
INFORME TECNICO

Desarrollo de software para una agencia de viajes

NOMBRE: Jesus Oviedo Ramos

Jorge Rodríguez Leiva

Gerardo Reyes Reyes

CARRERA: Analista Programador

ASIGNATURA: Programación Orientada a Objeto Seguro

PROFESOR: Claudio Andrés Muños Sepúlveda

FECHA: 17/12/2025

1- INTRODUCCIÓN

En este informe se describe el proceso del diseño, desarrollo e implementación de una solución de software en Python además de una Base de Datos para una agencia de viajes ficticia llamada "Viajes Aventura". aplicando metodologías ágiles, principios de programación segura y conexión a la base de datos.

2-CONTEXTO Y PROBLEMÁTICA

La agencia "Viajes Aventura" requiere un sistema que permita gestionar destinos turísticos, paquetes de viaje y reservas de clientes de forma segura y eficiente. Actualmente, no cuentan con una plataforma digital que centralice esta información, lo que dificulta la administración y la experiencia del cliente.

3-OBJETIVOS DEL PROYECTO

Objetivo General

Desarrollar un sistema de reservas para una agencia de viajes utilizando Python, incorporando autenticación segura y persistencia en base de datos.

Objetivos Específicos

- Gestionar destinos turísticos (crear, listar, modificar y eliminar).
- Administrar paquetes turísticos y su disponibilidad.
- Implementar un sistema de reservas para clientes autenticados.
- Aplicar metodologías ágiles en el desarrollo del proyecto.
- Diseñar y documentar el sistema mediante diagramas UML y BPMN.

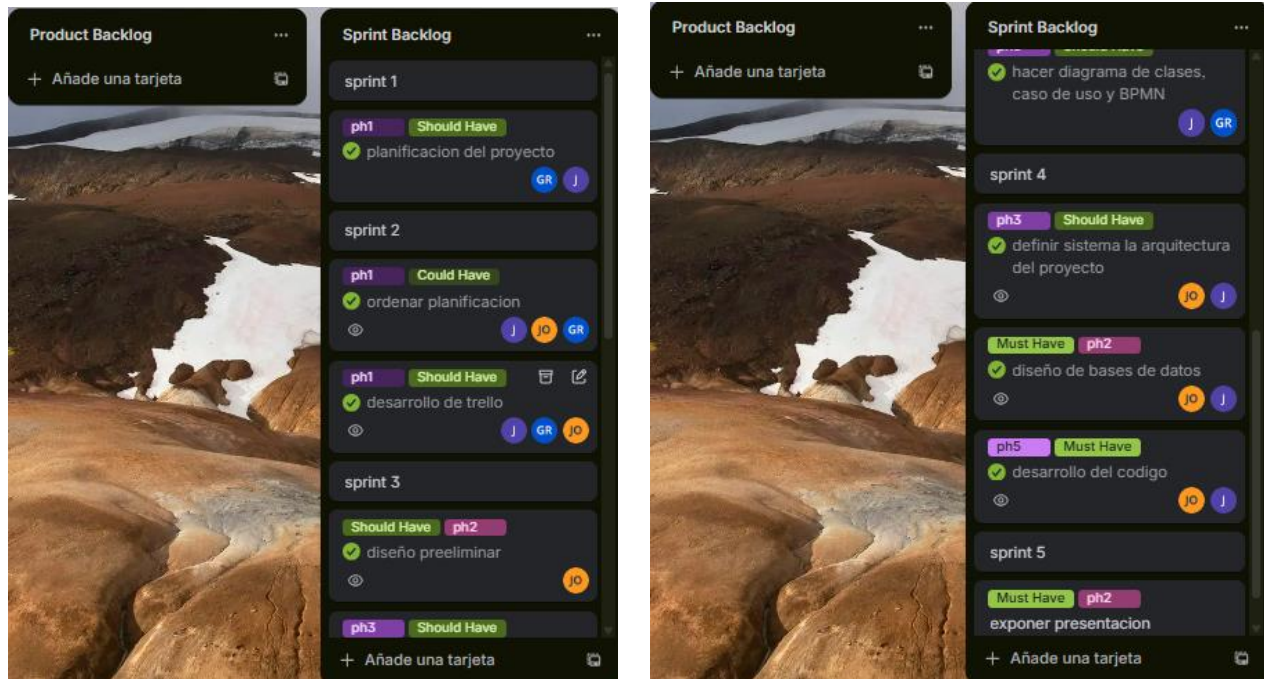
4-RECOLECCION DE REQUISITOS

Los requisitos fueron definidos a partir del caso planteado por el cliente simulado. Entre los principales se encuentran:

- Gestión completa de destinos.
- Creación y visualización de paquetes turísticos.
- Registro y autenticación de usuarios.
- Almacenamiento de información en una base de datos.

