ADMINISTRACIÓN
Y NEGOCIOS

Inacap

Informe 4

Integrantes: Orlando Aravena, Marco Badillo y Jorge Dominguez
Carrera: Analista Programador
Asignatura: Programación Orientada a Objeto TI3021/D-B50-N2-P13-C2/D
Profesor: Luis Rodrigo Arriagada Cerda
Fecha: 13-12-24

ADMINISTRACIÓN
Y NEGOCIOS

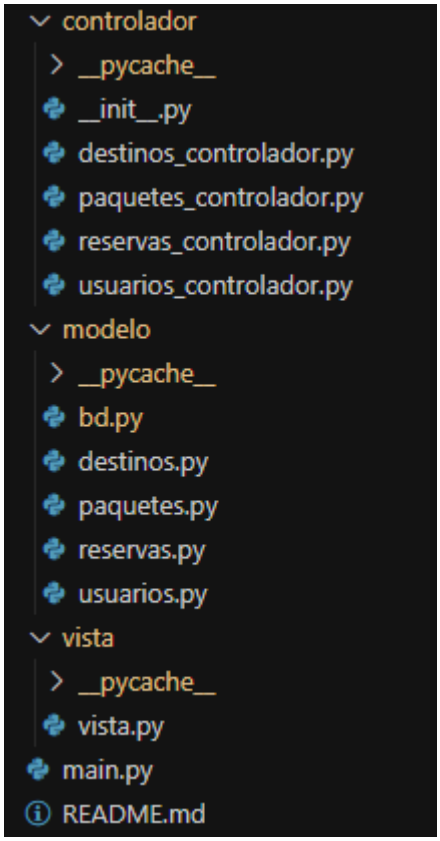
Inacap

Índice:

1 Introducción

El proyecto "Viajes-Aventura" es un sistema de software diseñado para gestionar las operaciones principales de una agencia de viajes, incluyendo la administración de usuarios, destinos, paquetes turísticos y reservas. Este sistema facilita la organización interna, reduce la posibilidad de errores humanos y mejora la experiencia del cliente al automatizar procesos clave.

El desarrollo del proyecto sigue el patrón arquitectónico Modelo-Vista-Controlador (MVC), asegurando una separación clara entre la lógica de negocio, los datos y la interfaz de usuario, lo que permite una mayor flexibilidad, mantenibilidad y escalabilidad del sistema.



2 Contexto

En el competitivo mercado actual, las agencias de viajes necesitan soluciones tecnológicas que les permitan gestionar sus operaciones de manera eficiente y atractiva. Un sistema bien diseñado puede:

1.- Simplificar la creación y gestión de paquetes turísticos.

2.- Facilitar la reserva de servicios por parte de los clientes.

3.- Garantizar la seguridad y fiabilidad de los datos.

El proyecto "Viajes-Aventura" busca abordar estas necesidades mediante un enfoque simple pero eficaz, utilizando Python como lenguaje base. La herramienta también está pensada para ser una base que pueda expandirse en el futuro, integrando más funcionalidades o adaptándose a nuevas tecnologías.

3 Alcance

El sistema "Viajes-Aventura" cubre las siguientes funcionalidades principales:

1.- **Gestión de Usuarios:**

Permite registrar, asegurando un control adecuado sobre los clientes y empleados de la agencia.

2.- **Gestión de Destinos:**

Facilita la organización de destinos disponibles para los paquetes turísticos.

3.- **Gestión de Paquetes Turísticos:**

Ofrece una manera de agrupar destinos y servicios adicionales en paquetes que los clientes pueden reservar.

4.- **Gestión de Reservas:**

Incluye la capacidad de registrar y rastrear las reservas realizadas por los usuarios.

El proyecto se limita, por el momento, a una interfaz basada en consola, sin incluir una interfaz gráfica ni una integración con plataformas externas. En el futuro, se podrían desarrollar módulos adicionales, como un portal web o una aplicación móvil, para ampliar la usabilidad del sistema.

