

EVIDENCIA CALIFICABLE # 3

Nombre la unidad de aprendizaje: Estructura de Datos y su Procesamiento

Licenciado en Tecnologías de Información

Semestre: 3 Grupo: 31

Nombre del profesor: Álvaro Francisco Salazar Gonzales

Nombre de los integrantes del equipo:

|  |  |
| --- | --- |
| **Nombre** | **Matricula** |
| Acosta Guzman Raymundo Tadeo | 2029973 |
| Barrientos Agüero Debanhi Nohemí | 2003270 |
| Euresti Cantú Ileana Aidé | 1995474 |
| Tello Salinas Diego Gerardo | 1996652 |

|  |  |
| --- | --- |
| **Contenido mínimo a evaluar** | **Cumplimiento** |
| **Índice** |  |
| **Introducción .- incluye valores UANL aplicados** |  |
| **Análisis y emisión de juicio** |  |
| **Conclusiones individuales** |  |
| **Conclusión del equipo** |  |
| **Identificación de sub resultados de aprendizaje ANECA.** |  |
| **Calificación Evidencia:** |  |
| **Firma del maestro** | |

San Nicolás de los Garza, ciudad universitaria (02/11/2022)

Estructura de Datos y su Procesamiento

Evidencia # 3

Licenciado en tecnologías de información

Nombre de los integrantes del equipo

|  |  |
| --- | --- |
| **Nombre** | **Matricula** |
| Acosta Guzman Raymundo Tadeo  **8** | 2029973 |
| Barrientos Agüero Debanhi Nohemí  **C:\Users\joart\AppData\Local\Microsoft\Windows\INetCache\Content.Word\9.jpg** | 2003270 |
| Euresti Cantú Ileana Aidé  **10** | 1995474 |
| Tello Salinas Diego Gerardo | 1996652 |

**Índice**

**Lista de objetivos por cumplir …………………………………………….4**

**Código………………………………………………………………………….5**

**Explicación del código……………………………………………………20**

**Capturas de pantalla ………………………………………………………22**

**Aportaciones de cada integrante………………………………………25**

**Conclusión grupal…………………………………………………………26**

**Individual……………………………………………………………………26**

**URL…………………………………………………………………………..27**

**Lista de objetivos por cumplir**

Un listado de objetivos específicos del problema (requerimientos) a cumplir por parte del proyecto.

ii) Código Python de la solución que incluya números de línea.

iii) Explicación textual de la aplicación de estructuras de datos (Listas, tuplas, diccionarios,

conjuntos, etc.) para proveer la funcionalidad solicitada refiriendo por números de línea las

secciones de código que la soportan.

iv) Capturas de pantalla de la interfaz de usuario para el script y su secuencia de uso para

demostrar el cumplimiento de la funcionalidad indicada.

v) Una sección donde indique la aportación específica de cada integrante del equipo en cuanto

al proyecto y/o documento.

vi) Una sección de conclusiones donde se presente la conclusión grupal después de realizar el

proyecto y una reflexión acerca de mejoras futuras que se le pudieran realizar.

vii) URL sintética para acceder al repositorio GIT en caso de ser necesaria su validación.

j) No deberá presentar faltas de ortografía.

k) Deberá incluir un índice que omita las portadas y el índice en sí, debiendo iniciar en la primera.

**Código**

# from modulos.layouts import +

#from modulos.randomKey import randomKey

# from modulos.date import +

#from modulos.randomKey import randomKey

#from .date import +

from time import strftime

from datetime import datetime

from collections import namedtuple

import xlsxwriter

import sqlite3

import os

import csv

listaClient = []

listaSala = []

reservaciones = []

GlbClient = None

db = 'coworking.sqlite3'

cnx = None

if not os.path.exists(db):

    cnx = sqlite3.connect(db)

    cnx.execute(

        "CREATE TABLE IF NOT EXISTS Clientes (id INTEGER PRIMARY KEY AUTOINCREMENT, nombre TEXT NOT NULL);")

    cnx.execute(

        "CREATE TABLE IF NOT EXISTS Salas (id INTEGER PRIMARY KEY AUTOINCREMENT, nombre TEXT NOT NULL, limite INTEGER NOT NULL, turno TEXT NOT NULL);")

    cnx.execute(

        "CREATE TABLE IF NOT EXISTS Reservaciones (id INTEGER PRIMARY KEY AUTOINCREMENT, fecha TEXT NOT NULL,id\_cliente INTEGER NOT NULL, id\_sala INTEGER NOT NULL, nombre\_evento TEXT NOT NULL);")

else:

    cnx = sqlite3.connect(db)

# menu()

# Partes del menu

# Reservaciones

# Registrar a un nuevo cliente

def newClient(idClient):

    idClient = idClient

    nameClient = input("Nombre del Cliente: ")

    return idClient, nameClient  # regresa lista (idClient, nombre)

# Registrar una sala

def newSala(idSala):

    idSala = idSala

    nameSala = input("Nombre de la Sala: ")

    capacidad = input("Capacidad de la Sala: ")

    # regresa lista (idSala, nombre de la sala, capacidad)

    return idSala, nameSala, capacidad

# Registrar reservacion de una sala

def newReservacion(listaClient, listaSala, reservaciones):

    nombreCliente = input("\nNombre del Cliente: ")

    if(validacionCliente(listaClient, nombreCliente)):

        nombreSala = input("Nombre de la Sala: ")

        if(validacionSala(listaSala, nombreSala)):

            nombreEvento = input("Nombre del evento: ")

            fechaEvento = input(

                "Fecha del evento en formato d/m/y. Ejemplo: 20/04/2019: ")

            fechaEvento = conversionFecha(fechaEvento)

            if(validacionDias(fechaEvento.day)):

                print("")

                """

                turno = input(

                    "Que turno desea reservar, Formato para turno, 'mañana, 'tarde', noche: ")

                if(turno == 'mañana' or turno == 'tarde' or turno == 'noche'):

                    # ('sala2', 'tarde', '18/09/2022', lista)

                    if(disponibilidadSala(nombreSala, turno, strFecha(fechaEvento), reservaciones) != True):

                        folio = randomKey()

                        return nombreCliente, nombreSala, nombreEvento, strFecha(fechaEvento), turno, folio

                    else:

                        print("sala no disponoble en ese turno en esa fecha\n")

                else:

                    print("formato del turno incorrecto\n")

                """

    return False

# a)    La reserva de la sala se debe hacer, por lo menos, dos días antes

def validacionDias(fechaApartada):

    if(fechaUnDia() != fechaApartada and fechaActual() != fechaApartada and fechaApartada > fechaActual()):

        print("Fecha disponible")

        return True

    else:

        print("Fecha no disponible.\n")

        print("Favor de reservar con dos dias de anticipacion\n")

        return False

# b)    Solamente pueden reservar una sala aquellos que son clientes registrados

def validacionCliente(listaClient, nameClient):

    if(len(listaClient) != 0):

        strClients = str(listaClient).strip("[]")

        strName = str(nameClient).strip("[]")

        if ",".join(strName) in ",".join(strClients):

            print("Cliente encontrado\n")

            return True

        else:

            print("cliente no registrado\n")

            return False

    else:

        print("ningun Cliente registrado\n")

        return False

# validacion de sala existente

def validacionSala(listaSala, nameSala):

    if(len(listaSala) != 0):

        strClients = str(listaSala).strip("[]")

        strName = str(nameSala).strip("[]")

        if ",".join(strName) in ",".join(strClients):

            print("Sala encontrada")

            return True

        else:

            print("Sala no registrada\n")

    else:

        print("ningun Sala registrada\n")

        return False

# validacion de sala con disponibilidad

def disponibilidadSala(sala, turno, fecha, listaReservacion):

    if(len(listaReservacion) != 0):

        for itemReservacion in listaReservacion:

            stritemReservacion = str(itemReservacion).strip("[]")

            strNameSala = str(sala).strip("[]")

            strFecha = str(fecha).strip("[]")

            strTurno = str(turno).strip("[]")

            if ",".join(strFecha) in ",".join(stritemReservacion):

                if ",".join(strNameSala) in ",".join(stritemReservacion):

                    if ",".join(strTurno) in ",".join(stritemReservacion):