5-PLANIFICACION DEL PROYECTO

El proyecto se desarrolló utilizando una metodología ágil basada en Scrum, dividiendo el trabajo en Sprint's.



Cada sprint incluyó tareas asignadas a los integrantes del equipo, con revisiones periódicas y entregas parciales.

6-ARQUITECTURA DEL SOFTWARE

El sistema fue desarrollado con una estructura ordenada, separando las distintas partes del programa para facilitar su comprensión y mantenimiento. La lógica del sistema, el acceso a la base de datos y la interacción con el usuario se encuentran organizadas en módulos independientes. Esta forma de trabajo permite que el código sea más claro, fácil de modificar y menos propenso a errores, además de facilitar futuras mejoras al sistema.

7-PATRONES DE DISEÑO UTILIZADOS

Se utilizaron patrones como:

Modelo Vista Controlador (MVC): para separar responsabilidades. El Modelo se encarga de manejar los datos y la lógica del negocio, la Vista muestra la información al usuario, y el Controlador recibe las acciones del usuario y coordina el funcionamiento del sistema.

DAO (Data Access Object): se utiliza para manejar el acceso a la base de datos. Permite separar las consultas SQL del resto del programa, haciendo que el código sea más limpio y seguro. Gracias a este patrón, cualquier cambio en la base de datos se puede realizar sin afectar la lógica principal del sistema.

8-DISEÑO DEL SISTEMA

Diagrama de Clases

```

graph TD
    Cliente((Cliente))
    Administrador((Administrador))
    Sistema((Sistema))
    SistemaPagos((Sistema de pagos))

    subgraph Sistema
        Registrarse([Registrarse])
        IniciarSesion([Iniciar Sesión])
        VerPaquetes([Ver paquetes])
        ConsultarReservas([Consultar Reservas])
        ReservarPaquetes([Reservar Paquetes])
        pagarReserva([pagar reserva])
        recibirComprobante([recibir comprobante])
        emitirComprobante([emitir comprobante])
        ListarDestino([Listar Destino])
        CrearPaquetes([Crear Paquetes])
        AgregarDestino([Agregar Destino])
        EliminarDestino([Eliminar Destino])
        ModificarDestino([Modificar Destino])
        EliminarPaquete([Eliminar Paquete])
        EditarPaquete([Editar Paquete])
        mostrarReservas([mostrar reservas])
        RealizarReserva([Realizar Reserva])
        ProcesarPago([Procesar Pago])
        ValidarPago([Validar Pago])
    end

    Cliente --> Registrarse
    Cliente --> IniciarSesion
    Cliente --> VerPaquetes
    Cliente --> ConsultarReservas
    Cliente --> ReservarPaquetes
    Cliente --> recibirComprobante

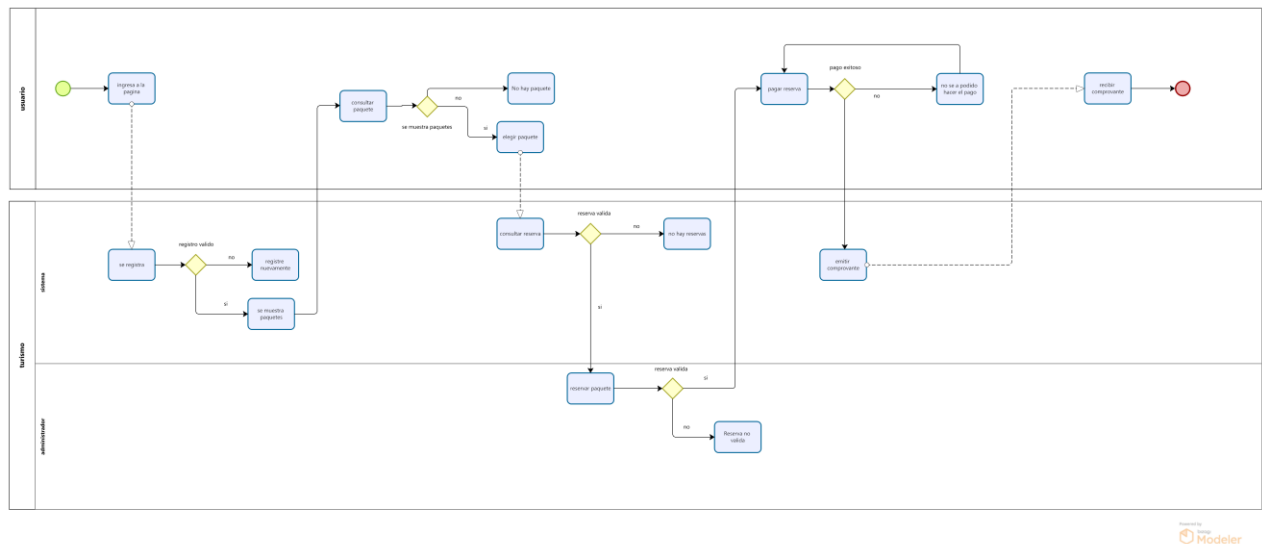
    Administrador --> ListarDestino
    Administrador --> CrearPaquetes
    Administrador --> AgregarDestino
    Administrador --> EliminarDestino
    Administrador --> ModificarDestino
    Administrador --> EliminarPaquete
    Administrador --> EditarPaquete

    Sistema --> ConsultarReservas
    Sistema --> mostrarReservas
    Sistema --> RealizarReserva

    SistemaPagos --> ValidarPago
    SistemaPagos --> emitirComprobante

    ReservarPaquetes --> pagarReserva
    pagarReserva --> ProcesarPago
    ProcesarPago --> ValidarPago
    ValidarPago --> emitirComprobante
    emitirComprobante --> recibirComprobante
  
