**Pseudocódigo Jurassic Park**

Módulo cliente\_app

Importar el módulo 'ventanas'

Importar el módulo 'menus'

Importar la función 'system' del módulo 'os'

Inicializar la variable 'opc\_menu\_cli' con una cadena vacía

Crear un objeto 'menu\_cliente' de la clase 'Menu' del módulo 'menus'

Mientras 'opc\_menu\_cli' sea diferente de "3", hacer lo siguiente:

Llamar a la función 'system' con el argumento "cls" para limpiar la consola

Llamar al método 'menu\_clientes' del objeto 'menu\_cliente'

Leer la entrada del usuario y asignarla a 'opc\_menu\_cli'

Crear un objeto 'ventana' de la clase 'Ventana' del módulo 'ventanas'

Si 'opc\_menu\_cli' es igual a "1", llamar al método 'mapa' del objeto 'ventana'

Si 'opc\_menu\_cli' es igual a "2", llamar al método 'vademecum\_guias' del objeto 'ventana'

Módulo empleados

Procedimiento clear:

Si el sistema operativo es 'nt' (Windows) entonces

Ejecutar el comando del sistema 'cls'

Sino

Ejecutar el comando del sistema 'clear'

Fin Si

Fin Procedimiento

Clase Empleados:

Constructor(idn, nombre, apellidos, dni, puesto, salario):

Asignar idn a self.idn

Asignar nombre a self.nombre

Asignar apellidos a self.apellidos

Asignar dni a self.dni

Asignar puesto a self.puesto

Asignar salario a self.salario

Fin Clase

Clase GestorEmpleado:

Constructor():

Inicializar self.empleados como una lista vacía

Procedimiento registra\_empleado(idn, nombre, apellidos, dni, puesto, salario):

Crear un nuevo objeto Empleados con los parámetros proporcionados

Añadir el nuevo empleado a la lista self.empleados

Procedimiento modifica\_empleado():

Solicitar al usuario el ID del empleado a modificar

Para cada empleado en self.empleados:

Si el ID del empleado coincide con el ID proporcionado por el usuario entonces

Mostrar todos los atributos y sus valores del empleado

Solicitar al usuario qué atributo desea modificar

Según el atributo seleccionado por el usuario:

Si el atributo es "1" (Nombre) entonces

Solicitar al usuario el nuevo nombre y asignarlo al empleado

Si el atributo es "2" (Apellidos) entonces

Solicitar al usuario los nuevos apellidos y asignarlos al empleado

Si el atributo es "3" (DNI) entonces

Solicitar al usuario el nuevo DNI y asignarlo al empleado

Si el atributo es "4" (Puesto) entonces

Solicitar al usuario el nuevo puesto y asignarlo al empleado

Si el atributo es "5" (Salario) entonces

Solicitar al usuario el nuevo salario y asignarlo al empleado

Fin Según

Fin Si

Fin Para

Fin Procedimiento

Fin Clase

Módulo Jurassic Park

Importar los módulos necesarios

Definir la función clear:

Si el sistema operativo es 'nt' (Windows) entonces

Ejecutar el comando del sistema 'cls'

Sino

Ejecutar el comando del sistema 'clear'

Fin Si

Fin Procedimiento

Inicializar las variables idn, id\_visitante, id\_alerta, bd, opc\_menu\_prin, tipo\_entrada

Crear objetos de las clases GestorEmpleado, GestorVisitantes, Menu y BaseDatos

Llamar al método crea\_tabla de base\_datos

Mientras True:

Llamar a la función clear

Llamar al método menu\_principal de el\_menu

Solicitar al usuario la opción del menú principal

Si la opción del menú principal es "1":

Llamar al método menu\_gestor\_empleados de el\_menu

Solicitar al usuario la opción del menú de empleados

Si la opción del menú de empleados es "1":

Solicitar al usuario los datos del nuevo empleado

Llamar al método registra\_empleado de new\_gestion\_emple con los datos del nuevo empleado