4 Arquitectura del Proyecto

El proyecto "Viajes-Aventura" está organizado siguiendo el patrón arquitectónico Modelo-Vista-Controlador (MVC). Este patrón permite separar las responsabilidades del sistema en tres componentes principales, facilitando el desarrollo, la prueba y el mantenimiento del código.

Estructura del Proyecto:

El sistema está dividido en tres carpetas principales:

1.- controlador: Contiene la lógica de negocio y maneja las interacciones entre la vista y el modelo.

destinos_controlador.py: Gestiona las operaciones relacionadas con los destinos.

paquetes_controlador.py: Controla la creación y manejo de paquetes turísticos.

reservas_controlador.py: Administra el flujo de las reservas realizadas.

usuarios_controlador.py: Maneja las acciones relacionadas con los usuarios.

2.- modelo: Define las clases y la conexión con la base de datos.

bd.py: Configuración y conexión a la base de datos.

destinos.py: Modelo de datos para los destinos.

paquetes.py: Modelo de datos para los paquetes turísticos.

reservas.py: Modelo de datos para las reservas.

usuarios.py: Modelo de datos para los usuarios.

3.- Vista: Proporciona la interfaz para la interacción del usuario.

vista.py: Implementa la interfaz basada en consola.

main.py: Es el punto de entrada del programa que conecta los controladores, modelos y vistas.

5 Descripción del Sistema

El sistema "Viajes-Aventura" tiene como objetivo ofrecer una solución completa para la gestión de una agencia de viajes. Las principales funcionalidades son las siguientes:

1.- Gestión de Usuarios:

Registro, edición y eliminación de usuarios.

Roles diferenciados para clientes y administradores.

2.- Gestión de Destinos:

Creación, edición y eliminación de destinos turísticos.

Organización de destinos por categorías o características clave.

3.- Gestión de Paquetes Turísticos:

Creación de paquetes que incluyen múltiples destinos.

Agregar servicios adicionales, como transporte o guías.

4.- Gestión de Reservas:

Registro de reservas realizadas por los clientes.

Control del estado de las reservas (pendiente, confirmada, cancelada).

Flujo del Sistema

El flujo típico del sistema comienza con el inicio de sesión del usuario. Dependiendo de su rol, se le muestran opciones para interactuar con destinos, paquetes o reservas. Un administrador puede gestionar toda la información, mientras que un cliente solo puede ver destinos y realizar reservas.

Tecnologías Utilizadas

Lenguaje de programación: Python.

Gestor de bases de datos: SQLite (se utiliza bd.py para la conexión).

Organización modular: Uso del patrón MVC para separar responsabilidades.

6 Diseño del Sistema

Diagramas UML

Diagrama de Clases:

Clases principales:

Usuario: Contiene atributos como nombre, email, y rol.

Destino: Incluye atributos como nombre, ubicación y descripción.

Paquete: Relaciona múltiples destinos y servicios adicionales.

Reserva: Almacena información de la reserva, como cliente, paquete y estado.

Relaciones:

Un usuario puede realizar varias reservas.

Un paquete puede incluir varios destinos.

Una reserva pertenece a un usuario y a un paquete.

Base de Datos

El sistema utiliza una base de datos SQLite, organizada de la siguiente manera:

Tablas:

Usuarios: Almacena los datos de los usuarios.

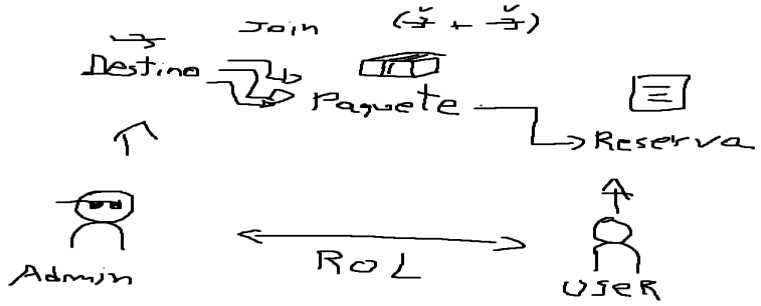
destinos: Contiene la información de los destinos turísticos.