```

Procesos de Negocio (BPMN)



9-DESARROLLO DE LA SOLUCION

El código fue desarrollado en Python aplicando programación orientada a objetos seguro. El código se encuentra estructurado en módulos y clases.

10-SEGURIDAD Y AUTENTICACION

El sistema implementa autenticación segura mediante hash de contraseñas, evitando el almacenamiento de contraseñas en texto plano, además al crear un nuevo administrador se le pide un código unico

11-PERSISTENCIA Y BASE DE DATOS

La información del sistema se almacena en una base de datos en appserver conectado a phpmyadmin. Las principales tablas corresponden a usuarios, destinos, paquetes turísticos y reservas.

12-PRUEBAS REALIZADAS

Se realizaron varias pruebas para validar el correcto funcionamiento del sistema y la integración entre sus componentes. Resultando en la correcta ejecución del programa en su totalidad

13-RESULTADOS Y DEMOSTRACION

El sistema desarrollado cumple con los requisitos planteados por el cliente, permitiendo gestionar destinos, paquetes y reservas de manera segura. E incluso se agregaron funciones nuevas y un mayor proceso de seguridad y verificación para administradores

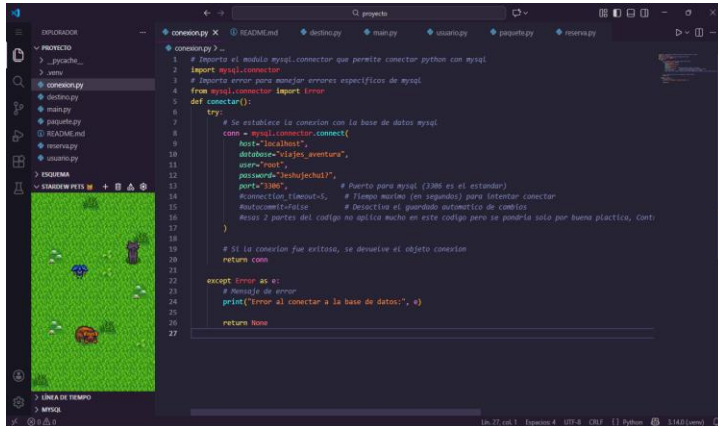
14-CONCLUSION

El desarrollo del proyecto permitió aplicar conocimientos de programación orientada a objetos seguros, Bases de Datos, seguridad y metodologías ágiles, logrando una solución funcional y escalable para una agencia de viajes.

15. Anexos

Código fuente en Python.