Llamar al método almacena\_info de base\_datos con los datos del nuevo empleado

Si la opción del menú de empleados es "2":

Crear un objeto de la clase Tablas

Llamar al método tabla\_empleados de tabla\_empleado con base\_datos como argumento

Si la opción del menú de empleados es "3":

Solicitar al usuario el ID del empleado a modificar

Intentar llamar al método modif\_info de base\_datos con el ID del empleado

Si ocurre una excepción, mostrar un mensaje de error al usuario

Si la opción del menú de empleados es "4":

Solicitar al usuario el ID del empleado a buscar

Intentar llamar al método busca\_info de base\_datos con el ID del empleado

Si ocurre una excepción, mostrar un mensaje de error al usuario

Si la opción del menú de empleados es "5":

Solicitar al usuario el ID del empleado a eliminar

Intentar llamar al método elimina\_info de base\_datos con el ID del empleado

Si ocurre una excepción, mostrar un mensaje de error al usuario

Si la opción del menú de empleados es "6":

Crear un objeto de la clase Horario con lista\_empleados como argumento

Llamar al método crea\_horario de tabla\_horario

Llamar al método consultar\_horario de tabla\_horario

Si la opción del menú principal es "2":

Llamar al método menu\_gestor\_visitantes de el\_menu

Solicitar al usuario la opción del menú de visitantes

Si la opción del menú de visitantes es "1":

Solicitar al usuario los datos del nuevo visitante

Llamar al método registro\_visit de new\_gestion\_visit con los datos del nuevo visitante

Llamar al método almacena\_info de base\_datos con los datos del nuevo visitante

Si la opción del menú de visitantes es "2":

Crear un objeto de la clase Tablas

Llamar al método tabla\_visitantes de tabla\_visitantes con base\_datos como argumento

Si la opción del menú de visitantes es "3":

Crear un objeto de la clase Tablas

Llamar al método informe de tabla\_visitantes con base\_datos como argumento

Si la opción del menú principal es "3":

Llamar al método menu\_gestor\_instalaciones de el\_menu

Solicitar al usuario la opción del menú de instalaciones

Si la opción del menú de instalaciones es "1":

Solicitar al usuario los datos de la nueva alerta

Llamar al método almacena\_info de base\_datos con los datos de la nueva alerta

Si la opción del menú de instalaciones es "2":

Crear un objeto de la clase Tablas

Llamar al método tabla\_instalaciones de tabla\_instalaciones con base\_datos como argumento

Si la opción del menú de instalaciones es "3":

Solicitar al usuario el ID de la alerta a eliminar

Intentar llamar al método elimina\_info de base\_datos con el ID de la alerta

Si ocurre una excepción, mostrar un mensaje de error al usuario

Si la opción del menú principal es "4":

Preguntar al usuario si desea guardar los cambios

Si el usuario desea guardar los cambios:

Llamar al método commit de base\_datos.conn

Llamar al método close de base\_datos.conn

Mostrar un mensaje de confirmación al usuario

Salir del programa

Fin Mientras

Módulo login

Importar los módulos necesarios

Crear una ventana de Tkinter con un tamaño de 1200x900, un título "Jurasic Park Soft" y un color de fondo beige

Definir la función abre\_menu\_empleado:

Destruir la ventana de Tkinter

Ejecutar el script de Python 'Jurassic Park.py' con subprocess.run

Definir la función abre\_menu\_cliente:

Destruir la ventana de Tkinter

Ejecutar el script de Python 'cliente\_app.py' con subprocess.run

Definir la función login\_empleado:

Obtener el nombre de usuario y la contraseña de las entradas correspondientes

Conectar a la base de datos 'usuarios.db' con sqlite3

Crear una tabla 'usuarios' si no existe, con las columnas 'usuario' y 'contraseña'

Insertar algunos usuarios y contraseñas en la tabla 'usuarios'