                        print('turno encontrado')

                        print(itemReservacion)

                        return True

                    else:

                        return False

                else:

                    return False

            else:

                return False

    else:

        # print("Ninguna reservacion\n")

        return False

# Consultar las reservaciones existentes para una fecha específica.

def consultaReservaciones(reservaciones):

    fecha = input("que fecha deseas consultar: ")

    fecha = conversionFecha(fecha)

    fecha = strFecha(fecha)

    print("++ Reporte de Reservaciones para el dia ", fecha, " ++++\n")

    print("Cliente\t", "Sala\t", "Evento\t", "Turno\t",)

    if(len(reservaciones) != 0):

        for itemReservacion in reservaciones:

            stFecha = str(fecha).strip("[]")

            if stFecha in ",".join(itemReservacion):

                print(itemReservacion[0], "\t", itemReservacion[1],

                      "\t", itemReservacion[2], "\t", itemReservacion[4])

            else:

                print("error")

        print("++++ FIN DEL REPORTE ++++++\n")

    else:

        print("No hay reservaciones aun")

        return False

# consultar la disponibilidad de las salas para una fecha especifica

def disponibilidadSalasfecha(reservaciones, salas):

    fecha = input("que fecha deseas consultar: ")

    fecha = conversionFecha(fecha)

    fecha = strFecha(fecha)

    print("++ Reporte de Reservaciones para el dia ", fecha, " ++++\n")

    print("Sala\t", "Turno\t")

    if(len(reservaciones) != 0):

        for itemSalas in salas:

            for itemReservacion in reservaciones:

                stFecha = str(fecha).strip("[]")

                stsala = str(itemSalas[1]).strip("[]")

                if stFecha in ",".join(itemReservacion) and stsala in ",".join(itemReservacion):

                    if not 'mañana' in ",".join(itemReservacion):

                        print(stsala, " Matutino")

                    if not 'tarde' in ",".join(itemReservacion):

                        print(stsala, " Tarde")

                    if not 'noche' in ",".join(itemReservacion):

                        print(stsala, " Noche")

                    print(itemReservacion)

                elif(not stsala in ",".join(itemReservacion)):

                    print(stsala, " Matutino")

                    print(stsala, " Tarde")

                    print(stsala, " Noche")

                else:

                    break

        #     if stFecha in ",".join(itemReservacion):

        #         print(itemReservacion[0], "\t", itemReservacion[1], "\t", itemReservacion[2], "\t", itemReservacion[4])

        #     else:

        #         print("error")

        # print("++++ FIN DEL REPORTE ++++++\n")

    else:

        print("No hay reservaciones aun")

        return False

# Editar evento existente

def editEvento(reservaciones):

    elemento = input("Nombre del evento a modificar: ")

    nuevoElemento = input("Nuevo Nombre: ")

    if(len(reservaciones) != 0):

        for itemReservacion in reservaciones:

            strElement = str(elemento).strip("[]")

            if strElement in ",".join(itemReservacion):

                copia = list(itemReservacion)

                copia[2] = nuevoElemento

                itemReservacion = tuple(copia)

                print(itemReservacion)

                print("Nombre cambiado")

                return reservaciones

            else:

                print("error")

    else:

        print("No hay reservaciones aun")

        return False

# Excel

def excelExport(reservaciones):

    print(reservaciones)

    Auto = namedtuple("Reservas", "sala, clientes, evento, turno")

    datos\_a\_grabar = dict()

    e = 0

    for strElement in reservaciones:

        datos\_a\_grabar[e] = Auto(

            strElement[1], strElement[0], strElement[2], strElement[4])

        print(datos\_a\_grabar)

        e = e + 1

    # Paso 3: Abrir, en modo de escritura, el archivo destino

    archivo = open("eventos.csv", "w", newline="")

    # Paso 4: Establecer una salida de escritura

    grabador = csv.writer(archivo)

    # Paso 5: Grabar el encabezado (OPCIONAL)

    grabador.writerow(("Clave", "Sala", "Cliente", "Evento", "Turno"))

    # Paso 6: Iterar sobre los elementos de los datos a grabar o bien pedir de golpe que se graben todos los elementos

    grabador.writerows([(clave, datos.sala, datos.clientes, datos.evento, datos.turno)

                       for clave, datos in datos\_a\_grabar.items()])

    archivo.close()

def excelImport():

    # Solamente si se está trabajando en otro código diferente al de la creación

    Auto = namedtuple("Reservas", "sala, clientes, evento, turno")

    datos\_a\_leer = dict()

    with open("eventos.csv", "r", newline="") as archivo:

        lector = csv.reader(archivo)

        next(lector)

        lista = []

        for clave, sala, clientes, evento, turno in lector:

            datos\_a\_leer[int(clave)] = Auto(sala, clientes, evento, turno)

            print(datos\_a\_leer)

    listOfValues = datos\_a\_leer.values()

    listOfValues = list(listOfValues)

fecha = datetime.now()

dt\_string = fecha.strftime("%d/%m/%Y %H:%M:%S")

# print("date and time =", dt\_string)

def fechaDosDias():

    fechaRes = fecha.day + 2

    return fechaRes

def fechaUnDia():

    diaActual = fecha.day

    return diaActual + 1

def fechaActual():

    diaActual = fecha.day

    return diaActual

def conversionFecha(fechaConvertida):

    # una\_fecha = '20/04/2019'

    fecha\_dt = datetime.strptime(fechaConvertida, '%d/%m/%Y')

    return fecha\_dt

def strFecha(fechastr):

    stringFecha = fechastr.strftime("%d/%m/%Y")

    return stringFecha

def randomKey():

    numero = 100

    folio = format(id(numero), "x")

    return folio

def validaInput(str):

    if not str:

        input("Este Valor no Puede Omitirse")

    else:

        return str

# Funcion para agregar un cliente a la cnx

def agregarCliente(nombre):

    cnx.cursor().execute('''INSERT INTO Clientes(nombre)

                  VALUES(?)''', (nombre,))

    cnx.commit()

    id = cnx.cursor().execute(

        '''SELECT MAX(id) id from Clientes''').fetchone()[0]

    return id

# Funcion para agregar una sala a la cnx

def agregarSala(nombre, ocupacion):

    for turno in ['Matutino', 'Vespertino', 'Nocturno']:

        cnx.cursor().execute('''INSERT INTO Salas(nombre,limite,turno)

                        VALUES(?,?,?)''', (nombre, ocupacion, turno))

    cnx.commit()

# Funcion para agregar una reservacion a la cnx

def agregarReservacion(nombre, fecha, sala, client):

    cnx.cursor().execute('''INSERT INTO Reservaciones(nombre\_evento,fecha,id\_sala,id\_cliente)

