

EVIDENCIA CALIFICABLE # 2

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 |

|  |  |
| --- | --- |
| **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 (18/10/2022)

Estructura de Datos y su Procesamiento

Evidencia #2

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 |

**Indice**

**Lista de objetivos por cumplir ………………………………………………**

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

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

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

**Aportaciones de cada integrante ………………………………………….**

**Conclucion grupal ……………………………………………………………**

**Individual ………………………………………………………………………**

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

**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 \*

listaClient = []

listaSala = []

reservaciones = []

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)

                    menu()

            elif(opcionResaervaciones == 2):

                editEvento(reservaciones)

                menu()

            elif(opcionResaervaciones == 3):

*# nueva funcion disponibilidad de la salas para una fecha*

                disponibilidadSalasfecha(reservaciones, listaSala)

                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)

                menu()

            elif(opcionReportes == 2):

*#nueva funcion exportar excel*

                excelExport(reservaciones)

                menu()

            else:

                menu()

            menu()

        elif(menuOpcion == 3):

*# Registro de clientes*

            cliente = newClient(randomKey())

            listaClient.append(cliente)

            print(listaClient)

            menu()

        elif(menuOpcion == 4):

*# Regristro de salas*

            sala = newSala(randomKey())

            listaSala.append(sala)

            print(listaSala)

            menu()

        elif(menuOpcion == 5):

            return 0

        else:

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

            menu()

menu()

from modulos.randomKey import randomKey

from .date import \*

import csv

from collections import namedtuple

*# 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: ")

    return *idSala*, nameSala, capacidad *#regresa lista (idSala, nombre de la sala, 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)):

                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():

    Auto = namedtuple("Reservas", "sala, clientes, evento, turno") *#Solamente si se está trabajando en otro código diferente al de la creación*

    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)

from datetime import datetime

from time import strftime

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

**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**

**conclusión Grupal**

Como conclusión grupal esperamos gratamente que el código lo que nos esté pidiendo para esta evidencia de aprendizaje este todo en orden ya que hemos estado en las clases y revisando las libretas para así poder cumplir satisfactoriamente con nuestros trabajos futuros.

**conclusión individual**

**Acosta Guzman Raymundo Tadeo**

Bueno para esta segunda evidencia fue complementar la primera para así poder que nos arrojase otros resultados y que nuestro programa este un poco más completo y formal, esto nos va servir para nuestro futuro profesional.

**Barrientos Agüero Debanhi Nohemí**

Como conclusión de esta evidencia poniendo en práctica dichas herramientas 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 los mismo pudieran concentrarse más en resolver la tarea o los problemas de manera que nuestro trabajo sea de manera clara y consiste a la hora de la lectura fácil para cualquier usuario.

**Euresti Cantú Ileana Aidé**

Gracias a este trabajo pudimos mejorar más nuestra primera evidencia nos apoyamos un poco de la libreta que nos enseña el profesor durante las clases, hemos puesto en práctica muchas cosas de ella y también nos ha ayudado pues a practicar más sobre los códigos en Python si hemos batallado un poco en agregar nuevas coas a la evidencia también en modificarla ya que si teníamos demasiadas cosas desordenadas pero gracias a las evidencias vamos poniendo más en práctica nuestra forma de programar.

**URL**