PROCESO GESTIÓN DEL TALENTO HUMANO

FORMATO INFORME MENSUAL EJECUCIÓN CONTRACTUAL

Riohacha,04/2022

Señor (a)
EUFEMIA MARGARITA PACHECO MAYA
SUPERVISOR(A) CONTRATO No. **CO1.PCCNTR.3464769**
COORDINADORA ACADEMICA
C.I.E.A
Riohacha

Asunto: Informe mensual de ejecución contractual
Mes 04 del año 2022

Referencia: No CO1.PCCNTR.3464769 del año 2022

ALFONSO JOSE PATERNINA DIAZ, identificado con la cédula de ciudadanía No.78.715.466 de,Monteria en mi calidad de Contratista del SENA, en Centro Industrial Y De Energias Alternativas, en cumplimiento del Contrato de Prestación de Servicios de la referencia, a continuación presento el Informe de actividades realizadas en el mes objeto de cobro.

Valor y forma de Pago: El valor total del presente contrato es máximo de **ONCE MILLONES SETECIENTOS MIL PESOS ($11.700.000).** Esta suma será pagada por el SENA al contratista de la siguiente manera: **a).** tres (3) pagos iguales por el valor de **TRES MILLONES NOVECIENTOS MIL PESOS M/CTE. ($3.900.000,00).**

Plazo: Será hasta el 30 de Abril de 2022.

OBJETO:
Prestar los servicios personales de carácter temporal de Artesanos Técnicos Tecnólogos y/o Profesionales que se desempeñen como instructores en los programas de Formación Titulada y Complementaria de la Programación Regular ejecutando horas directas a formación y otras actividades relacionadas con la Formación Profesional en el Centro Industrial y de Energías Alternativas de la Regional Guajira.

Obligaciones Especificas:

No	Obligaciones	Acciones realizadas	Evidencias
1	Ejecutar la Formación Profesional Integral de acuerdo con lo establecido en la Guía de Desarrollo Curricular GFPI-G-012 o normatividad vigente.	Se desarrollan proyectos para las fichas: (2279650, 2279649 y 2233187), de Programación para Analitica de Datos.	Reporte mensual de evidencias.
2	Realizar y entregar informes de Ejecución de la Formación Profesional Integral cuando el Centro de Formación lo requiera, con las novedades académicas, disciplinarias, para los comités de evaluación y seguimiento del Centro y como parte de la evidencia de seguimiento y cumplimiento de las obligaciones contractuales.	Reportes estadístico, informes mensuales de ejecución contractual, relación de asistencia.	Formato Reporte Mensual del Docente que reposa en Coordinación Académica
3	Evaluar las evidencias de aprendizaje (Desempeño, Producto y Conocimiento) – consignadas dentro del Portafolio de Evidencias que corresponda según procedimiento en SIGA y teniendo en cuenta las disposiciones tecnológicas LMS o Físicas definidas por la entidad.	Aprovechamiento de la plataforma Meet de google. Para el proceso formativo con el apoyo de las TIC. Formulando actividades de foros videos y cuestionarios, de formación para su debida realización y su evidencia en la plataforma.	ANEXO 1 INFORME DE ACTIVIDADES No. 1 ✓ Guías ✓ Actas ✓ Videos ✓ Trabajos
4	Entregar soportes e información de calidad requeridos en el proceso de ingreso de aprendices matriculados en los programas de Formación Complementaria tales como: Ficha de matrícula; Fotocopia del documento de identidad legible, y demás requisitos definidos por la entidad.	En el mes objeto de cobro no participe de ninguna actividad relacionada con este proceso.	NO APLICA.
5	Registrar, verificar y hacer seguimiento oportuno en el sistema de información definido por la entidad para la Gestión Académica mediante las siguientes actividades: a) Verificar el estado de totalidad	Durante el mes objeto de cobro no se realizo por parte ninguna de estas actividades.	N.A

	de los aprendices en el proceso de selección y matricula. b) Registrar juicios evaluativos del reconocimiento de aprendizajes previos, crear y asociar aprendices a la ruta de aprendizaje c) Comunicar al Coordinador Académico o supervisor de contrato sobre anomalías, inconsistencias, novedades de aprendices y hallazgos en el registro de la información.		
6	Aplicar y hacer cumplir el reglamento del aprendiz, así como el manual de convivencia del Centro de Formación.	Se imparte formación profesional integral a las fichas asignadas, garantizando el cumplimiento del reglamento del aprendiz	Sofiaplus, Reportes estadístico y control de asistencias.
7	Participar cuando el centro de formación lo requiera en actividades relacionadas con: a) Jornadas de diseño y desarrollo curricular b) Proyectos de investigación técnica y/o pedagógica, c) Proceso de selección, diseño de talleres e instrumentos que alimenten los bancos de pruebas para la selección de aprendices d) Proceso de Registro calificado para los programas en nivel tecnólogo del área temática objeto del presente contrato e) La promoción del portafolio de servicios e) Procesos de autoevaluación de programas de formación en nivel tecnólogo f) Seguimiento en la etapa productiva de los aprendices que le sean asignados.	Durante el mes objeto de cobro no se realizo por parte de la entidad ninguna actividad.	NO. APLICA
8	Tramitar las cuentas de cobro en las fechas definidas por el Centro de Formación y de acuerdo con el plan de pagos, así como la entrega de los informes y soportes requeridos según la normatividad vigente.	El tramite de la cuentas se realizo dentro del plazo establecido cargado los informes requeridos.	Plataforma Secopii

9	Aplicar al proceso de certificación de la norma de competencia "ORIENTAR PROCESOS FORMATIVOS CON BASE EN LOS PLANES DE FORMACIÓN CONCERTADOS, o la actualización. "ORIENTAR FORMACIÓN PRESENCIAL DE ACUERDO CON PROCEDIMIENTO TÉCNICO Y NORMATIVO" o en su defecto deberá aplicar al proceso de certificación de la norma en el caso que se encuentre vencido y aportar el respectivo certificado de aprobación, o la evidencia de la inscripción efectiva a mismo, en el informe de ejecución contractual antes de la terminación del contrato.	En el mes objeto de cobro no participe de ninguna actividad relacionada con este proceso.	NO.APLICA
10	Impartir Formación Profesional Integral en el área designada en sus diferentes modalidades y niveles con un alcance mínimo de 160 horas mensuales.	El reporte mensual Estadístico evidencia la intensidad horaria Impartida en la formación Profesional.	Formato Estadistico mes de Abril.
11	Desarrollar las actividades adicionales que se requieran para la ejecución del contrato, tales como ajustarse a las políticas de comunicación del SENA y aplicación de las TIC (Manejar los sitios Web oficiales, Manejo Secop II, utilizar correctamente los correos electrónicos, revisarlos periódicamente y contestar requerimientos); El contratista deberá ejecutar su contrato	Se desarrollan todas estas actividades, acudiendo a las capacitaciones requeridas para el buen uso de estas tecnologías.	Certificados en la plataforma sicontratista y asistencia a las capacitaciones.

	conforme al Sistema Integrado de Gestión y Autocontrol "SIGA" del SENA el cual se encuentra documentado en la plataforma CompromISO, para lo cual deberá conocer apoyar y participar en las actividades propias del SIGA y del MIPG, velar por la correcta utilización de la imagen institucional, reportar al área de comunicaciones de la Regional los eventos o actividades a realizar que requieran divulgación		

A continuación relaciono los desplazamientos que realicé previo a la presentación de este informe. Una vez finalizado cada desplazamiento presenté al ordenador del gasto el informe en el Formato Informe Legalización Desplazamiento Contratista GTH-F-087, en el que se describieron las actividades desarrolladas y los resultados de cada desplazamiento. Cada informe cuenta con el visto bueno del Supervisor.
Se lista a continuación el soporte de la legalización de los desplazamientos realizados, los cuales forman parte integral del presente informe de ejecución contractual.

