENCES grupos(id),  
  FOREIGN KEY (noticia_id) REFERENCES noticias(id)  
);
```

4.5 Diseño

Nuestra aplicación adopta una paleta de colores sofisticada basada en tonos neutros y acentos metálicos para transmitir un diseño minimalista, elegante y premium.

| Elemento | Color | Código HEX | Descripción |
|-------------------|-------------|------------|--|
| Fondo principal | Blanco | #FFFFFF | Base limpia para un diseño aireado. |
| Fondo alternativo | Gris claro | #D9D9D9 | Contraste sutil entre secciones. |
| Texto principal | Gris oscuro | #2C2C2C | Legibilidad óptima en todo tipo de pantalla. |
| Header/Footer | Gris medio | #5A5A5A | Estructura visual sólida y consistente. |
| Elementos CTA | Oro | #FFD700 | Botones, iconos destacados y acciones clave. |

| | | | |
|----------------------|-------|---------|---|
| Detalles secundarios | Plata | #CoCoCo | Líneas divisorias, iconos menores o elementos informativos secundarios. |
|----------------------|-------|---------|---|

Uso Ideal

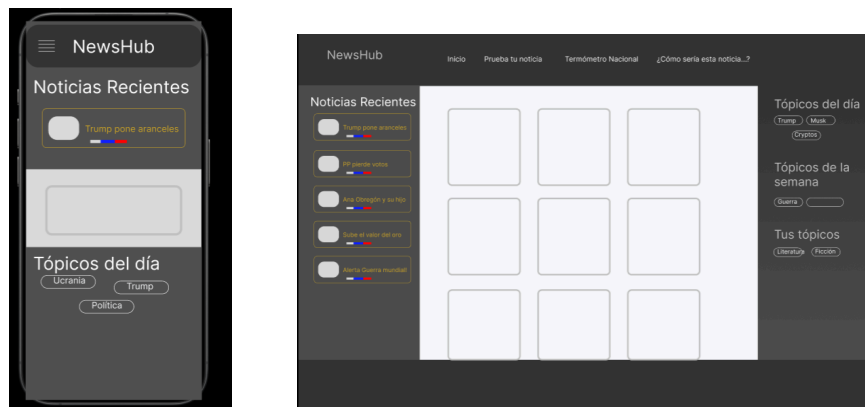
- Interfaces web limpias y sofisticadas.
- Aplicaciones con enfoque visual discreto y profesional.
- Proyectos que busquen transmitir lujo sin sobrecargar la experiencia de usuario.

Aplicación en Pantallas

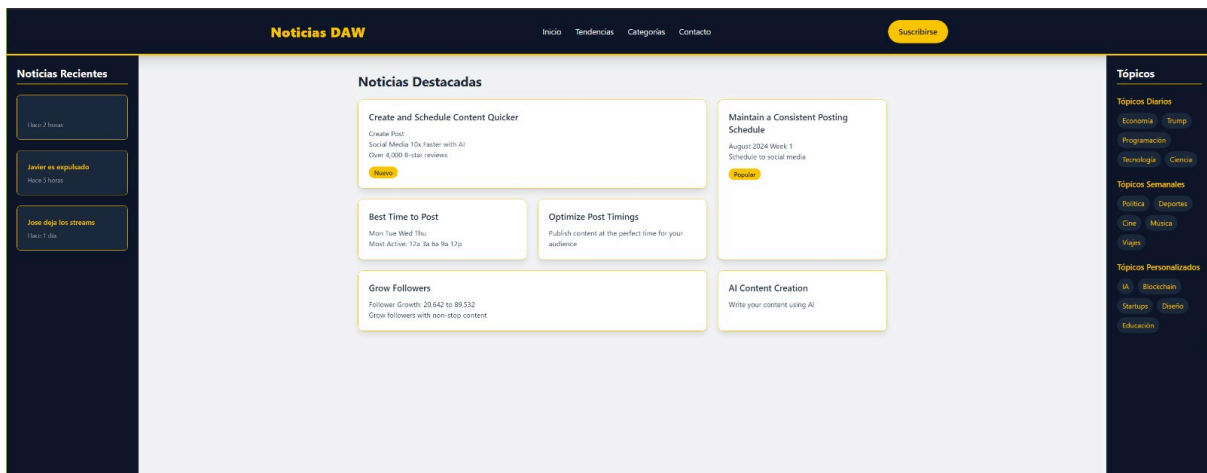
- **Secciones principales:** fondo blanco o gris claro para mantener claridad.
- **Texto:** gris oscuro para asegurar buena lectura.
- **Botones e iconos destacados:** en oro para atraer la atención sin perder elegancia.
- **Elementos secundarios o decorativos:** en plata, aportando jerarquía visual.
- **Header y Footer:** en gris medio para enmarcar el contenido y guiar la navegación.

Pantalla de diseño

Maquetas Figma



Prototipo HTML + CSS



Pantalla de inicio actual

5. Codificación

Stack Tecnológico

Backend (API)

- [Express.js](#)
- **Web scraping y unión de grupos:** Python
- **SGBD:** MySQL
- **ORM:** Sequelize (javascript)

Frontend

- **Framework:** Vue
- **Servidor de desarrollo:** Vite
- **UI Framework:** Tailwind CSS
- **Esencial:** HTML, CSS & JavaScript

Preparación del entorno de proyecto

A continuación, se enseñan los pasos para poder replicar el funcionamiento del escritorio.

Requisitos del sistema

- Node.js y npm
- Python
- Servidor Web (Apache o Nginx)
- Sistema Gestor de Base de Datos: MySQL

1. Actualice la máquina

