**Caso de uso: Registro de nuevo cliente.**

**Pseudocódigo:**

Nombre del caso de uso: Registro de usuario en una web de servicios de turismo

Actores: Usuario

Objetivo: Permitir que un usuario se registre en la web de servicios de turismo para poder acceder a los servicios ofrecidos.

Precondiciones:

- La web de servicios de turismo está disponible.

- El usuario no está registrado en la web de servicios de turismo.

Pasos del escenario principal:

1. El usuario accede a la página de registro en la web de servicios de turismo.

2. El usuario completa el formulario de registro con los siguientes datos:

a. Nombre y Apellido (cadena de caracteres)

b. DNI (entero)

c. CUIT (entero)

d. Domicilio (cadena de caracteres)

e. Email (cadena de caracteres)

f. Ciudad (cadena de caracteres)

g. País (cadena de caracteres).

3. El sistema valida los datos ingresados y registra al usuario. La validación incluye:

a. Verificar que el DNI y el CUIT son números válidos.

b. Verificar que el email es válido y no está asociado a otro usuario registrado.

c. Verificar que todos los campos obligatorios (nombre, apellido, DNI, CUIT, domicilio, email, ciudad, país) han sido completados.

d. Verificar que el usuario no está registrado previamente en la web de servicios de turismo.

Postcondiciones:

- El usuario está registrado en la web de servicios de turismo.

- El usuario puede acceder a los servicios ofrecidos por la web de servicios de turismo utilizando sus credenciales de registro.

Clase Cliente:

- Atributos:

a. Nombre y Apellido (cadena de caracteres)

b. DNI (entero)

c. CUIT (entero)

d. Domicilio (cadena de caracteres)

e. Email (cadena de caracteres)

f. Ciudad (cadena de caracteres)

g. País (cadena de caracteres)

h. Usuario (cadena de caracteres)

i. Contraseña (cadena de caracteres)

**Pseudocódigo para codificar el caso de uso:**

Paso 1: El usuario accede a la página de registro en la web de servicios de turismo.

En Python, esto se podría lograr mediante la creación de una función que abra la página de registro en el navegador web del usuario. Por ejemplo:

import webbrowser

def abrir\_pagina\_registro():

url = "https://www.tuserviciodeturismo.com/registro"

webbrowser.open(url)

Paso 2: El usuario completa el formulario de registro con los siguientes datos: Nombre y Apellido, DNI, CUIT, Domicilio, Email, Ciudad, País.

Para capturar los datos ingresados por el usuario, se puede utilizar la función input() en Python. Por ejemplo:

nombre = input("Ingrese su nombre: ")

apellido = input("Ingrese su apellido: ")

dni = input("Ingrese su DNI: ")

cuit = input("Ingrese su CUIT: ")

domicilio = input("Ingrese su domicilio: ")

email = input("Ingrese su correo electrónico: ")

ciudad = input("Ingrese su ciudad: ")

pais = input("Ingrese su país: ")

Paso 3: El sistema valida los datos ingresados y registra al usuario.

Para validar los datos ingresados por el usuario, se pueden utilizar varias técnicas de validación, como expresiones regulares o la verificación de tipos de datos. Aquí hay un ejemplo de cómo validar si el DNI ingresado es un número válido:

import re

def validar\_dni(dni):

patron = "^[0-9]{7,8}$"

if re.match(patron, dni):

return True

else:

return False

Una vez que los datos han sido validados, se pueden almacenar en una instancia de la clase Cliente, que contendrá todos los datos del usuario. Para esto, primero es necesario definir la clase Cliente:

class Cliente:

def \_\_init\_\_(self, nombre, apellido, dni, cuit, domicilio, email, ciudad, pais):

self.nombre = nombre

self.apellido = apellido

self.dni = dni

self.cuit = cuit

self.domicilio = domicilio

self.email = email

self.ciudad = ciudad

self.pais = país

Luego, se puede crear una instancia de la clase Cliente con los datos ingresados por el usuario:

cliente = Cliente(nombre, apellido, dni, cuit, domicilio, email, ciudad, pais)

Finalmente, se puede almacenar esta instancia en una base de datos o archivo para mantener un registro de todos los usuarios registrados en la web de servicios de turismo.

# Clase GuiaTuristico

class GuiaTuristico:

def \_\_init\_\_(self, nombre, apellido, dni, cuit, domicilio, email, ciudad, pais):

self.nombre = nombre

self.apellido = apellido

self.dni = dni

self.cuit = cuit

self.domicilio = domicilio

self.email = email

self.ciudad = ciudad

self.pais = pais

def publicar\_servicio(self, coordenadas, precio, titulo, descripcion, fecha, tipo):

# Paso 1

# El Guía Turístico accede a la sección de publicación de servicios en la web de servicios de turismo.

# No es necesario codificarlo, ya que es una acción manual del usuario.

# Paso 2

# El Guía Turístico completa el formulario de publicación de servicio con los siguientes datos:

# a. Coordenadas del viaje

# b. Precio por persona

# c. Título del servicio

# d. Descripción

# e. Fecha y Hora

# f. Tipo de servicio.

servicio = Servicio(coordenadas, precio, titulo, descripcion, fecha, tipo)

# Paso 3

# El sistema valida los datos ingresados y publica el nuevo servicio.

# La validación incluye:

# a. Verificar que las coordenadas del viaje sean válidas y se encuentren en el formato correcto.

# b. Verificar que el precio por persona sea un número válido.

# c. Verificar que el título y la descripción del servicio no excedan un límite de caracteres.

# d. Verificar que la fecha y hora sean válidas y no estén en el pasado.

# e. Verificar que el tipo de servicio esté entre las opciones permitidas.

servicio.validar\_datos()

# Paso 4

# El sistema muestra al Guía Turístico una confirmación de la publicación del servicio.

print("¡El servicio ha sido publicado con éxito!")

# Clase Servicio

class Servicio:

def \_\_init\_\_(self, coordenadas, precio, titulo, descripcion, fecha, tipo):

self.coordenadas = coordenadas

self.precio = precio

self.titulo = titulo

self.descripcion = descripcion

self.fecha = fecha

self.tipo = tipo

def validar\_datos(self):

# Validar que las coordenadas sean válidas y se encuentren en el formato correcto

if not isinstance(self.coordenadas, tuple) or len(self.coordenadas) != 2:

raise ValueError("Las coordenadas deben ser una tupla de 2 valores.")

# Validar que el precio por persona sea un número válido

if not isinstance(self.precio, (int, float)):

raise ValueError("El precio por persona debe ser un número.")

# Validar que el título y la descripción del servicio no excedan un límite de caracteres

if len(self.titulo) > 50:

raise ValueError("El título del servicio no puede exceder los 50 caracteres.")

if len(self.descripcion) > 500:

raise ValueError("La descripción del servicio no puede exceder los 500 caracteres.")

# Validar que la fecha y hora sean válidas y no estén en el pasado

if not isinstance(self.fecha, datetime.datetime) or self.fecha < datetime.datetime.now():

raise