Proyecto #3 – Entrega 2: Proceso y reflexión

Departamento de Ingeniería de Sistemas y Computación
ISIS 1226 – Diseño y programación O.O.

J. Camilo Mercado – 202021541; Carol Florido – 202111430; Elkin Cuello – 202215037

¿Qué cosas salieron bien y qué cosas salieron mal?

Respecto al proyecto 1, los aspectos principalmente salieron bien, en cuanto el diseño fue coherente con la implementación y los diagramas incluidos en el documento de diseño, por lo que se siguió el mismo diseño en todas las entregas (indicando que fue acertado), aún así existen aspectos que pudieron haberse hecho mejor, como hacer un diseño menos confuso y con clases altamente acopladas y gran cantidad de clases, eso pudo haberse solucionado usando un patrón desde el comienzo del proyecto o habiendo aplicado mejor el concepto de bajo acoplamiento alta cohesión. Esto último hizo que la implementación para la entrega 1 fuera tediosa y que existiesen conflictos entre las clases por el alto acoplamiento, pues si se modificaba solo un aspecto de una clase, por ejemplo, un caso que ocurrió, las habitaciones, había que modificar reservas, inventario, persistencias de reservas e inventario, la clase central PMS, y la aplicación, lo cual entorpeció el proceso de desarrollo en el grupo e hizo que existiesen problemas de comunicación entre los participantes, aún así se logró sobrellevar este inconveniente, pero sigue siendo un aspecto que se pudo haber mejorado.

Respecto al proyecto 2, de nuevo se puede decir que la mayoría de los aspectos salieron bien debido a que la implementación del diseño de interfaz en la entrega 1 fue completamente cubierto e incluso mejorado por la entrega 2, pues algo que funcionó fue hacer el diseño como mockup directamente con Swing en vez de usar otras herramientas, lo que ayudó a que fuera fácil la implementación de funcionalidades en el código. Aun así, nuevamente se presentó el problema de diseño del proyecto que fue no centralizar las clases y hacer muchas clases que al final era tedioso acceder a ellas por la cantidad que existen en el proyecto, sumando el mismo problema con las aplicaciones, pues se crearon 11 clases para hacer la interfaz gráfica, lo cuál generó un diseño todavía más complejo, y al no usar un patrón de diseño realmente se asemejó a la arquitectura “espagueti”, siguiendo realmente un anti-patrón. Este aspecto pudo haberse mejorado siguiendo un patrón que se ajustase a las necesidades del proyecto, con pocas clases y bajo acoplamiento. Aún así se logró un resultado satisfactorio para la implementación de interfaces gráficas en cuanto funcionaba correctamente manejando errores y excepciones, y evitando el detenimiento repentino del programa.

Por último, respecto al proyecto 3, se considera que los aspectos positivos predominan en la entrega en cuanto, a pesar de la cantidad de cambios y los problemas mencionados con el diseño, las funcionalidades de las pasarelas y la aplicación de huéspedes se hicieron ya centralizando clases y distribuyendo responsabilidades de una forma que redujera el acoplamiento entre las nuevas clases. Los reportes y gráficas, gracias a que la clase central PMS tenía una estructura clara y métodos con solo una responsabilidad se realizaron con los

datos que ya se tenían de estos sin mayor problema, y lo mismo ocurrió con las pruebas unitarias y de integración, pues se probaron los métodos específicos de los aspectos y se cubrió completamente las partes deseadas con resultados acertados. En cuanto aspectos que resultaron problemáticos, de nuevo, se encuentra la cuestión del alto acoplamiento entre las clases pre-existentes, haciendo que la implementación de cambios, para las habitaciones hizo que se tuvieran que modificar todas las apps dependientes (AppAdmin, App, AppRecep, Recepcionista, Reservas, Inventario, persistencias asociadas a esos y las interfaces), siendo haciendo complejo el desarrollo.

¿Qué decisiones resultaron acertadas y qué decisiones fueron problemáticas?

