Estructura del Informe

CAPITULO 1: Análisis del Problema

1. **Descripción del problema:** Las agencias de turismo manejan información de clientes, tours, reservas y pagos. Cuando estos datos se administran de forma manual o dispersa, se producen problemas como duplicidad de registros, pérdida de información, demoras en la atención y dificultades para generar reportes.
   El sistema propuesto busca resolver este problema permitiendo registrar, actualizar y consultar información de manera organizada y eficiente mediante estructuras de datos estudiadas en el curso. El sistema debe permitir registrar clientes, administrar tours, crear reservas vinculando un cliente y un tour, registrar pagos y generar reportes generales
2. **Requerimientos del sistema**
   A. **Funcionales**
   - **Gestión de clientes**
     - Agregar y actualizar clientes.
     - Buscar cliente por ID.
     - Eliminar cliente.
     - Listar todos los clientes registrados.
   - **Gestión de tours**
     - Agregar y actualizar tours.
     - Buscar tour.
     - Eliminar tour.
     - Listar tours disponibles.
   - **Gestión de reservas**
     - Crear reserva.
     - Cambiar estado (Pendiente, Pagado, Cancelado).
     - Listar reservas.
     - Validar que cliente y tour existan antes de reservar.
   - **Gestión de pagos**
     - Registrar pago asociado a una reserva.
     - Listar historial de pagos.
   - **Reportes**
     - Mostrar reservas por cliente.
     - Mostrar ingresos por tour.
     - Mostrar estados de reserva.

   B. **No funcionales**
   - **Usabilidad**: Debe ser fácil de usar mediante menús.
   - **Mantenibilidad**: Estructurado mediante módulos independientes.
   - **Eficiencia**: Respuesta rápida usando estructuras en memoria.
   - **Portabilidad**: Debe ejecutarse en cualquier ambiente Python.
   - **Simplicidad**: No requiere bases de datos externas.
3. **Estructuras de datos propuestas**
   Para almacenar la información se utilizarán **diccionarios de Python**, representando entidades:

- Clientes
- Tours
- Reservas
- Pagos

4. **Justificación de la elección**
   Los diccionarios permiten:
   - Búsqueda rápida por clave.
   - Estructuras simples y flexibles.
   - Fácil actualización y eliminación.
   - Perfecta representación de entidades tipo registro.
   - Permiten simular una base de datos pequeña sin requerir archivos.

   Por estas razones, son ideales para un prototipo CRUD modular como el requerido en este proyecto

Capítulo 2: Diseño de la Solución

1. **Descripción de estructuras de datos y operaciones:** El sistema usará varias estructuras de datos estudiadas en clase, cada una asignada a un módulo específico.
   A. Clientes
   - Operaciones:
     - Agregar/actualizar cliente
     - buscar cliente
     - eliminar cliente
     - listar clientes

   B. Tours
   - Operaciones:
     - Agregar/actualizar tour
     - Buscar tours
     - Eliminar tours
     - listar tours

   C. Reservas
   - Operaciones**:**
     - Crear reserva
     - Cambiar estado
     - Listar reservas

   D. Pagos
   - Operaciones**:**
     - Registrar pago
     - Listar Pago
2. **Algoritmos principales:(PSEINT)**
   a. Pseudocódigo para AGREGAR O ACTUALIZAR cliente

```
INICIO
  LEER id_cliente
  LEER nombre
  LEER dni
  LEER correo

  SI id_cliente existe en clientes
```

```
    actualizar datos
  SINO
    crear nuevo cliente
  FIN SI

  MOSTRAR "Cliente guardado correctamente"
FIN
```

b. Pseudocódigo para crear una reserva

```
INICIO
  LEER id_reserva
  LEER id_cliente
  LEER id_tour

  SI cliente NO existe
      MOSTRAR "Cliente no encontrado"
      FIN
  FIN SI

  SI tour NO existe
      MOSTRAR "Tour no encontrado"
      FIN
  FIN SI

  reservas[id_reserva] = {
     id_cliente,
     id_tour,
     estado = "Pendiente"
  }

  MOSTRAR "Reserva registrada"
FIN
```