Verificar si el nombre de usuario y la contraseña están en la base de datos

Si las credenciales son válidas, llamar a la función abre\_menu\_empleado

Sino, mostrar un mensaje de error en label\_error

Reproducir la música 'music\_login.mp3' con mi\_music.music

Crear un Frame para la imagen con un tamaño de 900x800 y un color de fondo beige

Cargar la imagen de fondo y redimensionarla a 900x800

Convertir la imagen a un formato adecuado para Tkinter con ImageTk.PhotoImage

Mostrar la imagen en un Label

Crear un Frame para el formulario con un color de fondo "#67604F"

Crear Labels y Entries para el nombre de usuario y la contraseña

Crear un Button para iniciar sesión, que llama a la función login\_empleado cuando se presiona

Crear un Label para mostrar mensajes de error

Crear un Frame para el botón del cliente con un color de fondo beige

Crear un Button que dice "Pincha aquí si eres cliente del parque", que llama a la función abre\_menu\_cliente cuando se presiona

Iniciar el bucle principal de Tkinter con root.mainloop

Módulo menus

Clase Menu:

Procedimiento menu\_principal:

Limpiar la pantalla

Mostrar "MENÚ PRINCIPAL" y las opciones del menú principal

Procedimiento menu\_gestor\_empleados:

Limpiar la pantalla

Mostrar "GESTOR DE EMPLEADOS" y las opciones del menú de empleados

Procedimiento menu\_gestor\_visitantes:

Limpiar la pantalla

Mostrar "GESTOR DE VISITANTES" y las opciones del menú de visitantes

Procedimiento menu\_gestor\_instalaciones:

Limpiar la pantalla

Mostrar "GESTOR DE INSTALACIONES" y las opciones del menú de instalaciones

Procedimiento menu\_clientes:

Limpiar la pantalla

Mostrar "BIENVENIDO A JURASSIC PARK" y las opciones del menú de clientes

Módulo mi\_music

Procedimiento music(cancion):

1. Inicializar el mezclador de música de pygame

2. Definir la ruta relativa al archivo MP3 desde el script Python como 'music/' concatenado con cancion

3. Obtener la ruta absoluta del directorio donde se encuentra el script Python

4. Definir la ruta absoluta del archivo MP3 como la combinación de la ruta del script y la ruta del archivo MP3

5. Cargar la música desde la ruta absoluta del archivo MP3

6. Reproducir la música

Fin del Procedimiento

Módulo tablas

Clase BaseDatos:

Procedimiento \_\_init\_\_:

1. Conectar a la base de datos SQLite en la ruta especificada

2. Definir la ruta relativa al archivo de la base de datos

3. Obtener la ruta absoluta del directorio donde se encuentra el script Python

4. Definir la ruta absoluta del archivo de la base de datos

5. Conectar a la base de datos SQLite en la ruta absoluta

6. Crear un cursor para la conexión de la base de datos

Procedimiento crea\_tabla:

1. Definir las sentencias SQL para crear las tablas 'empleados', 'visitantes' e 'instalaciones' si no existen

2. Ejecutar las sentencias SQL

3. Confirmar los cambios en la base de datos

Procedimiento almacena\_info(opc\_menu, \*atributos):

1. Definir un nuevo registro con los atributos proporcionados

2. Si opc\_menu es "1", insertar el nuevo registro en la tabla 'empleados'

3. Si opc\_menu es "2", insertar el nuevo registro en la tabla 'visitantes'

4. Si opc\_menu es "3", insertar el nuevo registro en la tabla 'instalaciones'

Procedimiento busca\_info(id):

1. Ejecutar una consulta SQL para seleccionar todos los registros de la tabla 'empleados' donde ID\_empleado es igual a id

2. Obtener el primer registro de la consulta

3. Imprimir todos los datos del registro

4. Esperar a que el usuario presione Enter para continuar

Procedimiento elimina\_info(opc\_menu, id):