ITEM	No DE LA ORDEN DE VIAJE	LUGAR DE DESPLAZAMIENTO	FECHAS DE DESPLAZAMIENTOS	
			INICIO	FINALIZACIÓN
1.	N.A	N.A	N.A	N.A
2.	N.A	N.A	N.A	N.A

Nota 1: *Por cada desplazamiento que haya realizado el contratista, adjuntará el respectivo informe que la soporte. En caso de haber realizado el desplazamiento en fecha posterior a la presentación del informe de ejecución contractual, deberá reportarlo en el siguiente informe de ejecución contractual.*

Para el trámite de la cuenta me permito adjuntar: Documentos electrónicos enunciados como evidencias del cumplimiento de las obligaciones contractuales y los desplazamientos realizados y el No.4473542316 y 4474845487, de las planillas, expedida por SOI correspondiente al mes de Marzo de 2022 y. (Decreto Ley 2106 de 2019 – " Decreto Ley Antitrámites")

Cordialmente,

Firma
ALFONSO JOSE PATERNINA DIAZ
C.C. No. 78.715.466 de Monteria

Recibí a satisfacción:

EUFEMIA MARGARITA PACHECO MAYA
SUPERVISOR(A) CONTRATO No. **CO1.PCCNTR. 3464769**
COORDINADORA ACADEMICA

INFORME DE ACTIVIDADES DE FORMACIÓN

PROGRAMA DE FORMACIÓN COMPLEMENTARIA

MES DE ABRIL DE 2022

PRESENTADO A:

Eufemia Margarita Pacheco Maya

Supervisor(a) Contrato **No. CO1.PCCNTR. 3464769 de año 2022**

PRESENTADO POR:

ALFONSO JOSE PATERNINA DIAZ

SERVICIO NACIONAL DE APRENDIZAJE - SENA

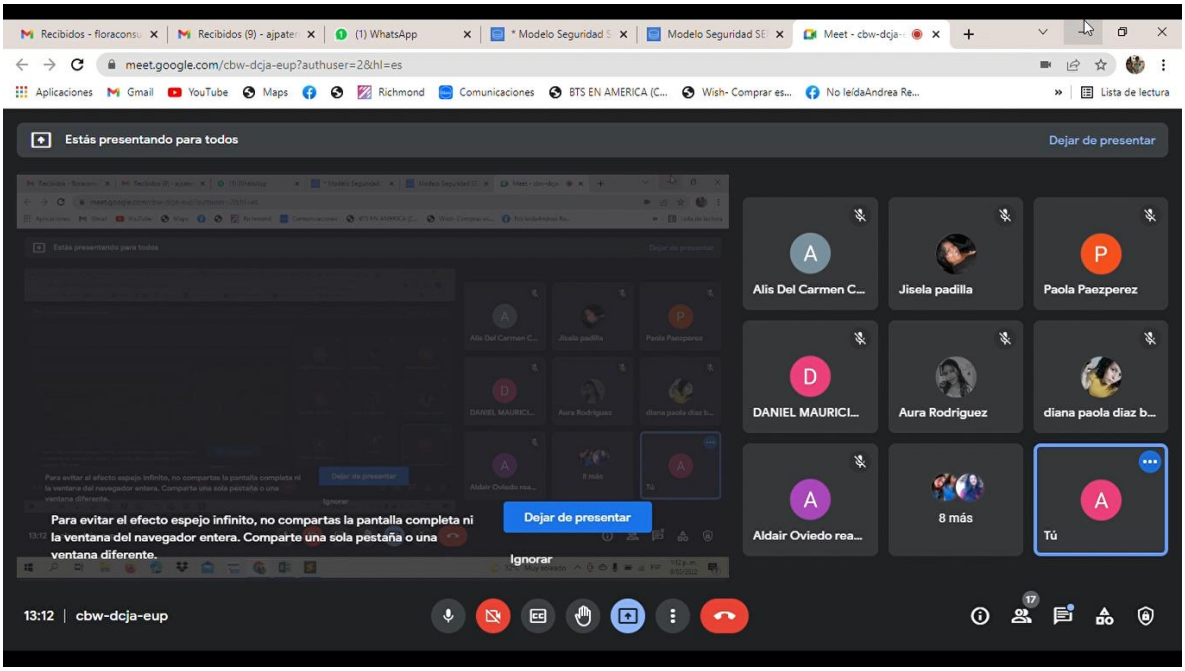
CENTRO INDUSTRIAL Y DE ENERGÍAS ALTERNATIVAS

RIOHACHA, LA GUAJIRA

EVIDENCIAS MES DE ABRIL

TÉCNICO EN PROGRAMACION PARA ANALITACA DE DATOS.

Ficha: 2233187

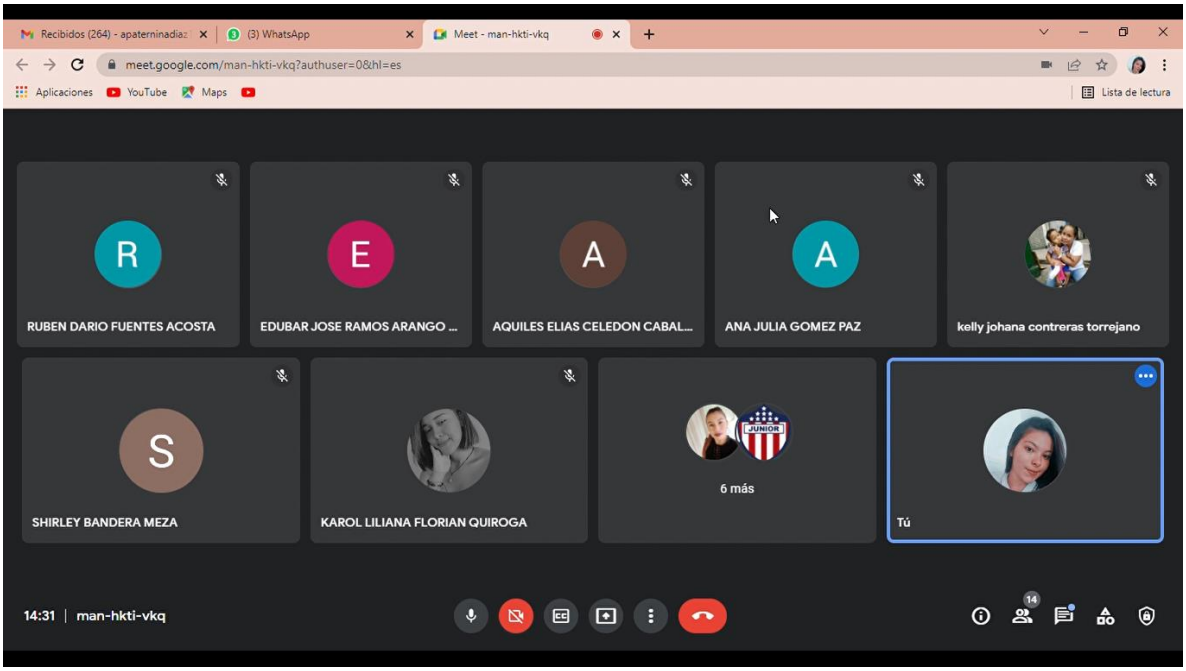


Figuras No1: Reunión Sincrónica, como crear una base de datos en SQlite3.
Ficha: 2233187.

EVIDENCIAS MES DE ABRIL

TÉCNICO EN PROGRAMACION PARA ANALITACA DE DATOS.

Ficha: 2279650



Figuras No3: Reunión Sincrónica, ANALISIS DISEÑO E IMPLEMENTACION DE UN PROTOTIPO PARA LA SEGURIDAD DEL CENTRO INDUSTRIAL Y ENERGIAS ALTERNATIVAS FASE DE INPLEMENTACION. Ficha: 2279650.



Figuras No4: Reunión Presencial, ANALISIS DISEÑO E IMPLEMENTACION DE UN PROTOTIPO PARA LA SEGURIDAD DEL CENTRO INDUSTRIAL Y ENERGIAS ALTERNATIVAS FASE DE INPLEMENTACION. Ficha: 2279650.



Figuras No5: Reunión Presencial, ANALISIS DISEÑO E IMPLEMENTACION Y DESARROLLO DE UN PROTOTIPO PARA LA SEGURIDAD DEL CENTRO INDUSTRIAL Y ENERGIAS ALTERNATIVAS FASE DE INPLEMENTACION .
Ficha: 2279649.



Figuras No6: Reunión Presencial, ANALISIS DISEÑO E IMPLEMENTACION DE UN PROTOTIPO PARA LA SEGURIDAD DEL CENTRO INDUSTRIAL Y ENERGIAS ALTERNATIVAS FASE DE IMPLEMENTACION. Ficha: 2279649.

INPLEMENTACION DE UN PROTOTIPO PARA LA SEGURIDAD DEL CENTRO INDUSTRIAL Y ENERGIAS ALTERNATIVAS POR MEDIO DEL RECONOCIMIENTO FACIAL “FASE DE IMPLEMENTACION”.

