**Proyecto ETL: Búsqueda en Bases de Datos**

Descripción General

Este proyecto implementa un pipeline ETL (Extracción, Transformación y Carga) diseñado para facilitar la búsqueda eficiente en bases de datos. A través de un flujo de trabajo automatizado, se extraen datos de una fuente, se limpian y transforman, y se cargan en una base de datos estructurada para permitir consultas optimizadas.

El propósito es construir un sistema que permita realizar búsquedas rápidas y precisas sobre grandes volúmenes de información, apoyando tareas como análisis exploratorio, generación de reportes y sistemas de recomendación.

Este proyecto está desarrollado con **Nuxt.js**, un framework basado en **Vue.js** que permite crear aplicaciones web universales con renderizado del lado del cliente y del servidor. El enfoque modular de Nuxt simplifica la creación de páginas, gestión de estado, peticiones al servidor y componentes reutilizables.

Objetivo del Proyecto

- Construir un pipeline ETL modular y mantenible.

- Organizar y transformar los datos para facilitar su indexación y búsqueda.

- Implementar estrategias para mejorar el rendimiento de consultas en bases de datos.

- Brindar soporte para búsquedas por palabras clave, coincidencias parciales y filtros personalizados.

Tema Central: Búsqueda en Bases de Datos

¿Qué es la búsqueda en bases de datos?

Es el proceso de consultar registros de una base de datos mediante criterios definidos (palabras clave, filtros, rangos, patrones, etc.). Es fundamental para sistemas de información, portales web, e-commerce, y análisis de datos.

Tipos comunes de búsqueda:

- Exacta: Busca coincidencias exactas en campos (e.g. nombre = "Juan").

- Parcial o difusa: Usa operadores como LIKE o algoritmos como Levenshtein.

- Full-Text Search: Usa índices especiales para búsquedas de texto completas.

- Por múltiples filtros: Combina condiciones como fecha, categoría, estado, etc.

🧰 Requisitos del Sistema

- Python 3.8 o superior

- Docker (opcional, pero recomendado)

- pip (administrador de paquetes de Python)

Estructura del Proyecto

tap\_pa\_bebd-main/

├── README.md ← Documentación completa

├── requirements.txt ← Dependencias de Python

├── docker-compose.yaml ← Orquestación con Docker

├── Dockerfile ← Imagen base para el contenedor

├── notebooks/

│ └── etl\_pipeline.ipynb ← Notebook demostrativo del ETL

└── src/

├── \_\_init\_\_.py

├── config.py ← Configuración general

├── data\_ingestion.py ← Extracción desde fuente (CSV, API, etc.)

├── data\_transformation.py ← Limpieza y normalización

└── data\_loading.py ← Carga en destino (base de datos o archivo)

Cómo se Aplica la Búsqueda en el Proyecto

1. Stores (Estado Global) – /stores

El proyecto probablemente utiliza Pinia como gestor de estado. Los archivos en stores/ definen stores que pueden ser accedidos desde cualquier componente o página.

2. Server (API Interna) – /server/api

En Nuxt 3, puedes definir endpoints directamente en /server/api, como si estuvieras creando un backend.

3. Components – /components

Son los bloques visuales reutilizables. Se usan dentro de pages/ o incluso en otros componentes.

pages/index.vue → muestra la vista principal y carga datos desde el backend.

Guarda esos datos en un store de Pinia, accesible desde cualquier parte.

Los components/ consumen los datos del store o se les pasan como props.

Scripts de Ejecución

bash

Copy

Edit

# Instalar dependencias

npm install

# Ejecutar en modo desarrollo

npm run dev

# Build para producción

npm run build

npm run start

Tecnologías Usadas

Nuxt.js 3

Vue 3 (Composition API)

Pinia (store)

TypeScript

Autores

Jander ensuncho perez

Juan fernando Miranda fabra

Yeison morelo chavez

Miguel gomez Caceres