Informe de Evaluación y Desarrollo del Sistema de Reservas de Hoteles

Isabella Ramírez

Diego Fernando Marín

Corporación Universitaria Remington

Facultad Ingeniería Sistemas

Santiago de Cali 2025

1. Evaluación detallada de los requerimientos

Como parte del proceso de validación de requerimientos, se tomó el repositorio de un compañero, según lo asignado por el profesor, y se realizó un análisis detallado de los requerimientos funcionales y no funcionales del sistema de reservas. A continuación, se presentan los resultados obtenidos:

Claridad: Se identificaron requerimientos bien definidos, aunque algunos carecen de detalles específicos, como los filtros avanzados de búsqueda y el tipo de cifrado de datos.

Completitud: Se encontraron omisiones en aspectos clave, como el manejo de estados de hoteles, autenticación antes de reservar, edición de datos personales y medidas adicionales de seguridad.

Consistencia: La mayoría de los requerimientos son consistentes con la entrevista realizada, aunque algunos elementos, como compatibilidad con navegadores y validaciones de seguridad en pagos, no fueron mencionados explícitamente.

Verificabilidad: Se determinó que los requerimientos pueden validarse mediante pruebas funcionales, auditorías de seguridad y evaluaciones de rendimiento.

Requerimientos	Claridad	Completitud	Consistencia	Verificabilidad
El sistema debe permitir registrar y actualizar la información de los hoteles, incluyendo nombre, ubicación y servicios ofrecidos.	☑	No incluye estados de los hoteles (activos/inactivos), fotos ni ofertas.	Compatible con la entrevista, pero falta el manejo de estados.	Se puede verificar registrando y actualizando hoteles.
El sistema debe permitir a los administradores gestionar la disponibilidad y precios de las habitaciones.	☑	No menciona control de temporadas o descuentos.	☑	Verificable probando ajustes de disponibilidad y precios.
El sistema debe permitir a los clientes buscar habitaciones aplicando filtros avanzados.	Claro, pero no especifica qué filtros estarán disponibles.	La entrevista menciona filtros detallados, pero aquí no se listan.	☑	Se puede probar aplicando distintos filtros.
El sistema debe garantizar que el proceso de reserva sea rápido y seguro.	Claro, pero falta especificar cómo se implementará la seguridad.	No menciona autenticación antes de reservar.	☑	Se puede verificar con pruebas de tiempos de respuesta y seguridad.
El sistema debe permitir el registro, autenticación y administración de usuarios.	Claro, aunque no menciona los tipos de roles.	No especifica los datos requeridos para el registro.	☑	Verificable probando registro y autenticación.
El sistema debe permitir a los clientes gestionar su perfil y visualizar su historial de reservas.	☑	No menciona edición de datos personales o eliminación de cuentas.	☑	Se puede probar con usuarios gestionando su perfil.
El sistema debe permitir que los clientes califiquen y comenten sobre su experiencia en los hoteles.	☑	☑	☑	Se puede probar ingresando calificaciones y comentarios.
Los comentarios deben ser visibles en la página del hotel correspondiente.	☑	No especifica si se pueden responder comentarios.	☑	Verificable revisando la página del hotel.
El sistema debe procesar las búsquedas y mostrar los resultados en menos de 3 segundos.	☑	☑	☑	Verificable con pruebas de carga.
Debe permitir la concurrencia de múltiples usuarios sin afectar el rendimiento.	☑	No menciona límites máximos de usuarios simultáneos.	☑	Se puede probar con pruebas de estrés.
Debe contar con cifrado para proteger datos personales y financieros.	Claro, pero no especifica qué tipo de cifrado.	No menciona medidas adicionales como autenticación en dos pasos.	☑	Verificable con auditorías de seguridad.
El acceso debe estar asegurado mediante autenticación de usuarios.	☑	☑	☑	Verificable probando autenticación.
La interfaz debe ser intuitiva y accesible desde cualquier dispositivo.	☑	No especifica compatibilidad con navegadores o sistemas operativos.	No se menciona en la entrevista, pero es buena práctica.	Se puede probar en distintos dispositivos.
Debe cumplir con estándares de accesibilidad para garantizar una experiencia inclusiva.	Claro, pero no especifica qué estándares.	No mencionado en la entrevista.	☑	Se puede evaluar con herramientas de accesibilidad.
El sistema debe poder manejar un crecimiento en el número de usuarios y transacciones sin comprometer su rendimiento.	☑	No menciona estrategias de escalabilidad.	No se menciona en la entrevista, pero es relevante.	Se puede probar con simulaciones de crecimiento.
Debe contar con documentación clara y actualizada para facilitar futuras mejoras y mantenimiento.	☑	No menciona qué incluirá la documentación.	No se menciona en la entrevista.	Verificable revisando documentación técnica.
El sistema debe integrarse con pasarelas de pago seguras.	Claro, pero no menciona cuáles.	No menciona validaciones de seguridad en los pagos.	☑	Se puede probar con transacciones reales.
Debe utilizar protocolos HTTP/HTTPS para la comunicación de datos.	☑	No menciona validaciones de certificados de seguridad.	No se menciona en la entrevista, pero es un estándar necesario.	Verificable con análisis de red.