En esta fase vamos a trabajar con Python y Sqlite3:

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes.

Python por ser un código abierto tiene una comunidad muy grande a nivel mundial y estos aportan librerías que ayudan al crecimiento de Python.

Describiremos algunas librerías que utilizaremos en esta aplicación.

OpenCV: Es una librería de computación visual creada por Intel, esta librería está disponible para múltiples plataformas como: Windows, Linux, Mac, Android, además cuenta con soporte para diferentes lenguajes como: Python, Java, C/C++, .Net, entre otros, maneja los conceptos básicos de la programación de aplicaciones enfocadas en el análisis y procesamiento de imágenes por computador.

La biblioteca OpenCV puede ser usado bajo licencia BSD para proyectos escolares o comerciales, las aplicaciones de esta librería incluyen, la robótica, análisis y procesamiento de imágenes o vídeos, seguimiento y detección de objetos, detección y reconocimiento de rostros, reconocimiento de placas de vehículos, análisis de formas, reconstrucción 3D, realidad aumentada, y mucho más.

Una biblioteca de código abierto que contiene implementaciones que abarcan más de 2500 algoritmos. Además, está especializada en el sistema de visión artificial y machine learning.

Imutils: Es un paquete que tiene una serie de funciones que ayudan a la Rotación de una imagen, la traslación y cambiar de tamaño entre otros, va de la mano con Opencv, Numpy y Matplotlib.

La traslación: es el desplazamiento de una imagen en la dirección *x* o *y* . Para traducir una imagen en OpenCV, debe proporcionar el cambio *(x, y)* , indicado como (t_x, t_y) para construir la matriz de traducción *M* :

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

La rotación: de una imagen en OpenCV se logra haciendo una llamada a y . Se debe tener más cuidado para proporcionar la coordenada *(x, y)* del punto alrededor del cual se va a rotar la imagen. Estas llamadas de cálculo pueden acumularse rápidamente y hacer que su código sea voluminoso y menos legible. La función en ayuda a resolver este problema.

Cambiar el tamaño: de una imagen en OpenCV se logra llamando a la función. Sin embargo, se debe tener especial cuidado para garantizar que se mantenga la relación de aspecto. Esta función de mantiene la relación de aspecto y proporciona los argumentos de palabras clave, por lo que la imagen se puede cambiar de tamaño al ancho/alto deseado mientras (1) se mantiene la relación de aspecto y (2) se asegura que las dimensiones de la imagen no tengan que ser calculadas explícitamente por El desarrollador.

PIL: Esta biblioteca proporciona una amplia compatibilidad con formatos de archivo, una representación interna eficiente y capacidades de procesamiento de imágenes bastante potentes.

La biblioteca de imágenes principal está diseñada para un acceso rápido a los datos almacenados en unos pocos formatos de píxeles básicos. Debería proporcionar una base sólida para una herramienta general de procesamiento de imágenes.

NumPy : permite la generación y manejo de datos extremadamente rápido. NumPy tiene su propia estructura de datos incorporada llamado arreglo que es similar a la lista normal de Python, pero puede almacenar y operar con datos de manera mucho más eficiente.

Tkinter: Es una librería del lenguaje de programación Python y funciona para la creación y el desarrollo de aplicaciones de escritorio. Esta librería facilita el posicionamiento y desarrollo de una interfaz gráfica de escritorio con Python. Tkinter es el paquete estándar de Python para interactuar con Tk.

Os: nos permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y nos permiten manipular la estructura de directorios (para leer y escribir archivos).

GESTOR DE BASES DE DATOS

SQLite: Se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

CREACION DE LA BASE DE DATOS TEST:

Las Tablas son las que almacenan la información del sistema, estas están formadas por columnas y filas donde las columnas son los atributos y las filas son los registros de cada uno de los usuarios, la suma de todas las tablas nos da como resultado la base de datos.

A continuación vamos hacer documentación detallada para que el código se pueda entender fácilmente para cualquier persona.

Importamos la librería sqlite3.

```
import sqlite3
```

conn = sqlite3.connect("test.db") (crea la base de datos y almacenamos la conexión en la variable conn)

cursor = conn.cursor()(creamos el objeto cursor para ejecutar acciones)

A continuación hacemos el escrip y lo almacenamos en sql”

La instrucción DROP TABLE IF EXISTS Borra la tabla si existe y la crea nuevamente.

```
sql = """
DROP TABLE IF EXISTS datos_personales;
DROP TABLE IF EXISTS ficha;
DROP TABLE IF EXISTS programa;
DROP TABLE IF EXISTS aprendiz;
DROP TABLE IF EXISTS instructor;
DROP TABLE IF EXISTS historial;
DROP TABLE IF EXISTS instructor_ficha;
DROP TABLE IF EXISTS empleado;
DROP TABLE IF EXISTS cargo;
DROP TABLE IF EXISTS departamento;
DROP TABLE IF EXISTS sexo;
```

Crea la tabla sexo con un atributo id_sexo de tipo entero con una longitud de 11 caracteres numéricos es llave primaria y no puede tener valores nulos, sexo es de tipo texto de 15 caracteres y no puede ser nulo.

```
CREATE TABLE sexo (

id_sexo INTEGER(11) PRIMARY KEY NOT NULL,

sexo VARCHAR(15) NOT NULL

);
```

Crea la tabla datos personales con un atributo id_persona de tipo entero con una longitud de 11 caracteres numéricos es llave primaria y no puede tener valores nulos, nombre es de tipo texto de 30 caracteres y no puede ser nulo, apellido es de tipo texto de 30 caracteres y no puede ser nulo,

Dirección es de tipo texto de 30 caracteres y no puede ser nulo, teléfono es de tipo texto de 10 caracteres y no puede ser nulo, correo es de tipo texto de 45 caracteres y no puede ser nulo, id_sexo es de tipo entero y no puede ser nulo, id_voz es de tipo entero y no puede ser nulo, id_face es de tipo entero y no puede ser nulo, id_tactil es de tipo entero y no puede ser nulo,

Su llave foránea id_sexo relacionada con id_sexo de la tabla sexo. “

```
CREATE TABLE datos_personales (

id_persona INTEGER(11) PRIMARY KEY NOT NULL,

nombre VARCHAR(30) NOT NULL,

apellido VARCHAR(35) NOT NULL,

direccion VARCHAR(30) NOT NULL,

telefono VARCHAR(10) NOT NULL,

correo VARCHAR(45) NOT NULL,

id_sexo INTEGER NOT NULL,

id_voz INTEGER NOT NULL,

id_face INTEGER NOT NULL ,

id_tactil INTEGER NOT NULL,

FOREIGN KEY(id_sexo) REFERENCES sexo(id_sexo));
```

Crea la tabla programa con un atributo id_programa de tipo entero, es llave primaria y no puede tener valores nulos, programa es de tipo texto de 50 caracteres y no puede ser nulo.

```
CREATE TABLE programa (

id_programa INTEGER PRIMARY KEY NOT NULL ,

programa VARCHAR(50) NOT NULL);
```

Crea la tabla ficha con un atributo id_ficha de tipo entero, es llave primaria y no puede tener valores nulos, ficha de tipo texto de 50 caracteres y no puede ser nulo, id_programa es de tipo entero de 11 caracteres numérico y no puede ser nulo, su llave foránea id_programa relacionada con id_programa de la tabla programa.

```
CREATE TABLE ficha (
id_ficha INTEGER(11) PRIMARY KEY NOT NULL ,
ficha VARCHAR(50) NOT NULL,
id_programa INTEGER(11) NOT NULL,
FOREIGN KEY(id_programa) REFERENCES programa(id_programa));
```

“Crea la tabla instructor con un atributo id_instructor de tipo entero, es llave primaria y no puede tener valores nulos, id_persona de tipo entero de 11 caracteres numéricos y no puede ser nulo, su llave foránea id_persona relacionada con id_persona de la tabla datos personales. “

```
CREATE TABLE instructor (
id_instructor INTEGER PRIMARY KEY NOT NULL ,
id_persona INTEGER(11) NOT NULL,
FOREIGN KEY(id_persona) REFERENCES datos_personales(id_persona));
```