                  VALUES(?,?,?,?)''', (nombre, fecha, sala, client))

    cnx.commit()

    folio = cnx.cursor().execute(

        '''SELECT MAX(id) id from Reservaciones''').fetchone()[0]

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(

        "+" + f"{'Reserva con Folio [{folio}]' : >50} Generado".format(folio=id) + f"{'+' : >52}")

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    return folio

# Funcion para editar una reservacion

def editarReservacion(folio, nombre):

    cnx.cursor().execute(

        '''UPDATE Reservaciones SET nombre\_evento = ? WHERE id = ?''', (nombre, folio))

    cnx.commit()

# Funcion para eliminar una reservacion

def eliminarReservacion(folio):

    today = datetime.datetime.now()

    eventDay = cnx.cursor().execute(

        '''SELECT fecha FROM Reservaciones Where id = ? ''', (folio,)).fetchone()

    if eventDay:

        delta = convertirFecha(eventDay[0]) - today

        print(delta.days)

        if(delta.days >= 3):

            opcion = input("[+ ¿Desea Borrar la Reservacion? Y/N +]")

            if opcion == 'Y' or opcion == 'y':

                cnx.cursor().execute('''Delete FROM Reservaciones WHERE id = ?''', (folio,))

                cnx.commit()

            if opcion == 'N' or opcion == 'n':

                return

        else:

            print("[+ Es Necesario 3 Dias Anticipacion para Eliminar Reservacion +]")

# Funcion para mostrar salas disponibles dada una fecha en especifico

def mostrarSalasDisp(fecha):

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(

        "+" + f"{'Salas Disponibles {fecha}' : >50}".format(fecha=fecha) + f"{'+' : >58}")

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    salas = cnx.cursor().execute(

        "SELECT id, nombre, turno FROM salas WHERE id NOT IN(SELECT id\_sala FROM Reservaciones where fecha != ?) ", (fecha,)).fetchall()

    print(f"{'Sala' : <40}{'Nombre' : <40}{'Turno' : <40}")

    for sala in salas:

        print(f"{sala[0] : <40}{sala[1] : <40}{sala[2] : <40}")

    print("\n\n")

    return len(salas)

# Funcion para mostrar clientes

def mostrarClientesDisp():

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(

        "+" + f"{'Clientes Disponibles' : >50}" + f"{'+' : >68}")

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    clientes = cnx.cursor().execute(

        "SELECT id, nombre FROM Clientes ").fetchall()

    print(f"{'Id' : <40}{'Nombre' : <40}")

    for cliente in clientes:

        print(f"{cliente[0] : <40}{cliente[1] : <40}")

    print("\n\n")

    return len(clientes)

def mostrarReserDisp():

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(

        "+" + f"{'Reservas Disponibles' : >50}" + f"{'+' : >68}")

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    reservas = cnx.cursor().execute(

        "SELECT id, nombre\_evento FROM Reservaciones ").fetchall()

    print(f"{'Id' : <40}{'Nombre' : <40}")

    for reserva in reservas:

        print(f"{reserva[0] : <40}{reserva[1] : <40}")

    print("\n\n")

    return len(reservas)

# Funcion para mostrar reservaciones dada una fecha en especifico

def obtenerReporteReservacion(fecha):

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(

        "+" + f"{'Reservaciones {fecha}' : >50}".format(fecha=fecha) + f"{'+' : >58}")

    print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print(f"{'Folio' : <35}{'Cliente' : <35}{'Evento' : <35}{'Turno' : <35}")

    eventos = cnx.cursor().execute('''SELECT sala.nombre, client.nombre, reserva.nombre\_evento, sala.turno from Reservaciones reserva join Salas sala on reserva.id\_sala = sala.id join Clientes client on client.id = reserva.id\_cliente where reserva.fecha = ?''', (fecha,)).fetchall()

    for evento in eventos:

        print(

            f"{evento[0] : <35}{evento[1] : <35}{evento[2] : <35}{evento[3] : <35}")

    print("\n\n")

# Funcion para exportar reservaciones dada una fecha en especifico a xlsx

def exportarReporteReservacion(fecha):

    eventos = cnx.cursor().execute('''SELECT sala.nombre, client.nombre, reserva.nombre\_evento, sala.turno from Reservaciones reserva join Salas sala on reserva.id\_sala = sala.id join Clientes client on client.id = reserva.id\_cliente where reserva.fecha = ?''', (fecha,)).fetchall()

    workbook = xlsxwriter.Workbook('reservaciones.xlsx')

    worksheet = workbook.add\_worksheet()

    for i in range(0,  1):

        worksheet.write(i, 0, 'Folio')

        worksheet.write(i, 1, 'Cliente')

        worksheet.write(i, 2, 'Evento')

        worksheet.write(i, 3, 'Turno')

    for i, row in enumerate(eventos):

        for j, value in enumerate(row):

            worksheet.write(i + 1, j, value)

    workbook.close()

def esFechaValida(fecha):

    try:

        if not fecha:

            return False

        datetime.strptime(fecha, '%d/%m/%Y')

        return True

    except ValueError:

        return False

def menu():

    print("1.-Reservaciones")

    print("2.-Reportes")

    print("3.-Regsitrar cliente nuevo")

    print("4.-Registrar una sala")

    print("5.-salir")

    menuOpcion = int(input("Seleccione una opcion: "))

    if(menuOpcion == 1):

        # reservaciones

        print("1.-Registrar una reservacion")

        print("2.-Modificar una reservacion")

        print("3.-Cosultar disponibilidad de salas para una fecha")  # funcion nueva

        opcionResaervaciones = int(input("Selecione una opcion"))

        if(opcionResaervaciones == 1):

            """

            reservacion = newReservacion(listaClient, listaSala, reservaciones)

            if(reservacion != False):

                reservaciones.append(reservacion)

                print(reservaciones)

            """

            nombreEvento = ''

            fechaEvento = ''

            while not nombreEvento or not fechaEvento or not esFechaValida(fechaEvento):

                nombreEvento = validaInput(input("Nombre del evento: "))

                fechaEvento = validaInput(

                    input("Fecha del evento en formato d/m/y. Ejemplo: 20/04/2019: "))

            salas = mostrarSalasDisp(fechaEvento)

            if(salas > 0):

                sala = validaInput(input("Selecciona una Sala Disponible."))

            else:

                input("No hay Salas Disponibles para Reservar")

            clientes = mostrarClientesDisp()

            if(clientes > 0):

                cliente = validaInput(

                    input("Selecciona un Cliente Disponible."))

                agregarReservacion(nombreEvento, fechaEvento, sala, cliente)

            else:

                input("No hay Clientes Disponibles para Reservar")

            menu()

        elif(opcionResaervaciones == 2):

            # editEvento(reservaciones)

            folio = ''

            nombre = ''

            mostrarReserDisp()

            while not folio or not nombre:

                folio = validaInput(input("Ingresa folio a Editar."))

                nombre = validaInput(input("Ingresa Nombre a Editar."))

            editarReservacion(folio, nombre)

            menu()

        elif(opcionResaervaciones == 3):

            # nueva funcion disponibilidad de la salas para una fecha

            # disponibilidadSalasfecha(reservaciones, listaSala)

            fechaEvento = ''

            while not fechaEvento and not esFechaValida(fechaEvento):

                fechaEvento = validaInput(input("Fecha del evento en formato d/m/y. Ejemplo: 20/04/2019: "))

            mostrarSalasDisp(fechaEvento)

            menu()

        else:

            menu()

    elif(menuOpcion == 2):