PaquetesDestino: Es la clase en la que se hace Join para unir varios destinos en 1 paquete

paquetes: Registra los paquetes turísticos creados.

reservas: Guarda información sobre las reservas realizadas.

Relaciones entre tablas:



7 Desarrollo

Metodología de Trabajo

El desarrollo de "Viajes-Aventura" se llevó a cabo siguiendo un enfoque iterativo e incremental. Cada funcionalidad se desarrolló por separado, probándola de manera individual antes de integrarse con el resto del sistema.

La estructura modular del proyecto permitió asignar roles específicos para cada componente (controlador, modelo y vista), facilitando la colaboración y reduciendo los riesgos de errores durante la integración.

Desafíos Técnicos

Algunos de los principales desafíos encontrados durante el desarrollo fueron:

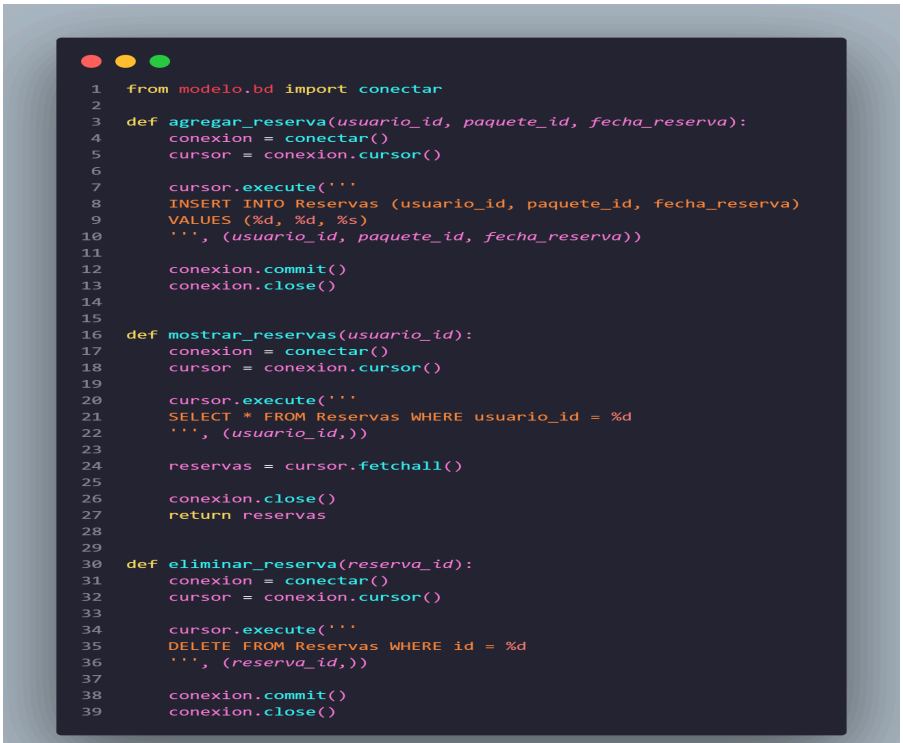
1.- Diseño de la base de datos: Definir una estructura relacional que permitiera gestionar las reservas, usuarios y paquetes sin redundancia ni pérdida de datos.

2.- Validaciones de datos: Garantizar que la información introducida en el sistema, como reservas y destinos, fuera consistente y válida.

3.- Conexión entre componentes: Asegurar una comunicación eficiente entre los modelos, controladores y la vista, respetando el patrón MVC.