“Crea la tabla aprendiz con un atributo id_aprendiz de tipo entero de 11 caracteres numéricos, es llave primaria y no puede tener valores nulos, id_persona de tipo entero de 11 caracteres numéricos y no puede ser nulo, id_ficha de tipo entero de 11 caracteres numéricos y no puede ser nulo, su llave foránea id_persona relacionada con id_persona de la tabla datos personales y la llave foránea id_ficha está relacionada id_ficha de la tabla ficha “

```
CREATE TABLE aprendiz (
id_aprendiz INTEGER(11) PRIMARY KEY NOT NULL ,
id_persona INTEGER(11) NOT NULL,
id_ficha INTEGER(11) NOT NULL,
FOREIGN KEY(id_persona) REFERENCES datos_personales(id_persona),
```

FOREIGN KEY(id_ficha) REFERENCES ficha(id_ficha));

Crea la tabla historial con un atributo id_historial de tipo entero de 11 caracteres numéricos, es llave primaria y no puede tener valores nulos, id_persona de tipo entero de 11 caracteres numéricos y no puede ser nulo, fecha de tipo fecha y hora no puede ser nulo, su llave foránea id_persona relacionada con id_persona de la tabla datos personales.

```
CREATE TABLE historial (

id_historial INTEGER(11) PRIMARY KEY NOT NULL ,

id_persona INTEGER(11) NOT NULL,

fecha DATETIME NOT NULL,

FOREIGN KEY(id_persona) REFERENCES datos_personales(id_persona));
```

Crea la tabla instructor_ficha con un atributo id_instructor de tipo entero de 11 caracteres numéricos no puede tener valores nulos, id_ficha de tipo entero de 11 caracteres numéricos y no puede ser nulo, su llave foránea id_ficha relacionada con id_ficha de la tabla ficha y id_instructor relacionada con id_instructor de la tabla instructor.

```
CREATE TABLE instructor_ficha (

id_ficha INTEGER(11) NOT NULL,

id_instructor INTEGER(11) NOT NULL,

FOREIGN KEY(id_ficha) REFERENCES ficha(id_ficha),

FOREIGN KEY(id_instructor) REFERENCES instructor(id_instructor));
```

Crea la tabla Cargo con un atributo id_cargo de tipo entero de 11 caracteres numéricos es llave primaria, no puede tener valores nulos, cargo de tipo texto de 20 caracteres y no puede ser nulo.

```
CREATE TABLE cargo (

id_cargo INTEGER(11) PRIMARY KEY NOT NULL ,

cargo VARCHAR(20) NOT NULL);
```

Crea la tabla Departamento con un atributo id_departamento de tipo entero de 11 caracteres numéricos es llave primaria, no puede tener valores nulos, departamento de tipo texto de 30 caracteres y no puede ser nulo.

```
CREATE TABLE departamento (

id_departamento INTEGER(11) PRIMARY KEY NOT NULL ,
```



```
departamento VARCHAR(30) NOT NULL);
```

Crea la tabla Empleado con un atributo id_empleado de tipo entero de 11 caracteres numéricos, es llave primaria y no puede tener valores nulos, id_persona de tipo entero de 11 caracteres numéricos y no puede ser nulo, id_cargo de tipo entero de 11 caracteres numéricos y no puede ser nulo, id_departamento de tipo entero de 11 caracteres numéricos y no puede ser nulo, su llave foránea id_persona relacionada con id_persona de la tabla datos personales , la llave foránea id_cargo está relacionada id_cargo de la tabla cargo y la llave foránea id_departamento está relacionada con id_de de la tabla cargo.

```
CREATE TABLE empleado (

id_empleado INTEGER(11) PRIMARY KEY NOT NULL ,

id_persona INTEGER(11) NOT NULL,

id_cargo INTEGER(11) NOT NULL,

id_departamento INTEGER(11) NOT NULL,

FOREIGN KEY(id_persona) REFERENCES datos_personales(id_persona),

FOREIGN KEY(id_cargo) REFERENCES cargo(id_cargo),

FOREIGN KEY(id_departamento) REFERENCES departamento(id_departamento));
```

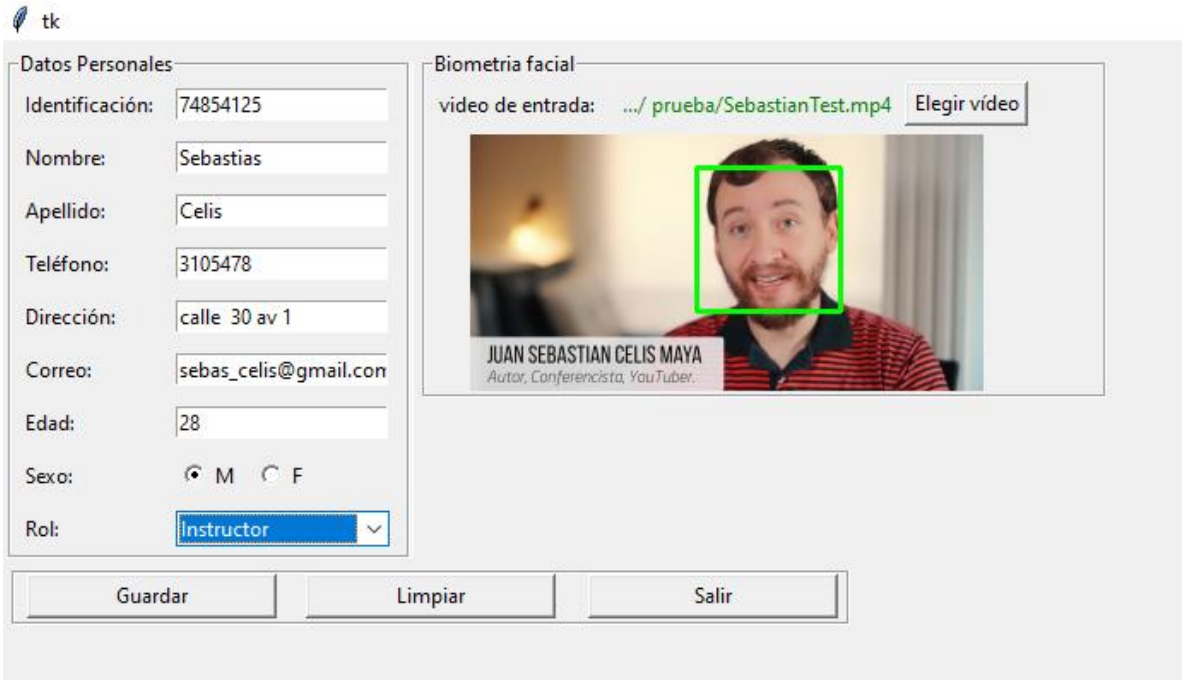
"""

Se ejecuta el escript o creación de la base de datos con todas sus tablas y manda un mensaje de creación de base de datos en caso contrario manda un excepción o error diciendo que ha ocurrido un error e imprime dicho error, finaliza cerrando la conexión y la base de datos.

```
try:
        cursor.executescript(sql)
        print('La base de datos a sido creada')
except sqlite3.Error as e:
        print('Ha ocurrido un error',e.args)
conn.close() “cierra la base de datos”
```

Formulario Datos Personales:

Se registra la información básica de los usuarios como: nombres, apellidos, teléfono, dirección, correo, edad, sexo, rol del usuario y la captura de la biometría facial del usuario.




```
from tkinter import *

from tkinter import ttk

from tkinter import messagebox

from tkinter import filedialog

from PIL import Image

from PIL import ImageTk

import cv2

import imutils

import os
```

```
import sqlite3


def guardar(): #Definimos una función guardar
    con = sqlite3.connect("test.db")#creamos la conexión a la base de datos
    cursor = con.cursor()#creamos el cursor para ejecutar alguna acción
```

sql= es una instrucción que inserta en la tabla datos_personales los atributos nombre, apellido, dirección, teléfono, correo y id_sexo.

```
    sql = """
    INSERT INTO `datos_personales` (`id_persona`, `nombre`, `apellido`, `direccion`, `telefono`, `correo`, `id_sexo`)
    VALUES (?, ?, ?, ?, ?, ?, ?)#valores a almacenar son 7
    """
```

Se extrae de la entrada de cada valor y se lo asigna a cada variable

```
    cedula= entry_cedula.get()
    nombre = entry_nombre.get()
    apellido = entry_apellido.get()
    telefono = entry_telefono.get()
    direcccion = entry_direccion.get()
    correo = entry_correo.get()
    edad = entry_edad.get()
    sex = sexo.get()
```