```
sudo apt update && sudo apt upgrade
```

2. Clone el repositorio

<https://github.com/javizuurc/NewsHub.git>

```
cd NewsHub
```

3. Elimine el fichero package-lock.json (para evitar problemas)

```
sudo rm -rf package-lock.json
```

4. Instale las dependencias de node

```
npm install
```

5. En el directorio /api cree el fichero '.env'

Configure sus variables de entorno siguiendo la estructura que le proporcionamos:

```
OPENAI_API_KEY =
```

```
DB_NAME = database
```

```
DB_USER = root
```

```
DB_PASS = 11092001
```

```
DB_HOST = 127.0.0.1
```

```
DB_DIALECT = mysql
```

```
DB_LOGIN = false
```

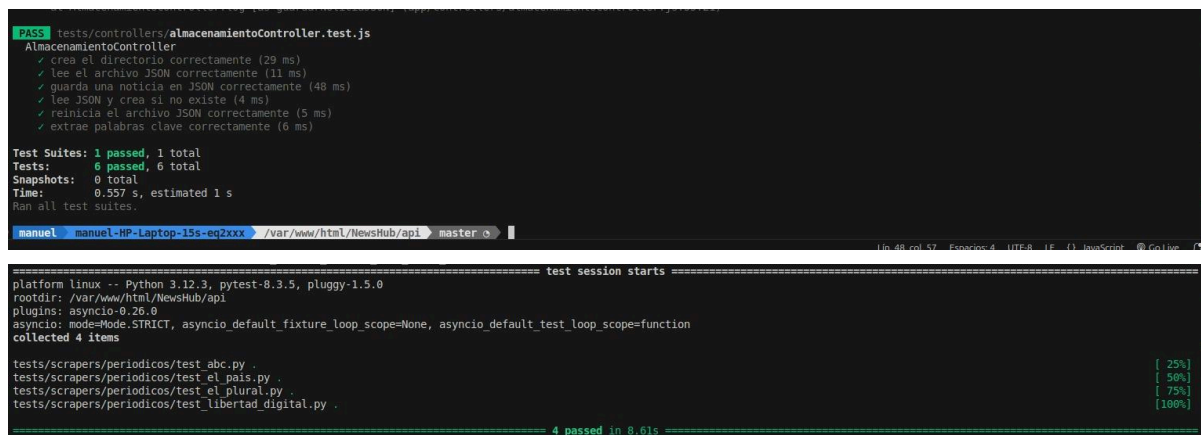
además de otro al mismo nivel del proyecto con este parámetro:

```
VITE_API_URL=http://localhost:3000
```

6. Inicie la API

```
node server.js
```


6. Prueba unitarias



```
tests/controllers/almacenamientoController.test.js
PASS tests/controllers/almacenamientoController.test.js
  AlmacenamientoController
    ✓ crea el directorio correctamente (29 ms)
    ✓ lee el archivo JSON correctamente (11 ms)
    ✓ guarda una noticia en JSON correctamente (48 ms)
    ✓ lee JSON y crea si no existe (4 ms)
    ✓ reinicia el archivo JSON correctamente (5 ms)
    ✓ extrae palabras clave correctamente (6 ms)

Test Suites: 1 passed, 1 total
Tests: 6 passed, 6 total
Snapshots: 0 total
Time: 0.557 s, estimated 1 s
Ran all test suites.

platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0
rootdir: /var/www/html/NewsHub/api
plugins: asyncio-0.26.0
asyncio: mode=Mode.STRICT, asyncio_default_fixture_loop_scope=None, asyncio_default_test_loop_scope=function
collected 4 items

tests/scrapers/periodicos/test_abc.py . [ 25%]
tests/scrapers/periodicos/test_el_pais.py . [ 50%]
tests/scrapers/periodicos/test_el_plural.py . [ 75%]
tests/scrapers/periodicos/test_libertad_digital.py . [100%]

