DISEÑO DE GESTIÓN DE ERRORES:

Es crucial tener en cuenta los factores que pueden afectar el funcionamiento adecuado de nuestro programa, por lo tanto, hemos identificado los posibles errores que pueden ocurrir al implementar la aplicación. Las principales fuentes de errores que hemos detectado son los errores de entrada del usuario, como errores de escritura, entre otros, así como posibles errores en la carga de datos. En general, consideramos que la mejor manera de manejar los errores es a través del uso de excepciones.

Las excepciones se utilizarían en los siguientes casos:

1. Autenticación de usuario: Antes de ingresar a la aplicación, se solicita al usuario que ingrese un nombre de usuario y contraseña. Es posible que la persona que ejecute el programa ingrese credenciales incorrectas o que no existan en el sistema. En caso de no encontrar la información correspondiente, se mostraría un mensaje indicando que el usuario es incorrecto o no existe, o que la contraseña es incorrecta. Esta lógica se implementa directamente en la parte del controlador (controller) para lanzar un error cuando no se encuentra un usuario válido.

2. Manejo de archivos: Existe la posibilidad de que, al solicitar al usuario o al administrador que carguen un archivo, este no se encuentre o no exista. En este caso, se utilizaría una excepción del tipo FileNotFoundException. Sin embargo, incluso si el archivo se carga correctamente, podría haber problemas al leerlo. En ese caso, se lanzaría una excepción con un mensaje indicando "Error leyendo el archivo". En el diagrama de clases UML y en el código, esta excepción se representa mediante la clase Formato Archivo Exception.

3. Error de persistencia: Esta excepción se lanzaría cuando el estado de la aplicación no se pueda ejecutar correctamente. Por ejemplo, si se intenta abrir la aplicación después de haberse cerrado, se mostraría un mensaje de error indicando "Imposible restaurar el estado del programa". Del mismo modo, al intentar cerrar el programa, si no se puede guardar el estado actual, se mostraría un mensaje de error. Este tipo de excepción se maneja mediante la clase PersistenceException.

COMPONENTES LÓGICOS DE LA ETAPA 3:

Para esta tercera parte del proyecto, fue necesario agregar código nuevo al ya existente, siguiendo los lineamientos de los principios SOLID. Por otro lado, los nuevos requerimientos fueron hechos teniendo en cuenta el funcionamiento previo del programa, por esto mismo, decidimos que la mejor forma de abordar los nuevos requerimientos era por medio del desarrollo de nuevas clases, tanto del modelo como de la vista.

Añadimos diferentes funcionalidades al código como una interfaz destinada únicamente a los usuarios, una facturación electrónica, entre otras.

PROCESO DE DESARROLLO:

La fase inicial del proyecto resultó ser la más desafiante en nuestra opinión. Requirió un análisis exhaustivo del enunciado que describía el programa que debíamos desarrollar a lo largo del semestre, así como un arduo trabajo para proponer posibles soluciones y establecer las relaciones entre las clases.
En esta etapa inicial del proyecto, además del análisis y diseño, también tuvimos que implementar el programa utilizando como base el lenguaje JAVA. Durante esta implementación, nos apoyamos en los diagramas de UML que habíamos creado en las etapas de análisis y diseño, para establecer las relaciones entre los componentes del sistema.

El siguiente paso consistió en desarrollar la interfaz visual de nuestra aplicación. Hasta ese momento, todo el trabajo se había realizado en la consola del entorno de desarrollo integrado (IDE), por lo que ahora teníamos que crear una interfaz gráfica de usuario (GUI). Utilizando bibliotecas de desarrollo de GUI como Java Swing y Java AWT, logramos crear una interfaz estéticamente agradable y amigable para el usuario. Además, logramos conectar las funciones implementadas en el modelo con la interfaz gráfica para que fueran plenamente funcionales.

Por último, era necesario realizar ajustes adicionales para hacer la aplicación más completa. Estos ajustes incluían la implementación de una interfaz para los usuarios del hotel y la integración de un sistema de pago electrónico. Además, se hizo necesario establecer un sistema de pruebas para facilitar la detección de errores en el código. Esta fase fue crucial, ya que fue fundamental adaptarse a los cambios que pudieran surgir en la aplicación en cualquier momento.

Como reflexión final, hemos apreciado la importancia de desarrollar proyectos de esta naturaleza y de seguir estos pasos. Ya que nos ha preparado para comprender el funcionamiento del desarrollo de proyectos desde la perspectiva de un ingeniero de sistemas.

¿QUE SALIÓ BIEN Y QUE SALIÓ MAL?
Dos de los aspectos negativos que resaltamos de nuestro proceso, es que debido a un mal proceso de diseño inicial nuestro código resultó con un leve nivel de acoplamiento. De igual forma, en el diseño inicial no tuvimos en cuenta el TAD que íbamos a utilizar para los datos, lo que resultó en un proceso más complicado, si tuviéramos que cambiar algo del proceso que hicimos sería, sin duda alguna, el usar mapas en vez de listas, esa fue una decisión no acertada.

En la otra mano, de los aspectos positivos resaltamos la funcionalidad del código, ya que aún siendo una implementación complicada, logramos sacar adelante los requerimientos que nos pedían. Asimismo, la efectividad en nuestro trabajo gracias a la organización con la que llevábamos el proyecto, al tener todos nuestros documentos y archivos ordenados correctamente, fue más fácil el entendimiento del código, lo cual ayudó a los cambios que tenían que ser realizados a lo largo de la implementación .

ASPECTOS A MEJORAR:
En primera instancia, el ponerle mayor atención al diseño. Esto debido a que si el diseño es errado en un inicio, se genera un efecto bola de nieve que acaba repercutiendo en altas horas del proyecto.
De igual forma, ser conscientes del tiempo que se tiene para el desarrollo de las etapas, el que parezca fácil no significa que lo sea. En un punto del desarrollo tuvimos que desvelarnos para hacer la entrega debido a un despiste con el plazo de la entrega, lo que pudo dar pie a que se cometieron errores.

DIAGRAMA UML DEL PROYECTO:
https://app.diagrams.net/#G1nz3jAm2q8i619AIAxj1GKYxQz_nPQ32J