UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

MANUAL TÉCNICO DEL PROYECTO IPC2 FASE 1

14 DE JUNIO DEL 2023

JONATAN JOSUE VASQUEZ PASTOR

Introducción

Se desarrollo una aplicación que permite gestionar todas las actividades de USAC Cinema. Esta aplicación deberá tener diferentes módulos utilizando listas simplemente enlazada, listas circulares y listas doblemente enlazadas. La aplicación cuenta con un módulo para programar las proyecciones de películas en las salas de cine dentro del campus universitario. permite la selección de películas de diferentes géneros, épocas y países, y establecer fechas y horarios para las proyecciones.

Se realizo una clase usuario y otros usuarios usando listas enlazadas doble haciendo un CRUD para gestionar los datos

```python
class Usuario:
    def __init__(self,rol, nombre, apellido, telefono, correo, contra):
        self.rol = rol
        self.nombre = nombre
        self.apellido = apellido
        self.telefono = telefono
        self.correo = correo
        self.contra = contra
        self.siguiente = None

    def mostrar(self):
        print('\n---DATOS--')
        print(f'rol: {self.rol}\nnombre: {self.nombre} {self.apellido}\ntelefono:
        {self.telefono}\ncorreo: {self.correo}')
        print('------\n')


class Usuarios:
    def __init__(self):
        self.cabeza = None

    def add_usuario(self, rol, nombre, apellido, telefono, correo, contra):
        nuevo_usuario = Usuario(rol, nombre, apellido, telefono, correo, contra)

        if self.cabeza is None:
            self.cabeza = nuevo_usuario
        else:
            nodo_actual = self.cabeza
            while nodo_actual.siguiente is not None:
                nodo_actual = nodo_actual.siguiente
            nodo_actual.siguiente = nuevo_usuario

    def buscar_usuario(self, correo, contra):
        actual = self.cabeza
        while actual is not None:
            if actual.correo == correo:
                if actual.contra == contra:
                    actual.mostrar()
                    return True
            actual = actual.siguiente
        return False

    def rol_usuario(self, correo, contra):
        actual = self.cabeza
        while actual is not None:
            if actual.correo == correo:
                if actual.contra == contra:
```

Se realizo una clase película y películas usando listas enlazadas doble haciendo un CRUD para gestionar los datos

```python
class Pelicula:
    def __init__(self,categoria, titulo, director, anio, fecha, hora):
        self.categoria = categoria
        self.titulo = titulo
        self.director = director
        self.anio = anio
        self.fecha = fecha
        self.hora = hora
        self.next = None
        self.previous = None

    def mostrar(self):
        print(f'Categoria: {self.categoria}\nTitulo: {self.titulo}\nDirector: {self.
        director}\nAnio: {self.anio}\nFecha: {self.fecha}\nHora: {self.hora}\n')

class Peliculas:
    def __init__(self):
        self.cabeza = None

    def vacia(self):
        return self.cabeza is None

    def add_pelicula(self, categoria, titulo, director, anio, fecha, hora):
        nueva_pelicula = Pelicula(categoria, titulo, director, anio, fecha, hora)
        if self.vacia():
            self.cabeza = nueva_pelicula
        else:
            pelicula_actual = self.cabeza
            while pelicula_actual.next is not None:
                pelicula_actual = pelicula_actual.next
            pelicula_actual.next = nueva_pelicula
            nueva_pelicula.previous = pelicula_actual

    def buscar_pelicula(self, titulo, hora, fecha):
        actual = self.cabeza
        while actual is not None:
            if actual.titulo == titulo:
                if actual.hora == hora:
                    if actual.fecha == fecha:
                        actual.mostrar()
                        return True
            actual = actual.next
        return False

    def datos_pelicula(self, titulo, hora, fecha):
        actual = self.cabeza
        while actual is not None:
            if actual.titulo == titulo:
```

se realizó una clase cine y cines usando listas enlazadas doble haciendo un CRUD para gestionar los datos

```
class Cine:
    def __init__(self, nombre, sala_numero, asientos):
        self.nombre = nombre
        self.sala_numero = sala_numero
        self.asientos = asientos
        self.next = None
        self.previous = None

    def mostrar(self):
        print(f'Nombre Cine: {self.nombre}\nNo. sala: {self.sala_numero}\nAsientos:
        {self.asientos}\n')



class Cines:
    def __init__(self):
        self.cabeza = None

    def vacia(self):
        return self.cabeza is None

    def add_cine(self, nombre, sala_numero, asientos):
        nuevo_cine = Cine(nombre, sala_numero, asientos)
        if self.vacia():
            self.cabeza = nuevo_cine
        else:
            cine_actual = self.cabeza
            while cine_actual.next is not None:
                cine_actual = cine_actual.next
            cine_actual.next = nuevo_cine
            nuevo_cine.previous = cine_actual

    def modificar_cine(self, cine, numero_sala):
        actual = self.cabeza
        while actual is not None:
            if actual.cine == cine:
                if actual.sala_numero == numero_sala:
                    actual.mostrar()
                    print('-------------INGRESE LOS DATOS NUEVOS------------')
                    nombre = input('Nombre del Cine: ')
                    actual.nombre = nombre
                    sala = input('Sala No.: ')
                    actual.sala_numero = sala
                    asientos = input('Asientos: ')
                    actual.asientos = asientos
                    return True
            actual = actual.next
        return False
```