1. Si opc\_menu es "1", ejecutar una sentencia SQL para eliminar el registro de la tabla 'empleados' donde ID\_empleado es igual a id

2. Si opc\_menu es "3", ejecutar una sentencia SQL para eliminar el registro de la tabla 'instalaciones' donde ID\_alerta es igual a id

3. Si no se eliminó ningún registro, imprimir un mensaje indicando que el registro no existe

4. Si se eliminó un registro, imprimir un mensaje indicando que el registro se eliminó correctamente

5. Esperar a que el usuario presione Enter para continuar

Procedimiento modif\_info(id):

1. Ejecutar una consulta SQL para seleccionar todos los registros de la tabla 'empleados' donde ID\_empleado es igual a id

2. Pedir al usuario que introduzca el atributo que quiere cambiar y su nuevo valor

3. Ejecutar una sentencia SQL para actualizar el atributo del registro en la tabla 'empleados' donde ID\_empleado es igual a id

4. Imprimir un mensaje indicando que el registro se modificó correctamente

5. Esperar a que el usuario presione Enter para continuar

Fin de la Clase

Clase Tablas:

Procedimiento \_\_init\_\_:

1. Crear una tabla PrettyTable

2. Limpiar las filas de la tabla

Procedimiento muestra\_tabla(base\_datos, campo):

1. Ejecutar una consulta SQL para seleccionar todos los registros de la tabla especificada

2. Obtener todos los registros de la consulta

3. Para cada registro en los registros, añadir el registro a la tabla

4. Imprimir la tabla

5. Esperar a que el usuario presione Enter para continuar

Procedimiento tabla\_empleados(base\_datos):

1. Definir los nombres de las columnas de la tabla

2. Llamar al procedimiento muestra\_tabla con la base de datos y "empleados" como argumentos

Procedimiento tabla\_visitantes(base\_datos):

1. Definir los nombres de las columnas de la tabla

2. Llamar al procedimiento muestra\_tabla con la base de datos y "visitantes" como argumentos

Procedimiento tabla\_instalaciones(base\_datos):

1. Definir los nombres de las columnas de la tabla

2. Llamar al procedimiento muestra\_tabla con la base de datos y "instalaciones" como argumentos

Procedimiento informe(base\_datos):

1. Definir los nombres de las columnas de la tabla

2. Ejecutar una consulta SQL para seleccionar el tipo de entrada, el número de visitantes y los ingresos de la tabla "visitantes", agrupados por el tipo de entrada

3. Obtener todos los registros de la consulta

4. Para cada registro en los registros, añadir el registro a la tabla

5. Imprimir la tabla

6. Ejecutar una consulta SQL para contar el número de visitantes

7. Obtener el primer registro de la consulta y asignarlo a total\_clientes

8. Ejecutar una consulta SQL para sumar el precio de todas las entradas

9. Obtener el primer registro de la consulta y asignarlo a total\_ingresos

10. Imprimir el número total de clientes y los ingresos totales

11. Esperar a que el usuario presione Enter para continuar

Fin de la Clase

Clase Horario hereda de Tablas:

Procedimiento \_\_init\_\_(lista\_empleados):

1. Llamar al constructor de la clase padre

2. Imprimir "HORARIO"

3. Definir las horas y la lista de empleados

4. Crear una tabla de horario vacía y una lista para los nombres de los empleados

5. Crear una matriz para las tareas

6. Llamar al procedimiento crea\_tabla\_horario

Procedimiento crea\_tabla\_horario:

1. Para cada empleado en la lista de empleados, añadir el nombre del empleado a la fila de nombres de empleados

2. Definir los nombres de las columnas de la tabla con la fila de nombres de empleados

3. Para cada hora en las horas, añadir una fila a la tabla con la franja horaria

Procedimiento crea\_horario:

1. Limpiar la matriz de tareas

2. Para cada empleado en la lista de empleados, añadir el nombre del empleado a la fila de nombres de empleados

3. Definir los nombres de las columnas de la tabla con la fila de nombres de empleados

4. Para cada hora en las horas, crear una lista de tareas con la franja horaria y pedir al usuario que introduzca la tarea de cada empleado en esa franja horaria

