Reporte del Proyecto

Base de Datos

- **Archivo:** comunicacion_contextual.py
  Propósito: Este código se encarga de interactuar con una base de datos en Supabase para cargar el historial de conversaciones de un usuario, verificar si el bot ya ha saludado al usuario en el historial y determinar si el bot debe saludar al usuario o preguntar si tiene alguna duda. Se utilizan funciones para cargar el historial, verificar si ya se ha saludado, obtener la hora de la última interacción y determinar si el bot debe realizar ciertas
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: datetime, supabase, dotenv
- **Archivo:** config.py
  Propósito: Este código tiene como propósito principal la gestión de configuraciones de un bot, utilizando Supabase como base de datos. - Primero, se cargan las variables de entorno necesarias para la conexión con Supabase. - Se define un decorador `login_requerido` que verifica si un usuario está autenticado antes de ejecutar una función. - Se implementan funciones para cargar y guardar configuraciones en la tabla "configuracion_bot" de Supabase
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: datetime, twilio.rest, flask, dotenv, supabase, functools
- **Archivo:** config_helper.py
  Propósito: El código proporciona funciones para cargar y guardar configuraciones en una base de datos de Supabase. La función `cargar_configuracion(nombre_nora)` recupera la configuración asociada a un nombre específico desde la tabla "configuracion_bot" en Supabase. Si no se encuentra la configuración, se devuelve un valor predeterminado. La función `guardar_configuracion(nombre_nora, data)` guarda la configuración proporcionada en la base de datos de Sup
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: supabase, dotenv
- **Archivo:** debug_integracion.py
  Propósito: Este código tiene como propósito revisar y verificar la configuración de un sistema en una base de datos Supabase, así como también revisar la configuración de algunas variables de entorno. Para lograr esto, se definen las tablas esenciales en Supabase y se verifica si contienen datos. Luego, se revisa la configuración de la tabla 'settings' para determinar si ciertas opciones están habilitadas o deshabilitadas. Finalmente
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: twilio.rest, supabase, dotenv, clientes.aura.handlers.handle_keywords, clientes.aura.handlers.handle_ai
- **Archivo:** error_logger.py
  Propósito: Este código tiene como propósito registrar errores en una base de datos en Supabase. Primero, se configura la conexión con Supabase utilizando las credenciales almacenadas en variables de entorno. Luego, se define la función `registrar_error(origen, mensaje_error)` que recibe como parámetros el origen del error y una descripción del

mismo. Dentro de la función, se crea un diccionario con la información del error y se intenta insert
Rutas: Ninguna
Archivos CSS: Ninguno
Archivos HTML: Ninguno
Tablas y columnas:
Archivos .py usados: datetime, supabase, dotenv

- **Archivo:** history.py
  Propósito: Este código tiene como propósito principal guardar un mensaje en la tabla `historial_conversaciones` de una base de datos en Supabase. Para ello, se define una función llamada `guardar_en_historial` que recibe como parámetros el número del remitente o destinatario, el contenido del mensaje, el origen del mensaje (ya sea 'usuario' o 'bot') y el nombre del asistente Nora AI. El mensaje se guarda con información
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: datetime, supabase, dotenv
- **Archivo:** memoria.py
  Propósito: Este código se encarga de interactuar con una base de datos Supabase para almacenar y recuperar información de memoria de usuarios. Las funciones `obtener_memoria(usuario_id)` y `guardar_memoria(usuario_id, data)` se encargan de buscar y almacenar respectivamente la memoria asociada a un usuario en la tabla "memoria_usuarios" de Supabase. La función `obtener_memoria(usuario_id)` devuelve un diccionario con la memoria
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: supabase, dotenv
- **Archivo:** settings_loader.py
  Propósito: Este código tiene como propósito cargar configuraciones desde la tabla `configuracion` en Supabase. Primero se configura la conexión a Supabase utilizando las credenciales obtenidas de variables de entorno. Luego, la función `cargar_settings` busca y devuelve las configuraciones específicas para un nombre dado (por defecto "aura"). Si no se encuentran configuraciones para ese nombre, se retornan valores predeterminados. En caso de algún error durante la
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: supabase, dotenv
- **Archivo:** startup_check.py
  Propósito: Este código tiene como propósito realizar una serie de verificaciones al iniciar la aplicación de Nora AI. En primer lugar, se carga la configuración de Supabase a partir de variables de entorno. Luego, se definen funciones para verificar la existencia de ciertas tablas en Supabase y para asegurarse de que las variables de entorno necesarias estén configuradas. Finalmente, la función `inicializar_nora()` inicia la verificación del ent
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: supabase, dotenv
- **Archivo:** twilio_sender.py
  Propósito: Este código tiene como propósito enviar mensajes a través de Twilio y registrar los envíos en una tabla llamada `twilio_logs` en Supabase. Para lograr esto, se configuran las credenciales de Twilio y Supabase a partir de variables de entorno. Además, se define una función `registrar_envio` que registra los mensajes enviados en Supabase

y una función `enviar_mensaje` que envía mensajes a través de
Rutas: Ninguna
Archivos CSS: Ninguno
Archivos HTML: Ninguno
Tablas y columnas:
Archivos .py usados: twilio.rest, dotenv, datetime, supabase, clientes.aura.utils.error_logger

Etiquetas

- **Archivo:** contactos.py
  Propósito: Este código tiene como propósito interactuar con una base de datos a través de Supabase para gestionar los datos de contactos. - La función `obtener_datos_contacto(numero)` busca un contacto en la base de datos a partir de un número de teléfono normalizado y devuelve sus datos si existe, o inicializa datos predeterminados si no se encuentra. - La función `actualizar_datos_contacto(numero, nombre, foto_perfil, ia_activada, etiquetas
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: datetime, supabase, dotenv, utils.normalizador
- **Archivo:** historial.py
  Propósito: Este código tiene como propósito principal guardar mensajes de conversaciones en un historial y actualizar la información de contacto en una base de datos en Supabase. La función `guardar_en_historial` recibe como parámetros el remitente, el mensaje, el tipo de mensaje (por defecto "recibido"), el nombre, si la inteligencia artificial está activada (por defecto True) y una lista de etiquetas. Normaliza el número de tel
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: datetime, supabase, dotenv, clientes.aura.utils.normalizador

Otros

- **Archivo:** google_sheets.py
  Propósito: Este código tiene como propósito autenticarse con una cuenta de servicio de Google y obtener un servicio para trabajar con Google Sheets. Para lograrlo, se definen las variables de alcance necesario y se configuran el ID de la hoja de cálculo y el rango a leer o escribir. La función `get_gsheet_service()` valida que las variables de entorno requeridas estén configuradas y luego crea las credenciales necesarias a partir de esas variables
  Rutas: Ninguna
  Archivos CSS: Ninguno
  Archivos HTML: Ninguno
  Tablas y columnas:
  Archivos .py usados: googleapiclient.discovery, google.auth.transport.requests, google.oauth2.service_account