Se crea un tupla con la colección de variables y se le asigna la variable datos.

```
    datos = (cedula,nombre,apellido,direcccion,telefono,correo,sex)
    try:
```

se ejecuta el sql y se le pasan los datos, si se ejecuta correctamente guarda la información en un registro de la tabla datos_personales.

```
        cursor.execute(sql,datos)
            con.commit() #Actualiza el registro que se ingresó en la tabla datos_personales de la base de datos TEST
            print('El registro se guardo satisfactoriamente')
        except sqlite3.Error as e:
            print('Ha ocurrido un error',e.args[0])

        con.close()#cierra la conexión de la base de datos.
def limpiar():
        pass
def salir():
        pass

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 0
```

Función de detección de frame hace una copia del frame en la variable auxframe convierte el frame a escala de grises le pasa la máscara al clasificador de faces y halla los diferentes contornos y dibuja un rectángulo alrededor del rostro.

```
def deteccion(frame):
        global count
        personName = entry_cedula.get()
        dataPath = 'C:/Users/sofia paternina/Desktop/Aquiles_Proyecto/Reconocimiento/Data'
        personPath = dataPath + '/' + personName
        if not os.path.exists(personPath):
            print('Carpeta creada:',personPath)
            os.makedirs(personPath)
```

```
        auxFrame = frame.copy()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = faceClassif.detectMultiScale(gray,1.3,7)
        for (x,y,w,h) in faces:
                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                rostro = auxFrame[y:y+h,x:x+w]
                rostro = cv2.resize(rostro,(150,150),interpolation=cv2.INTER_CUBIC)
                cv2.imwrite(personPath +'/rostro_{}.jpg'.format(count),rostro)
                count += 1
        if count > 399:
                cap.release()

        return frame
```

Función visualizar toma frames con el método imutils. resize le cambia el tamaño al frame le aplica la función detección(frame) este devuelve el frame y lo convierte de BGR2RGB, el frame lo convierte en un vector, después la convierte en una foto imagen y se la configura a un label y la visualiza 19 veces

```
def visualizar():
        global cap
        if cap is not None:
                ret, frame = cap.read()
                if ret == True:
                        frame = imutils.resize(frame,width=300)
                        frame = deteccion(frame)
                        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                        im = Image.fromarray(frame)
                        img = ImageTk.PhotoImage(image= im)
```

```
            lbl_vid.configure(image= img)
            lbl_vid.image = img
            lbl_vid.after(19,visualizar)
        else:
            lbl_vidPath.configure(text= "Aún no seleciona un video", fg= "red")
            lbl_vid.image= ""
            cap.release()
```

Función elegir video escoge un video de acuerdo al formato que se le asigno en este caso extensiones mp4 o avi.

```
def elegir_video():
    global cap

    if cap is not None:
        lbl_vid.image = ""
        cap.release()
        cap = None

    video_path = filedialog.askopenfilename(filetypes= [
        ("all video format", ".mp4"),
        ("all video format", ".avi")])
    if len(video_path) > 0:
        lbl_vidPath.configure(text= ".../"+video_path[-25:], fg="green")
        cap = cv2.VideoCapture(video_path)
        visualizar()
    else:
        lbl_vidPath.configure(text= "Aún no seleciona un video", fg= "red")
```

```
cap = None
```

Comienza la raíz o root se definió las ventanas, los marcos, las etiquetas, las entradas y los botones de la aplicación.

```
root = Tk()

root.geometry("650x345+200+100")


lblF_dp = LabelFrame(root, text='Datos Personales')

lblF_dp.place(x= 5,y= 5, width= 235)

lblF_vid = LabelFrame(root, text='Biometria facial')

lblF_vid.place(x= 245,y= 5, width= 400)

# label frame para la seccion botones

lblF_btn = LabelFrame(root)

lblF_btn.place(x=5, y=310, width=490, height=32)


#labels

lbl_cedula = Label(lblF_dp, text="Identificación:", fg="black")

lbl_cedula.grid(row=0, column=0, padx=5, pady=5, sticky=W)

lbl_nombre = Label(lblF_dp, text="Nombre:", fg="black")

lbl_nombre.grid(row=1, column=0, padx=5, pady=5, sticky=W)

lbl_apellido = Label(lblF_dp, text="Apellido:", fg="black")

lbl_apellido.grid(row=2, column=0, padx=5, pady=5, sticky=W)

lbl_telefono = Label(lblF_dp, text="Teléfono:", fg="black")

lbl_telefono.grid(row=3, column=0, padx=5, pady=5, sticky=W)

lbl_direccion = Label(lblF_dp, text="Dirección:", fg="black")

lbl_direccion.grid(row=4, column=0, padx=5, pady=5, sticky=W)
```

```
lbl_correo = Label(lblF_dp, text="Correo:", fg="black")
lbl_correo.grid(row=5, column=0, padx=5, pady=5, sticky=W)
lbl_edad = Label(lblF_dp, text="Edad:", fg="black")
lbl_edad.grid(row=6, column=0, padx=5, pady=5, sticky=W)
lbl_sexo = Label(lblF_dp, text="Sexo:", fg="black")
lbl_sexo.grid(row=7, column=0, padx=5, pady=5, sticky=W)
lbl_rol = Label(lblF_dp, text='Rol:')
lbl_rol.grid(row= 8, column= 0,padx=5, pady=5, sticky=W)

# ------ Video -----
lbl_vidInfo = Label(lblF_vid, text= "video de entrada:")
lbl_vidInfo.grid(row=0, column=0, padx=5,pady=5)
lbl_vidPath = Label(lblF_vid, text= "Aún no seleciona un video", fg="red")
lbl_vidPath.grid(row=0, column=1, padx=5,pady=5)
lbl_vid = Label(lblF_vid)
lbl_vid.grid(row=1, column=0, columnspan=3)

#Entradas
entry_cedula = Entry(lblF_dp)
entry_cedula.grid(row=0, column=1, padx=5)
entry_nombre = Entry(lblF_dp)
entry_nombre.grid(row=1, column=1, padx=5)
entry_apellido = Entry(lblF_dp)
entry_apellido.grid(row=2, column=1, padx=5)
entry_telefono = Entry(lblF_dp)
```

```
entry_telefono.grid(row=3, column=1, padx=5)

entry_direccion = Entry(lblF_dp)

entry_direccion.grid(row=4, column=1, padx=5)

entry_correo = Entry(lblF_dp)

entry_correo.grid(row=5, column=1, padx=5)


entry_edad= Entry(lblF_dp)

entry_edad.grid(row=6,column=1, padx=5)


sexo = IntVar()

entry_sexo = Radiobutton(lblF_dp, text="M", value=1, variable=sexo)

entry_sexo.grid(row=7, column=1, padx=5, sticky= W)

entry_sexo2 = Radiobutton(lblF_dp, text="F", value=2, variable=sexo)

entry_sexo2.grid(row=7, column=1, padx=5, )


entry_rol=ttk.Combobox(lblF_dp, state="readonly", width=17,)

entry_rol["value"]=["Aprendiz","Administrador","Instructor","Visitante"]

entry_rol.grid(row=8,column=1, padx=5)


# botones

b_guardar = Button(lblF_btn, text="Guardar", command=guardar)

b_guardar.place(relx=0.015, y=0, relwidth=0.3)

b_limpiar = Button(lblF_btn, text="Limpiar", command=limpiar)

b_limpiar.place(relx=0.35, relwidth=0.3)

b_salir = Button(lblF_btn, text="Salir", command=salir)
```

```
b_salir.place(relx=0.69, relwidth=0.3)


btn_ver_video = Button(lblF_vid, text="Elegir vídeo",command=elegir_video, anchor= E)

btn_ver_video.grid(row=0, column=2, sticky= E)

root.mainloop()
```

RECONOCEDOR FACIAL.