        # Reportes

        print("1.-Reporte en pantalla de reservaciones de salas para una fecha")

        print("2.-Exportar reporte tabular en Excel")

        opcionReportes = int(input("Selecione una opcion"))

        if(opcionReportes == 1):

            # consultaReservaciones(reservaciones)

            fechaReporte = ''

            while not fechaReporte and not esFechaValida(fechaReporte):

                fechaReporte = validaInput(

                    input("Fecha de reserva en formato d/m/y. Ejemplo: 20/04/2019: "))

            obtenerReporteReservacion(fechaReporte)

            menu()

        elif(opcionReportes == 2):

            # nueva funcion exportar excel

            # excelExport(reservaciones) TODO

            fechaReporte = ''

            while not fechaReporte and not esFechaValida(fechaReporte):

                fechaReporte = validaInput(

                    input("Fecha de reserva en formato d/m/y. Ejemplo: 20/04/2019: "))

            exportarReporteReservacion(fechaReporte)

            menu()

            menu()

        else:

            menu()

        menu()

    elif(menuOpcion == 3):

        # Registro de clientes

        # cliente = newClient(randomKey())

        # listaClient.append(cliente)

        # print(listaClient)

        name = ''

        while not name:

            name = validaInput(input("Nombre del Cliente: "))

        id = agregarCliente(name)

        menu()

    elif(menuOpcion == 4):

        # Regristro de salas

        # sala = newSala(randomKey())

        # listaSala.append(sala)

        # print(listaSala)

        nameSala = ''

        capacidad = ''

        while not nameSala or not capacidad:

            nameSala = validaInput(input("Nombre de la Sala: "))

            capacidad = validaInput(input("Capacidad de la Sala: "))

        if int(capacidad) > 0:

            agregarSala(nameSala, int(capacidad))

            menu()

        else:

            input("La Capacidad de la Sala debe ser mayor a 0")

    elif(menuOpcion == 5):

        return 0

    else:

        print("++\tSeleccione una opcion valida\t++")

        menu()

menu()

**Explicación del código**

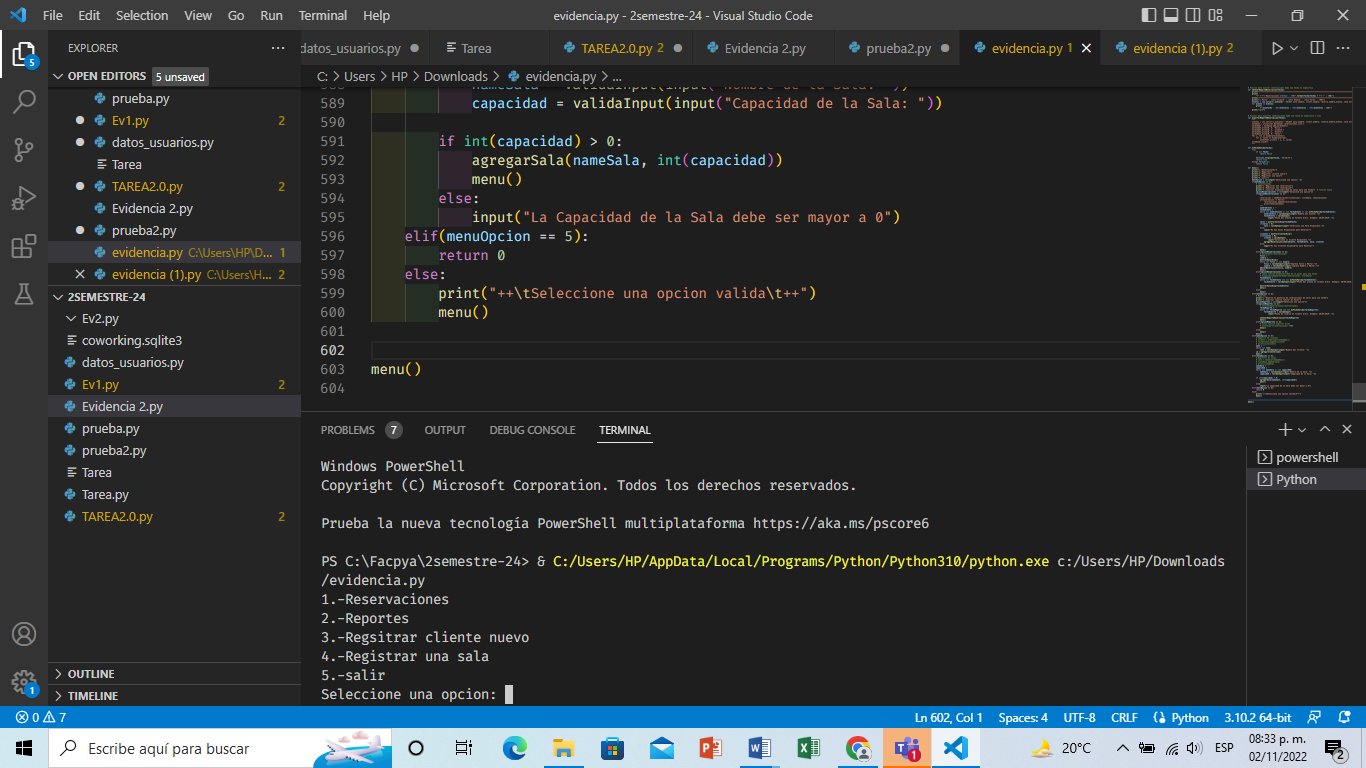
El código consta de un archivo principal (main.py) el cual cuenta con las importaciones de las dependencias de las funciones de los demás archivos el cual se estarán utilizando por todo el proyecto, seguido de la función principal que es el menú, este nos conectara dependiendo de la opción seleccionada a la función adecuada, también en este mismo archivo inicializaremos las variables principales las cuales estaremos utilizando por todo el programa, las cuales son listas anidadas estas mismas se estarán modificando en la ejecución del código.

