Descargar las dependencias para el funcionamiento de dotnet

Despues de descargar el repositorio:

Se ingresa en la raiz Backend para ejecutar los comendos para la descarga de las dependencias necesarias para el funcionamiento del proyecto,(la lista de dependencias se pueden encontrar en el archivo “dependencies.txt” en la misma carpeta Backend)

- dotnet add package Pomelo.EntityFrameworkCore.MySql --version 8.0.6
- dotnet add package Microsoft.EntityFrameworkCore.Design --version 8.0.6
- dotnet add package Microsoft.EntityFrameworkCore.Tools --version 8.0.6
- dotnet add package System.Net.Http.Json --version 9.0.0
-

Para verificar el tener instalados estas dependencias se puede usar el comando:

- Dotnet list package

PRUEBA TÉCNICA - GODOY CÓRDOBA
DOCUMENTACIÓN TÉCNICA: APLICACIÓN CATFACTS + GIPHY

1. INTRODUCCIÓN

El presente documento busca explicar las funcionalidades desarrolladas en el actual proyecto, el cómo se utilizan y se aplican para dar solución a los problemas y requerimientos planteados.

Resumen

El proyeccto desarrollado se divide principalmente en 2(dos) áreas principales, Frontend y Backend donde el primero se destina al ambiente grafico o interfaz con el que interactúa el usuario para visualizar los requerimientos:

- Visualizar el “fact” o “dato” de los gatos obtenido del API : catfact.ninja/fact
- Presentar una imagen teniendo como paramento 3 palabras del texto anterior y una llave suministrada en el documento
- Visualizar un historial de los datos anteriores, la información que se presentaba era:
    - Fecha
    - Texto completo
    - Las 3(tres) palabras usadas
    - URL resultante
- Poder cambiar entre las pantallas de historial e imágenes/Texto

En el área del Backend se crearon Endpoints que harían los procesos de interactuar con las API y desarrollar las actividades establecidas. También se agregaron unas funcionalidades para el mejor funcionamiento del programa, entre ellos está la paginación del historial (límite de entradas que se muestran) para mejorar la organización cuando halla mayor cantidad de datos y una opción para eliminar valores del historial, esto para dar más oportunidad del usuario para interactuar con los datos.

BACKEND

Desarrollado en framework .NET, se crearon diferentes carpetas para organizar las funcionalidades:

- Controllers : crear los endpoints responsables de manejar la lógica de las API's.
- Migrations: gestiona estructura de la Base de datos a usar en el proyecto.
- Models: para crear una estructura de datos para implementarlo en la base de datos.

Endpoints Controllers:

- Get api/fact: Obtiene un dato aleatorio sobre gatos desde una API externa, este dato es devuelto al frontend con la información dentro de Body. En caso de error da codigo 500.
- Post api/Gif: Busca un gif usando las 3 palabras que recibe por medio el body , el el body recibe toda la sentencia además de las 3 palabras usadas, estos valores son guardados en la base de datos usando el modelo SearchHistory.cs donde se guarda en la estructura y se inserta en la base de datos, el valor que devuelve al frontend es la url resultante.
- Get api/history: Obtiene el historial de las últimas 20 búsquedas realizadas, ordenadas por fecha descendente.
- Delete History/delete: Elimina un registro específico del historial de búsquedas. Al recibir el id único del fact y usar este mismo para eliminarlo en la base de datos y la pantalla del historial

FRONTEND

Desarrollado en framework REACT para una interfaz reactiva y uso de CSS para mejoras graficas.

Funciones:

- FetchCatFact: buscar el echo de los gatos, envía la solicitud al backend y al recibir la respuesta se guarda el valor de la sentencia y luego se guardan y envían las 3 primeras palabras del echo para buscar el gif.
- FetchGif: buscar el gif, es invocado en la función FetchCatFact utiliza la sentencia entera y las tres palabras, como respuesta define un URL que es enseñado en el HTML en la etiqueta src.
- FetchHistory: guarda la respuesta del historial, estas respuestas son publicadas en la pestaña history usando .map para recorrer todo el arreglo y publicar 1 a la vez.

- Eliminar: usando la lógica anterior de publicación del historial a cada entrada publicada se le asigna un id único el cual es usado para eliminar el dato especifico en la base de datos.
- HandleRefreshGif: vuelve a buscar un GIF para el mismo dato de gatos actual.
- HandleRefreshAll: obtiene un nuevo dato de gatos y su GIF correspondiente.

EL HTML divide en 2 páginas alternando entre ellas, dependiendo el estado de de la variable “activeTab” si es “current” muestra el texto y la imagen y “History ” muestra el historial de las entradas regitradas

Diagrama de funcionamiento