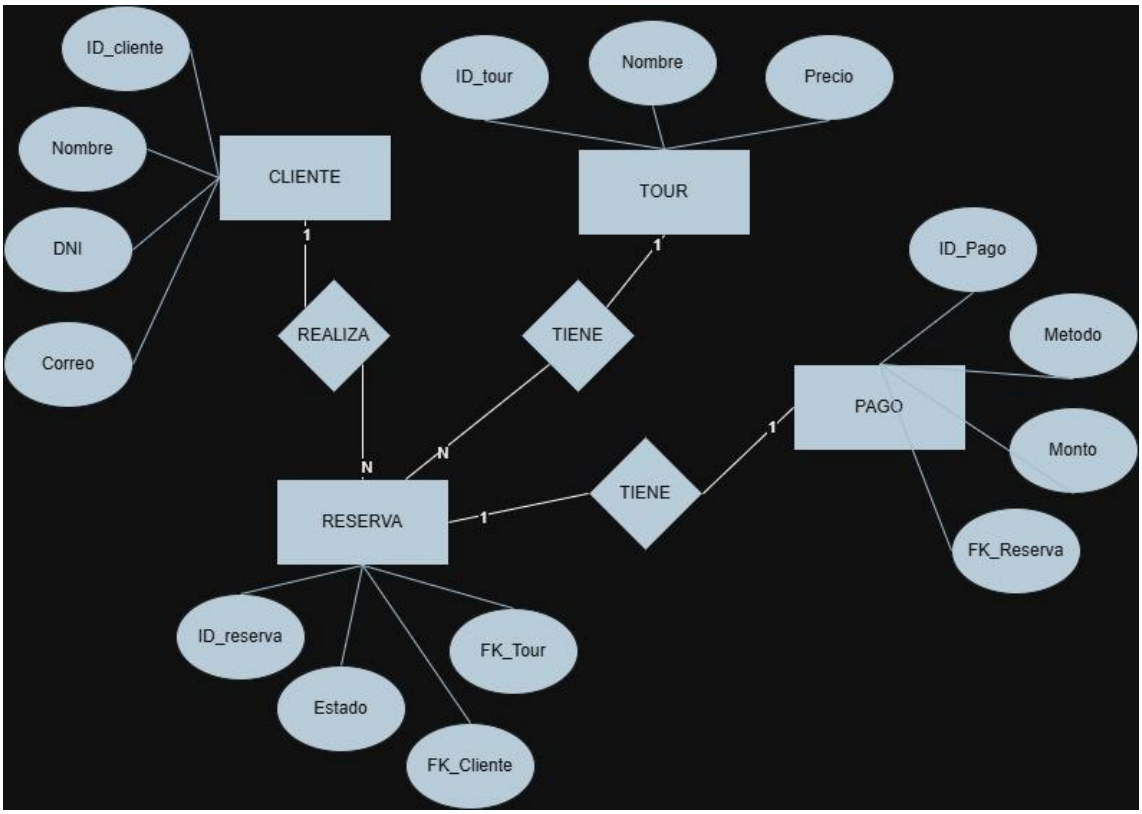
c. Pseudocódigo para CAMBIAR ESTADO de una reserva

```
INICIO
  LEER id_reserva

  SI reserva no existe
      MOSTRAR "No existe"
      FIN
  FIN SI

  LEER nuevo_estado
  SI nuevo_estado ES valido (Pendiente / Pagado / Cancelado)
      actualizar estado
      MOSTRAR "Estado actualizado"
  SINO
      MOSTRAR "Estado inválido"
  FIN SI
FIN
```

3. Diagramas de Flujo



4. **Justificación del diseño:**
   - El sistema utiliza módulos independientes para mejorar mantenibilidad.
   - El uso de diccionarios permite alta velocidad en operaciones CRUD.
   - El menú general facilita la navegación.
   - La estructura modular evita errores y hace el código escalable.
   - Los algoritmos siguen un flujo simple, con validación en cada operación