Las decisiones acertadas fueron las siguientes:

- *Distribución de responsabilidades individuales para los métodos*: al hacer que cada método hiciese solo una tarea se logró habilidad para la implementación en grupo y el entendimiento del código por cada miembro del grupo.
- *Establecer formatos antes de cada implementación*: Antes de cada implementación se acordaban aspectos estéticos y funcionales para la uniformidad del trabajo, lo cuál ayudó en gran medida para que cada miembro del grupo pudiera desarrollar correctamente sus partes que dependían de las de los otros compañeros y con una estética uniforme que le da atributos de calidad al proyecto.
- *Buen diseño inicial y acogerse fielmente a este durante el desarrollo*: Al diseñar se tuvo en cuenta todos los aspectos necesarios, y al seguir fielmente el diseño, se facilitó el desarrollo de partes dependientes y no se tuvo modificaciones significativas de este a lo largo de los desarrollos de los proyectos.

Las decisiones problemáticas fueron las siguientes:

- *Generar un diseño inicial con muchas clases sin un patrón:* Esto resulto en un alto acoplamiento y el uso de un anti-patrón como se mencionó en el literal anterior, lo cual entorpeció en gran medida el desarrollo y causo problemas de desarrollo especialmente en las partes dependientes del proyecto. Además, implementar muchas clases hizo que el proyecto fuese robusto, desaprovechando memoria en cuanto una clase se tenía que instanciar una y otra vez para poder usarla desde otras clases.
- *No documentar métodos complejos:* Esto ocasionó que al implementar entregas siguientes fuese difícil volver a entender el código para hacer las modificaciones necesarias para los nuevos requerimientos de cada entrega.

¿Qué tipo de problemas tuvieron durante el desarrollo de los proyectos y a qué se debieron?

El principal problema que se tuvo por desconocimiento del uso de github desde el IDE de eclipse, por lo que el manejo de las ramas, y merge-conflict, causo que algunas veces secciones de código se borraran o sobrescribieran cuando no debían hacerlo, igualmente, el uso de JARS en el proyecto 3 fue conflictivo en cuanto fue complejo usarlo con ruta relativa

y no absoluta de cada máquina, y luego subirlo a github debido a que no se contaba con que el gitinore esta haciendo que los JARS no se subieran en el commit, lo que ocasionó el entorpecimiento en el desarrollo por fallas con estas tecnologías. Otras veces, había problema con la interpretación del enunciado, por ejemplo, en la entrega 1 donde no se especificaba el uso de impuestos, fue una observación que nos hicieron en la entrega. Luego, en la implementación de los indicadores, los nombres no eran dicientes y la interpretación fue propia del grupo de lo que creíamos que buscaban visualizar, aún así esto solo causo ambigüedad en la toma de decisiones, pero no en la implementación. Por último, existió la dificultad de que cada uno diseñó el proyecto sin realmente conocer el negocio por no haber tenido experiencia en proyectos como estos, por lo que la mayoría de los errores fueron causados en el desarrollo, pero se pudieron solucionar, aún así., se gastó tiempo considerable en estos. Por último, al usar diferentes IDEs en el desarrollo, el manejo de ellos paths relativos y absolutos fue un problema en la primera entrega del proyecto. En las siguientes entregas se solucionó cambiando la configuración de la lectura de todos los archivos.

Utilidad para grupos que hipotéticamente hicieran los proyectos nuevamente

Se reconoce que con la reflexión se pueden hacer las siguientes recomendaciones para la implementación de estos proyectos en un futuro a modo de conclusión:

- Desde el inicio adherirse a un patrón de diseño que implemente pocas clases con bajo acoplamiento y alta cohesión, preferiblemente centralizada con el propósito de facilitar el desarrollo
- Los métodos deben cumplir solo una tarea y esto facilita el entendimiento como la actualización del código
- Usar paths relativos en vez de absolutos con System.getProperty("user.dir") para ayudar a encontrar los archivos independientes de la máquina y el IDE que se use
- Incluir JARS no externos sino subirlos al proyecto y usarlos dentro de este
- Usar herramientas de gestión de código con git, como git kraken y evitar el uso de la consola para hacer commits y push en el código debido a que no se tiene control con el manejo de conflictos
- Tener buena comunicación grupal respecto a lo que se quiere diseñar e implementar para que todos estén en una misma página
- En lo posible documentar los métodos principales para que, al actualizar el código sea fácil identificar cómo funciona el método