5. Añadir la lista de tareas a la tabla y a la matriz de tareas

Procedimiento consultar\_horario:

1. Imprimir la tabla

2. Esperar a que el usuario presione Enter para continuar

Fin de la Clase

Módulo ventanas

Clase Ventana:

Método inicializar:

Crear una nueva ventana con fondo beige y tamaño 900x600

Crear un nuevo marco en la ventana con fondo beige y tamaño 300x200

Empaquetar el marco en la ventana

Método ruta\_absoluta(foto, tipo\_imagen='dinosaurios'):

Definir la ruta del archivo como la unión de 'images', 'tipo\_imagen' y 'foto'

Definir la ruta del script como el directorio del script Python actual

Definir la ruta absoluta como la unión de la ruta del script y la ruta del archivo

Devolver la ruta absoluta

Método mapa:

Crear un objeto de música con 'music\_mapa.mp3'

Definir la fuente del mapa como "Sitka Display" de tamaño 36

Definir la ruta de la imagen del mapa como la ruta absoluta de 'mapa\_guia.jpg' en 'mapas'

Abrir la imagen del mapa y redimensionarla a 450x600

Convertir la imagen del mapa a un formato adecuado para tkinter

Definir la ruta de la imagen de la leyenda como la ruta absoluta de 'leyenda\_mapa.jpg' en 'mapas'

Abrir la imagen de la leyenda y convertirla a un formato adecuado para tkinter

Crear una etiqueta para mostrar la imagen del mapa y posicionarla en la fila 1, columna 1

Crear una etiqueta para mostrar la imagen de la leyenda y posicionarla en la fila 0, columna 0

Crear una etiqueta para mostrar el título "Mapa Parque Jurásico" y posicionarla en la fila 0, columna 1

Configurar el fondo de la etiqueta del mapa como beige

Iniciar el bucle principal de la ventana

Módulo visitantes

Clase Visitantes:

Constructor(id\_visitante, nombre\_visitante, apellidos\_visitante, dni\_visitante, edad, tipo\_entrada, precio):

Asignar id\_visitante a self.id\_visitante

Asignar nombre\_visitante a self.nombre\_visitante

Asignar apellidos\_visitante a self.apellidos\_visitante

Asignar dni\_visitante a self.dni\_visitante

Asignar edad a self.edad

Asignar tipo\_entrada a self.tipo\_entrada

Asignar precio a self.precio

Fin Clase

Clase GestorVisitantes:

Constructor():

Inicializar self.visitantes como una lista vacía

Procedimiento registro\_visit(id\_visitante, nombre\_visitante, apellidos\_visitante, dni\_visitante, edad, tipo\_entrada, precio):

Crear un nuevo objeto Visitantes con los parámetros proporcionados

Añadir el nuevo visitante a la lista self.visitantes

Procedimiento consultar\_visit():

Para cada visitante en self.visitantes:

Imprimir el visitante

Fin Para

Fin Procedimiento

Fin Clase

Clase Ventana:

Método vademecum\_guias:

Reproducir la música 'music\_guia.mp3'

Definir una función interna crea\_imagen que toma un camino de imagen:

Abrir la imagen en el camino dado

Redimensionar la imagen a 275x275

Convertir la imagen a un formato adecuado para tkinter

Devolver la imagen convertida

Definir los caminos de las imágenes para varios dinosaurios usando el método ruta\_absoluta

Crear las imágenes para los dinosaurios usando la función crea\_imagen

Para cada dinosaurio:

Crear una etiqueta para mostrar la imagen del dinosaurio y posicionarla en el marco

Crear una etiqueta para mostrar la descripción del dinosaurio y posicionarla en el marco

Iniciar el bucle principal de la ventana

Crear una instancia de la clase Ventana