**Universidad Mariano Gálvez Guatemala, Campus**

**Jutiapa**

**Facultad de Ingeniería en Sistemas**

# Curso: Programación 1



**Catedrático: Ing. Ruldin Ayala**

**Nombre del estudiante: Lester David Payes Méndez**

# Carné: 0905-24.22750

**Proyecto EmDijes1**

**Descripción y Diseño**

**EmDijes1** es una aplicación de escritorio desarrollada en C# (.NET 8, Windows Forms) cuyo objetivo es brindar acompañamiento espiritual y mental personalizado a los usuarios. El sistema detecta la emoción del usuario mediante un escaneo facial utilizando AWS Rekognition, realiza preguntas adaptadas a esa emoción, genera un resumen espiritual profundo usando la API de OpenAI (modelo gpt-4o o gpt-4.5-turbo) y almacena todo el proceso en una base de datos local. Además, sugiere canciones y videos de YouTube relacionados con la emoción detectada.

El diseño sigue el principio de **separación de responsabilidades**:

* Los datos y entidades están en Models.
* La lógica de negocio y acceso a servicios externos en Services.
* La interfaz gráfica en Forms.
* La configuración centralizada en archivos y clases específicas.
* Esta estructura facilita el mantenimiento, la escalabilidad y la comprensión del código, permitiendo futuras integraciones o migraciones a otros entornos.

**APIs y Servicios Utilizados**

* **AWS Rekognition**:  
  Se utiliza para el análisis facial y la detección de emociones. Al iniciar la aplicación, se toma una imagen del usuario (por webcam o archivo) y se envía a AWS Rekognition, que devuelve la emoción predominante (feliz, triste, enojado, sorprendido, disgustado, etc.).  
  Esto permite personalizar la experiencia desde el primer momento.
* **OpenAI API (gpt-4-turbo)**:  
  Se emplea para generar reflexiones, consejos, oraciones y respuestas empáticas personalizadas.  
  Se eligió este modelo porque, tras pruebas, ofrece respuestas profundas, empáticas y de alta calidad, muy cercanas a GPT-4, pero con mayor velocidad y menor coste.  
  Los prompts enviados a la IA han sido cuidadosamente diseñados para obtener resultados útiles y personalizados.
* **.NET 8 / C# 12**:  
  Plataforma y lenguaje principal, aprovechando las últimas características del lenguaje y el framework.
* **No se usa API de YouTube**:  
  Los videos se muestran embebidos usando la URL pública en un control web.

No se usa porque como no quiero obtener nombre de un video su miniatura y así según URL ni nada de eso por el estilo, si no que solo quiero un video según la emoción se usa lo que ya explique anterior mente para por nombre de canto se busque el video o ya sea por medio de URL completos por eso en la app hay dos botones para estos casos.

**Estructura del Proyecto**

**Carpeta Models**

Contiene las clases que representan los datos principales de la aplicación:

* **RespuestasUsuario**  
  Clase que almacena las respuestas del usuario a las preguntas personalizadas.  
  Propiedades:
  + EmocionDetectada: emoción detectada por AWS Rekognition.
  + Respuesta1 a Respuesta7: respuestas a cada pregunta.
* **ResumenUsuario**  
  Representa el resumen espiritual generado por la IA.  
  Propiedades:
  + FechaRegistro: fecha y hora de la sesión.
  + Emocion: emoción detectada.
  + Versiculo: versículo bíblico sugerido.
  + Reflexion: reflexión espiritual personalizada.
  + Consejo: consejo cristiano adaptado.
  + Oracion: oración personalizada.
  + Canciones: lista de canciones recomendadas.
* **OpenAISettings**  
  Almacena la configuración necesaria para conectarse a la API de OpenAI.  
  Propiedades:
  + ApiKey: clave de acceso a OpenAI.
  + Model: modelo a utilizar (por ejemplo, gpt-4o).
* **AWSSettings**  
  Almacena la configuración para los servicios de AWS.  
  Propiedades:
  + AccessKey: clave de acceso AWS.
  + SecretKey: clave secreta AWS.
  + Region: región de AWS.

**Carpeta Services**

Contiene la lógica de negocio y los servicios que interactúan con APIs externas o la base de datos:

* **ServicioOpenAI**  
  Se encarga de comunicarse con la API de OpenAI para obtener respuestas y resúmenes personalizados.  
  Métodos:
  + ObtenerRespuestaAsync(prompt): envía un prompt y recibe la respuesta de la IA.
* **AWSRekognitionService**  
  Encapsula la lógica para enviar imágenes a AWS Rekognition y obtener la emoción detectada.  
  Métodos:
  + DetectarEmocionAsync(imagen): analiza la imagen y devuelve la emoción predominante.
* **DatabaseService**  
  Gestiona la conexión y operaciones CRUD con la base de datos local.  
  Métodos:
  + GuardarResumen(ResumenUsuario resumen): guarda el resumen generado.
  + GuardarRespuestasUsuario(RespuestasUsuario respuestas): guarda las respuestas individuales.
  + ObtenerConnectionString(): obtiene la cadena de conexión desde la configuración.
* **ResumenUsuarioService**  
  Proporciona métodos para consultar y guardar resúmenes en la base de datos.  
  Métodos:
  + ObtenerTodos(): obtiene todos los resúmenes.
  + ObtenerPorFecha(fecha): filtra por fecha.
  + ObtenerPorId(id): busca por identificador.

**Carpeta Forms**

Contiene todos los formularios (ventanas) de la interfaz gráfica de usuario (UI):

* **FormularioPreguntas**  
  Muestra preguntas personalizadas según la emoción detectada, recoge respuestas y muestra apoyo de la IA tras cada respuesta.  
  Permite avanzar pregunta por pregunta y almacena las respuestas.
* **FormularioResumen**  
  Muestra el resumen espiritual generado por la IA, permite ver videos de YouTube sugeridos y guardar el resumen en la base de datos.  
  El usuario puede interactuar con los enlaces de canciones y videos(web browser).
* **HistorialForm**  
  Permite consultar el historial de resúmenes guardados, filtrando por fecha o emoción.  
  Muestra los datos en un DataGridView y permite revisar sesiones anteriores.

**Configuración**

* **Configuracion.cs**  
  Clase para leer la configuración desde el archivo appsettings.json.  
  Permite centralizar y proteger claves y parámetros sensibles.
* **appsettings.json**  
  Archivo de configuración donde se guardan las claves y parámetros necesarios para la aplicación, como la API Key de OpenAI, las claves de AWS y la cadena de conexión a la base de datos.
* **.gitignore** sirve para indicar a Git qué archivos y carpetas no deben incluirse en el control de versiones por ejemplo en mi proyecto es para no subir las Apis ni el servidor sql etc**.**