2. Código Python implementado

El sistema ha sido desarrollado en **Python**, aplicando principios de **POO** para una gestión eficiente y escalable en reservas hoteleras. Se describe las principales clases y funcionalidades implementadas:

Clases Principales

1. **Usuario**: Clase base que modela atributos y métodos comunes para clientes y administradores.
    - **Cliente**: Hereda de la clase Usuario y permite gestionar reservas personales, verificar disponibilidad y realizar pagos.
    - **Administrador**: Hereda de Usuario y está encargado de gestionar hoteles, usuarios y generar reportes administrativos.
2. **Hotel**: Clase que representa hoteles registrados en el sistema, con información detallada sobre sus habitaciones y servicios.
3. **Habitación**: Contiene atributos como tipo de habitación (individual, doble, suite), precio, y disponibilidad.
4. **Reserva**: Se encarga de manejar la gestión de reservas, incluyendo fechas de entrada y salida, cálculo de costos y estado de la reserva.
5. **Pago**: Administra los pagos asociados a las reservas, incluyendo métodos de pago y comprobantes.

Estructura del Proyecto

El proyecto sigue una estructura organizada de carpetas y archivos para mantener la claridad y modularidad del código:

- **docs/sistema_reservas/models/**: Contiene las clases que definen el modelo de datos del sistema, como Usuario, Cliente, Administrador, Hotel, Habitacion, Reserva y Pago.
- **docs/sistema_reservas/main.py**: Archivo principal que ejecuta y valida el sistema. Sirve como punto de entrada para probar las funcionalidades integradas.

Ejemplo de Ejecución del Proyecto

Para ejecutar el sistema, usar el siguiente comando en la terminal:

$ python docs/sistema_reservas/main.py

Este comando ejecutará la lógica principal y permitirá validar la creación de usuarios, gestión de reservas, pagos, etc.

3. Conclusiones sobre la calidad del diseño y la implementación

El diseño del sistema es adecuado y refleja una correcta aplicación de POO. El diagrama UML muestra cómo las clases están organizadas de manera lógica para cumplir con los principales requerimientos funcionales. Por ejemplo, las clases Administrador y Cliente heredan de Usuario, lo que ayuda a reutilizar código y facilita posibles mejoras futuras.

Además, las relaciones entre clases como Reserva, Hotel y Habitación aseguran que la gestión de disponibilidad y precios esté correctamente implementada.

En cuanto a los requisitos no funcionales, el diseño también parece orientado a soportar escalabilidad y rendimiento, aunque sería importante detallar más aspectos relacionados con la seguridad, como el cifrado de datos y la autenticación.

Finalmente, aunque el diseño sugiere que la interfaz puede ser accesible e intuitiva, habría que asegurarse de implementar estándares de usabilidad para garantizar una buena experiencia de usuario.