MEMORIA - PRÁCTICA 3

Grupo: 3B

Integrantes:

Cristian Ruiz Martín

Pablo Rubia Arias

Mikolaj Zabski

Ignacy Borzestowski

Olivier Gabana Gómez

José Francisco Ruiz Sierras

Tecnologías usadas

Cloudinary: es el servicio de gestión de imágenes en la nube. Se seleccionó por sus capacidades y la facilidad de integración mediante su API, permitiendo almacenar y acceder a los archivos multimedia de forma segura y escalable.

Node.js: es un entorno de ejecución de JavaScript de código abierto y multiplataforma, diseñado para crear aplicaciones del lado del servidor y de red de alto rendimiento.

React: es una biblioteca de JavaScript de código abierto desarrollada por Facebook, diseñada para construir interfaces de usuario (UI) de manera eficiente y escalable.

Nominatim: es un servicio de geocodificación abierto y gratuito vinculado a OpenStreetMap (OSM). Su propósito principal es convertir direcciones físicas (como "123 Calle Principal, Ciudad X") en coordenadas geográficas (latitud y longitud) y viceversa.

Replanteamiento de decisiones anteriores

En el backend han sido necesarias varias modificaciones:

- Hemos cambiado todas las rutas que acceden a objetos singulares:
    - antes el id se proporcionaba como parámetro de consulta (/entrada?id=id_string)
    - ahora se proporciona como un parámetro URL (/entry/id_string)
- Combinamos los gestores de búsqueda parametrizados en uno solo, recibiendo las peticiones en "/search". De esta forma, el usuario puede buscar un objeto proporcionando varios parámetros de búsqueda a la vez.
- En lugar de depender de un único parámetro para filtrar los objetos por fecha (createdAt), dividimos el parámetro en dos para que sea posible buscar todos los objetos creados en el periodo comprendido entre las dos fechas proporcionadas.
- Cambiamos el objeto «Versión» para que sea posible asignar ids de objetos «Media». Es decir, ahora se pueden asignar imágenes a una versión para que se muestren en la parte del frontend.
- Nuevos atributos para las rutas de los servicios externos (Cloudinary & Nominatim).

Estructura del proyecto: Frontend

```
├── deno.lock
├── docker-compose.yml
├── Dockerfile
├── eslint.config.js
├── index.html
├── package.json
├── package-lock.json
├── public
│   └── vite.svg
├── src
│   ├── api
│   │   ├── Api.js
│   │   ├── CommentApi.js
│   │   ├── EntryApi.js
│   │   ├── MediaApi.js
│   │   ├── VersionApi.js
│   │   └── WikiApi.js
│   ├── App.jsx
│   ├── components
│   │   ├── Comentario.jsx
│   │   ├── ConfirmationModal.jsx
│   │   ├── EntradaCard.jsx
│   │   ├── Footer.jsx
│   │   ├── Header.jsx
│   │   ├── SearchResultsList.jsx
│   │   ├── ToastMessage.jsx
│   │   ├── VersionCard.jsx
│   │   ├── Version.jsx
│   │   └── WikiCard.jsx
│   ├── context
│   │   ├── ToastContext.jsx
│   │   └── ToastProvider.jsx
│   ├── layout
│   │   └── MainLayout.jsx
│   ├── main.jsx
│   ├── pages
│   │   ├── AdvancedSearchPage.jsx
│   │   ├── EntradaPage.jsx
│   │   ├── FormVersionPage.jsx
│   │   ├── FormWikiPage.jsx
│   │   ├── HomePage.jsx
│   │   ├── VersionPage.jsx
│   │   └── WikiPage.jsx
│   └── styles
│           └── theme.js
└── vite.config.js
```

Requisitos principales

- Creación, edición y mantenimiento de wikis.
- Añadir, modificar, corregir y eliminar entradas de una wiki.
- Visualización de imágenes y mapas.
- Gestión de versiones que permite volver a cualquier modificación pasada.
- Búsqueda de entrada por:
    - Título

    - Autor
    - Fecha de Creación
    - Wiki

Aparte, nos hemos encargado de incluir búsquedas adicionales sobre:

- Búsqueda de wikis por:
    - Título
    - Descripción
    - Categoría
    - Fecha de Creación
- Búsqueda de versión por:
    - Contenido
    - Editor
    - Fecha de Creación
- Búsqueda de comentario por:
    - Contenido
    - Autor
    - Fecha de Creación
    - Calificación

Las búsquedas de texto (título, descripción, etc.) son parciales, es decir, el resultado de la búsqueda no va a mostrar únicamente el texto buscado, si no todo lo que lo contenga.

Base de datos

La base de datos se encuentra alojada en Atlas, la URI mediante la cual accede el backend es la siguiente: "mongodb+srv://admin:8fdCAkfmVQCRDRfK@cluster0.rfz8f.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0".

Es necesario especificar que el nombre de la base de datos, DB_NAME = “laWikiCopia”. Si se quiere ver la base de datos a través de Compass la URL es la siguiente: “mongodb+srv://admin:8fdCAkfmVQCRDRfK@cluster0.rfz8f.mongodb.net/” Los configs ya vienen configurados en el archivo comprimido con codigo fuente.

Instrucciones de instalación/despliegue

Para lanzar la aplicación completa basta con ejecutar

docker compose up --build

en la raíz del directorio con el código fuente. Esta carpeta debe contener un archivo “docker-compose.yml”.

Funcionalidad del frontend

Se ha usado Cloudinary para almacenamiento de los datos multimedia (imágenes). Se accede a las imágenes mediante el parámetro que poseen las entradas media_ids, que nos devuelve todos los ids de las imágenes que están relacionadas a una entrada (versión) dada. También se ha añadido imágenes a las wikis.

En cuanto al servicio externo nominatim, hemos empleado llamadas directas a la API con el parámetro address para especificar el lugar que queremos visualizar en mapas. Para minimizar las llamadas a la API se ha optado por una solución con caché que almacena los datos de coordenadas en almacenamiento de sesión.