El código realiza el proceso de reconocimiento facial para hacer el reconocimiento debemos tener previamente almacenada los frames o (fotos de los usuarios) en nuestra base de datos, paso a seguir el usuario posa para la cámara de alta resolución y esta le toma varias fotos y con la ayuda de la librería Opencv esta compara los contornos del usuario con los contornos de los usuarios registrados en nuestra base de datos si lo encuentra en dicha consulta nos da como resultado la identificación de este y el debido acceso a las instalaciones del SENA.

```
import cv2

import os

import imutils

from funciones import identifica

dataPath = 'C:/Users/sofia paternina/Desktop/Aquiles_Proyecto/Reconocimiento/Data'

imagePaths = os.listdir(dataPath)

print('imagePaths=',imagePaths)


#face_recognizer = cv2.face.EigenFaceRecognizer_create()

#face_recognizer = cv2.face.FisherFaceRecognizer_create()

face_recognizer = cv2.face.LBPHFaceRecognizer_create()


#leer el modelo
```

```
#face_recognizer.read('Modelos/modeloEigenFace.xml')
#face_recognizer.read('Modelos/modeloFisherFace.xml')
face_recognizer.read('Modelos/biometriaFacialSena_LBPHFace.xml')

# x = face_recognizer.getLabels()
# print(x[0])
#cap = cv2.VideoCapture('Videos de prueba/SebastianTest.mp4')
#cap = cv2.VideoCapture('Nayelis.mp4')

#camara Laptop
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')

while True:
        ret, frame = cap.read()
        if ret == False: break
        frame = imutils.resize(frame,width=300)
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        auxFrame = gray.copy()

        faces = faceClassif.detectMultiScale(gray,1.3,7)

        for (x,y,w,h) in faces:
                rostro = auxFrame[y:y+h,x:x+w]
```

```
                rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
                result = face_recognizer.predict(rostro)

                cv2.putText(frame,'{}'.format(result),(x,y-5),1,1.3,(255,255,0),1,cv2.LINE_AA)

#               if result[1] < 5700: # EigenFace
#               if result[1] < 450:  # FisherFace
                if result[1] < 71:   # LBPHFace
                        cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-
25),2,1.1,(0,255,0),1,cv2.LINE_AA)
                        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
                        #print(imagePaths[result[0]])
                        cedula = imagePaths[result[0]]
                else:
                        cv2.putText(frame,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
                        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
        cv2.imshow('frame',frame)
        k = cv2.waitKey(1)
        if k == 27:
                break
print(cedula)
identifica(int(cedula))
cap.release()
cv2.destroyAllWindows()
```

ENTRENAMIENTO PARA PREDICCION DE LOS USUARIOS:

El entrenamiento de los usuarios almacenamos las imágenes en una carpeta y se identifica con el número de cedula y también crea una lista de todos los usuarios en el orden de entrada, puede usar tres modelos de entrenamiento de acuerdo al número de usuarios.

entrenando EigenFace: para Entrenar miles de usuarios.

face_recognizer = cv2.face.EigenFaceRecognizer_create()

entrenando FisherFace: para entrenar cientos de usuarios

face_recognizer = cv2.face.FisherFaceRecognizer_create()

entrenando LBPHFace: para entrenas decenas de usuarios en este caso utilizaremos este método para realizar nuestro entrenamiento.

face_recognizer = cv2.face.LBPHFaceRecognizer_create()

al entrenar nuestro prototipo los tiempos de respuestas van a ser más rápidos, precisos y con un margen de error mínimo, ya que el programa se vuelve autónomo al momento de tomar decisiones.

```
import cv2

import os

import numpy as np


dataPath = 'C:/Users/sofia paternina/Desktop/Aquiles_Proyecto/Reconocimiento/Data'

peopleList = os.listdir(dataPath)

print('Lista de personas: ',peopleList)


labels = []

facesData = []

label = int(peopleList[0])


#print(label)
```

```
for nameDir in peopleList:
        personPath = dataPath + '/' + nameDir
        print('Leyendo las imágenes')

        for fileName in os.listdir(personPath):
                #print('Rostros: ', nameDir + '/' + fileName)
                labels.append(label)
                facesData.append(cv2.imread(personPath+'/'+fileName,0))
                #image = cv2.imread(personPath+'/'+fileName,0)
                #cv2.imshow('image',image)
                #cv2.waitKey(10)
        label = int(nameDir)
        print(label)

#print('labels= ',labels)

#entrenando EigenFace
#face_recognizer = cv2.face.EigenFaceRecognizer_create()   #miles

#entrenando FisherFace
#face_recognizer = cv2.face.FisherFaceRecognizer_create()   #cientos

#entrenando LBPHFace
face_recognizer = cv2.face.LBPHFaceRecognizer_create()    #decenas
```

```
print('Entrenando...')

face_recognizer.train(facesData,np.array(labels))


#almacenar

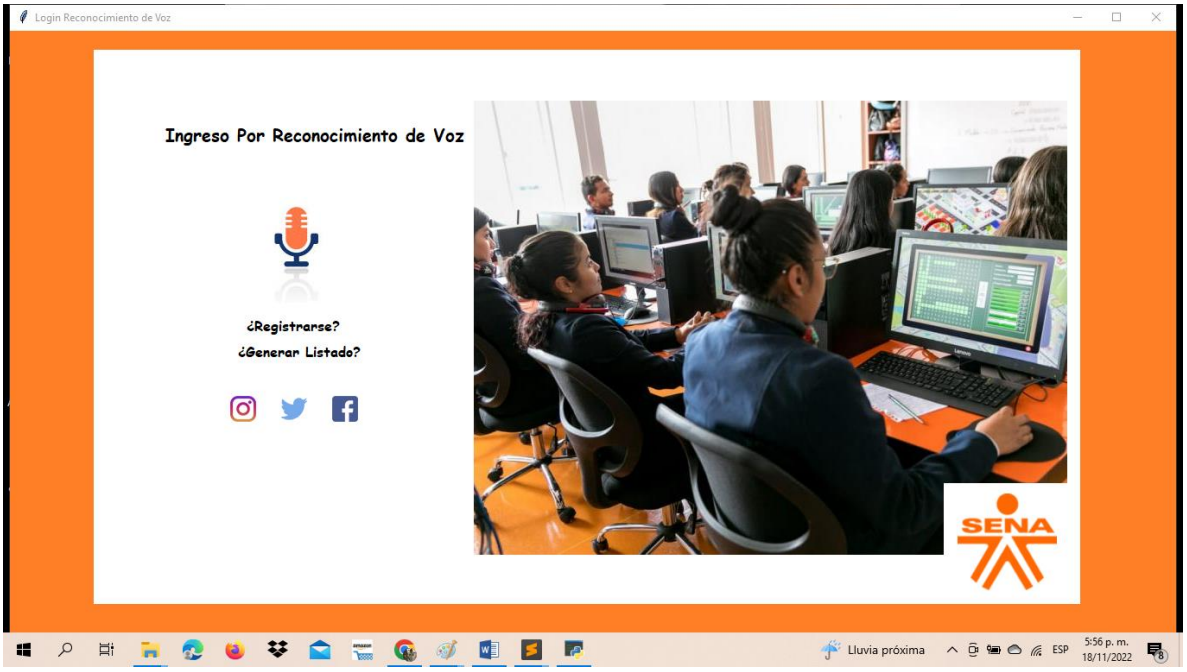
#face_recognizer.write('Modelos/modeloEigenFace.xml')

#face_recognizer.write('Modelos/modeloFisherFace.xml')

face_recognizer.write('Modelos/modeloLBPHFace2.xml')


print('modelo almacenado')
```