4 passed in 8.61s
```

7. Conclusiones

A lo largo del desarrollo de este proyecto, **NewsHub**, hemos experimentado un importante crecimiento tanto a nivel técnico como profesional. Este trabajo nos ha permitido aplicar de manera estructurada los conocimientos adquiridos durante nuestra etapa cursando el ciclo de Desarrollo de Aplicaciones Web, enfrentándonos a un proceso completo e integral de análisis, diseño, codificación y documentación de una solución web funcional.

Desde el primer momento, nos propusimos ir más allá de cumplir los requisitos académicos, asumiendo el reto de construir y explorar en base a una plataforma que ofreciera una alternativa real a la forma en que los usuarios consumen noticias. El objetivo principal de NewsHub ha sido permitir a los internautas identificar diferentes enfoques sobre una misma noticia. Dando una alternativa sin precedentes en España del consumo de noticias.

A pesar de nuestra inexperiencia, hemos abordado el desarrollo con responsabilidad, curiosidad y una fuerte motivación por aprender. Esto nos ha llevado a tomar decisiones técnicas fundamentadas, tratar de implementar de la mejor forma posible buenas prácticas y buscar siempre la mejor solución posible en cada fase del proyecto, cometiendo errores, pero aprendiendo de ellos.

En este sentido, NewsHub no solo ha sido una herramienta de aprendizaje, sino también una oportunidad para desarrollar una idea con propósito. Pensamos que NewsHub puede convertirse en una iniciativa con una clara utilidad social, ayudando a mejorar la calidad del consumo informativo, especialmente en un contexto donde la veracidad y objetividad de la información son cada vez más cuestionadas.

Esperamos que el esfuerzo, dedicación y entusiasmo depositados en NewsHub queden reflejados en el resultado final, y que este sea el inicio de futuros desarrollos más ambiciosos.

8. Glosario

(Términos técnicos y conceptos clave del proyecto).

A

- **API REST:** Interfaz de comunicación entre el frontend y backend usando HTTP/JSON. En NewsHub, se usan endpoints para procesar noticias.
- **Axios:** Librería JavaScript para hacer peticiones HTTP a APIs (usada en el backend con Node.js).

B

- **BeautifulSoup:** Librería de Python para extraer datos de HTML (empleada en el web scraping de periódicos como El País o ABC).
- **Backend:** Parte del sistema que procesa lógica y datos. En NewsHub, usa Node.js (API) y Python (scraping).

C

- **CORS** (Cross-Origin Resource Sharing): Mecanismo para permitir peticiones entre dominios diferentes (configurado en la API con el middleware cors).

D

- **Dotenv:** Librería para gestionar variables de entorno (ej: claves de API o credenciales de BD).

E

- **Express:** Framework de Node.js para construir la API RESTful de NewsHub.
- **Endpoint:** Ruta específica de una API (ej: /api/noticias/almacenar-bbdd para guardar noticias en la base de datos).

F

- **Frontend:** Parte visual de la aplicación.
- **Fake News:** Noticias falsas, uno de los problemas que NewsHub busca mitigar mediante el contraste de fuentes.

J

- **Jest:** Framework para testing de JavaScript (usado en pruebas de la API).
- **JSON** (JavaScript Object Notation): Formato de intercambio de datos (ej: archivo noticias.json generado por los scrapers).

N

- **Node.js:** Entorno de ejecución para JavaScript en el backend (base de la API de NewsHub).
- **Nodemon:** Herramienta para reiniciar automáticamente el servidor durante el desarrollo.

M

- **MySQL:** Sistema Gestor de Bases de Datos relacionales.

O

- **ORM** (Object-Relational Mapping): Técnica para manipular bases de datos con objetos (en NewsHub, Sequelize para MySQL).

P

- **Python:** Lenguaje usado para el web scraping (scripts en /scrapers/).
- **Prensa Amarilla:** Sinónimo de prensa sensacionalista.

S

- **Scraping:** Extracción automática de datos de páginas web (ej: noticias de El País o Libertad Digital).
- **Sequelize:** ORM para interactuar con la base de datos desde Node.js.
- **Sesgo mediático:** Tendencia ideológica en la cobertura de noticias (enfoque principal de NewsHub).

T

- **Testing:** Proceso de verificación del código. NewsHub usa Jest y SuperTest para pruebas.

W

- **Web Scraping:** Automatización para extraer información específica de webs (en este caso, noticias).