se realizo una clase boleto y boletos usando listas enlazadas haciendo un CRUD para gestionar los datos y realizar compras de boletos

```
class Boleto:
    def __init__(self, pelicula, fecha, hora, no_boletos, no_sala, nombre,  nit,
    direccion, usuario, listaBoletos):
        self.pelicula = pelicula
        self.fecha = fecha
        self.hora = hora
        self.no_boletos = no_boletos
        self.no_sala = no_sala
        self.nombre = nombre
        self.nit = nit
        self.direccion = direccion
        self.usuario = usuario
        self.next = None
        self.previous = None
        self.total = 42
        self.listaBoletos = listaBoletos

    def mostrar(self):
        print('---------------FACTURA-------------')
        print(f'Pelicula: {self.pelicula}\nFecha: {self.fecha}\nHora: {self.hora}\nNo.
        Sala: {self.no_sala}\nNo. boletos: {self.no_boletos}\nNombre: {self.nombre}
        \nNIT: {self.nit}\nDireccion: {self.direccion}\nUsuario: {self.usuario}')
        print('----------LISTA ASIENTOS-----------')
        for bol in self.listaBoletos:
            print(bol)
        print('--------------Total---------------')
        print('Q', self.total*int(self.no_boletos))
        print('---------------------------------')


class Boletos():
    def __init__(self):
        self.cabeza = None

    def vacia(self):
        return self.cabeza is None

    def add_compra(self, pelicula, fecha, hora, no_boletos, no_sala, nombre, nit,
    direccion, usuario, listaBoletos):
        nuevo_boleto = Boleto(pelicula, fecha, hora, no_boletos, no_sala, nombre, nit,
        direccion, usuario, listaBoletos)
        if self.vacia():
            self.cabeza = nuevo_boleto
        else:
            boleto_actual = self.cabeza
            while boleto_actual.next is not None:
                boleto_actual = boleto_actual.next
            boleto_actual.next = nuevo_boleto
```

Librerías que se utilizaron y archivos creados que se utilizaron.

```
import xml.etree.ElementTree as ET
from pelicula import Peliculas
from usuario import Usuarios
from cine import Cines
from tkinter import filedialog
from boleto import Boletos
```

Funciones para leer xml de usuario, cines, peliculas

```
def leerXML_usuarios(ruta_xml):

    tree = ET.parse(ruta_xml)
    root = tree.getroot()
    lista_usuarios = Usuarios()

    for elemento in root.iter('usuarios'):
        for el in elemento.findall('usuario'):
            rol = el.find('rol').text
            nombre = el.find('nombre').text
            apellido = el.find('apellido').text
            telefono = el.find('telefono').text
            correo = el.find('correo').text
            contra = el.find('contrasena').text

            lista_usuarios_fin.add_usuario(rol, nombre, apellido, telefono, correo,
            contra)

def leerXML_cines(ruta_xml):
    tree = ET.parse(ruta_xml)
    root = tree.getroot()


    for cines in root.iter('cine'):
        nombre = cines.find('nombre').text
        #print('Nombre: ', nombre)
        for cine in cines.iter('salas'):
            for sala in cine.findall('sala'):
                numero = sala.find('numero').text
                asientos = sala.find('asientos').text

                lista_cines.add_cine(nombre,numero, asientos)
```

```
def modo_cliente(usuario):
    salir = True

    while salir:
        print('1-Listado Peliculas\n2-Peliculas Favoritas\n3-Comprar
        Boletos\n4-Boletos Comprados\n5-salir')
        opcion = input('Ingrese una opcion: ')
        if opcion == '1':
            lista_peliculas.imprimir_pelicula()
        elif opcion == '2':
            print("se muestra listado peliculas favoritas")
        elif opcion == '3':
            lista_cines.imprimir_cines()
            print('INGRESE LOS DATOS DE LA PELICULA Y SALA')
            titulo = input('Titulo: ')
            hora = input('Hora: ')
            fecha = input('Fecha: ')
            pasar = lista_peliculas.buscar_pelicula(titulo, hora, fecha)
            if pasar:
                no_boletos = input('No. boletos: ')
                no_sala = input('No. Sala: ')
                sala_exis = lista_cines.buscar_sala(no_sala)
                if sala_exis:
                    nombre = input('Nombre: ')
                    nit = input('nit: ')
                    direccion = input('direccion: ')

                    lista = []
                    for bols in range(int(no_boletos)):
                        dato = '202007092'+str(bols)
                        lista.append(dato)

                    lista_boletos.add_compra(titulo, fecha, hora, no_boletos, no_sala,
                    nombre, nit, direccion, usuario, lista)

                else:
```