Memoria

Cuarta Entrega Grupo 1A

URL de la Aplicación:

https://lawiki.ecosys.eu/

Tecnologías y Herramientas utilizadas:

- **Proveedor Cloud:** AWS (Amazon Web Services)
- **Lenguajes:**
  - **JavaScript/TypeScript**: Utilizados principalmente para desarrollar componentes en React y Next.js.
  - **CSS**: Usado para la estilización de los componentes y la interfaz de usuario.
- **Frameworks y Bibliotecas:**
  - **React (v18.3.1)**: Utilizado para la construcción de componentes reutilizables y el manejo de la interfaz de usuario reactiva.
  - **Next.js (v15.0.3)**: Framework basado en React, empleado para la renderización del lado del servidor (SSR) y la generación de sitios estáticos (SSG), optimizando así el rendimiento y el SEO.
  - **Bootstrap (v5.3.3)**: Framework CSS utilizado para la construcción rápida de diseños responsivos y componentes estandarizados.
  - **Leaflet (v1.9.4)** y **React-Leaflet (v5.0.0)**: Utilizados para la integración de mapas interactivos en la aplicación.
  - **Cloudinary (v2.5.1)**: Usado para gestionar y optimizar las imágenes cargadas.
  - **Azure**: Usado para las traducciones de artículos.
- **Dependencias y sus Usos**
  - **bootstrap (v5.3.3):**
    - Facilita la creación de diseños responsivos y estéticamente atractivos.
  - **cloudinary (v2.5.1):**
    - Maneja la optimización y transformación de imágenes en la nube.
  - **leaflet (v1.9.4) y react-leaflet (v5.0.0):**
    - Renderiza mapas interactivos en aplicaciones React.
  - **next (v15.0.3):**
    - Aporta herramientas para la generación de páginas y la gestión de rutas con una arquitectura basada en componentes.
  - **react (v18.3.1) y react-dom (v18.3.1):**
    - Son el corazón de la aplicación, permitiendo construir y renderizar la interfaz de usuario de manera eficiente.

Requisitos del caso de estudio

Gestión de imágenes en la nube

- **Requisito:** Las imágenes de las wikis y artículos debían cargarse, optimizarse y transformarse de manera eficiente para mantener la calidad y reducir el tamaño.

- **Solución**: Cloudinary Se empleó para la gestión y optimización de imágenes, permitiendo la transformación dinámica y la carga adaptativa según el dispositivo del usuario.

Mapas Interactivos

- **Requisito:** La aplicación debía incluir mapas interactivos para mostrar ubicaciones específicas, de forma dinámica y personalizada.

- **Solución**: Leaflet y React-Leaflet: Estas bibliotecas se utilizaron para integrar mapas interactivos con capas personalizadas. Su integración con React permitió actualizaciones dinámicas basadas en el estado de la aplicación.

Interfaz de Usuario

- **Requisito:** La aplicación debía adaptarse a distintos tamaños de pantalla y ofrecer una experiencia de usuario coherente y estética.

- **Solución**: Bootstrap**:** Este framework CSS permitió construir rápidamente diseños responsivos con componentes predefinidos, asegurando un diseño limpio y uniforme en diferentes dispositivos.

Traducciones Automáticas

- **Requisito:** Los artículos de la aplicación necesitaban ser traducidos automáticamente a diferentes idiomas.

- **Solución**: Azure: La API de traducción de Azure permitió automatizar la traducción de contenido a varios idiomas, garantizando compatibilidad con una audiencia diversa.

Alojamiento Web

- **Requisito**: La web debe estar disponible de manera online para todo el mundo.
- **Solución**: AWS: Fue la solución como infraestructura en la nube, ya que ofrece servicios escalables (Aunque no lo necesitemos en nuestro proyecto actual)

Entidades de la base de datos:

1.Articles

Propiedades (columnas):

- `id` (clave primaria, generada automáticamente)
- `name` (texto, opcional)
- `short_text` (texto, opcional)
- `text` (texto, opcional)
- `author` (texto, opcional, ID del autor relacionado con `User`)
- `email` (texto, opcional, email del autor)
- `images` (lista de URLs, opcional)
- `googleMaps` (URL o texto, opcional)
- `date` (fecha y hora, opcional)
- `wikiID` (texto, ID relacionado con `Wiki`)
- `versions` (relación con versiones, opcional, lista de versiones asociadas)

Relaciones:

- Muchos a uno con `User` (author).
- Muchos a uno con `Wiki` (wikiID).
- Uno a muchos con `Comment` (los comentarios del artículo).