Conexion a la Base de Datos

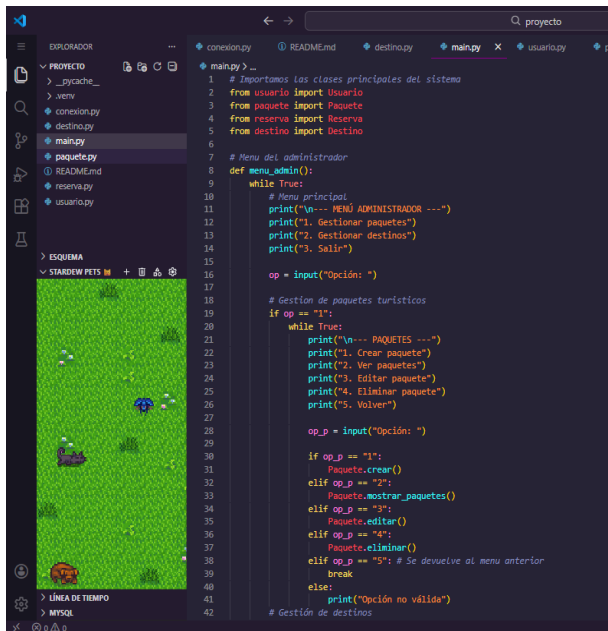


```

1 # Importa el modulo mysql.connector que permite conectar python con mysql
2 import mysql.connector
3 # Importa error para manejar errores especificos de mysql
4 from mysql.connector import error
5 def conectar():
6     try:
7         # Se establece la conexion con la base de datos mysql
8         con = mysql.connector.connect(
9             host="localhost",
10            database="viajes_aventura",
11            user="root",
12            password="desajuchul",
13            port="3306", # Puerto para mysql (3306 es el estandar)
14            # Tiempo maximo (en segundos) para intentar conectar
15            # Autocommit=False # Desactiva el guardado automatico de cambios
16            # Mas 2 partes del codigo no aplica mucho en este codigo pero se ponian la sola por buena practica, Contr
17        )
18        # Si la conexion fue exitosa, se devuelve el objeto conexion
19        return con
20    except error as e:
21        # Mensaje de error
22        print("Error al conectar a la base de datos:", e)
23        return None
24
25
26
27

```

Código principal main



```

1 # Importamos las clases principales del sistema
2 from usuario import Usuario
3 from paquete import Paquete
4 from reserva import Reserva
5 from destino import Destino
6
7 # Menu del administrador
8 def menu_admin():
9     while True:
10         # Menu principal
11         print("\n--- MENÚ ADMINISTRADOR ---")
12         print("1. Gestionar paquetes")
13         print("2. Gestionar destinos")
14         print("3. Salir")
15
16         op = input("Opción: ")
17
18         # Gestión de paquetes turísticos
19         if op == "1":
20             while True:
21                 print("\n--- PAQUETES ---")
22                 print("1. Crear paquete")
23                 print("2. Ver paquetes")
24                 print("3. Editar paquete")
25                 print("4. Eliminar paquete")
26                 print("5. Volver")
27
28                 op_p = input("Opción: ")
29
30                 if op_p == "1":
31                     Paquete.crear()
32                 elif op_p == "2":
33                     Paquete.mostrar_paquetes()
34                 elif op_p == "3":
35                     Paquete.editar()
36                 elif op_p == "4":
37                     Paquete.eliminar()
38                 elif op_p == "5": # Se devuelve al menu anterior
39                     break
40                 else:
41                     print("Opción no válida")
42
43             # Gestión de destinos

```

Capturas de la ejecución del sistema.

```
=== VIAJES AVENTURA ===
1. Iniciar sesión
2. Registrarse
3. Salir
Opción: 1
Nombre de usuario: admin
Contraseña: admin123
Bienvenido admin (admin)

--- MENÚ ADMINISTRADOR ---
1. Gestionar paquetes
2. Gestionar destinos
3. Salir
Opción: 1

--- PAQUETES ---
1. Crear paquete
2. Ver paquetes
3. Editar paquete
4. Eliminar paquete
5. Volver
Opción: 5

--- MENÚ ADMINISTRADOR ---
1. Gestionar paquetes
2. Gestionar destinos
3. Salir
Opción: 2

--- DESTINOS ---
1. Crear destino
2. Ver destinos
3. Editar destino
4. Eliminar destino
5. Volver
Opción: █
```

Menú de administrador

```
=== VIAJES AVENTURA ===
1. Iniciar sesión
2. Registrarse
3. Salir
Opción: 1
Nombre de usuario: iñaqui
Contraseña: iñaqui123
Bienvenido iñaqui (user)

--- MENÚ USUARIO ---
1. Ver paquetes
2. Reservar paquete
3. Ver mis reservas
4. Cancelar reserva
5. Salir
Opción: █
```

Menú de usuario