Fragmento de Código Representativo

El siguiente fragmento muestra cómo el controlador de reservas conecta la lógica de negocio con el modelo:



```python
from modelo.bd import conectar

def agregar_reserva(usuario_id, paquete_id, fecha_reserva):
    conexion = conectar()
    cursor = conexion.cursor()

    cursor.execute('''
    INSERT INTO Reservas (usuario_id, paquete_id, fecha_reserva)
    VALUES (%d, %d, %s)
    ''', (usuario_id, paquete_id, fecha_reserva))

    conexion.commit()
    conexion.close()


def mostrar_reservas(usuario_id):
    conexion = conectar()
    cursor = conexion.cursor()

    cursor.execute('''
    SELECT * FROM Reservas WHERE usuario_id = %d
    ''', (usuario_id,))

    reservas = cursor.fetchall()

    conexion.close()
    return reservas


def eliminar_reserva(reserva_id):
    conexion = conectar()
    cursor = conexion.cursor()

    cursor.execute('''
    DELETE FROM Reservas WHERE id = %d
    ''', (reserva_id,))

    conexion.commit()
    conexion.close()
```

8 Pruebas

Pruebas Realizadas

El sistema se sometió a diversas pruebas para garantizar su correcto funcionamiento:

Pruebas Unitarias:

Cada controlador y modelo fue probado de manera individual, verificando que los métodos cumplieran con su propósito.

Ejemplo: Crear un destino y comprobar que se guarda correctamente en la base de datos.

Pruebas de Integración:

Se probó la interacción entre los diferentes módulos del sistema (modelo, controlador y vista).

Ejemplo: Crear un paquete que incluya destinos previamente registrados y realizar una reserva para dicho paquete.

Pruebas de Usuario:

Simulación de escenarios comunes para garantizar que los flujos del sistema funcionan sin interrupciones.

Ejemplo: Registro de un usuario, búsqueda de destinos y realización de una reserva.

Resultados

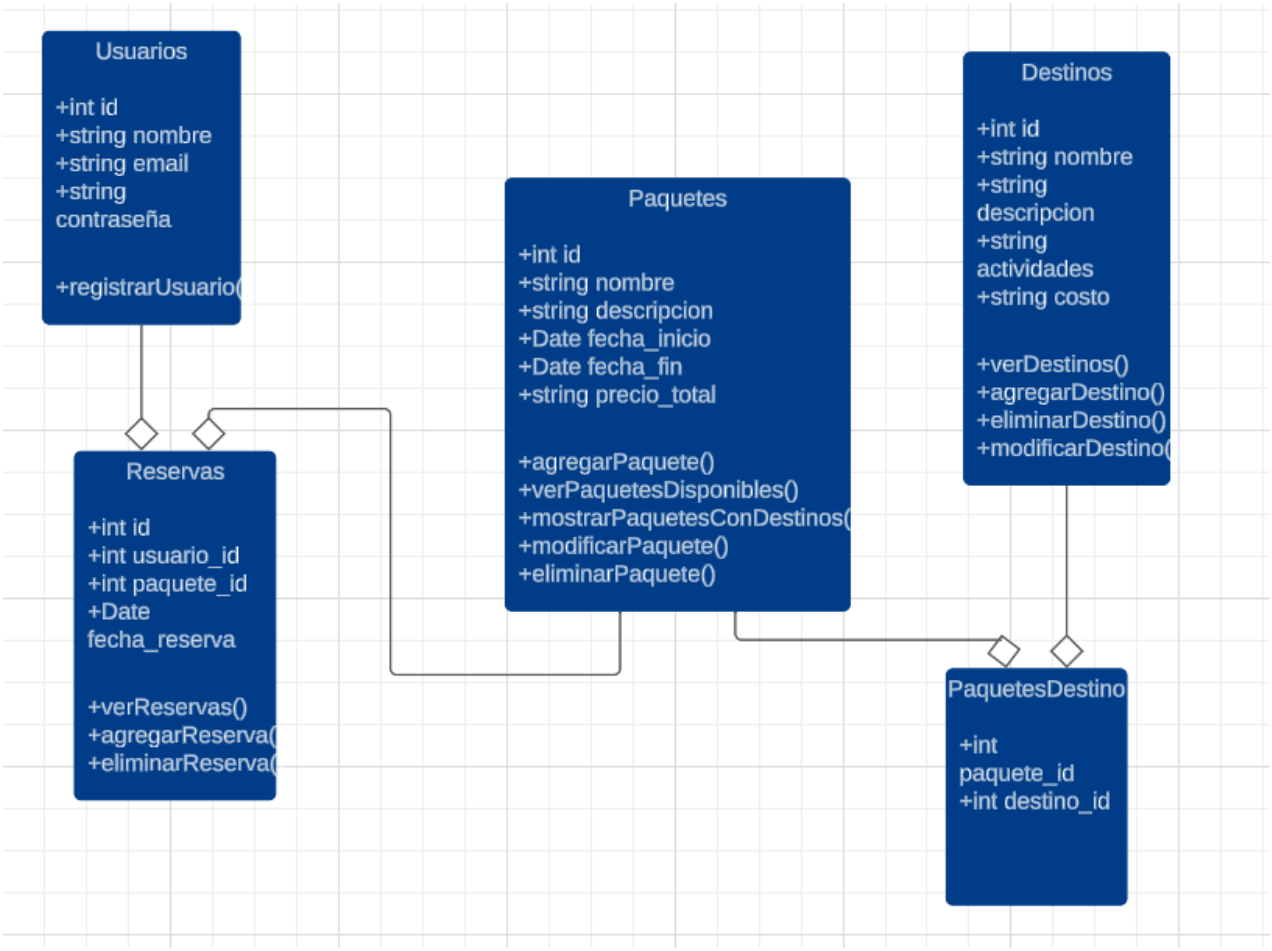
Las pruebas confirmaron que las funcionalidades principales del sistema operan correctamente. Sin embargo, se detectaron áreas de mejora, como la validación de entradas de datos en la interfaz y el manejo de errores en la conexión a la base de datos.