2. Comments

Propiedades (columnas):

- `id` (clave primaria, generada automáticamente)
- `date` (fecha y hora, opcional)
- `content` (texto, opcional)
- `article_id` (clave foránea, relacionada con `Articles`)
- `author_id` (clave foránea, relacionada con `User`)
- `rating` (flotante, opcional, valoración del comentario)
- `destination_id` (texto, opcional, destino relacionado)

Relaciones:

- Muchos a uno con `Articles` (article_id).
- Muchos a uno con `User` (author_id).

3. Notifications

Propiedades (columnas):

- `id` (clave primaria, generada automáticamente)
- `date` (fecha y hora, opcional)
- `title` (texto, obligatorio)
- `body` (texto, opcional)
- `opened` (booleano, indica si fue abierta)
- `user_id` (clave foránea, relacionada con `User`)

Relaciones:

- Muchos a uno con `User` (user_id).

4. Users

Propiedades (columnas):

- `id` (clave primaria, generada automáticamente)
- `name` (texto, obligatorio)
- `gmail` (email, único y obligatorio)
- `googleID` (texto, único y obligatorio)
- `rating` (flotante, obligatorio, valoración promedio del usuario)
- `nComentarios` (entero, obligatorio, número de comentarios realizados)
- `level` (texto, valores posibles: "redactor" o "admin")

Relaciones:

- Uno a muchos con `Articles` (autor).
- Uno a muchos con `Comments` (autor).
- Uno a muchos con `Notifications` (receptor).

5. Wikis

Propiedades (columnas):

- `id` (clave primaria, generada automáticamente)
- `name` (texto, opcional)
- `description` (texto, opcional)
- `author` (texto, opcional, ID relacionado con `User`)
- `bg_image` (URL o texto, opcional)
- `logo` (URL o texto, opcional)

Relaciones:

- Muchos a uno con `User` (author).
- Uno a muchos con `Articles` (relación inversa a `wikiID` en `Articles`).

CREDENCIALES PARA ACCEDER A LA BASE DE DATOS:

USERNAME**: wikiAdmin**
PASSWORD: **_P5JP53hT8**
DATABASE_URL:
cluster0.w2vxl.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
DATABASE_NAME: **laWiki2**

Instrucciones de Instalación y Despliegue

Para poder usar la aplicación debemos seguir los siguientes pasos:

Despliegue local en Windows

Abrir Power shell en el directorio ***/laWiki/app*** y ejecutar ***.\run_app.ps1***. Este se despliega aplicación con ajustes apropiados en el ***http://localhost:3000***

Despliegue automática en la Nube AWS

Hacer push en la rama main del repositorio **https://github.com/illyaro/laWiki.git**
Este se ejecutará un script de despliegue automático. (ver fichero ***/laWiki/.github/workflows/deployAWS.yml***)

Despliegue manual en la nube AWS

1. Es aconsejable agregar memoría SWAP a la EC2 si el contenedor tiene poca memoria ram (< 4GB) para prevenir fallos durante la instalación de las dependencias de node
    a. **fallocate -l 4G /swapfile**
    chmod 600 /swapfile
    mkswap /swapfile
    swapon /swapfile
    swapon --show
    free -h
    echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
2. Crear entorno virtual de python con el nombre **venv** en la runa **~/venv**
3. Instalar versión v22.11.0 del node
    a. curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
    b. nvm install v22.11.0
4. Instalar y configurar nginx
    a. sudo apt-get install nginx
    b. crear fichero en ***/etc/nginx/sites-enabled/laWiki*** con contenido
    server {

server_name <tu_ip> <tu_nombre_del_dominio>;
 # Front end
location / {
proxy_pass http://127.0.0.1:3000;
}