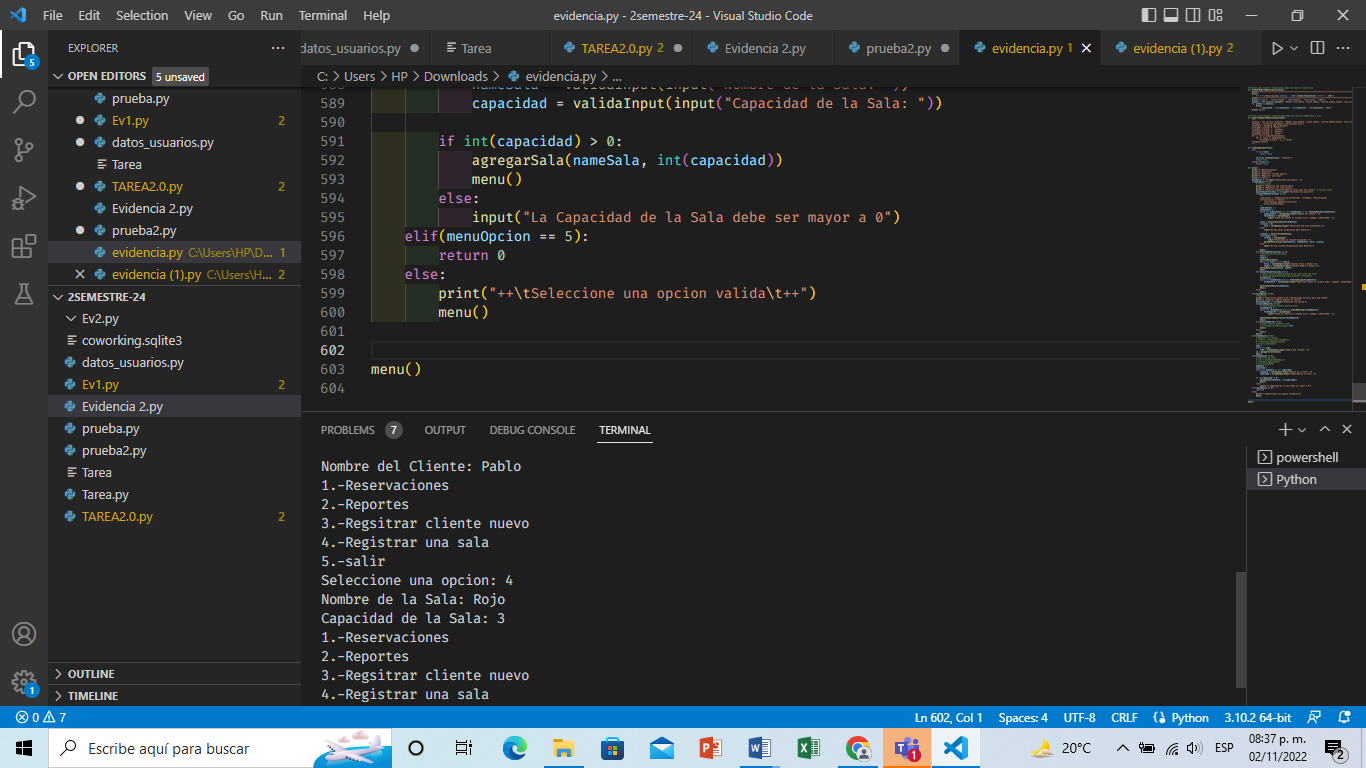
En la carpeta de los módulos contaremos con los demás archivos el cual cada uno tienes sus propias funciones para ejecutarse en determinadas circunstancias empezaremos los la de randomKey.py en este archivo tenemos una de las funciones principales y mas usadas para generar una clave única para nuestros registros, esta función se estará ejecutando cada vez que se crea un nuevo cliente o sala o también para generar folios alas reservaciones.

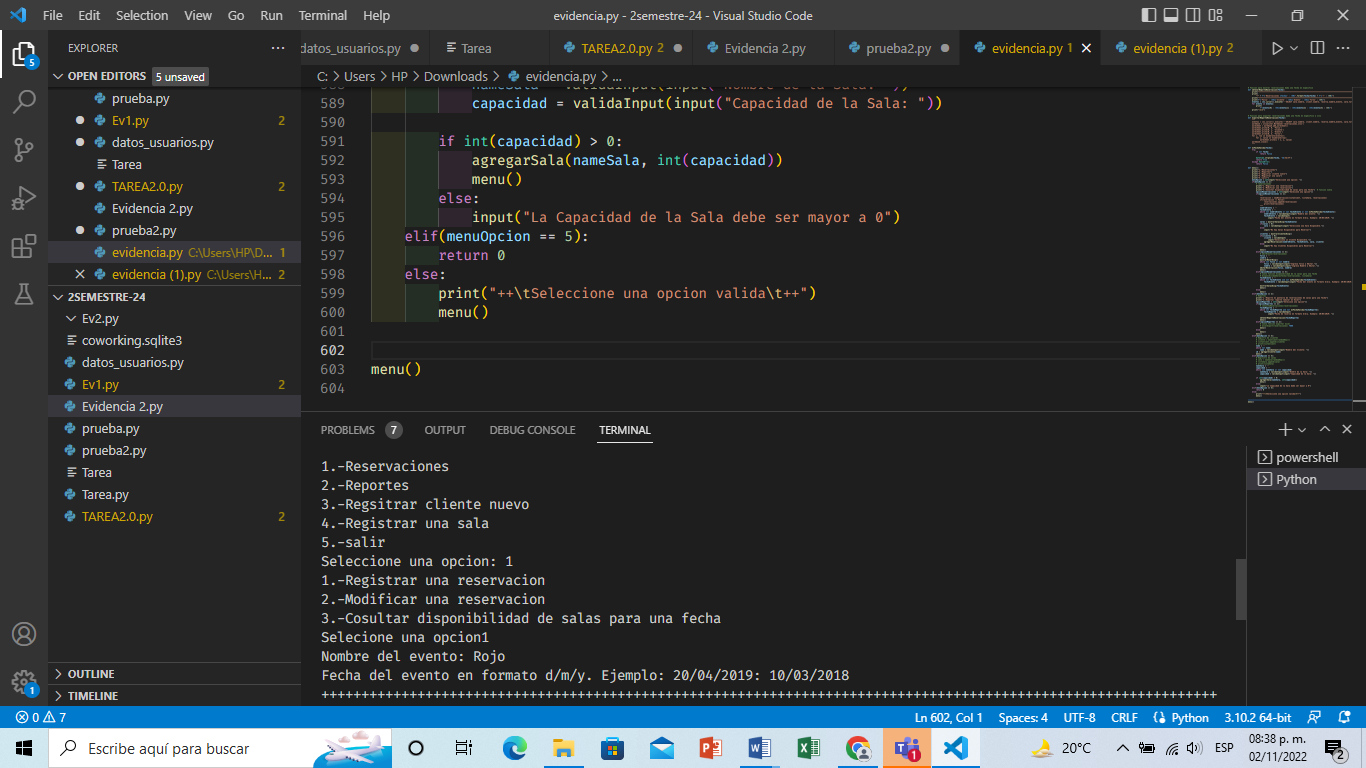
En el segundo archivo que es el de date.py, tendremos las funciones de declaraciones de fecha como por ejemplo la de la fecha actual, que le dirá al programa la fecha en la que se encuentra para posteriormente realizar comparaciones con fechas validas, las funciones de fechaUnDia() o fechaDosDias() los las que establecen los días hábiles para reservar una sala disponible, las función de conversiónFecha(), nos permite realizar una conversión de algun string de fecha a un formato de fecha valido, para posterior mente ser evaluado o guardado en algun arreglo o lista, la funcion de strFecha() nos permite realizar lo anverso lo ya mencionado, le tendremos que pasar una fecha para que nos devuelva un string de fecha con formato valido, todo esto es posible con la librería datetime y time.