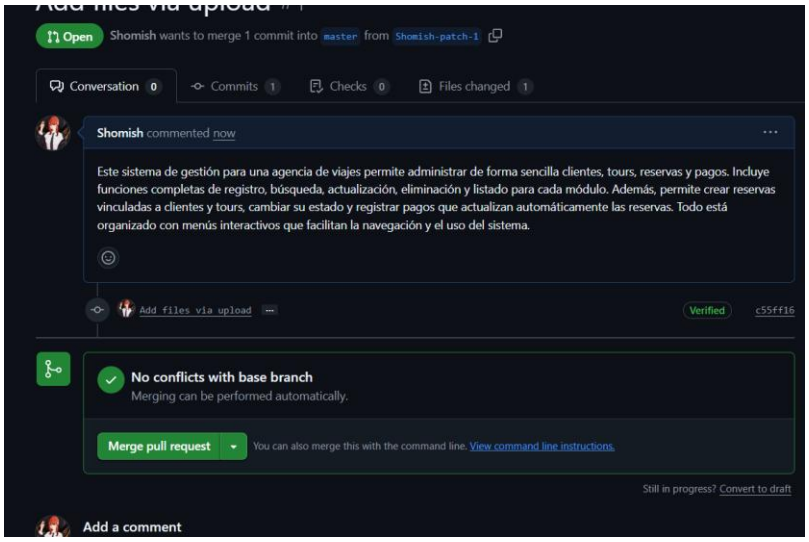
Capítulo 3: Solución Final

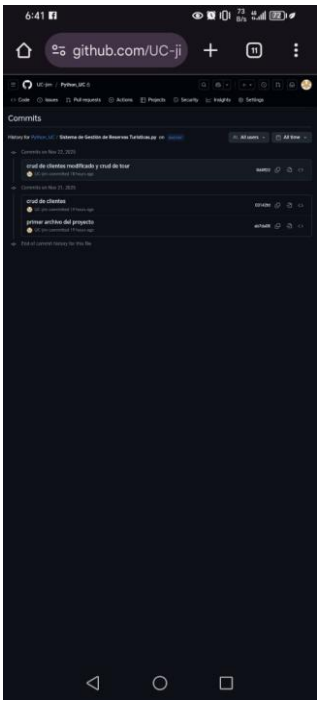
1. **Código limpio, bien comentado y estructurado.**
2. **Capturas de pantalla de las ventanas de ejecución con las diversas pruebas de validación de datos**
3. **Manual de usuario**

Capítulo 4: Evidencias de Trabajo en Equipo

1. **Repositorio con Control de Versiones (Capturas de Pantalla)**
   - Registro de commits claros y significativos que evidencien aportes individuales (proactividad).
   - Historial de ramas y fusiones si es aplicable.
   - Enlace a la herramienta colaborativa

Imágenes de commit


Open
Shomish wants to merge 1 commit into master from Shomish-patch-1
Conversation 0
Commits 1
Checks 0
Files changed 1
Shomish commented now
Este sistema de gestión para una agencia de viajes permite administrar de forma sencilla clientes, tours, reservas y pagos. Incluye funciones completas de registro, búsqueda, actualización, eliminación y listado para cada módulo. Además, permite crear reservas vinculadas a clientes y tours, cambiar su estado y registrar pagos que actualizan automáticamente las reservas. Todo está organizado con menús interactivos que facilitan la navegación y el uso del sistema.
Add files via upload
Verified
c55ff16
No conflicts with base branch
Merging can be performed automatically.
Merge pull request
You can also merge this with the command line. View command line instructions.
Still in progress? Convert to draft
Add a comment


6:41
github.com/UC-ji
Commits


Shomish Add files via upload
91ea652 · 9 minutes ago
5 Commits
README.md
Readme.SistemaTurismo
10 minutes ago
Sistema de Gestión de Reservas Turísticas.py
crud de clientes modificado y crud de tour
20 hours ago
SistemaAgenciaViajes.py
Add files via upload
9 minutes ago
README
Proyecto: Sistema de Gestión para Agencia de Viajes
Este proyecto implementa un sistema interactivo en Python diseñado para organizar las operaciones principales de una agencia de viajes. El sistema funciona mediante menús y permite realizar registros, búsquedas, actualizaciones y eliminaciones dentro de cuatro módulos fundamentales:
Módulos incluidos
Clientes: Registro y administración de datos personales como nombre, DNI y correo.
Tours: Gestión de tours disponibles, incluyendo nombre y precio.
Reservas: Creación de reservas vinculadas a clientes y tours, con control de estados (Pendiente, Pagado, Cancelado).
Pagos: Registro de pagos asociados a reservas, actualizando automáticamente el estado correspondiente.
Funcionalidades principales
CRUD completo para clientes y tours