9 Anexos

Código fuente:

[1zcan/Viajes-Aventura: Evaluación n°4 de Programación orientada a objetos Inacap]

Diagrama UML:



Bitácora:

Fecha	Nombre de estudiante	Actividad realizada	Observaciones
05-12-24	Orlando Aravena	Creación del modelo para la gestión de destinos.	Completo y Probado
06-12-24	Marco Badillo	Desarrollo del controlador de reservas.	Se detectaron algunos errores
08-12-24	Jorge Dominguez	Implementación de la conexión con la base de datos.	Correcto, sin problemas
08-12-24	Orlando Aravena	Diseño de la interfaz en la vista (archivo vista.py).	Se planean mejoras estéticas futuras.
10-12-24	Marco Badillo	Pruebas unitarias de la funcionalidad de paquetes turísticos.	Todo en orden.
10-12-24	Jorge Dominguez	Revisión e integración de módulos (modelo, vista y controlador).	Sin errores.
11-12-24	Todo el equipo	Elaboración de la documentación técnica del proyecto.	En proceso.
12-12-24	Todo el equipo	Validación de entradas en la interfaz del usuario.	Validaciones agregadas exitosamente.

10 Conclusión

El proyecto "Viajes-Aventura" logró cumplir con los objetivos propuestos, desarrollando un sistema funcional que permite gestionar de manera eficiente los procesos de una agencia de viajes. A través de la implementación del patrón MVC, se logró una estructura de código clara, modular y fácil de mantener.

Logros

Gestión efectiva de usuarios, destinos, paquetes y reservas.

Base sólida para futuras expansiones del sistema, como la integración de una interfaz gráfica o funcionalidades avanzadas.

Mejoras Futuras

Implementar un sistema de autenticación robusto para los usuarios.

Desarrollar una interfaz gráfica o aplicación web para mejorar la experiencia del usuario.

Añadir funcionalidades como reportes de ventas y estadísticas sobre destinos más populares.