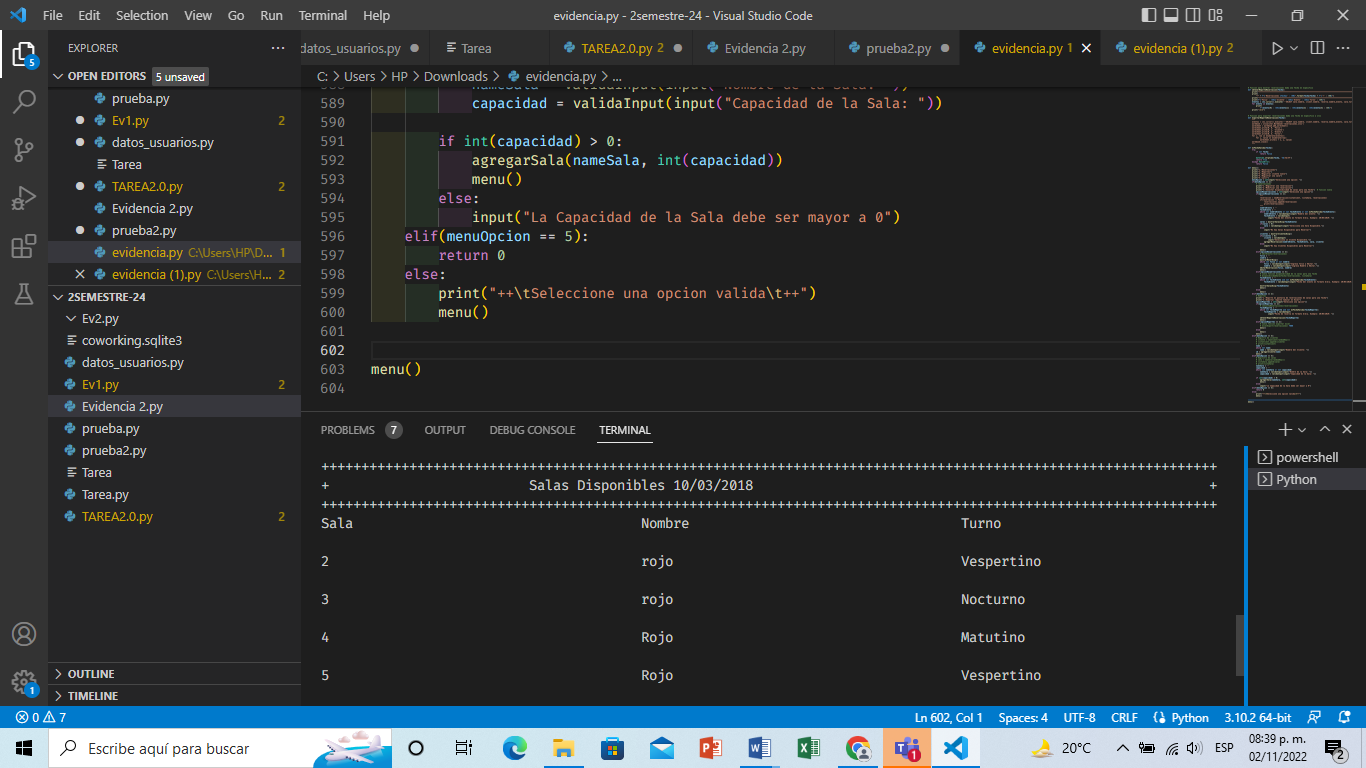
Ahora nos vamos con lo más importante que es el archivo layouts.py, en este archivo se encuentran las mayorías de las funciones que pedirá el menú y se estarán ejecutando mas de una vez en primera parte tendremos las dependencias de las demás funciones de los demás archivos como el randomKey o date por primera parte tendremos la función de registrar a un nuevo cliente donde dentro de la función pediremos que se nos indique el nombre y regresaremos tanto el id como el nombre del nuevo cliente para posteriormente ser guardado en una lista de clientes que instanciaremos en el archivo de main.py, por segunda parte tendremos lo la función de registrar una sala nueva, es la misma manera en la que se maneja la función del nuevo cliente siguiendo los mismos parámetros, pero preguntando como nuevo la capacidad de la misma, para después ser retornados los valores ya solicitados y guardarlos en una lista en el archivo de main.py. Después tenemos la función de registrar una reservación de una sala la cual funciona como las anteriores funciones pero diferenciándose de las demás primeras, ya que esta dentro de la misma función llamara a otras funciones que harán validaciones por ellas la función recibe 3 parámetros la cual es la lista de cliente, lista de las salas y las reservaciones por primera parte recibe el nombre del cliente y hace una validación para asegurarse que el cliente exista de esta manera aseguramos que la persona que esta pidiendo una reservaciones este de alta en el programa, la segunda parte es pedir el nombre de las sala, para posteriormente volver hacer una validación de existencia de la sala, en el caso de que sea verdadero seguirá pidiendo con los demás datos como el nombre del evento, fecha del evento y después esta variable se le hará una conversión con una función ya mencionada que es la función conversiónFecha() seguido de eso se hará una validación de que se este rentando por lo menos en un plazo de 2 días antes, con la función que veremos posteriormente, una ves que sea verdadero pediremos el turno del evento el cual se reparte por mañana, tarde, noche una vez puesto el turno correcto se pasara a realizar uso de la función de randomKey, para generar un folio único y seguido de eso se regresara de la función los datos guardados en una lista. La siguiente función nos permite validar la fecha y de que se este realizando en un plazo de dos días de anticipación. La función siguiente es la de la validación de clientes registrados nos permite ver o comparar si es que hay algun cliente registrado o no en el sistema. La función a continuación nos permite validar la sala existente siguiendo la misma lógica que la de validación de clientes. La función siguiente es la de validación de sala con disponibilidad a esta función se le pasan 4 parámetros los cuales son la lista de las salas, el turno que se desea consultar, la fecha y la lista de reservación siguiendo casi la misma lógica que la de los clientes esta estará haciendo tres validaciones para poder determinar dado a una fecha el turno consultado si coincide con alguna reservación de las ya creadas. La función siguiente es una de las mas interesantes ya que nos desplegara en pantalla cuales son las fechas con disponibilidad de rentar una sala sin desplegarnos las ya ocupadas, de esta manera podemos consultar que salas estas disponibles en ciertas fechas. Función para editar un evento existente, consta de un solo parámetro que es la de las listas de las reservaciones y posteriormente pedirle al usuario un nuevo nombre y pedir el nombre del evento a modificar hacer una validación y comprobación para modificar la tupla convertirá en lista modificar el elemento y nuevamente regresar al estado original. Función de excelExport esta función recibe como parámetro la lista de las reservaciones que posteriormente se almacenara en un diccionario y para posterior mente crear un archivo en formato csv y guardar los datos en las misma. Función de ExcelImport, esta función nos permite leer un archivo de Excel extraer los datos y guardarlos en un diccionario para posteriormente pasarlos a la lista de reservaciones y darle continuidad al programa en el ultimo guardado de datos.