 # Wiki microservice v1
location /api/v1/wikis {
proxy_pass http://127.0.0.1:13000/api/v1/wikis;
}

 # Wiki microservice v2
location /api/v2/wikis {
proxy_pass http://127.0.0.1:13000/api/v2/wikis;
}

 # Article microservice
location /api/v1/articles {
proxy_pass http://127.0.0.1:13001/api/v1/articles;
}

 # Translations
location /api/v1/translate {
proxy_pass http://127.0.0.1:13001/api/v1/translate;
}

```
        # Comments microservice
        location /api/v1/comments {
        proxy_pass http://127.0.0.1:13002/api/v1/comments;
        }

        # Notifications microservice
        location /api/v1/notifications {
        proxy_pass http://127.0.0.1:13003/api/v1/notifications;
        }

        # Users microservice
        location /api/v1/users {
        proxy_pass http://127.0.0.1:13004/api/v1/users;
        }
}
```

   c. configurar el fichero **/etc/nginx/nginx.conf** agregando en el objeto **http {}** linea de código **client_max_body_size 20M;**
   d. reinicializar nginx con
      sudo systemctl stop nginx
      sudo systemctl start nginx
2. Modificar el fischer **/etc/hosts** agregando linea 127.0.0.1 <tu_nombre_del_dominio>
3. Instalar certificado ssl con Certbot. sigue instrucciones detalladas en https://certbot.eff.org/instructions?ws=nginx&os=pip
   a. sudo apt update
      sudo apt install python3 python3-venv libaugeas0
   b. sudo python3 -m venv /opt/certbot/
      sudo /opt/certbot/bin/pip install --upgrade pip
   c. sudo /opt/certbot/bin/pip install certbot certbot-nginx
   d. sudo ln -s /opt/certbot/bin/certbot /usr/bin/certbot
   e. sudo certbot --nginx
   f. verificar que el contenido del fichero /etc/nginx/sites-enabled/laWiki es similar al proporcionado con el código de fuente **/laWiki/etc_nginx_sites-enabled_laWiki**
4. Reinicializar de nuevo nginx
   sudo systemctl stop nginx
   sudo systemctl start nginx
5. Copiar el repositorio (o la carpeta /laWiki/app) en EC2
6. Cambiar los enlaces internos asegurando que estas usando el dominio registrado en el documento **/laWiki/app/front_end/.dev.production**, por defecto es https://lawiki.ecosys.eu.
   En caso de que nombre del usuario del sistema es distinto del ‘ubuntu’, corregir la runa del entorno dentro del script **/laWiki/app/aws_start.sh**
7. Arrancar el servidor
   Entrar en el directorio /laWiki/app y ejecutar en la consola estándar de linux
   source ./aws_start.sh

La página estará disponible en 1-2 minutos accediendo por URL del nombre del dominio.

8. Parar el servidor
   Entrar en el directorio /laWiki/app y ejecutar en la consola estándar de linux
   source ./aws_stop.sh

Descripción de la API REST

Hemos usado FASTAPI para desarrollar los endpoints de la API REST. Aquí se muestran los endpoints de cada elemento.

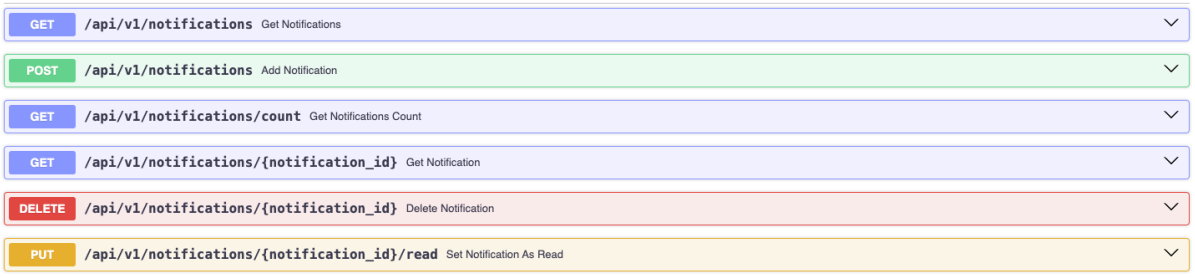
Articles:

Respecto a la última entrega se han añadido los endpoints para la traducción.

Wikis:

Comments

Notifications:



Users:

Esta parte es completamente nueva, por lo que los endpoints son todos nuevos.