**Entendimiento del proyecto**

Con base a un manual de instrucciones se diseñó con énfasis en programación un sistema de gestión para el **Hotel El Turista**, en el marco del curso de Algoritmia y Programación. Este desarrollo responde a una necesidad real de **digitalizar y automatizar los procesos operativos del hotel**, que actualmente son gestionados de forma manual.

Antes de iniciar con el diseño del algoritmo principal, realizamos un **proceso de levantamiento de requisitos** guiado por una serie de preguntas clave. Estas preguntas nos permitieron comprender mejor las necesidades del usuario (en este caso, el recepcionista y el administrador del hotel), definir los objetivos del software, las funcionalidades mínimas viables, las restricciones tecnológicas, y los criterios de calidad esperados.

Este entendimiento previo no solo orientó nuestra lógica de programación, sino que nos permitió estructurar un algoritmo **claro, modular y validado**, en el que cada función responde a un requerimiento real. El siguiente documento presenta las respuestas detalladas a ese proceso de análisis, antes de continuar con la lógica y codificación del sistema.

**Objetivos y propósitos**

* **Cuál es el objetivo principal de este programa para el hotel?**

El objetivo principal es digitalizar y optimizar la gestión operativa del Hotel El Turista, mediante un software de consola en Python que permita registrar huéspedes, administrar reservas, controlar ingresos y salidas, emitir facturación y generar reportes administrativos de forma automatizada, precisa y segura.

* **¿Qué problema estamos resolviendo con este sistema?**

Se resuelve la ineficiencia del sistema manual actual, que depende de registros físicos propensos a errores, pérdida de información y lentitud operativa. El sistema mejora la trazabilidad, la disponibilidad de datos en tiempo real y la capacidad de generar reportes confiables para la toma de decisiones.

* **¿Quién lo va a usar? (¿el recepcionista?, ¿el administrador?, ¿el huésped?**

El sistema será usado principalmente por los **recepcionistas** para gestionar las operaciones diarias (registro, reservas, check-out), y por el **administrador** para acceder a reportes e indicadores de desempeño. El huésped no interactúa directamente con el software.

**¿Qué cosas necesita hacer esa persona desde el sistema?**

* El **recepcionista** debe poder: registrar huéspedes, realizar reservas, verificar disponibilidad, registrar salidas y generar facturas.
* El **administrador** debe poder: autenticarse, acceder a reportes estadísticos, consultar ingresos, disponibilidad, y generar gráficos con matplotlib.

**Funcionalidades y requisitos**

* **¿Qué cosas obligatoriamente debe hacer el sistema?**
* Registrar y validar datos de huéspedes.
* Realizar reservas verificando disponibilidad.
* Registrar check-out y generar facturas.
* Acceder a reportes administrativos.
* Generar gráficos estadísticos con Matplotlib.
* Exportar datos a archivos CSV.
* Registrar eventos en un archivo de log.
* Proteger el módulo de administración con autenticación.
* **¿Hay cosas que sería chévere tener, pero que no son urgentes ni necesarias?**
* Interfaz gráfica (GUI) en lugar de consola.
* Envío automático de confirmaciones de reserva por correo.
* Registro de observaciones personalizadas por huésped.
* Backups automáticos de los registros.
* Control de horarios o turnos de recepcionistas.
* **¿Qué cosas el sistema no va a hacer?**
* No permitirá que el huésped interactúe directamente.
* No gestionará pagos electrónicos ni conexión con bancos.
* No permitirá mascotas ni habitaciones múltiples.
* No se integra con otros sistemas externos (por ahora).

**Prioridades y alcance**

**Si el tiempo fuera poco:**

* **¿cuáles son las funciones más importantes que sí o sí debe tener?**
* Registro de huéspedes con validación.
* Reserva de habitaciones con disponibilidad.
* Registro de salida y facturación.
* Acceso del administrador a estadísticas básicas.
* Exportación de datos (CSV).
* **¿Qué funciones se podrían hacer después si hay tiempo?**
* Visualización avanzada de gráficos con Matplotlib.
* Generación automática de reportes históricos.
* Implementación de filtros avanzados por fechas o tipos de huéspedes.
* Personalización de habitaciones (servicios, precios dinámicos).
* **¿Hay alguna versión futura del programa que se piensa mejorar más adelante?**

Sí. En una versión futura se podría:

* Migrar a una interfaz gráfica amigable.
* Incluir una base de datos relacional.
* Permitir gestión de usuarios múltiples con roles y permisos.
* Agregar funcionalidades para marketing (registro de clientes frecuentes, promociones).