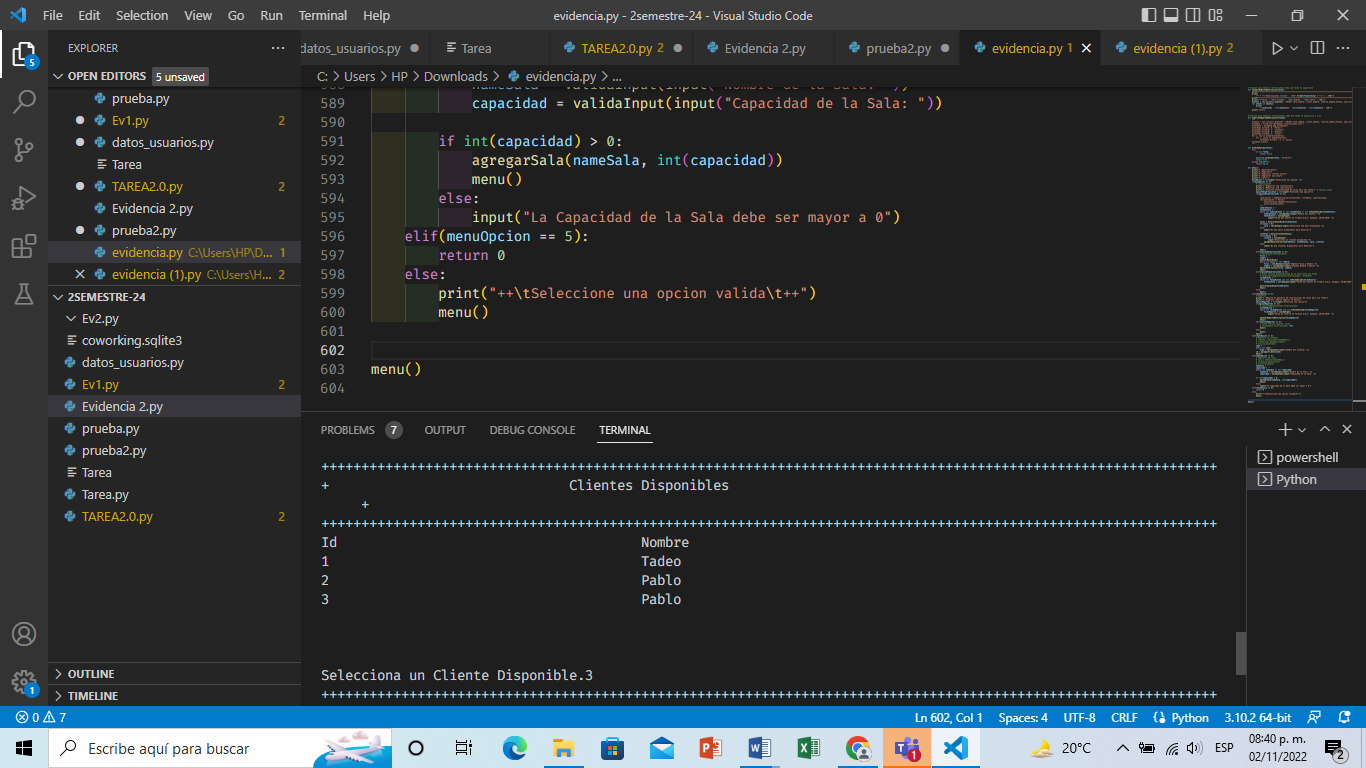
**Capturas de pantalla**

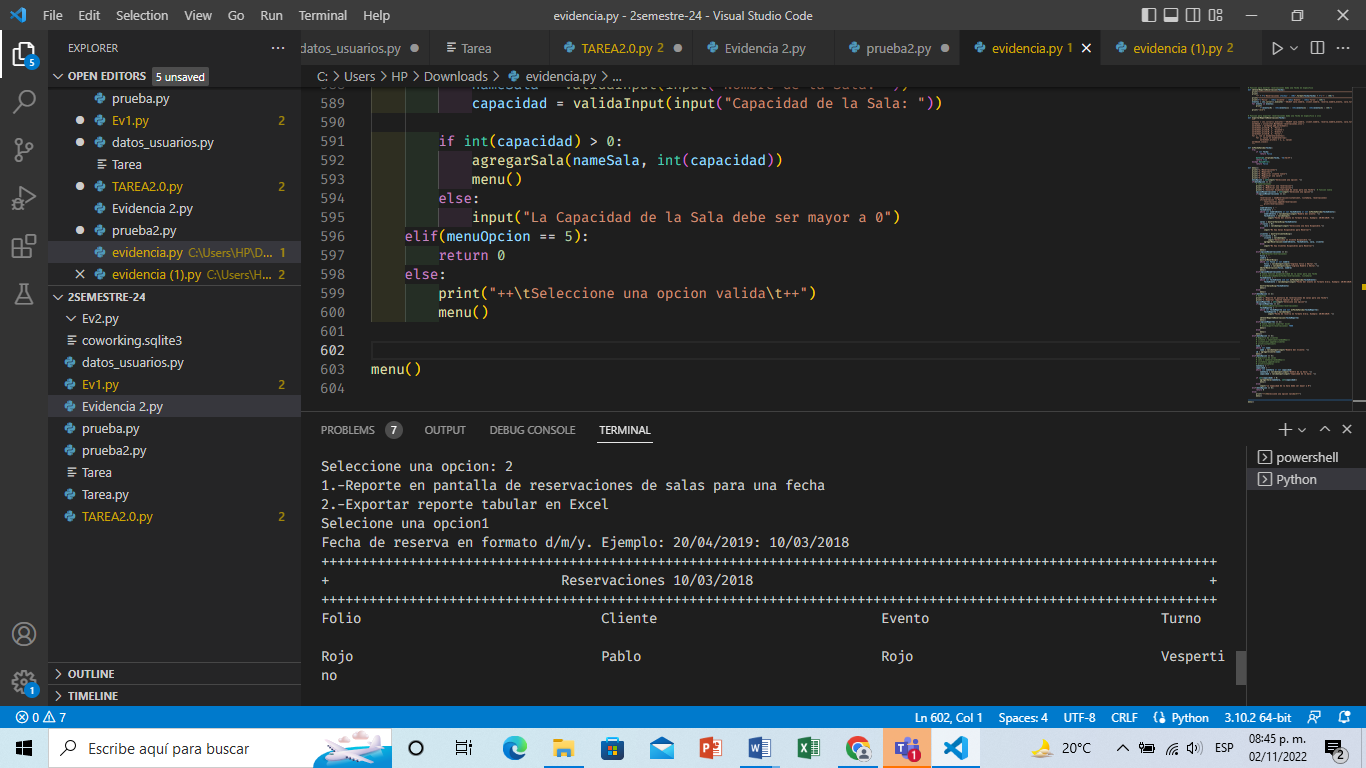
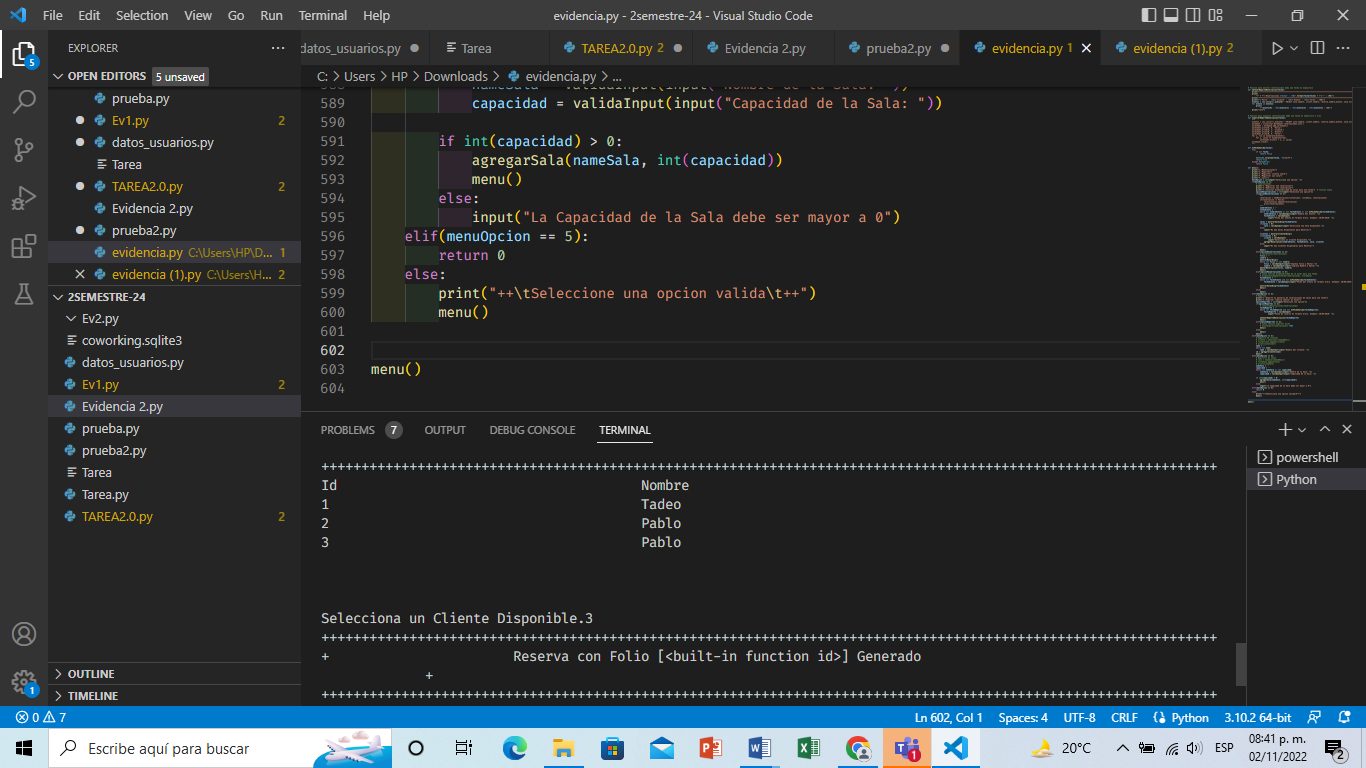