RECONOCIMIENTO DE VOZ:




```
import tkinter

from tkinter import ttk

from threading import Thread
```

```
import mysql.connector
from tkinter import *
from tkinter import messagebox
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
import os

import pyttsx3
import engineio
import time

import pyaudio
import wave

import speech_recognition as sr
from tabulate import tabulate
from openpyxl import Workbook
import pandas as pd
import matplotlib.pyplot as plt

import xlrd
import mysql.connector



engineio = pyttsx3.init()
voices = engineio.getProperty('voices')
engineio.setProperty('rate', 130)
engineio.setProperty('voice',voices[0].id)
```

```
release = True

limit = 100

fila =1



def speak(text):

        engineio.say(text)

        engineio.runAndWait()




def submit_2():


        Intervalo = 1024

        Formato = pyaudio.paInt16

        Canal = 2

        Baudio = 44100

        Duracion = 5

        Salida = "audio.mp3"


        p = pyaudio.PyAudio()


        Grabando = p.open(format=Formato, channels=Canal, rate=Baudio, input=True,
frames_per_buffer=Intervalo)

        speak("Digame cual es su usuario y contraseña")


        speak(" Grabando.....")
```

```
a = []

for i in range(0, int(Baudio / Intervalo * Duracion)):
        data = Grabando.read(Intervalo)
        a.append(data)

speak(" Procesando.....")

Grabando.stop_stream()
Grabando.close()
p.terminate()

wf = wave.open(Salida, 'wb')
wf.setnchannels(Canal)
wf.setsampwidth(p.get_sample_size(Formato))
wf.setframerate(Baudio)
wf.writeframes(b''.join(a))
wf.close()
r = sr.Recognizer()
with sr.AudioFile("audio.mp3") as source:
        audio = r.listen(source)
        text = r.recognize_google(audio, language='es-ES')
        texto = text
        time.sleep(1.5)
        f = open("login.txt", "w")
        f.write(text)
        f.close()
```

```
        lista = texto
        palabras = lista.split()
        usuario = palabras[0]
        contraseña = palabras[1]
        db = mysql.connector.connect(host="localhost", user="root", password="", db="sena")
        cursor = db.cursor()

        cursor.execute("SELECT * FROM tabla WHERE Usuario=%s and
Password=%s",(usuario,contraseña))
        row = cursor.fetchone()
        if row==None:
                speak("Error al ingresar a la plataforma. verifique si esta en el sistema o intente de
nuevo")
                messagebox.showinfo(message="Error al ingresar a la base de datos", title="Login
Reconocimiento de Voz")

        else:
                imprimir = "Has ingresado correctamente a la plataforma usuario," + usuario
                speak(imprimir)
                messagebox.showinfo(message="Has ingresado correctamente", title="Login
Reconocimiento de Voz")
                new_window()



def new_window():

        f = open("login.txt", "r")
        palabra = f.read()
```

```
palabras = palabra.split()
usua = palabras[0]
passw = palabras[1]
f.close()
root2 = Toplevel()
root2.geometry("1350x730")
imprimir =  "Bienvenido Al Centro Industrial Del Sena Usuario " + " " + usua
speak(imprimir)
Micro2=PhotoImage(file="icon\\micro.png")

main_bg=Label(root2, image=bkg_image)
main_bg.pack()
login_bg=Label(main_bg, image=login_image2, bd=0)
login_bg.place(x=95, y=40)
login_frame=Frame(login_bg, bg="white", height=600, width=400)
login_frame.place(x=40, y=20)

login_info=Label(root2, text=imprimir, bg="white", font=("Comic Sans Ms", 15, "bold"))
login_info.place(x=150, y=60)
db = mysql.connector.connect(host="localhost", user='root', password='', db="sena")
cursor = db.cursor()
q2=usua
query = "SELECT ID, Nombre, Apellido, Cargo FROM tabla WHERE Usuario LIKE '%" + q2 +
"%' OR Cargo LIKE '%" + q2+ "%'"
cursor.execute(query)
rows = cursor.fetchall()
print(tabulate(rows, headers=['ID', 'Nombre', 'Apellido', 'Cargo'], tablefmt='psql'))
wb =Workbook()
```

```
        hoja = wb.active
        hoja.title ="Listado De Ingreso Al Sena Industrial"
        hoja.append(('ID', 'Nombre', 'Apellido', 'Cargo'))
        global fila
        for registro in rows:
                hoja.append(registro)
                fila += 1


        wb.save(filename = "Listado.xlsx")
        book = xlrd.open_workbook("Listado.xlsx") # excel nombre de archivo
        sheet = book.sheet_by_name("Listado De Ingreso Al Sena Industrial") # Nombre de la hoja
en el archivo de Excel
        query = "INSERT INTO tabla2(ID, Nombre, Apellido, Cargo) VALUES(%s,%s,%s,%s)"


        for r in range(1, sheet.nrows):
                ID = sheet.cell(r, 0).value
                Nombre = sheet.cell(r, 1).value
                Apellido = sheet.cell(r, 2).value
                Cargo = sheet.cell(r, 3).value
                values = (ID, Nombre, Apellido, Cargo)
                cursor.execute(query, values)



        cursor.close()
        db.commit()
        db.close()
        os.system("del login.txt")
        os.system("del audio.mp3")
```

```
        os.system("del Listado.xlsx")



def check():
        root2 = Toplevel()
        root2.geometry("1350x730")
        user = user_entry.get()
        passw = pass_entry.get()
        Micro2=PhotoImage(file="icon\\micro.png")


        main_bg=Label(root2, image=bkg_image)
        main_bg.pack()
        login_bg=Label(main_bg, image=login_image2, bd=0)
        login_bg.place(x=95, y=40)
        login_frame=Frame(login_bg, bg="white", height=600, width=400)
        login_frame.place(x=40, y=20)


        login_info=Label(root2, text="Menu de Registro", bg="white", font=("Comic Sans Ms", 15,
"bold"))
        login_info.place(x=150, y=60)
        login_info=Label(root2, text="Ingrese su ID", bg="white", font=("Comic Sans Ms", 13,
"bold"))
        login_info.place(x=180, y=150)
        ent = Entry(login_frame, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10,
"bold"))
        ent.place(x=30, y=130)
```

```
        line=Frame(login_frame, height=3, width=250)
        line.place(x=30, y=150)



        ####################################


        login_info2=Label(root2, text="Ingrese Su Nombre", bg="white", font=("Comic Sans Ms", 13, "bold"))
        login_info2.place(x=180, y=230)
        ent2 = Entry(login_frame, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10, "bold"))
        ent2.place(x=30, y=210)
        line2=Frame(login_frame, height=3, width=250)
        line2.place(x=30, y=230)


        ################################


        login_info4=Label(root2, text="Ingrese Su Apellido", bg="white", font=("Comic Sans Ms", 13, "bold"))
        login_info4.place(x=180, y=320)
        ent4 = Entry(login_frame, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10, "bold"))
        ent4.place(x=30, y=300)
        line4=Frame(login_frame, height=3, width=250)
        line4.place(x=30, y=320)


        ################################


        login_info5=Label(root2, text="Ingrese Su Rol", bg="white", font=("Comic Sans Ms", 13, "bold"))
```

```
        login_info5.place(x=180, y=420)
        ent5 = Entry(login_frame, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10,
"bold"))
        ent5.place(x=30, y=400)
        line5=Frame(login_frame, height=3, width=250)
        line5.place(x=30, y=420)

        ##################################

        def grab():
                user = user_entry.get()
                Intervalo = 1024
                Formato = pyaudio.paInt16
                Canal = 2
                Baudio = 44100
                Duracion = 5
                Salida = "audio2.mp3"

                p = pyaudio.PyAudio()

                Grabando = p.open(format=Formato, channels=Canal, rate=Baudio, input=True,
frames_per_buffer=Intervalo)
                speak("Digame cual es su usuario y contraseña")

                speak(" Grabando.....")

                a = []
```

```
        for i in range(0, int(Baudio / Intervalo * Duracion)):
                data = Grabando.read(Intervalo)
                a.append(data)


        speak(" Procesando.....")


        Grabando.stop_stream()
        Grabando.close()
        p.terminate()


        wf = wave.open(Salida, 'wb')
        wf.setnchannels(Canal)
        wf.setsampwidth(p.get_sample_size(Formato))
        wf.setframerate(Baudio)
        wf.writeframes(b''.join(a))
        wf.close()
        r = sr.Recognizer()
        with sr.AudioFile("audio2.mp3") as source:
                audio = r.listen(source)
                text = r.recognize_google(audio, language='es-ES')
                texto = text
                time.sleep(1.5)
                f = open("registro.txt", "w")
                f.write(text)
                f.close()
        speak("El proceso ha finalizado, puede dar click a la opcion, agregar.")


def add():
```

```
                f = open("registro.txt", "r")
                palabra = f.read()
                palabras = palabra.split()
                usua = palabras[0]
                passw = palabras[1]
                f.close()

                db = mysql.connector.connect(host="localhost", user="root", password="",
db="sena")
                cursor = db.cursor()
                codigo = ent.get()
                nombre = ent2.get()
                apellido = ent4.get()
                rol = ent5.get()

                query = "INSERT INTO tabla(ID, Nombre, Apellido, Usuario, Password, Cargo)
VALUES(%s,%s,%s,%s,%s,%s)"
                cursor.execute(query,(codigo, nombre, apellido, usua, passw, rol))
                db.commit()
                query = "SELECT ID, Nombre, Apellido, Usuario, Password, Cargo FROM tabla"
                cursor.execute(query)
                rows = cursor.fetchall()
                speak("Los datos han sidos ingresados correctamente, su usuario y contraseña
son: ")
                imprimir = "Usuario: "+ usua + " " + "Contraseña: " + passw
                messagebox.showinfo(message=imprimir, title="Login Reconocimiento de Voz")
                os.system("del registro.txt")
                os.system("del audio2.mp3")
```

```
        login_info6=Label(root2, text="Ingresa Usuario y contraseña Dando Click Al Boton", bg="white", font=("Comic Sans Ms", 13, "bold"))
        login_info6.place(x=520, y=150)
        login_btn1=Button(root2, bg="white", command = grab, image=Micro, bd=0, relief="flat")
        login_btn1.place(x=720, y=220)
        login_btn3=Button(root2, bg="white", command = add, image=Add, bd=0, relief="flat")
        login_btn3.place(x=650, y=420)


def Listado():
        root3 = Toplevel()
        root3.geometry("1350x730")


        main_bg2=Label(root3, image=bkg_image)
        main_bg2.pack()
        login_bg2=Label(main_bg2, image=login_image2, bd=0)
        login_bg2.place(x=95, y=40)
        login_frame2=Frame(login_bg2, bg="white", height=600, width=400)
        login_frame2.place(x=40, y=20)


        login_info=Label(root3, text="Listado De Personas Que Ingresaron Al Plantel", bg="white", font=("Comic Sans Ms", 15, "bold"))
        login_info.place(x=120, y=60)
        login_info7=Label(root3, text="Ingrese su Usuario", bg="white", font=("Comic Sans Ms", 13, "bold"))
        login_info7.place(x=180, y=150)
        ent6 = Entry(login_frame2, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10, "bold"))
        ent6.place(x=30, y=130)
```

```
        line6=Frame(login_frame2, height=3, width=180)
        line6.place(x=30, y=150)



        ######################################


        login_info7=Label(root3, text="Ingrese Su Contraseña", bg="white", font=("Comic Sans Ms", 13, "bold"))
        login_info7.place(x=180, y=230)
        ent7 = Entry(login_frame2, relief="flat", width=37, fg="black", font=("Comic Sans Ms", 10, "bold"), show="*")
        ent7.place(x=30, y=210)
        line7=Frame(login_frame2, height=3, width=180)
        line7.place(x=30, y=230)


        ################################


        def Generar():
                user = ent6.get()
                passw = ent7.get()


                try:
                        db = mysql.connector.connect(host="localhost", user=user, password=passw, db="sena")
                        cursor = db.cursor()
                        speak("Ha ingresado correctamente a la base de datos, Generando listado")
                        style1 = ttk.Style()
```



```
                    style1.configure("mystyle.Treeview", highlightthickness=0, bd=0,
background='light blue', font=("Comic Sans Ms", 9, "bold"))

                    trv = ttk.Treeview(root3, columns=(1,2,3,4), show="headings", height="8",
style="mystyle.Treeview")
                    trv.place(x=430, y=140)

                    trv.heading(1, text="ID")
                    trv.heading(2, text="Nombre")
                    trv.heading(3, text="Apeliido")
                    trv.heading(4, text="Cargo")
                    query = "select * from tabla2"
                    cursor.execute(query)
                    rows= cursor.fetchall()
                    for i in rows:
                        trv.insert('','end', value=i)

            except:
                    speak("Error al ingresar a la base de datos, intente de nuevo")


            def clear1():
                    db = mysql.connector.connect(host="localhost", user='root', password='',
db="sena")
                    cursor = db.cursor()

                    if messagebox.askyesno("Confirmar eliminación", "¿Quiere eliminar los
campos"):
                        query = "DELETE FROM tabla2"
```

```
                    cursor.execute(query)
                    db.commit()
                    print("lo que sea")
                    query = "SELECT ID, Nombre, Apellido, Cargo FROM tabla2"
                    cursor.execute(query)
                    rows = cursor.fetchall()
                    trv.delete(*trv.get_children())
                    for i in rows:
                            trv.insert('', 'end', value=i)

        login_btn5=Button(root3, bg="white", command = clear1, image=clear, bd=0,
relief="flat")
        login_btn5.place(x=420, y=420)

    login_btn4=Button(root3, bg="white", command = Generar, image=login_button, bd=0,
relief="flat")
    login_btn4.place(x=180, y=420)


def facebook():
    chrome_options = webdriver.ChromeOptions()
    chrome_options.add_argument("disable-notifications")
    driver = webdriver.Chrome(chrome_options=chrome_options)
```

```
        driver.get("https://www.facebook.com/SENA/")

def twitter():
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_argument("disable-notifications")
        driver = webdriver.Chrome(chrome_options=chrome_options)
        driver.get("https://twitter.com/SENAComunica")

def instagram():
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_argument("disable-notifications")
        driver = webdriver.Chrome(chrome_options=chrome_options)
        driver.get("https://www.instagram.com/accounts/login/?next=/senacomunica/")

root = Tk()
root.geometry("1350x730")
root.title("Login Reconocimiento de Voz")
text = StringVar()


fb_icon=PhotoImage(file="icon\\facebook.png")
instagram_icon=PhotoImage(file="icon\\instagram.png")
twitter_icon=PhotoImage(file="icon\\twitter.png")
login_button=PhotoImage(file="button\\login.png")
login_image=PhotoImage(file="image\\login_bg2.png")
login_image2=PhotoImage(file="image\\login_bg3.png")
bkg_image=PhotoImage(file="image\\dock2.png")
clear=PhotoImage(file="button\\clear.png")
```



```
Micro=PhotoImage(file="icon\\micro.png")

Add=PhotoImage(file="button\\add.png")


main_gb = Label(root, image=bkg_image)

main_gb.pack()


login_bg=Label(main_gb, image=login_image, bd=0)

login_bg.place(x=95, y=20)

login_frame =Frame(login_bg, bg="white", height=600, width=400)

login_frame.place(x=40, y=20)

login_info = Label(login_frame, text="Ingreso Por Reconocimiento de Voz", bg="white",
font=("Comic Sans Ms", 15, "bold"))

login_info.place(x=40, y=60)

user_label = Label(login_frame, text="", bg="white",fg="black", font=("Comic Sans Ms", 10,
"bold"))

user_label.place(x=50, y=150)

user_entry = Entry(login_frame, relief="flat",width=37, fg="black", font=("Comic Sans Ms", 10,
"bold"))

user_entry.place(x=50, y=180)



pass_label = Label(login_frame, text="", bg="white",fg="black", font=("Comic Sans Ms", 10,
"bold"))

pass_label.place(x=50, y=220)

pass_entry = Entry(login_frame, relief="flat",width=37, fg="black", font=("Comic Sans Ms", 10,
"bold"),show="*")

pass_entry.place(x=50, y=250)


rem_box = Button(login_frame,text="¿Registrarse?", bg="white",command=check,
state="normal", relief="flat", font=("Comic Sans Ms", 12, "bold"))
```