**Limitaciones y restricciones**

* **¿Podemos usar cualquier herramienta o lenguaje? ¿Solo Python]?**

El proyecto está limitado al uso de **Python en consola**, tal como lo especifica el documento. Además, se deben utilizar librerías como pandas, matplotlib, y funciones de entrada/salida de archivos. No se permite el uso de interfaces gráficas ni otros lenguajes de programación.

* **¿Tenemos un tiempo límite para entregar el proyecto?**

Sí. Según las instrucciones del profesor, existen dos entregas claves:

* **Primera entrega:** Semana 10 (componentes 1 al 7)
* **Entrega final:** Semana 16 (proyecto completo + sustentación)
* **¿Hay reglas o condiciones especiales que debemos tener en cuenta?**

Sí. Algunas de las condiciones más relevantes son:

* No se permiten mascotas en el sistema de reservas.
* Las habitaciones son exclusivamente individuales o suites.
* Se debe usar GitHub como repositorio oficial del proyecto.
* La autenticación del administrador debe leerse desde un archivo externo.

**Interfaz y Experiencia del usuario**

* **Ya hay algún dibujo o idea de cómo debe verse el programa en consola?**

Sí. En el documento aparece una ilustración que representa un menú de inicio en consola. Este debe ser visualmente amigable, con opciones claras y navegación simple para los usuarios.

* **¿Cómo debe sentirse la persona que lo usa? ¿Fácil? ¿Claro? ¿Rápido?**

El programa debe sentirse **intuitivo, fluido y confiable**. La navegación debe ser clara, con mensajes orientadores, validaciones visibles y tiempos de respuesta rápidos. El usuario debe tener la seguridad de que cada acción está siendo procesada correctamente.

* **¿El programa debe funcionar para personas con alguna discapacidad?**

No se menciona como requisito explícito. Sin embargo, dado que se trabaja por consola, se pueden seguir buenas prácticas como:

* Colores de alto contraste
* Mensajes de texto suficientemente descriptivos  
  Esto puede facilitar el uso a personas con dificultades visuales leves.

**Pruebas y calidad**

* **¿Cómo sabremos si el sistema funciona bien?**

Mediante la realización de pruebas funcionales y de validación con diferentes escenarios:

* Registro de datos con errores.
* Reservas sin disponibilidad.
* Facturación correcta según tipo de habitación y noches.
* Reportes coherentes con los datos reales.
* **¿Qué cosas debe cumplir para decir que está bien hecho?**
* Que cada función haga exactamente lo que debe hacer.
* Que los datos estén bien validados y almacenados.
* Que los reportes y gráficos sean correctos.
* Que el usuario no se sienta perdido o confundido.
* Que todos los requisitos del documento se hayan cumplido.
* **¿Quién revisará si todo está funcionando correctamente?**

El **profesor (Product Owner)** es quien asumirá el rol de cliente final. Él será quien verifique que el sistema cumple los requerimientos y evaluará tanto la funcionalidad como la documentación y la sustentación del grupo.

**Riesgos y mitigación**

* **¿Qué podría salir mal durante el proyecto?**
* Pérdida de archivos o versiones por mala gestión de GitHub.
* Falta de comunicación entre los miembros del grupo.
* Errores de última hora al ejecutar el sistema.
* Desacuerdos sobre responsabilidades.
* Que no se comprendan bien los requisitos.
* **¿Cómo podemos prepararnos para que eso no pase?**
* Usar GitHub desde el inicio con backups regulares.
* Hacer reuniones semanales de seguimiento.
* Mantener un log claro de tareas asignadas.
* Estudiar bien el documento y resolver dudas a tiempo.
* Dividir bien los módulos y probarlos individualmente antes de integrarlos.
* **¿Qué hacemos si en el último momento algo falla?**
* Tener un **plan de recuperación rápida**, como versiones anteriores del código guardadas.
* Establecer un **responsable de pruebas finales.**
* Si hay errores menores, documentarlos y explicarlos con claridad durante la sustentación.
* Priorizar que el sistema funcione en sus funciones esenciales.

**Colaboración y comunicación**

* **¿Cómo nos vamos a comunicar con el profesor y entre nosotros?**
* Usaremos **grupos de WhatsApp** o correo institucional para comunicarnos entre nosotros.
* Para preguntas al profesor, se usarán los canales establecidos en el curso (correo, foros o clases sincrónicas).
* **¿Dónde vamos a guardar el proyecto? (Ej: en GitHub)**

Todo el proyecto será almacenado y actualizado en un **repositorio en GitHub**, creado por el líder del grupo y compartido con los demás integrantes.

* **¿Quién se encarga de subir los archivos? ¿Quién organiza las tareas?**
* El **líder del grupo** será responsable de subir y gestionar los archivos en GitHub.
* Las tareas serán organizadas en un **acta de responsabilidad**, donde cada miembro asumirá un rol (programación, documentación, pruebas).