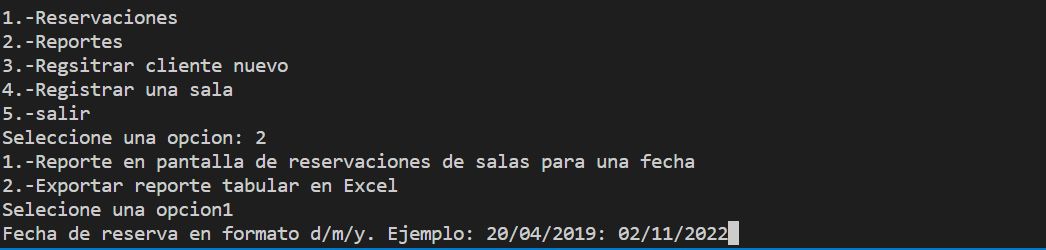


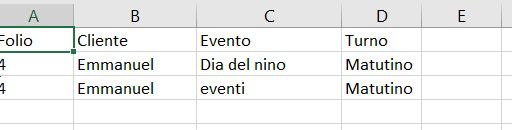
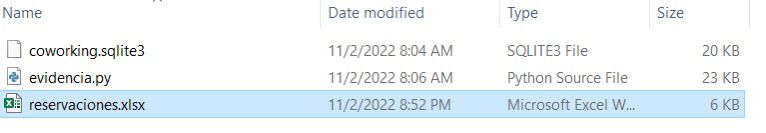








****

****

**Aportaciones de cada integrante**

Acosta Guzman Raymundo Tadeo

° Empezando con el código el cual cuenta con las importaciones de las dependencias de las funcionalidades de los otros archivos el cual se estarán usando por todo el plan, seguido de la funcionalidad primordial que es el menú, este nos conectara dependiendo de la alternativa escogida a la funcionalidad idónea, además en este mismo documento inicializaremos las cambiantes primordiales las cuales estaremos usando por todo el programa, las cuales son listas anidadas estas mismas se estarán modificando en la ejecución del código.

Barrientos Agüero Debanhi Nohemí

Como mayor parte de aportación el documento layouts.py, en este documento se hallan las mayorías de las funcionalidades que pedirá el menú y se estarán ejecutando más de una vez en primera parte tendremos las dependencias de las otras funcionalidades de los otros archivos como el randomKey o date por primera parte tendremos la capacidad de registrar a un nuevo comprador donde en la funcionalidad pediremos que se nos indique el nombre y regresaremos tanto el id como el nombre del nuevo comprador para después ser guardado en una lista de consumidores que instanciaremos en el documento de main.py, por segunda parte tendremos lo la capacidad de registrar una sala nueva, es la misma forma en la que se maneja la funcionalidad del nuevo comprador siguiendo los mismos límites.

Euresti Cantú Ileana Aidé

Para la Funcionalidad de excelExport esta funcionalidad obtiene como parámetro la lista de las reservaciones que después se almacenara en un diccionario y para siguiente mente generar un documento en formato csv y guardar los datos en la misma. Funcionalidad de ExcelImport, esta funcionalidad nos posibilita leer un documento de Excel sustraer los datos y guardarlos en un diccionario para más adelante pasarlos a la lista de reservaciones y darle continuidad al programa en el último guardado de datos.

Tello Salinas Diego Gerardo

Como mayor aportación en la evidencia fue reescribir algunas partes del código anterior y la corrección de errores que el código iba presentando cuando queríamos exportar los datos que nos daba el usuario a la base de datos para poder tener como resultado la evidencia 3 que se nos mencionó sería utilizada como la base de nuestro PIA.

**Conclusión Grupal**

En esta evidencia de aprendizaje agregamos y mejoramos algunas cosas que hacían falta para ello todos los integrantes del equipo dimos nuestro granitos de arena para que así esta evidencia saliera adelante, previamente concluimos con esta evidencia satisfactoriamente haciendo uno que otro ajuste como lo es la exportación del reporte a Excel e así para poder seguir con el PIA.

**Conclusión individual**

**Acosta Guzman Raymundo Tadeo**

Para esta ev de aprendizaje # 3 se tiene el propósito cumplido con el objetivo de construir y elaborar nuestro programa ya que cada vez se le van agregando cosas nuevas para su funcionalidad y mejoría a la hora de hacer nuestras consultas, claro que para esto no estuvo fácil, tuvimos que ver las libretas para seguir adelante con el proyecto ya que todo esto que estamos haciendo es para nuestro futuro PIA y algo está mal nos puede perjudicar a todos, pero estamos seguros que saldrá bien.

**Barrientos Agüero Debanhi Nohemí**

Como conclusión de esta evidencia nosotros como equipo requerimos poner en práctica dichas herramientas vistas en clase con el programa ya conocido dando a mejorar nuestro trabajo que abarcan un amplio y variado grado de variantes que se desarrollaron con el objetivo de ser más accesibles y entendibles por la mayoría de programadores, de manera que entre los mismos pudieran concentrarse más en resolver la tarea o los problemas de manera que nuestro trabajo que ya sea de manera clara y consiste a la hora de la lectura fácil para cualquier usuario.

**Euresti Cantú Ileana Aidé**

En esta evidencia batallamos un poco más de lo esperado ya que si se nos olvidó cosas pero al repasar la libreta que el profesor nos brindó pudimos recordar algunas cosas y también poner otras en práctica algunas si funcionaban y otras no tanto esta evidencia nos a ayudado mucho a estar practicando aunque si tenemos algunos errores pero cada evidencia lo vamos mejorando.

**Tello Salinas Diego Gerardo**

Durante esta evidencia 3 se implementó la práctica de una base de datos en Python, reforcé mis habilidades para programar trabajando con tuplas, diccionarios y otros temas que vimos en clase también; mismos que se habían utilizado también para la realización de las dos evidencias anteriores. De igual manera corregí algunas dudas que fueron surgiendo sobre ciertos temas en base a la investigación necesaria, teniendo como resultado la evidencia 3 que se nos mencionó sería utilizado como la base de nuestro PIA.

**URL**