rem_box.place(x=130, y=280)

rem_box = Button(login_frame,text="¿Generar Listado?", bg="white",command=Listado, state="normal", relief="flat", font=("Comic Sans Ms", 12, "bold"))

rem_box.place(x=120, y=310)

login_btn2=Button(login_frame, bg="white", command=submit_2, image=Micro, bd=0, relief="flat")

login_btn2.place(x=160, y=160)

share_frame=Frame(login_frame, bg="white", height=50, width=500)

share_frame.place(x=0, y=380)

margin=Label(share_frame, bg="white", width=12)

margin.pack(side=LEFT)

fb_ico=Button(share_frame, bg="white", command=facebook, image=fb_icon, bd=0, relief="flat")

fb_ico.pack(side=RIGHT)

margin=Label(share_frame, bg="white", width=3)

margin.pack(side=RIGHT)

twitter_ico=Button(share_frame, bg="white",command=twitter, image=twitter_icon, bd=0, relief="flat")

twitter_ico.pack(side=RIGHT)

margin=Label(share_frame, bg="white", width=3)

margin.pack(side=RIGHT)

instagram_ico=Button(share_frame, bg="white",command=instagram, image=instagram_icon, bd=0, relief="flat")

instagram_ico.pack(side=RIGHT)

margin=Label(share_frame, bg="white", width=3)

margin.pack(side=RIGHT)

speak("Hola, Bienvenido al programa de reconocimiento de voz, aprendiz del sena.")

root.mainloop()

Anexamos dos carpetas con todo el código fuente y una breve descripción de los dos prototipos.